Decentralising Active Network Management through Distributed Constraint Optimisation

Dimitrios Athanasiadis

Submitted for the degree of Doctor of Philosophy

Institute for Energy and Environment

Department of Electronic and Electrical Engineering

University of Strathclyde

Glasgow G1 1XW

Scotland, UK

February 2017

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed:

Date:

Abstract

Electricity networks are becoming more complex due to the introduction of distributed energy sources. The power grid will face fundamental changes in its structure and behaviour. In addition, new technologies will be required in order to maintain the stability of the network.

A key enabling technology for the future power networks is autonomous distribution networks which operate in cooperation with distributed grid intelligence and active network management. Local control can be achieved through fast control and communications and needs to be coordinated with the overall system controls. The primary issues for distribution network operation with a high penetration of distributed resources is power flow management as well as voltage control.

Active Network Management can be described as the control and management of generation and load in distribution networks. The main concept is to satisfy network limits such as voltage, power and frequency while at the same time increase the connected generation output with the minimum cost. There is a need for coordination between generators to maintain the balance in the network and avoid investment costs.

This thesis considers the introduction of Distributed Constraint Optimisation as a way of providing Active Network Management. It is an agent-based coordination method that is able to coordinate generators' output without violation of network constraints. This method, which is drawn from the Artificial Intelligence community and was previously used for smaller problems such as meeting scheduling, is studied and evaluated for use in power systems networks in order to provide solutions in a decentralised way.

Case studies consider both DC and AC power flow management and the solution of the economic dispatch problem. DC power flow management under DCOP provides optimal solutions for radial networks while AC power flow management is examined from a theoretical standpoint until the limitations of distributed constrained optimisation software are addressed.

Acknowledgements

I would like to thank my supervisor Prof. Stephen McArthur for his unbelievable dedication to my research. I want to thank him for his time, patience and knowledge, without which this research would not have been possible. Moreover, I would like to thank him because he believed in me, especially in difficult times when I thought there was a dead-end in my research. Finally, I would like to thank him for giving me the opportunity to undertake such an interesting PhD project.

Furthermore, special thanks to Dr. Ivana Kockar for her guidance and advice. I would like to express to her my gratitude and my respect.

Moreover, I would like to thank EPSRC for the financially support in order to finish my PhD.

Finally, I'd like to thank all my colleagues at the University for their help in bringing this PhD to completion.

Contents

List of Figures

List of Tables

List of Publications

1. S. D. J. McArthur, P. C. Taylor, G. W. Ault, J. E. King, D. Athanasiadis, V. D. Alimisis, M. Czaplewski. The Autonomic Power System – Network Operation and Control Beyond Smart Grids. In *IEEE Power Engineering Society Innovative Smart Grid Technologies Europe, 2012*, Berlin, Germany, October 2012.

2. Dimitrios Athanasiadis, S. D. J. McArthur. Active Network Management using Distributed Constraint Optimisation. In *IEEE Power Engineering Society General Meeting 2013,* Vancouver, Canada, July 2013.

3. Dimitrios Athanasiadis, Ivana Kockar, S. D. J. McArthur. Distributed Constraint Optimisation for Flexible Network Management. In *IEEE Power Engineering Society Innovative Smart Grid Technologies Europe, 2013,* Copenhagen, Denmark, October 2013.

Glossary of Abbreviations

AC	Alternating Current
ADM	Alternating Direction Method
ADOPT	Asynchronous Distributed Optimisation
A-DPOP	Approximate DPOP
AEP	American Electric Power
AFB	Asynchronous Forward Bounding
AMS	Agent Management Service
ANM	Active Network Management
APP	Auxiliary Problem Principle
BnB-ADOPT	Asynchronous Branch-and-Bound
CBR	Case-Based Reasoning
CCGT	Combined cycle gas turbines
CHP	Combined heat and power generation
CO_2	Carbon Dioxide
ConcFB	Concurrent Forward Bounding
CPA	Current Partial Assignment
CSP	Constraint Satisfaction Problem
DACR	Distribution Automation Circuit Reconfiguration
DaCSA	Divide and Coordinate Subgradient Algorithm
DALO	Distributed Asynchronous Local Optimisation
DBA	Distributed breakout algorithm
DC	Direct Current
DCOP	Distributed Constrained Optimisation

DER	Distributed Energy Resources
DF	Directory Facilitator
DFS	Depth- First Search
DG	Distributed Generation
DisCSP	Distributed Constraint Satisfaction
DMS	Distribution Management System
DPOP	Dynamic Programming Optimisation Protocol
DSM	Demand Side Management
DSO	Distribution System Operator
DSTATCOM	Distributed Static Compensator
DVR	Dynamic Voltage Restore
EDPOP	Stochastic Uncertainty using Collaborative Sampling
EMS	Energy Management System
FACTS	Flexible AC Transmission System
FIPA	Foundation for Intelligent Physical Agents
FPP	Flexible Plug and Play
GWh	Gigawatt-Hour
HC	Hydrocarbons
ICT	Information and communications technology
IEA	International Energy Agency
IED	Intelligent Electronic Device
INEA	Informatization Energy Engineering Automation
JADE	Java Agent Development Framework
KOPT	K-optima

kW	Kilowatt
LIFO	Last In First Off
MAS	Multi-Agent Systems
MGM	Maximum Gain Message
MPC	Model Predictive Control
MW	Megawatt
NCBB	No-Commitment Branch and Bound
NO_x	Nitrogen Oxide
O-DPOP	Open Dynamic Programming Optimisation
OECD	Organization for Economic Co-operation and Development countries
OPF	Optimal Power Flow
OptAPO	Optimisation by Asynchronous Partial Overlay
PCPM	Predictor-Corrector Proximal Multiplier Method
P-DPOP	Private Dynamic Programming Optimisation Protocol
PF	Power Flow
PFM	Power Flow Management
PM_{10}	Particulate Matters
PV	Photovoltaic
SCADA	Supervisory Control and Data Acquisition
SHEAP	Shetland Heat Energy and Power
SHEPD	Scottish Hydro Electric Power Distribution
SO_2	Sulphur Dioxide
SSTS	Solid-State Transfer Switch
STATCOM	Static Synchronous Compensator

SynchBB	Synchronous Branch and Bound
UCTE	Union for the Coordination of Transmission of Electricity
VVO	Volt -VAR Optimisation
WSN	Wireless Sensor Networks

Chapter 1

Introduction

1.1 Introduction to the research

Electricity networks in 2050 will be more complex and more difficult to design and manage, in terms of the technical, social and environmental issues. Power system networks are growing and changing significantly. The global expansion of sustainable sources of energy will have an impact on future power systems. The increased penetration of distributed generation will provide diversity to electrical power generation and load. The grand challenge for the future networks is the appropriate design in order to provide flexibility, controllability, stability and scalability.

Future power system grids will experience fundamental changes in their system structure and behaviour. Moreover, they will require enhanced control and automation, and IT-driven management functions. For this reason, new control technologies are needed in order to provide flexibility, dispatchability and voltage stability.

Future grids will integrate a wide variety of distributed generation. The connection of generation at the distribution level affects the flow of real and reactive power and voltage. The positive impacts can be: a) voltage support; b) improved system reliability; c) loss reduction; and, d) transmission and distribution capacity release [ZMA+15]. The integration of renewable energy sources but also energy storage and active participation of consumers brings more complexity to the power grid. Traditionally, many technologies were used to balance the system such as: a) transformer tap-changers; b) line regulators; and, c) shunt capacitors. Power systems were previously controlled by SCADA systems and control rooms, however, future technologies include multi-agent systems and active network management.

In future power systems, the Smart Grid technologies will bring in new types of entities such as distributed generation, Virtual Power Plants, customers with Advanced Metering Infrastructure, demand aggregators and microgrids. These entities are different from those in the current system in two aspects. First, the number of such entities is very large, and

they are distributed throughout the system and may be geographically far away from each other. Secondly, enabled by their hardware and software capabilities, these entities tend to behave autonomously to maximise their own benefits, and such an objective is not consistent with that of the current centralised dispatch. In view of both aspects above, it is difficult for the system operator to centrally dispatch the various types of entities by using the current operational framework. To better accommodate a large number of autonomous entities, and to take advantage of the potential benefits for individual entities to make active responses, decentralisation of the current framework needs to be considered.

Active Network Management (ANM) provides the smart solution for managing production and load in real time without significant expenses around upgrading the power systems infrastructure. The benefits of ANM include the detection of faults in the system, the ability to introduce more renewable energy generation in the network without systems infrastructure investments, the use of energy storage and the prevention of outages. [MKF+14]

Against this background, this research is focused on decentralised and distributed control techniques under ANM. The approaches include distributed constraint satisfaction and optimisation, and use of search algorithms in distributed constrained optimisation (DCOP). Previously, only distributed constraint satisfaction techniques were used in power systems [DMD+09]. The authors provided an algorithm which was able to change the output of power generation in order to maintain the balance of the network and minimise the cost. However, the technique was not decentralised as it involved a central authority that needed to know all the information from generators in order to provide a solution. For this reason, this thesis will introduce 3 case studies of a distributed and decentralised method that is able to control the generators' output without the need of a central authority, and that provide an optimal solution. The case studies include DC power flow management in a small example of 3 buses in order to present the process of DCOP, an 11kV network of 40 buses and finally an AC power flow management study.

Through the analysis and comparison of distributed intelligent methods, the most appropriate algorithms are evaluated and identified for applications such as power flow management and thermal constraint management.

1.2 Novelty of the research

The research presented in this thesis analyses the benefits of using decentralised and distributed techniques in power systems. The research provides all the details of how this can be implemented in power systems networks.

This thesis provides the following contributions to knowledge:

- An investigation of the challenges involved with the application of distributed techniques in power systems
- An investigation of active network management schemes in academia and industry
- An analysis of the DCOP framework where the agents can optimise a global function in a distributed fashion using local communication
- An analysis of all the existing DCOP algorithms
- An analytical example of applying the DCOP algorithms in power systems
- A case study indicating how the DC power flow management can be performed under the DCOP framework
- Analysis of the results and comparison between the algorithms in order to understand the differences among them regarding the computation time and messages that are exchanged
- A case study of how the AC power flow management can be obtained under the DCOP framework
- An analytical description of all the challenges regarding the FRODO software
- Proposals for avoiding the limitations of DCOP software
- Proposals on how to apply the DCOP algorithms without the use of distributed optimisation software

1.3 Thesis Overview

This thesis is arranged into six chapters. Chapter 2 illustrates the move from centralised to decentralised control. Centralised, decentralised and distributed techniques are detailed in order to understand the importance of distributed control. Examples of applications of these techniques are presented in order to compare and contrast the differences of the techniques. Additionally, Active Network Management Schemes are presented.

Chapter 3 follows with the introduction of Distributed Constraint Satisfaction to understand the background of the technique by providing examples and also algorithms. Chapter 4 illustrates the main concept of the thesis which is the Distributed Constraint Optimisation technique. Analytical descriptions of the technique are presented along with examples and algorithms within the framework.

Chapter 5 presents three different case studies and their analysis and evaluation. The first case study considers DC power flow management in a small network of 3 buses and the second case study provides an 11kV network of 40 buses. Algorithms for solving the economic dispatch are provided in order to understand the differences between them, regarding the time of computation and the number of messages that are exchanged. The third case study provides an AC power flow management example and analyses the problems of applying DCOP in the specific example. Furthermore, an analytical description of the FRODO software is provided in order to understand the software that is used to solve the optimisation problems. Finally, Chapter 6 summarises the work presented in the thesis and provides the future research required in the DCOP area.

Chapter 2

From Centralised to Distributed Control

2.1 Introduction

The electric power system is a physical network that plays a significant role in the lives of people across the world. In addition, the existing power grid is a system that contains components that have been operational for almost 50 years. The grid will face a number of different challenges over the next years and there is a need to address and face these challenges. The challenges include: a) increasing energy prices; b) increased availability of different generation; c) reduced system inertia; d) electric vehicles not only for loads but also as energy storage; e) penetration of distributed generation; f) a range of different distributed generation sources; and, g) increased demand side response [MTA+12].

In the move to a decarbonized energy network, the heat and transport sectors will be largely integrated into the electricity system. The grand challenge in future energy networks is to deliver the fundamental changes in the electrical power system that will support this transition [MTA+12]. These challenges need to be tackled in order to provide reliability, security of supply and consequently decreased costs.

The future power system, the smart grid, is expected to operate under an updated philosophy with a significant increase in the level of monitoring, communication, and control and coordination [Arn11]. There is no standard definition of smart grid, however, a digital technology that allows the communication between the utilities and the customers makes the grid smart. Furthermore, it's the use of advanced control technologies, integrated communications and improved interfaces. *Figure 2.1*, illustrates a future smart grid concept.

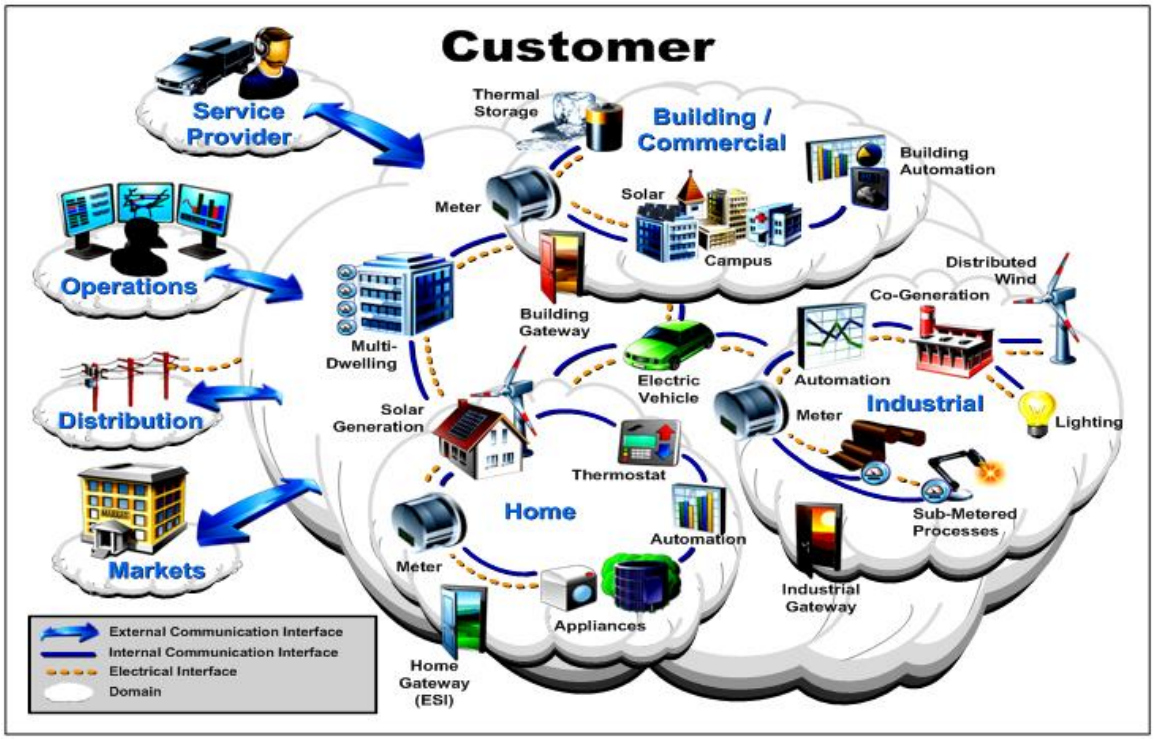


Figure 2.1: Smart Grid concept [NIST10]

One of the most important emerging challenges facing the grid is the need to incorporate more renewable generation. The renewable generation relies mostly on wind power and solar technology which makes it even harder for the system to balance generation and load as they are unpredictable sources of energy. Moreover, resource locations require renewable generators to be located far from the existing electricity grid which increases the cost due to the connection to the grid. The increased penetration of renewable energy sources will change the design and operation of distribution systems and will increase the cost for many customers.

We are moving from a highly-centralised to a more decentralised energy system relying on more distributed generation, energy storage and a more active involvement of consumers through demand response. The expansion of distributed generation will provide benefits to the utilities and the consumers but at the same time will provide challenges to the utilities. There is a need to combine the traditional practices with future techniques.

The integration of the new grid technologies will impose fundamental changes to the existing grid structure. The existing technologies are going to expand in both number and type. Moreover, the desired level of reliability will depend on the degree of control, automation and information technology.

The evolution of the power grid will enable the participation of electricity consumers. New emerging technologies such as smart meters and power line carriers embrace the communication with consumers and allow them to make decisions based on the real-time grid conditions. The integration of demand-response technology introduces new consumer-driven dynamical systems each with its own control loop.

Normally, the power systems network consists of meshed transmission networks and radial distribution networks. The distinction between these two networks lies in the fact that different requirements are needed for each network. The integration of distributed energy sources provides significant challenges on power flows and voltages of the network.

Although the transmissions system continues to introduce new control technology, perhaps the most evident upgrades appear in the distribution system. For instance, in the measurement and communication infrastructure, SCADA systems are quickly entering distribution. Also, smart meters, phasor measurement units, and dynamic line ratings have been the centre of attention in both academia and industry.

Taking into account a large-scale deployment of distributed generation, an active distribution network is needed to adapt the existing passive and less intelligent one. Designing the future grid should be based on the main requirements according to future circumstances. Firstly, the network needs to be efficient and flexible to cope with arising problems in operation such as bidirectional power flow, voltage rise, short-circuit current increase, or stability increase. The structure of the network should be designed in an adjustable and scalable way for varying needs. Another required characteristic of the network is intelligence in order to self-adjust and to be adaptable in autonomous operation.

Hence, the balancing between supply and demand could be controlled precisely in both normal and disturbance states. [POW15]

Flexible AC Transmission System (FACTS) technology is used mainly for power system control [GA12]. The integration and expansion of these devices will provide flexible and efficient solutions to the network. Other devices that are used in power system networks are a) Distributed Static Compensator (DSTATCOM) [AGM+10], b) Dynamic Voltage Restorer (DVR) [NNN+04], c) Solid-State Transfer Switch (SSTS) [CC14].

Another perceptive of the distribution system integrates the use of wireless sensor networks (WSNs). The power system networks are expanding and the need for new control schemes is essential in order to study the behaviour of the distribution systems. New techniques need to provide scalability and allow different smart grid applications to be integrated into the power system networks [ZR09] [ZHQ12]. Many techniques have been proposed for managing these networks and these include: a) Clustering, where the network can be studied if it is divided into different subnetworks b) Partitioning, which is the almost the same control scheme as clustering [ES10] and c) multilevel partitioning [KK98].

In [ES10] and [SJG11] it is identified that distributed control techniques can be used as an alternative approach to traditional control schemes and provide advantages in communication between the control entities.

Noncentralised intelligent techniques may provide the required flexibility to support the evolving control functions within the future electricity grid. These techniques can be decentralised with respect to power systems community but distributed for the Artificial Intelligent community. The idea of control hierarchy is based on the allocation of different units based on different time frames. [MM14].

The following sections will analyse the different types of techniques that are used in power systems.

2.2 Drivers for Centralised Control – Moving from Centralised to Distributed Generation

2.2.1 Centralised Generation

Central generation is the production of the electricity (bulk power) by power plants. These power plants include gas or coal or nuclear plants. Moreover, large hydro plants can be considered. Currently, the electric power system is operated in a centralised way in the UK.

However, electricity generation was generated by small power plants at the beginning of the 20th century. The technological advances in AC systems enabled the transition to bigger power plants hence the development of distributed energy was mainly for back-up services. [GE14]

There were many reasons behind the centralised electricity control. These include: a) high energy efficiency where larger power plants were capable of handling higher pressures and temperatures of steam used in electricity generation; b) environmental reasons where the use of transmission networks made it possible to build the generation power plants outside of cities hence removing pollution; and, c) pursuit of reliability where large power plants were connected to transmission networks.

Today, the technology improvements in distributed energy sources have resulted in the expansion of small scale renewable technologies as compared to the central power stations. Environmental concerns forced the electricity companies to take into account distributed energy sources and for them to be integrated within power system networks simultaneously with large central power plants.

2.2.2 Disadvantages of centralised generation

Several studies have tried to present and provide the disadvantages of using centralised generation [PDH^{+}05]. One of the main disadvantages of centralised generation is the transmission and distribution costs. Normally, the costs are higher for small customers at a low voltage level and smaller for industrial customers in medium voltage [IEA02]. These

costs are affected by line losses where the energy is lost when flowing in transmission and distribution lines, and also by conversion losses due to specifications of the network.

Regarding the energy efficiency, power stations built in the 1960s used high temperatures and pressures. This method provided the energy efficiency but at the same time resulted in wear and tear of steam turbines delivering lower than expected operating life [Hir89]. To tackle this problem and avoid the use of high pressures cooling systems were introduced. However, it is more difficult to transport steam and heat rather than electricity hence justifying the use of distributed generation close to consumption point.
The biggest disadvantage of centralised generation is the potential of failures that cause significant costs and also a lack of reliability and controllability. [Las11]

Even though centralised SCADA systems provide benefits in terms of power system control of hardware and software, some changes are needed in order to have a control capability with the integration of distributed generation. New communications infrastructure is needed in order to provide a reliable and sustainable framework, response times for the detection of switch changes needs to be faster which may prevent sequences being carried out in the required time and risk of single point failures need to be avoided.

One of the biggest disadvantages of the centralised generation is the investment in transmission and distribution networks. Over the next decades, the infrastructure of transmission and distribution networks needs to be upgraded. The total cost of investment until 2030 for Organization for Economic Co-operation and Development countries (OECD) is between 3,000 to 3,500 billion dollars according to the International Energy Agency (IEA) [IEA03]. To avoid these costs, the use of distributed generation could play a significant role. According to IEA scenarios, the cost of using renewable energy sources could be up to 3,000 billion dollars.

The transition from a centralised approach to a more decentralised approach will enhance new opportunities to provide security performance of the future electricity system.

2.2.3 Centralised Control Techniques

Optimal Power Flow and Economic Dispatch: The OPF is an intelligent load flow technique that adjusts the power system control settings while solving the load flows and satisfying network constraints. It uses advanced techniques that are able to satisfy power systems network constraints and provide optimal solutions. Lin presented a distributed optimal power flow with discrete control variables of large distributed power systems [LL08]. The algorithm contains two basic features: a) an algorithm to solve continuous distributed optimal power flow; and, b) an ordinal optimisation strategy to choose the appropriate discrete control variable solution. Mudumbai et al. presented a distributed control for the optimal economic dispatch of a network of heterogeneous power generators [MDC12]. In this algorithm, the frequency of generators is adjusted in order to balance any load fluctuations. It is proven that the algorithm is able to provide optimal results in a network without losses. Cai et al. introduced an economic dispatch using multi-agent systems [CNM12] and finally, Zhang and Chow solved the economic dispatch using incremental cost consensus algorithm [ZC11].

2.3 Distributed Generation

2.3.1 Introduction and advantages

The opposite of the centralised generation is called distributed energy production. There is a definition for decentralised generation from the European Union: “Decentralised generation can be defined as generation plants connected to the distribution system where the distribution system is the high-voltage, medium-voltage and low-voltage networks as opposed to the extra high-voltage and high-voltage transmission system” [EU09].

Decentralised technology is not a new concept and many utilities were using small units for decades. The changes in regulation, the subsidies and also the environmental concern contributed to the expansion and rapid growth of renewable energy sources. Also, the technological growth of these units and the consumer’s environmental concerns contributed to the move from centralised production of electricity.

Power systems can be more reliable by considering the load as an important energy asset. Traditionally, the observation of the system can be managed only through a central authority. However, the centralised scheme does not provide the adequate flexibility and reliability. These limitations can be avoided while still providing system level coordination through the use of decentralised controls based on local information. All elements of the power system including source, loads, and the network itself have influence, interaction, and coupling to all other elements. Local control is able to provide flexibility even if the connection with the system is lost.

2.3.2 Decentralised Generation impact on decentralised control

Despite the advantages of using distributed generation, there are some challenges that need to be tackled with the increase penetration of renewable energy sources. The current infrastructure of power systems was not built with the ability to integrate a large percentage of distributed generation. The increase penetration of distributed generation will drive a significant and radical change to the architecture of the electricity network, especially at the distribution level where the vast majority are connected.

Regarding the technical issues of the penetration of distributed generation, an analytical description is provided.

Capacity issues: The introduction of distributed generators at the distribution level influence the equipment such as cables, lines and transformers. One of the biggest problems affects the transformers and their behaviour. Most of the times, distributed generation is able to cover the demand in distribution networks but occasionally it can produce more generated electricity than is needed in order to cover the demand. This amount of electricity will flow through the lines and has to be exported to the medium voltage network or to the high voltage network depending on where the excess power is available. Therefore, the transformers need to able to handle this reverse flow. This situation is more noticeable during peak hours where production forecast from peaking distributed generators is key while determining the specifications of the equipment, as capacities will be added when the total power flow is already significant [PCF^{+}06].

Voltage issues: Typically, the distributed generators are connected to the distribution networks where there are low voltage networks and rarely to medium voltage networks. This affects the flow of the power and voltage profile which is increased and has to be within predefined acceptable limits.

Protection: According to [JAK+00] additional protection schemes are needed to avoid internal faults and islanding. Furthermore, according to [IEEE00], islanding is "A condition in which a portion of the utility system that contains both load and distributed resources remains energized while isolated from the remainder of the utility system."

The main advantages of islanding are: a) control of minimization of generation-load imbalance in an island and b) quick response of the system by integrating the islanded section into the whole power system network.
The biggest concern of islanding is the safety of personnel as they can be injured due to the presence of distributed generation. Furthermore, islanding can cause significant damage to the utility equipment. The main damage of the equipment can be triggered by the voltages and frequency which are not operated in satisfactory standards. If the grid has enough capacity to balance any voltage or frequency aberrations caused by the distributed generation, then customers close to the islanding will not be impacted. However, if the grid in the affected area does not have enough capacity to correct the aberrations, then customers close to islanding may have noticed imbalances in voltage and frequency levels.

Transmission and distribution losses: The difference between centralised and decentralised generation is based on losses. The main benefit of distributed generation is the limited number of losses as they are connected to the distribution network. However, a high penetration of generators in a specific area will have a consequence in losses and despite the fact that the connections are in distribution level the losses will eventually increase [MRG+02].

Reactive power flow: Reactive power flow generated by distributed generation units may affect the network losses and voltage, either positively or negatively. If the reactive power is generated near the system load then the losses are decreased, however, if the reactive

power has to be delivered to the transmission network then the losses are increased. Therefore, network management is needed in order to concern the reactive power [BCF08].

Voltage and current transients: Normally, fluctuations of the voltages can occur when distributed generation sources are either powered on or off. The effect of this situation may provide a destabilisation of the system.

Power quality: The introduction of distributed generation can cause significant changes to the power quality. The effect on power quality depends on various reasons. The type of DG can affect the power quality and also the size of the DG. Normally, DGs are installed to provide backup generation hence improve the power quality of the system. However, significant disadvantages arise on power quality with the expansion of distributed generation. The main disadvantages include: a) voltage regulation (the use of inverters and induction generators are causing violations of voltage because they're not capable to provide reactive power hence reduce the voltage levels), b) voltage flicker (fluctuation of energy from wind turbines and photovoltaic sources, and connection and disconnection of induction generators from the network), c) voltage dips (can be caused at the start-up of induction generators) and d) harmonics (DGs based on induction and synchronous generators such as are PV panels are able to change the harmonics impedance). [HSH10]

Climate change and reduce emissions: Even though distributed generation is cleaner than centralised large generation plants it does not mean that it is always a clean generation.

2.4 Decentralised and Distributed control techniques

2.4.1 Decentralised control

One of the main aspects of decentralised control is that there is no communication between distributed energy sources. This feature mainly affects the performance of the power system network. In any abnormal situation, each source will be responsible only for its part of the network while there is no communication between them.

The existing decentralised control techniques include:

a) Droop-based methods where the advantage lies in the absence of communication and control decisions are based on local measurements [BBV+07], [LVL04]. These methods are used for frequency and voltage control. Moreover, since there is only local measurement and no communication then control of the system can be achieved with the allowance for small errors. This small error or small droop as it is called can be acceptable if it doesn't violate the predefined limits [VGL+09]. However, these techniques have several disadvantages [GVM+11] such as: 1) poor transient performance; 2) lack of robustness due to inability to account for load dynamics; 3) absence of black-start for system restoration; 4) inability to impose a fixed system frequency independent of system loading conditions; and, 5) partial coupling of real and reactive power. Several approaches have been implemented in order to overcome all these limitations such as: 1) Adaptive decentralised droop controller to preserve power sharing stability of paralleled inverters [MS08], 2) Sharing of the nonlinear load in parallel-connected three-phase converters [BBE01] and 3) Autonomous load sharing of voltage source converters [SL05].

b) Master/slave control technique: This method is flexible regarding the connection and disconnection of distributed energy units, however, it requires a dominant unit for satisfactory operation [LMM05].

c) Robust servomechanism control method: This control method introduces a servomechanism controller for the operation of a single distributed generation islanded system. However, despite the fact that the method provides stability, it cannot be used in power system networks with various distributed energy sources. [KDI10]

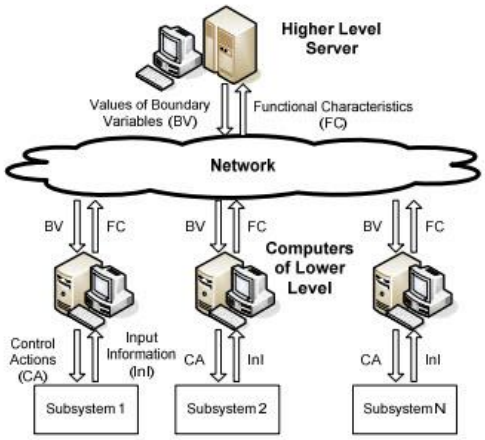


Figure 2.2: Configuration of the distributed control system [Sou07]

2.4.2 Distributed control

Distributed control techniques are based on communication between the distributed energy units. Before analysing these techniques, a detailed description needs to be presented. The main idea behind distributed control was to provide solutions for integrating large amounts of electricity generation without the disruption of the utility network [Las02], [HAI+07].

Large power systems consist of power systems of independent countries and independent electricity markets. The most important problem, which should be solved in the creation of such a system, is the ability to achieve global optimality of interconnection. This problem can be solved in distributed control systems by having a hierarchical structure and containing controllers, belonging to different hierarchical levels of the system. *Figure 2.2* illustrates the configuration of a hierarchical distributed control system. [Sou07]

In the solution of power flow and optimal power flow this system operates in each iteration in the following way: a) Data on parameters of functional characteristics of all subsystems are delivered from computers of lower level to the upper-level computer ; b) The solution is executed by computer at the upper level and data of the values of boundary variables are sent to lower level computers; and, c) Calculation of the values of internal variables of subsystems is executed by computers at the lower level.

The calculations in the lower lever computers are executed simultaneously and data delivery between these computers are subject to boundary variables.

The hierarchical control of microgrids contains three control levels according to the Union for the Coordination of Transmission of Electricity (UCTE). *Figure 2.3,* illustrates these three control levels: Primary control, Secondary control and Tertiary control.

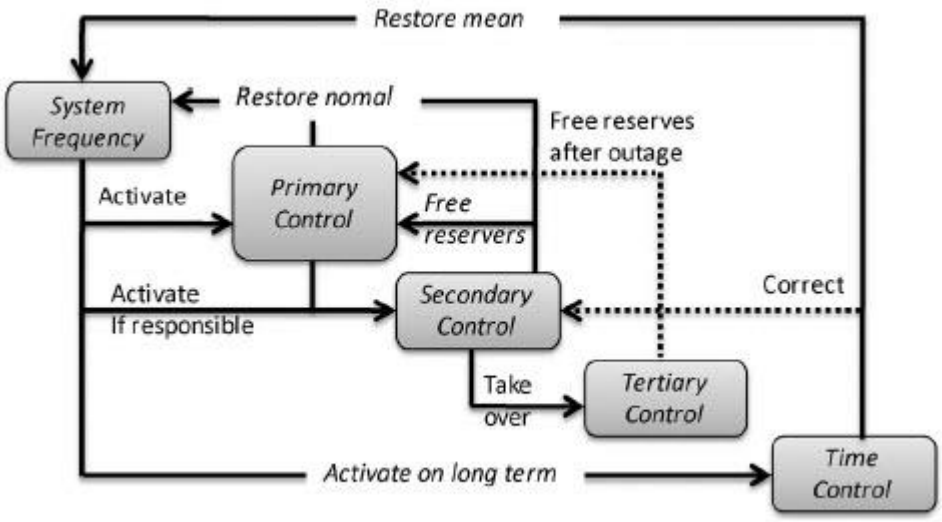


Figure 2.3: Hierarchical control levels of microgrid [EN10]

Primary control is the first control level and is the fastest response of the system. Primary control is responsible for ensuring the voltage and frequency do not violate their statutory limits. Moreover, this type of control includes power sharing and islanding detection. In addition, due to the absence of communication, all the signals are measured locally and typically the droop-control method is used. [GCL+13]. The disadvantage of this method is that there is a trade-off between power sharing and voltage regulation. Normally, this problem can be solved in the second type of control level, however, in the literature, some authors discuss the use of communication in primary control level such as active load sharing using communication [SWL+06] and master-slave methods [CS06].

Secondary control, known as central control or energy management system (EMS) is responsible for power quality, mitigating long-term voltage and frequency deviations, and

coordination of the individual distributed energy sources units for their optimal operation by determining the set points for the primary control. Mainly it is responsible for keeping the required values of the electrical levels in the microgrid. Moreover, it can contain a synchronisation control loop for connection or disconnection from the distribution system. The EMS is communicating directly with the distribution management system (DMS) through a communication link. Furthermore, secondary control objectives may include power quality control, voltage profile control, reactive power sharing, and loss reduction.

Secondary control is used to compensate frequency and amplitude deviations. In this type of control, it is ensured that frequency and voltage deviations are regulated toward zero in any change of generation or load within the microgrid. These deviations are monitored and compared to the reference values of frequency and voltage and the values are sent to the units in order to restore the output voltage and frequency. Typically, the secondary level of control is used in power systems in order to correct the frequency of the grid.

In the literature, different types of methods have been introduced for the secondary control level. Savaghebi et al. proposed a control scheme for voltage unbalance compensation in an islanded droop-controlled microgrid [SJV$^+$12]. Mehrizi and Iravani proposed a potential-function based control of a microgrid in islanded and grid-connected mode. More analytically, a central controller defines the functions of each distributed energy units. The central controller is communicating with the units which take the measurements and provides them with set points [MSI10].

The secondary control should be designed to operate on a slower time than that of the primary control to: a) decouple secondary control from primary control and b) reduce the required communication bandwidth between the system components. Galvin electricity initiative [Gal07] defines master controller specifications for perfect power systems during major disturbances such as the transition from grid-connected mode to islanded mode.

Tertiary control is responsible for the power flows between the microgrid and the grid. It is the slowest level of control and defines the long-term set-points based on distributed energy units.

In [GCL+13], Guerrero et al. presented the architecture for intelligent grids. Regarding the control levels of microgrids, they defined inner control loops as an additional type of control. This control level includes the regulation issues of each mode and linear and nonlinear control loops are performed in order to balance the output voltage and manipulate the current to keep the system stable.

2.4.3 Distributed Control Techniques

In this section 4 different types of techniques will be analysed. These include: a) Agent-based techniques b) Distributed Model Predictive Control-Based Techniques c) Consensus-Based Techniques and d) Decomposition-Based Techniques

Agent-Based Techniques: A distributed control strategy that belongs to the agent-based framework is multi-agent systems (MAS). Multi-agent systems is one of the most promising approaches to tackle the control issues within power network [WW99], [Woo09], [MDC+07], [MDCD+07]. A lot of work has been done regarding thermal constraints, voltage control and distributed optimisation for decentralised control schemes [TMA+08], however, there is still a missing gap where all the issues need to be solved concurrently. MAS offer a way to develop flexible, extensible and robust hardware and software systems; as well as being a powerful modelling tool [MDC+07], [MDCD+07]. MAS have already been applied to Smart Grid applications such as voltage control and power flow management [BM07]. Although these projects have provided advances in single power system control functions, there is a need for methods to coordinate multiple functions to achieve holistic control. Multi-agent systems are able to provide autonomy where no human interaction is needed, a social ability where there is a common language between the agents and awareness where they are able to communicate each other and respond to the environment changes. There are two different categories of agents and these include: a) decentralised agent architecture where there are different types of agents; and, b) hierarchical agents which consist of different layers of agents. Agents programming in

power systems is usually carried out in the Java Agent Development framework (JADE) [BCP+03]. The Foundation for Intelligent Physical Agents' (FIPA) standards have been used by developers in multi-agent community. FIPA aims to define specifications and standards that can be used to support interoperability between agent-based systems developed by the different companies and organisations [FIPA02]. *Figure 2.4*, presents an Agent platform on a COM6xx with AuRA-NMS control approaches.

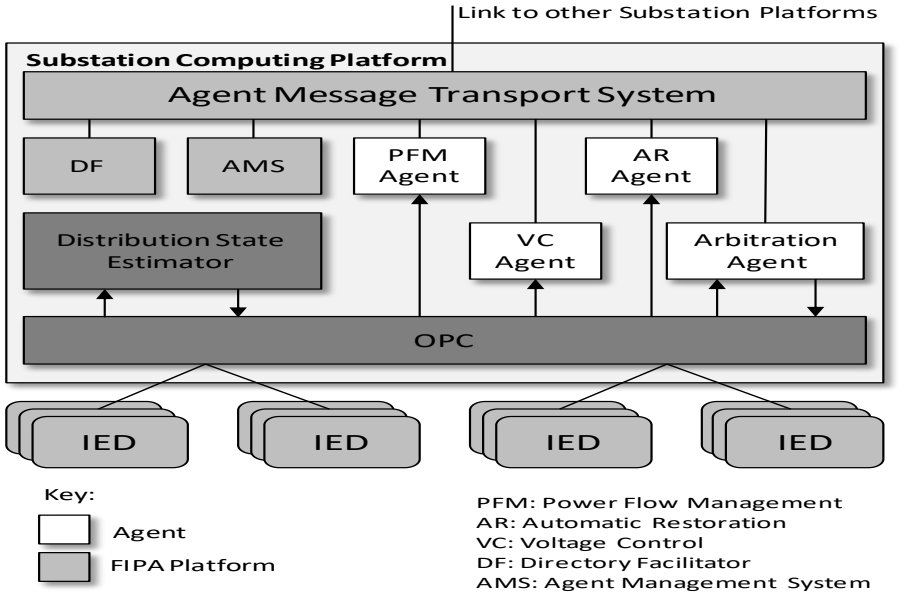


Figure 2.4: Agent platform on a COM6xx with AuRA-NMS control approaches [DDA+10]

Each agent platform includes two utility agents: the agent management service (AMS) agent, which is compulsory, and the directory facilitator (DF) agent, which is optional. The AMS contains a directory of agents registered with the MAS platform. The DF maintains a directory of agents and the services they can offer other agents.

Despite the fact that multi-agent systems can provide flexibility, reactivity, pro-activeness and scalability there are some disadvantages that occasionally prevent the use of this technique. A common ontology and a globally accepted standard is missing, which support the interoperability of the individual hardware pieces and information systems. MAS also have been used in microgrids for control and management. Logenthiran et al

proposed a multi-agent system for energy resource scheduling of an islanded power system with distributed resources, which consists of integrated microgrids and lumped loads [LSK11]. Dimeas and Hatziargyriou also presented a multi-agent system for microgrids and the system is being operated in Kyfnos island in Greece [HAI+07], [DH04].

Distributed Model Predictive Control-Based Techniques: Model predictive control (MPC) is an online optimisation technique that performs real-time optimisation to determine the control inputs [Mac02]. The main advantage of this method lies in the manner of the constraints that are incorporated and also the stability and robustness that they provide [Ros03]. For meshed electricity grids, separate controllers are used to control functions in the system rather than one central controller due to computational constraints. MPC optimisation is used in many applications. Wen et al. used a centralised MPC approach for optimal coordinated voltage control for power system voltage stability using an Euler state predictor [WWT+04] while Negenborn et al used a nonlinear MPC for emergency voltage control using pattern search [NLS+09]. Finally, Beccuti et al. presented a Lagrangian decomposition algorithm for optimal emergency voltage control [BDA+10]. The MPC voltage controllers can be categorised either in centralised or decentralised control. The centralised approach is able to provide optimal solutions, despite the computational cost and communication failures. However, the MPC decentralised methods are not able to provide optimal solutions.

As for the microgrids, Falahi et al. presented an MPC-based strategy for reactive power control in unbalanced microgrids [FBE13]. This method predicts the voltage profile in the next time steps and adjusts the voltage and reactive power set points to achieve a smooth voltage profile. Furthermore, Tan et al. presented a coordinated control and energy management of distributed generation inverters in a microgrid where they solve the problem in a slower time frame and they reduced the computational cost [TSC+13].

Consensus-Based Techniques: These techniques are able to solve distributed optimisation problems and provide scalability and extendibility [OFM07]. The technique is purely distributed and the global optimality is reached using limited communication between the

neighbour units. Xu and Liu, presented a multi-agent based load restoration algorithm for microgrids [XL11]. The main idea of this algorithm lies in the communication with neighbouring agents to find the global optimality using an average consensus theorem. Moreover, Dou and Liu presented a multi-agent based hierarchical hybrid control for smart microgrids [DL13]. This control scheme contains an upper-level energy management agent, several middle-level control agents and many lower level control agents. Different control strategies are designed in order to maintain the voltages in predefined limits and maximise the economic and environmental benefits.

Decomposition-Based Techniques: These types of techniques are based on decomposing the original optimisation problem into a number of subproblems that are solved iteratively until convergence. There are many decomposition techniques such as: a) auxiliary problem principle (APP) where the solution can be obtained by solving a sequence of auxiliary problems; b) predictor-corrector proximal multiplier method (PCPM); and, c) alternating direction method (ADM) where the algorithm solves convex optimisation problems by breaking them into smaller pieces, each of which are then easier to handle [CCM+13].

2.4.4 Applications of distributed techniques

In this section applications of distributed techniques are discussed covering voltage coordination, and frequency coordination.

Voltage Coordination: The rapid penetration of distributed generation led to the issue of voltage coordination. Several techniques have been introduced in order to tackle this problem. Vovos et al. presented and compared distributed and centralised methods to control distribution network voltages in terms of the capacity of the distributed generation that could be integrated in the existing networks [VKW+13]. Anand et al presented a distributed control solution to ensure proportional load sharing and improve voltage regulation in low-voltage DC microgrids [AFG13]. Finally, Vaccaro et al. introduced a decentralised and cooperative architecture for optimal voltage regulation in smart grids [VVZ11]. According to their results, smart controllers are used in order to detect voltage violations since the performances of the buses and the global performance of the grid is

monitored and computed by exchanging information between local nodes. These controllers calculate local bus variables and the distributed optimisers are performing calculations based on these variables in order to estimate the appropriate actions in order to minimise the objective function. Robbins et al. presented a two stage algorithm where in the first stage the units are injecting reactive power to the system in order to avoid voltage violations [RHG13]. However, if the problem still exists then, in the second stage, local controllers request additional react power injection from neighbouring units.

Frequency Coordination: The objective of frequency control is to have different distributed energy units converge to a common frequency. Each unit may have its own minimum and maximum values for power and terminal voltage. An approach which is typically used for frequency regulation of DER units is droop control but there are many disadvantages of this method as they mentioned earlier in the Chapter. Dai et al. presented a coordinated primary frequency control among non-synchronous systems connected by a multi-terminal high-voltage direct current grid [DPS+12]. The control scheme modifies the power injections from AC areas into the DC grid to make the system reacts to load imbalances. The control scheme is using algorithms from the control theory community such as consensus problem algorithms.

Finally, Shafiee et al. presented a distributed secondary control method for islanded microgrids [SVG12]. Their novel approach is different to the conventional approach of using a central controller by introducing a distributed network control system. The approach restores the frequency and the voltage but at the same time ensures reactive power sharing.

2.4.4 Conclusions

Power systems are classified as large-scale and highly non-linear systems. Centralised controllers are not adapted to control such systems because the global information of the entire system is not fully available in a centralised way to allow coordinating the control activity of the overall system. Additionally, centralised controllers are technically and economically very difficult to design and implement for power systems. On the other

side, decentralised control schemes are proposed since power systems can be characterised by an interconnection of many subsystems. In fact, global central controllers can be substituted by local controllers designed especially for each sub-system. The main goal of decentralised control is to find some feedback laws to adapt the interactions from the other subsystems where no state information is transferred. The advantage of decentralised controller design is to reduce complexity and minimise the amount of information that is exchanged, which leads to better feasibility for the control implementation. However, decentralised control cannot access the entire systems information like centralised control. Therefore, interconnections between subsystems need to be analysed, so that their influence on the system performance can be properly addressed by the control.

Typically, power system networks are operated in a centralised way and decentralised technology is taking into consideration. However, with the rapid increase of the renewable energy sources, new control schemes are needed in order to manage the direction of the power flows. There is a need for new intelligent methods that will be able to contribute to this transition in order to provide the grids of the future.

Furthermore, regarding the communications between network units, new schemes are required because the existing centralised schemes will not be able to provide the adequate communication after the expansion of renewable sources. Decentralised schemes provide local communication, hence there is no need for communication between the different parts of the system.

In addition, there is a difference in protection schemes on both centralisation and decentralisation. The difference lies in the technology that is used in the distribution feeders. Centralised approaches use circuit breakers while decentralised approached use autoreclosers. Autoreclosers are able to provide a fast solution of fault isolation because they isolate the fault only on the faulty device. Finally, decentralised intelligence schemes represent a desirable answer for protection, reliability, and control of today's power systems.

The majority of control and monitoring tools in present power systems are provided by Energy Management Systems (EMS) and are based on steady state system model, which cannot capture the dynamics of power system very well. This limitation is primarily due to the dependency of EMSs on slow update rates of the SCADA systems. The challenge

in implementing dynamic estimation and global control schemes is the unavailability of a fast, reliable and secure communication network.

Today, central control systems are facing many challenges that are related to the number of control variables, the complexity of the system and the demand of adaptability to a dynamic and uncertain environment. Consequently, the software developed for these applications is more complex and extend in size. Current control and management of the operations is achieved by SCADA systems and also Distributed Control Systems. However, more intelligent approaches are needed for monitoring and controlling the electricity grid. For this reason, Active Network Management is a promising area that uses flexible network customers autonomously and in real-time to increase the utilisation of network assets without breaching operational limits, thereby reducing the need for reinforcement, speeding up connections and reducing costs.

2.5 Introducing Active Network Management

Power industries need to provide a more visible electricity grid to deliver reliable power. In addition, they have to provide new intelligent technologies that incorporate renewable generation and mainly low carbon technologies to satisfy the demand. One way to achieve the above assumptions is to invest in grid infrastructure which is costly and mainly will affect the consumers. However, there is a way of avoiding all these costs with the solution of active network management.

ANM is a solution that allows a significant number of renewable energy generators to be connected in distribution networks without the need of network reinforcement. A solution for monitoring and controlling a network in real-time to introduce new connections and to ensure it satisfies operational constraints. ANM provides the minimization of new grid infrastructure investment while having the flexibility to adapt for future expansions. Furthermore, the real-time condition monitoring of system components such as power flows and voltages across the network is ensured while satisfying network constraints on transmission and distribution systems.

There is momentum in electricity industry to incorporate ANM schemes. The key reason behind this implementation involves: a) technological improvements (monitoring, communications and computational technology is faster, reliable and cost effective with

ANM); b) facilitating renewables (renewable generation is a top priority); c) operational solutions have the potential to be more cost-effective and less carbon intensive; and d) unpredictable and complex power flows become more manageable through ANM schemes.

The architecture of an ANM scheme can be defined as either decentralised or centralised. The difference between the two is found in the integration method, with remote devices performing algorithms in decentralised schemes and centralised schemes being more embedded within central network control.

An ANM scheme is considered to be decentralised when Programmable Logic Controllers (PLCs) or relays and Remote Terminal Units (RTUs) are sharing information among the devices. These device monitors are able to provide control signals based on the data they receive by incorporating algorithms in a decentralised way. The local control is performed automatically and autonomously.

A centralised ANM scheme can be defined by the coordination of the ANM actions from a central algorithm. The control room NMS will receive input signals from the SCADA system with which the scheme is integrated, and the ANM algorithm then processes these inputs to determine any required actions and issues appropriate control signals to the ANM-connected customer assets via the SCADA system.

2.5.1 Challenges for the industry

Power industries are facing challenges that need to be tackled in order to bring forward the new smart technologies. Moreover, they have to provide the appropriate electricity demand, they have to minimise the upgrades of the grid and they have to integrate the new technologies faster, smarter and economically.

The ANM philosophy lies on the move from the fit-and-forget passive approach to network planning, towards an active approach where autonomous real-time control of parameters maintains the network within secure operating conditions. The most common network constraints are generally thermal and voltage constraints, where control of generator real power export, reactive power generation or absorption, or transformer tap changer configurations can be used to manage power flow and voltage levels.

According to Smarter Grid Solutions [MKF+14], ANM will provide many benefits to the power industry and companies regarding grid management and control, such as: a) identification of voltage and thermal constraints through controlled devices; b) maximization of renewable energy generation; c) estimation of real-time ratings based on weather conditions and environmental constraints; d) coordination of devices to provide security of supply; and, e) implementation of pre-fault and post-fault actions in response to changes in the grid.

Furthermore, smart grid technology requires autonomous and reliable applications. Solutions must be scalable and configurable and able to support the integration of external systems using standard protocols. Also, new emerging solutions need to tackle all the changes to grid configuration in operational timescales.

One of the main advantages of using active network management scheme is the monitoring and controlling of the grids in real-time. ANM is able to provide the appropriate technologies to maintain the electrical balance in the grid. These technologies may include: a) integration of distributed energy sources; b) predictable results over short and long timescales; c) coordination between different devices to ensure real-time information and measurements; and, d) real-time monitoring techniques.

Finally, regarding the loss of the communication, this can be categorised in local loss and loss of the central network. The local loss will have an impact to all the AMN customers serviced by the failed communication and ANM customers will either be disconnected or constrained to a safe output level. The loss of central network communications is more critical and is different in centralised and decentralised ANM architecture. In centralised architecture, customers will either disconnected or constrained to a safe output level while in decentralised architecture some customers will be affected and others not.

It is with great interest that many projects across UK, Europe and USA are deployed regarding smart grids area and especially active network management schemes. Some of these projects are: 1) AuRA-NMS; 2) NINES; 3) Orkney ANM Scheme; 4) UK Power Networks Flexible Plug and Play; 5) ADINE; 6) ADDRESS; 7) ACTIVE Network; 8) GRID4EU; 9) Ecogrid; 10) IDE4L; 11) DISCERN; 12) E-DEMA; and, 13) S3C. All these

projects will be presented and analysed in order to provide the benefits of using such technologies.

2.5.2 UK ANM projects

In this section, UK projects will be analysed and presented. These projects include a) Autonomous Regional Active Network Management System (AuRA-NMS) b) Orkney ANM Scheme c) Northern Isles New Energy Solutions (NINES) and d) UK Power Networks Flexible Plug and Play.

2.5.2.1 Autonomous Regional Active Network Management System (AuRA-NMS)

The Autonomous Regional Active Network Management System (AuRA-NMS) project was a collaboration of 7 universities (Imperial College London, University of Strathclyde, University of Manchester, University of Bath, University of Edinburgh, Loughborough University and Durham University), two distribution network operators (EDF Energy and SP Energy Networks) and one manufacturer (ABB) [Gre09].

The Autonomous Regional Active Network Management System is autonomous and regional because it is decentralised and devolved from the network control centre and operates across a region [Gre09]. AuRA-NMS represents an active network management solution that can be applied to a variety of networks and deal with applications such as power flow management, steady state voltage control, restoration and minimization of losses [DMM$^+$09]. The AuRA-NMS solution was designed to provide flexibility and extensibility [DM07]. Flexibility is the ability to reconfigure the network management system in any change such as network topology, connection or removal of new generation or energy storage, protection and control equipment and installation or removal of measurement and monitoring equipment. Extensibility is the ability of the active network management to add new control philosophies and techniques in the future and replace existing control techniques. Furthermore, designed to be safe and secure also to have the ability to exhibit graceful degradation in performance during adverse or unanticipated network conditions [DMM$^+$09]. Power flow management involves the management of distributed generation such that the thermal limits of plant and equipment are not

exceeded. Thermal limits place a limitation on the firm DG connections that a network can support without having to be reinforced. ANM techniques, such as curtailment of generator output, offers a means of connecting more DG than the level allowed by conventional power system planning techniques [DDA+10] [DM07]. The tests, simulations and the corresponding results show that by modifying the standard formulation of the OPF problem, it is possible to detect and solve network thermal overloads whilst adhering to a last-in first-out commercial connection agreement. It is worth noting that the OPF priority is to balance network power flows whilst meeting line inequality constraints. Therefore, the commercial agreements are adhered to only when the network technical constraints permit. Although industrial experience proves the OPF solution is accurate through employment as a planning tool, the move to an operational implementation has some implications attached. OPF has the necessary generic characteristics in that it can cope with any network topology. In the case study investigations, it is assumed that all data necessary for load flow convergence was available. In reality, the data maybe skewed or in fact erroneous.

One of the key problems encountered when connecting generation to distribution networks is variation in voltage outside statutory limits, in particular, voltage rise. The control measures considered for coordinated voltage control within AuRA-NMS were: changing transformer tap position; changing PF set-point of distributed generation; and control of the real power output of DG. *Figure 2.5* illustrates the power flow management software with constraint satisfaction programming.

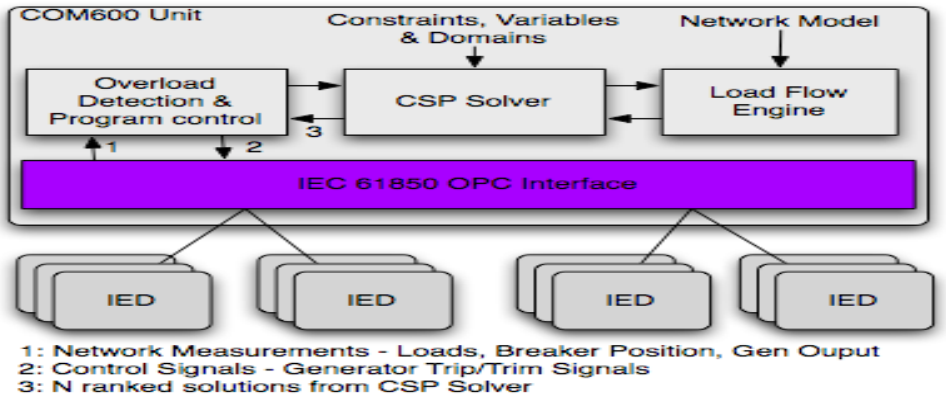


Figure 2.5: Power flow management software running on a COM6xx unit using constraint satisfaction programming [DDA+09]

A key part of the AuRA-NMS is that it integrates a number of disparate approaches to a number of different power system management and control tasks. T. Xu et al. developed a technique for voltage control using case-based reasoning [XPG+9] [TXM+08] and M. Dolan et al. [DDA+09] developed techniques for managing power flows in active distribution networks within thermal constraints using constraint programming and current tracing. These were all integrated in AuRA-NMS [DDM+09]. In order for CBR to be successful, the relative importance of case features, i.e. measurements, must be characterised to allow case matching. This means CBR can be relatively immune to errors in less important measurements. It can return several ranked solutions to a given problem, also possibly meeting the requirement for graceful degradation. The key challenge in the CBR approach is the method for a population of the case base and determining the relative importance of features. It is these that will determine the performance of the CBR system as a whole. The prototype CBR software is able to: detect a voltage excursion and formulate a problem case; retrieve the N most similar cases from the case base based on the problem case, and evaluate whether or not the proposed solution will remove the fault from the network using online simulation.

The CSP approach offers a number of ranked solutions. Should the preferred solution not mitigate the effects of an overload because of model error, the software can then try the next best solution, curtailing more generation until the overload is mitigated. As long as the overload can be mitigated before the protection trips thermally overloaded plant, then the approach will degrade gracefully. OPF can provide only one solution. If the solution is not valid, then a power flow management scheme may fail to operate and result in a trip due to overload. In the case of OPF, last-in first-out access arrangements are not expressed explicitly but as an artificial cost associated with each generator [DDM+09] [DDC09].

2.5.2.2 Orkney ANM scheme

The Orkney Smart Grid involves the application of a novel ANM scheme to enable increased renewable generator connections to the existing Orkney network. A collaborative project between the University of Strathclyde and Scottish Hydro Electric

Power Distribution plc resulted in the initial specification of the Orkney active network management (ANM) scheme [CAF+10].

The existing network arrangement was established to meet local demand, with the minimum of operator intervention and was not designed for the volume of generation it is presently facing. The local grid infrastructure and two 33 kV submarine cable connections to the mainland to export this energy have had all available capacity allocated. The Smart Grid, using ANM algorithms, has released a minimum of a further 15 MW of grid connection capacity [CAT05] [ZB07]. In the Orkney ANM scheme, the generators are grouped in a way to curtail their output generation if the constraints are violated according to the Last In First Off (LIFO) principle.

The Orkney ANM scheme makes use of a simple and deterministic set of rules and a recursive algorithm to ensure constraints are met. The ANM scheme could involve the application of sensitivity factors [Bia96], current tracing [DMD+09], the use of optimisation techniques [CAF+10] or artificial intelligence techniques [GCJ+09].

Demand customers on Orkney are not likely to experience any drop in quality of supply as a result of active management scheme introduction. System security issues have been assessed and it is thought that the active management scheme addresses concern over this issue.

2.5.2.3 UK Power Networks Flexible Plug and Play

The Flexible Plug and Play (FPP) project was initially started in 2012 and officially finished at the end of 2014 by UK Power Networks and funded by Ofgem. The total cost of the project was £9.7 million [UKP14]. The purpose of the project was to provide the integration of distributed generation in distribution networks without the need of grid investment. The aim of the project was to allow flexibility in generation output without violating network constraints. A distribution network in East of England was chosen which had the appropriate characteristics for applying an active network management scheme. The area contained 90MW of connected wind generation at 33kV, a combined heat and power generation plant and additional 57MWs of generation were under construction or under request for planning. One of the reasons for picking this network

was the fact that the electricity network was about to reach its operational limits hence make it ideal for introducing an active network management scheme [GA14].

After 3 years of trials, the project has significantly achieved many goals such as:

- 45 Distributed generation connections requested
- 39 connection offers of 176MW were issued
- 14 customer acceptances of the flexible connection

The trials managed to address a number of challenges while deploying the ANM. Two of the most important issues include the mitigation of thermal constraint on 33 kV overhead line and the mitigation of reverse power flow constraint of a 132/33 kV grid transformer.

The trials prove that the ANM scheme was able to deal with abnormal network events and also the capability of accommodating connection of new generation within the project timescales [FCC+11].

ANM has been proven to manage distributed generation connections even when it is integrated with other smart solutions. Smarter Grid Solutions was the company involved in the project, providing active power flow and active voltage management solutions. Additionally, the ANM scheme coordinates smart devices such as 1) Automatic Voltage Control relays; 2) Dynamic Line Rating relays; and, 3) Quadrature Booster Control System. These applications were successfully deployed and provided smart decisions for delivering additional generation capacity. Furthermore, the connection costs were reduced by up to 90% for 6 new distributed generation customers.

2.5.2.4 Northern Isles New Energy Solutions (NINES)

The NINES project was a three-year project led by Scottish Hydro Electric Power Distribution (SHEPD) in cooperation with Hjaltland Housing Association. Other contributors to the project include the University of Strathclyde, Shetland Heat Energy and Power (SHEAP) and Smarter Grid Solutions [DAF+12]. The aim of the project is to increase the amount of wind generation that is able to connect while reducing the amount of electricity demand and reducing of fossil fuels generation in Shetland Islands. The

Shetland Islands are not connected to the Great Britain electricity system hence they electricity is generated locally. Demand varies from 11MW to 48MW and heating can be provided only by oil and electricity as there is no gas supply. The electricity generation is provided mainly by 2 power stations which are able to produce a maximum of 88MW. Moreover, there is one wind farm of 3.68MW capacity and various small-scale wind generators. *Figure 2.6* illustrates the overview of systems architecture [OFM+13]. This system contains a) replacement of storage and water heaters in 1000 homes with modern storage heaters in order to balance the electricity network b) addition of electric boiler (4MW) to the heating system c) integration of ANM scheme to allow more renewable generated electricity to be added to the network d) installation of 1MW battery and e) introduction of new commercial arrangements to encourage businesses to change the times at which they use most energy.

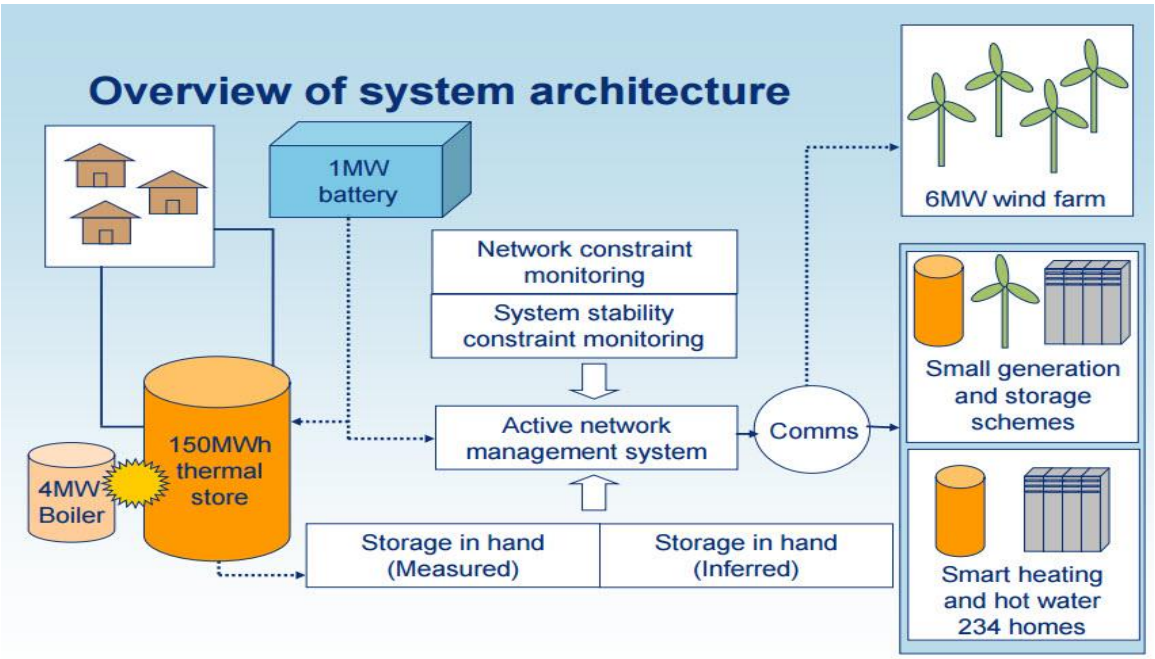


Figure 2.6: Overview of NINES system architecture [OFM+13]

The Shetland's community will benefit from this project as it will provide a) secure and reliable supply b) cleaner environment and c) lowest electricity costs in the future.

The ANM scheme in Shetland produces a schedule every 24 hours based on the demand and generation forecasts for all the controlled devices with respect to network constraints,

device constraints, system stability rules and balancing and scheduling rules. Based on the schedules, the ANM scheme is able to maximise the generation capacity of renewable energy sources whilst reducing the production from convention electricity generation. Smarter Grid Solutions provide the power flow and voltage management algorithms to perform real-time control of the devices and to ensure that the power flows and voltages are under the regulatory limits [CAV+08].

2.5.3 European ANM projects

In this section, EU projects will be analysed and presented. These projects include 1) Active Distribution network with full integration of Demand and distributed energy Resources (ADDRESS) 2) Active Distribution Network (ADINE) 3) GRID4EU 4) Ideal Grid for All (IDE4L) 5) Distributed Intelligence for Cost-effective and Reliable Solutions (DISCERN) 6) S3C and 7) Smart Grids Model Region Salzburg (SGMS).

2.5.3.1 ADDRESS Project

The ADDRESS project was initially started in 2008 and lasted 5 years until 2013. A European project which was funded by the European Commission and gathered 25 partners from 11 European countries. Partners included universities (University of Manchester, Universidad Pontificia Comillas, Università di Siena, Università di Cassino), network operators (ENEL Distribuzione, UK Power Networks, Iberdrola Distribución Eléctrica, Vattenfall), electric equipment manufacturers (ABB, Landis+Gyr, ZIV), home appliance manufacturers (Philips, Electrolux, RLtec) and Information and communications technology (ICT) providers [BAV+11].

The main goal of the project was to integrate demand and renewable energy sources to the smart grids. In addition, one of the objectives of the project was the active participation of consumers in power system markets. Furthermore, ADRESS project embraced the vision of the Smart Grids European Technology Platform to provide reliability, accessibility, flexibility and economy. Regarding reliability, the project aimed to provide all the emerged sustainable energy technologies in real-time network management and increase

the power system efficiency. As for the accessibility, the project proposed solutions to remove regulation barriers against the active participation of consumers and the integration of distributed energy sources. In terms of flexibility, the project proposed solutions for the real-time optimisation of energy flows and flexible solutions for enabling active demand. Finally, as for the economy aspect, the project enhanced the participation of power industry to increase the competitiveness and provide local and global savings [RMJ+08]. *Figure 2.7*, illustrates the overview of ADDRESS architecture.

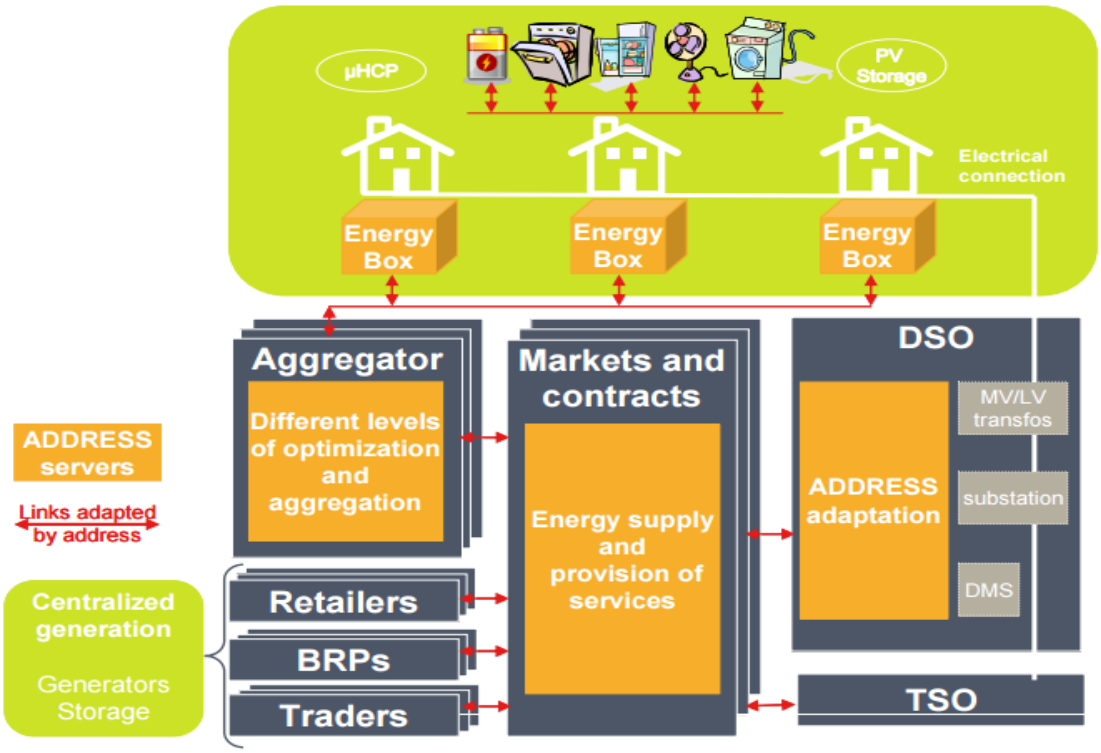


Figure 2.7: Overview of ADDRESS architecture [BAV+11]

2.5.3.2 ADINE Project

ADINE project was initially started in 2007 until 2010 as European project and involved Hermia LTD (coordinator), ABB Ltd Distribution Automation, AREVA T&D Ltd, Compower AB, Lund University and Tampere University of Technology [SRJ+10]. The main concept of the project was to provide an Active Network Management scheme which integrates the distributed generation into existing networks. The project provided solutions for protection relay and fault location applications, voltage control techniques, centralised

voltage control with SCADA systems and coordinated protection planning. The main objective of the project was to provide the reliable connection of distributed energy sources, to ensure the required power quality, to maximise the existed generation output in the chosen networks and finally to provide all the technical solution in protection and voltage control with real-time test studies and simulations. *Figure 2.8*, depicts the overview of the active distribution network in the ADINE project.

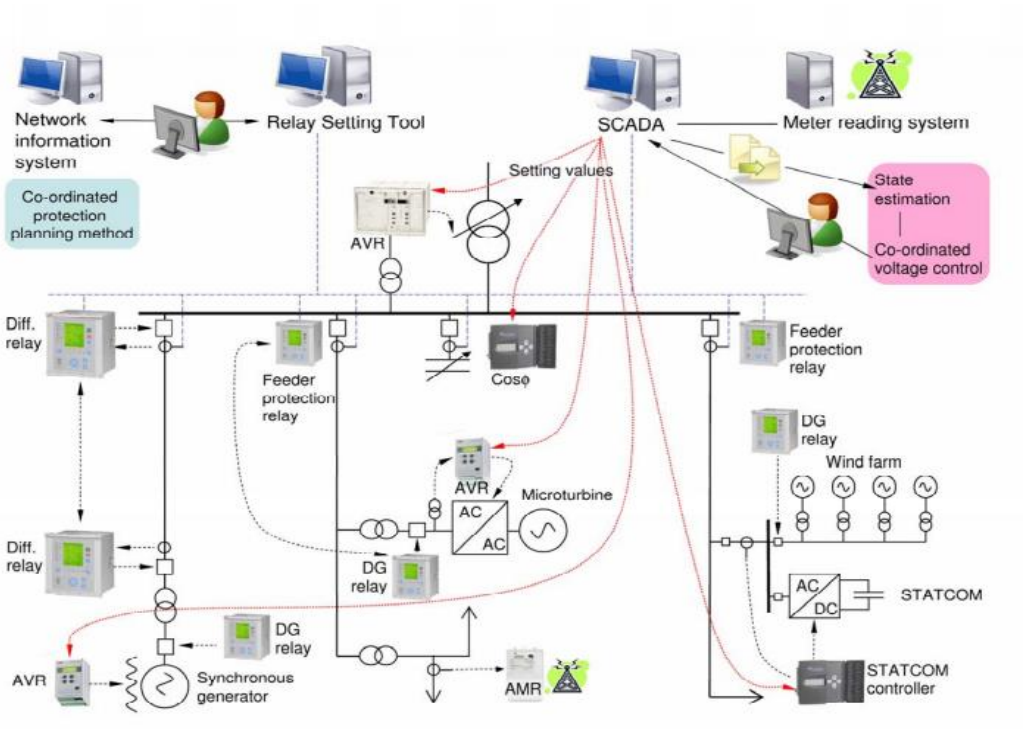


Figure 2.8: Overview of the active distribution network in the ADINE project [SRJ+10]

ABB and Tampere University of Technology provided new protection and voltage control solutions with the use of new IED's and DMS software. Compower and Lund University proved how a microturbine will contribute to the improvement of voltage quality and AREVA introduced a STATCOM controller in order to improve the power quality. The protection schemes which involved in the project included anti-islanding protection, differential and line protection and management of protection settings. In addition, the protection of distribution network involved the application of communication-based relays and fault location. The voltage control schemes which proposed in ADINE provided two solutions for low voltage level and for medium voltage level. For low

voltage level, the use of a microturbine was proposed and for medium voltage level, a voltage control application was developed on SCADA. The proposed voltage control algorithm included to basic function. The first function, which was called basic control was used in order to provide an acceptable level of voltage when a rise or drop was detected in a network node. The second function of the scheme, which was called restoring control, was in charge to provide the adequate power factor of distributed generation when the network state allowed it in order to normalise the voltages. Finally, a STATCOM controller was introduced in a distribution network in order to increase the power quality of the network by a) compensating harmonics and reactive power b) eliminating negative sequence currents c) reducing voltage flicker d) stabilising the voltage level and e) improving the network recovery during line fault [EC12].

2.5.3.3 GRID4EU Project

GRID4EU is the biggest funded project in smart grids area and is funded by the European Commission. The project started in 2011 and will finish in 2016. The project partners include 6 distribution system operators (ERDF, Enel Distribuzione, Iberdrola, CEZ Distribuce, Vattenfall Eldistribution and RWE) and 27 other partners from manufacturing industry, research academia, and energy suppliers. The project will identify and test the potential of smart grids in areas such as renewable energy integration, electric vehicle development, grid automation, energy storage, energy efficiency and load reduction [EC12].

The main objectives of the project are: “a) Develop and test innovative technologies b) Define standards through the set-up of demonstrators c) Guarantee the scalability of new technologies d) Guarantee the replicability over Europe and e) Analyse Smart Grid Cost-benefits. As for the main research and development challenges these include a) Using more renewable energy sources connected to distribution networks b) Implementing more efficient participation of customer to electricity markets c) Secure energy supply d) MV/LV network Supervision and Automation e) Improving peak load management through increased interactions between network operation and electricity customers f) Demand Side Management (DSM), Storage and Micro Grids” [EC12].

2.5.3.4 IDE4L Project

IDE4L is a European Commission funded project which was started in 2013 and will finish in 2016. Project coordinator of the programme is Tampere University of Finland but also other academic partners (DTU, Aachen University, Universidad Carlos III de Madrid, and KTH) and energy companies (IREC, OSTKRAFT, Schneider Electric, Telvent) are included [EC13].

The objectives of the project are : "a) to provide the distribution networks of the future that are able to integrate sustainable energy sources in an efficient use b) the development of distribution network automation systems and c) to develop the appropriate applications that will be able to monitor and control the whole network" [EC13].

The aim of the project is to create a framework that will be able to enhance the active distribution network based on existing technologies and future requirements. As for the challenges of this project, these include: a) control management of distributed energy sources b) efficient exploitation of existing networks c) reliability and quality of the power d) reduction of uncertainties such as electric vehicles and e) handling conflicts between electricity market needs and technical constraints [EC13].

The project is divided in three phases. Phase 1, defines the whole concept of the project where scenarios of penetration of renewable energy sources are taking into account. Phase 2, contains the methods and functionality that will contribute in order to build the IDE4L grid and Phase 3 includes the demonstration sites in Denmark, Italy, and Spain where all the functions and applications will be tested in real environment [EC13].

Finally, as for the expected results, the project will provide all the functionalities and methods in order to allow large-scale penetration of renewable energy sources, will improve the cost efficiency of distribution networks and will provide distribution network reliability.

2.5.3.5 DISCERN Project

The DISCERN project which is funded by the European Commission started in 2013 and will finish in 2016. The project includes 11 partners such as: RWE Deutschland AG, ABB**,**

Research Centre for Energy Resources and Consumption (CIRCE), Gas Natural Fenosa, Iberdrola, DNV KEMA, Oldenburg Research and Development Institute for Information Technology Tools and Systems (OFFIS), The Royal Institute of Technology (KTH), Southern Electric Power Distribution (SEPD), Vattenfall Eldistribution and ZIV [ECO13].

The main objectives of the project include: "a) Establishment of 5 different demonstration projects focused on MV and LV networks in order to compare and contrast technical solutions for monitoring and control of the distribution network b) Identification of all the technological applications for monitoring and controlling of the distribution network c) Mapping of relevant standards and contribution to standardisation activities and d) Development of cost-effective applications to increase the intelligence of distribution networks" [ECO13].

One of the main challenges of the project is to provide the observability of LV and MV networks. As the electricity production and consumption is changing with the integration of renewable energy sources and active demand, the distribution networks face a challenge to adopt these changes and ensure the quality of power supply. In order to provide a reliable and cost-effective electricity network, the distribution system operators (DSO's) need to provide all the technical solutions to tackle the dynamically changing constraints. DSO's, need to provide the appropriate level of intelligence to ensure security and reliability.

Based on the recommendation from DISCERN, DSOs will be enabled to implement solutions that have been tested and validated in various countries and circumstances [ECO13].

The project will aim at giving DSOs the tools to answer complex questions like:

- How much intelligence do I need in my distribution network to ensure a cost effective and reliable operation of the network?
- What is the most cost-effective solution to implement this intelligence in the network?

- How should the ICT-infrastructure be designed to serve the requirements of a DSO?

2.5.3.6 S3C Project

S3C Project is funded by the European Commission, started in 2011 and finished at the end of 2015. The project involves 7 partners: Vision on Technology (VITO), B.A.U.M. Consult GmbH, Energy Research Center of the Netherlands (ECN), Technical Research Institute of Sweden (SP), Ricerca sul Sistema Energetico (RSE), EDP Distribuição and Informatization Energy Engineering Automation (INEA). The main objective of the project is to provide the design, implementation and use of technology in combination with the active participation of consumers. Furthermore, the project will promote the smart energy behaviour of households by providing all the appropriate tools [ECO15].

S3C stands for Smart Consumer, Smart Customer, and Smart Citizen. Smart Consumer involves the end-user who is interested in reducing the energy bill by keeping comfort levels of energy services on an equal level. Smart Costumer involves the end-user whose main concern is to become a prosumer, meaning to produce energy or provide energy services. Finally, Smart Citizen involves the end-user whose contribution is based on the environmental support.

The S3C project's overall objective is to enable the 'smart' energy behaviour of energy customers in Europe by providing the appropriate technology with the evaluation of test cases, and pilot projects [ECO15]. Based on these, the S3C consortium has developed a practical toolkit that is able to be used by anyone who is involved in smart energy projects.

2.5.3.7 SGMS Project

The Smart Grids Model Region Salzburg Project comprises a total of 23 projects and received funds from the Climate and Energy Fund (KLI.EN) and the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) through their programmes (Energy of the Future, New Energies 2020). The project is supported by energy sector teams (Salzburg AG, Salzburg Netz GmbH), a property developer (Salzburg Wohnbau),

a technology vendor (Siemens), consulting services (Fichtner) and research institutions (Austrian Institute of Technology, Vienna University of Technology, CURE) [SGM13].

The main objective of the project is to provide a smart grid system in Salzburg which will integrate a high penetration of renewable energy sources without network congestions. Moreover, technology applications in voltage control, electric vehicles, and residential customers are provided in order to reduce network cost and provide balance in generation and consumption. *Figure 2.9*, illustrates the control concept of a low voltage network in Köstendorf, a municipality in Austria.

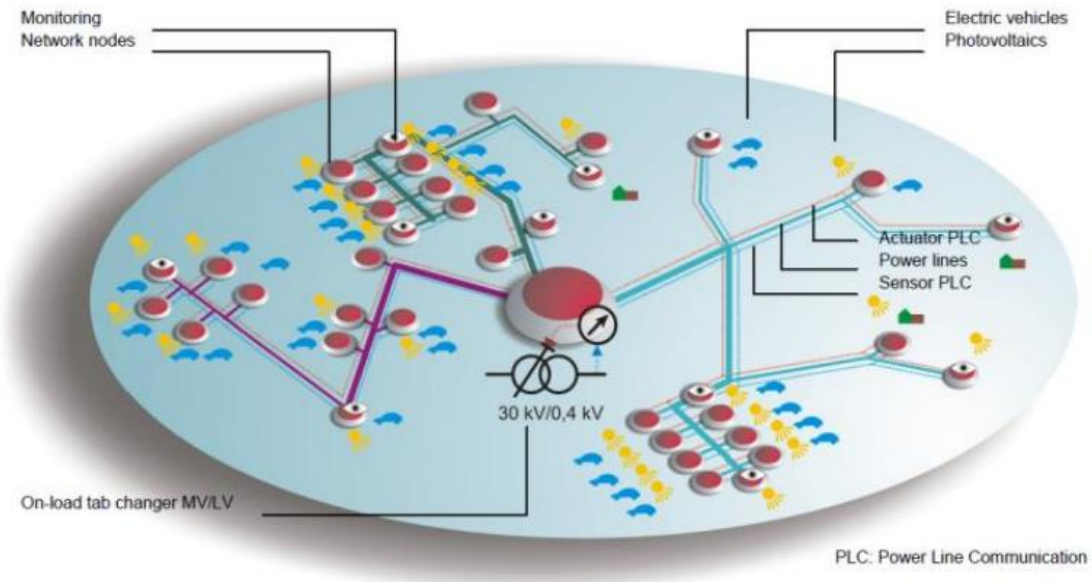


Figure 2.9: Control concept in the low voltage network in Köstendorf [SGM13]

The main objectives of all the projects in the region of Salzburg are the following:

- "Ensure a flexible and efficient grid infrastructure which is coordinated with the interest of the customers
- Increased use of renewable energies
- Reduction of periods with peak loads
- Increase of experience and innovation in this field
- Reduction of the emissions of CO_2 of the consumption of resources"

Intelligent control solutions are used in order to increase the integration of renewable energy sources to the existing network. A prototype was developed for medium voltage networks and the results proved that an increase of approximately 20% in generating capacity is realistic [SGM13]. In addition, a voltage control scheme was implemented and tested in the community of Köstendorf.

The project also includes intelligent system integration for electromobility. Also, the project involves the integration of electric vehicles and how they affect the power grid. Additionally, the project provides the implementation of adaptive charging. Before the implementation of this project, three-phase charging was considered in order to avoid system overloads. Finally, vehicle-to-grid delivery of electricity is not considered by the participants of the project as they consider it economically unfeasible.

Also, the project involves the integration of residential customers into the grid. Demand response technologies are used in the project, providing customers the participation with an active role in the electric power system [SGM13].

Finally, the project involves the integration of buildings. The integration of buildings in the smart grids provides a comfortable, intelligent and integrated smart infrastructure. Buildings and technologies that involve storage devices will play a significant role in the future. This can be achieved by thermal inertia, which makes it possible to shift energy use off-peak periods without causing a loss of comfort.

2.5.4 USA and Canada ANM projects

In this section, Northern American projects will be analysed and presented. These projects include a) American Electric Power (AEP) Ohio gridSMART and b) British Columbia Institute of Technology (BCIT) Intelligent Microgrid.

2.5.4.1 AEP Ohio gridSMART Project

The AEP Ohio gridSMART project was started in 2010 and finished in 2014. Partners included Lockheed Martin, UT-Battelle LLC, General Electric, S&C Electric Company, Schweitzer Engineering Laboratories and Silver Spring Networks. The aim of the project

was to provide the appropriate infrastructure in Ohio State that is able to decrease energy consumption, demand costs and Co_2 emissions [AEP14].

The project integrated multiple technologies to the infrastructure of the Ohio area including: “a) Advanced Metering Infrastructure (AMI) b) Distribution Automation Circuit Reconfiguration (DACR) c) Volt-VAR Optimisation (VVO) and d) Consumer Programs (CP). These technologies were combined with consumers’ communication and information sharing, demand response and dynamic pricing and consumer products such as electric vehicles and smart appliances” [AEP14].

The objectives of the project included: “a) Reduction of energy demand by 15MW b) Reduction of energy consumption by 18.000 MWh c) Reduction of CO_2 emissions by 16650 tons d) Improvement of system efficiency and reliability by 30-40 percent and e) Integration of 100kW storage resources. After the completion of the project the following objectives have been achieved: a) Establishment of secure and reliable smart grid infrastructure that is able to reduce energy consumption hence provide reliability and system efficiency b) Deployment of technologies and smart grid business models that should contribute to the extension of the smart grids nationally and c) Education of consumers in business models that provided tools and information of reducing consumption and peak demand” [AEP14].

The opportunity for consumers to participate in various programs allowed them to: “a) Receive real-time information about electricity usage. b) Manage their electricity usage and lower their consumption. c) Reduce cost with the same level of comfort and service and d) Reduce outage times through the automation of circuit reconfiguration. As for the AEP Ohio, the benefits included: a) Reduction of costs regarding field visits through the integration of smart products b) Improvement of system reliability c) Improvement of employee safety d) Improvement of customer satisfaction and e) Reduction of peak demand” [AEP14].

2.5.4.2 BCIT Intelligent Microgrid

BCIT Intelligent Microgrid is a collaboration work between BCIT and British Columbia (BC) Hydro in order to provide Canada's first smart Microgrid. The project started in 2007 and was deployed at BCIT's Burnaby campus [Far08]. The Microgrid project is a research and development platform where existing and future technologies in telecommunication, smart metering, and intelligent appliances are employed to develop and qualify the most robust, cost-effective and scalable solutions required to facilitate the evolution and the emergence of the Smart Grid. *Figure 2.10*, illustrates the topology of the Microgrid**.**

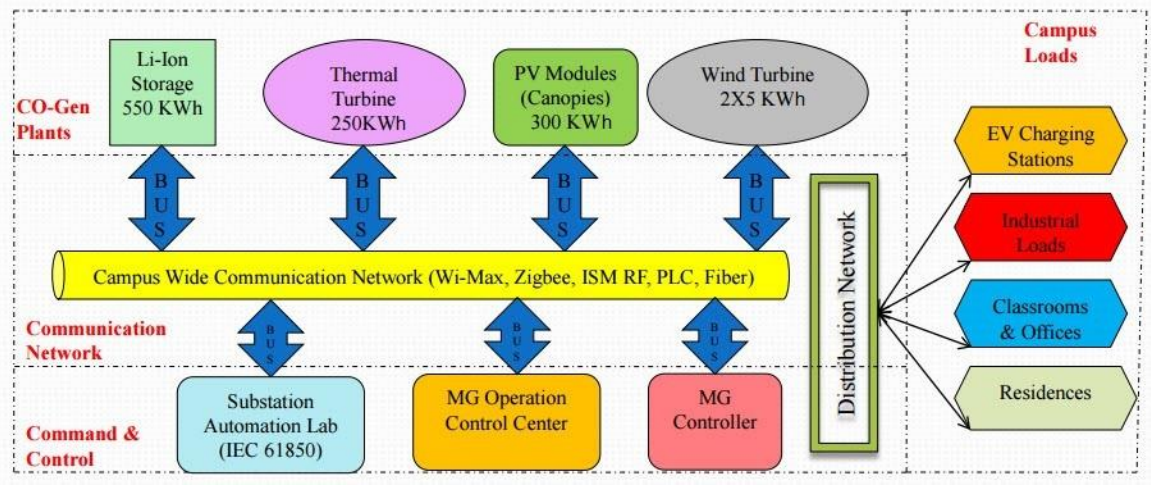


Figure 2.10: Microgrid Topology [Far08]

The project is divided in 3 phases. Phase 1 was completed in 2010 which involved: "a) Construction of the Microgrid b) Development of Load control devices c) Integration of solar modules and wind turbine d) Protection of BCIT campus". Phase 2 will run until the end of 2015 and will mainly focus on research and development of the project. Finally, Phase 3 will provide the Microgrid's commercialization and will run until the end of 2017 [Far08]. The objectives of the project include: "a) Development and demonstration of smart grid technological solutions b) Implementation of these technologies in Intelligent Grid c) Commercial success of the project through demonstration of products for application to national and international customers and d) Establishment of BCIT as a leader in smart grid research in Canada".

2.5.5 Review of the ANM activities within the projects

Most of the UK and European projects incorporate active network management schemes, however, some of them used centralised instead of decentralised techniques. As for the AuRA-NMS project, a CSP approach was able to provide ranked solutions however it was centralised and the curtailment of the generators was proposed as a solution to cover the demand in the network. Furthermore, the implementation of AuRA-NMS requires new communication investments such as installing new IEC61850 IEDs and COM600 units.

The Orkney ANM project used an ANM in order to avoid network refurbishments, which would have an economic impact. According to [KEM2012], the possible construction of a subsea cable between the Orkney islands and the mainland would have cost £30million. In contrast, the development of the ANM cost only £500k. According to the results, the Orkney Active Network Management scheme is able to provide additional wind generation without any further network refurbishments to a network that was at full capacity. It is one of the first smart grid projects that enables ANM technology. [KA13]

The FPP project trials have proven that ANM can be used as a tool to manage DGs. The applications for power flow and voltage management are able to provide solutions not only for the existing DGs but also for any new ones by providing all the smart decisions that will be able to integrate additional capacity. The significant challenge of the scheme was to integrate DGs without violating network constraints and provide safety in voltage and frequency limits. The results prove that ANM is able to provide the adequate voltage limits by controlling the real and reactive power of DG. However, the ANM solution was centralised a not decentralised.

The NINES project hasn't finished yet and trials are expected to run until the end of 2016 and include the participation of domestic customer properties. The implementation of the ANM scheme was based on the Orkney project, so the same voltage and power flow management tools were used to facilitate additional DGs, hence it was considered a centralised scheme. However, the difference lies in the fact that the project integrates the active participation of customers.

As for the European and USA projects, the only project which incorporates ANM schemes and has finished is ADINE, while IDE4L and DISCERN are about to finish by the end of 2016.

The aim of the ADINE project was to develop new control schemes for distribution network management. The ADINE project has provided several technical solutions that make integration of DG easier. The project integrated centralised voltage control techniques and co-ordination of protection relays. Some issues arise for the voltage control at low and medium level, however, solutions that were based on power electronics and automation were used to avoid and overcome these issues. Results from the project proved that active network management schemes are feasible but only in specific applications.

Based on the analysis of the above UK and European projects regarding ANM, centralised techniques were used for control functions either applied to power flow management or voltage control. For that reason, Distributed Constraint Satisfaction and Distributed Constraint Optimisation are proposed as alternative techniques for power flow management and considered the appropriate frameworks that are able to control network constraints by providing decentralised solutions.

2.6 Moving to Distributed Constrained Optimisation

The increased penetration of distributed generation becomes a valuable option for distribution systems to operate in a secure and cost-effective way without relying on network reinforcement. The principle of ANM is to address congestion and voltage issues via short-term decision-making policies. Moreover, ANM schemes maintain the system within operational limits in real-time by relying on the curtailment of generation and curtailment of renewable energy must be considered as a last choice. Also, communications failure needs to be considered in order to avoid unsatisfactory output levels of generation.

For that reason, Constraint Satisfaction, Distributed Constrained Satisfaction, and Distributed Constrained Optimisation are proposed as different frameworks opposed to

ANM. These frameworks are able to provide decentralised solutions and can be flexible and scalable.

As for the communications losses, the difference of the above frameworks is that all the decisions are taken into consideration locally in order to minimise the communication failures. With that in mind, when a communication loss occurs in one of the branches of the network, then the rest of the branches are able to provide solutions without violating the constraints of the network. Also, there is no need of a central authority as each decision is taken from each individual part of the network in real-time by informing only the local neighbours.

DCOP has gained attention in Artificial Intelligence community due to its capability of addressing complex and naturally distributed problems. Additionally, in power systems, there is an effort to provide decentralised and distributed techniques for control and especially, power flow management. For that reason, DCOP can be considered the adequate framework of providing decentralised and distributed solutions in power systems.

Chapter 3

Constraint Satisfaction (CSP) and Distributed Constraint Satisfaction (DisCSP)

3.1 Introduction

In this chapter, Constraint Satisfaction (CSP) and Distributed Constraint Satisfaction (DisCSP) problems and algorithms are analysed in order to understand the frameworks and their effectiveness in real-world problems. The main goal of CSP problems is to find a value for each variable so that the solution will satisfy the constraints. In DisCSP problems, the variables and constraints are distributed among the agents and the main goal is to find a value assignment to variables that satisfies the constraints while providing consistency among the agents.

A proper understanding of the algorithms is needed as an updated version of them is used in Distributed Constraint Optimisation (DCOP).

3.2 Constraint Satisfaction

A CSP is a problem where the goal is to find a consistent assignment of values to variables. Formally, a CSP problem can be defined as a tuple

$< V, D, F >$ such that:

- $V = \{V_1, \dots, V_n\}$ is a set of decision variables such that each variable is controlled by an agent.
- $D = \{d_1, \dots, d_n\}$ is a set of domains of the variables.
- $F = \{f_1, \dots, f_n\}$ is a set of constraints on their values.

A typical example of CSP is a puzzle called n-queens. *Figure 3.1* illustrates the 4-Queens example of CSP. The main goal is to place 4 queens on a 4x4 classboard so that they do not hit each other.

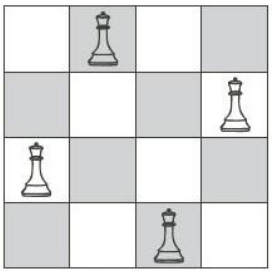


Figure 3.1: Example of CSP [YH00]

In order to place correctly the queens on the chessboard, only one queen can be placed in each row or column. The CSP can be formalised as a problem where there are four variables x_1, x_2, x_3 and x_4 each of which corresponds to the position of the queen. The domain of every variable is {1,2,3,4} and the solution of the problem can be reached by a combination of the values of these variables. As for the constraints, no queen can attack any other queen, so $Q_i \neq Q_j$ where Q_i is a queen that is placed in column i and Q_j a queen that is placed in row j.

There are two different groups of algorithms for solving CSP problems. The first group contains the search algorithms (backtracking and iterative improvement algorithms) and the second group contains the consistency algorithms.

In backtracking algorithm, a partial solution is constructed where a value assigned to a domain of variables that satisfies all of the constraints. The variables are added to the partial solution in order to reach a complete solution. At the moment where there is none value that satisfies all the constraints with the partial solution, the value of the most recently added variable to the tentative solution is changed.

In iterative improvement algorithms, all variables have tentative initial values and no partial solution is constructed. A solution that contains all variables is obtained by using hill-climbing search [RN03]. In hill-climbing search, the algorithm starts with an arbitrary solution to a problem and then attempts to find a better solution by incrementally changing a single element of the solution. Since these algorithms are hill-climbing search algorithms, occasionally they will be trapped in local-minima. Local-minima are states that violate some constraints, but the number of constraint violations cannot be decreased

by changing any single variable value. In iterative improvement algorithms, an error can be reconsidered without performing another search, and the same error can be reconsidered again. Finally, the quality of the solution is not guaranteed in these type of algorithms.

In consistency algorithms, the algorithms can be classified by the notion of k-consistency [Mac92]. A CSP is k-consistent if given any instantiation of any $k-1$ variables satisfying all the constraints among those variables, it is possible to find an instantiation of any kth variable such that the k values satisfy all the constraints among them. If there are n variables in a CSP and the CSP is k-consistent for all $k \leq n$, then a solution can be obtained immediately without any backtracking.

3.3 From Constraint Satisfaction to Distributed Constraint Satisfaction

A distributed constraint satisfaction problem is a constraint satisfaction problem in which variables and constraints are distributed among multiple automated agents. A DisCSP is a problem to find a consistent assignment of values to variables. Formally, a DisCSP is a tuple $< A, V, D, F >$ such that:

- $A = \{A_1, \dots, A_m\}$ is a set of agents.
- $V = \{V_1, \dots, V_n\}$ is a set of decision variables such that each variable is controlled by an agent.
- $D = \{d_1, \dots, d_n\}$ is a set of domains of the variables.
- $F = \{f_1, \dots, f_n\}$ is a set of functions (normally hard constraints) where f_i is any function $d_{i1} \times \dots \times d_{im} \rightarrow \Re$ which denotes how much cost is assigned to each possible combination of values of the variables.

The main difference between the distributed CSP and CSP is the distribution of variables and constraints between the agents. The agents can communicate with each other and send messages if they know the exact address of other agents. Yokoo et al. developed an algorithm for solving distributed CSP problems called asynchronous backtracking in which the agents can run asynchronously and simultaneously [YDI+92] [Yok94]. However, they tried to improve the algorithm with a more efficient approach which was

called asynchronous weak-commitment search which can revise a bad decision without exhaustive search [Yok95]. This algorithm managed to eliminate the disadvantages of backtracking algorithms and iterative improvement algorithms. The experimental results on various example problems show that this algorithm is 3 to 10 times more efficient than other algorithms (the min-conflict backtracking, iterative improvement algorithm) [YDI+98].

3.4 Application problems of DisCSP's

Various problems can be formalised as DisCSPs. For instance, Ghedira and Verfaillies presented a resource allocation problem as a DisCSP [GV92]. In their model, there are two different type of agents: task agents and constraint agents. In this formalisation, an agent that has the control of a variable does not have knowledge of the constraints associated with its variable. In that event, a task agent needs to communicate with the corresponding constraint agent in order to clarify if the constraints are satisfied or not. *Figure 3.2*, illustrates the resource allocation problem as a formalisation of DisCSP [GV92]. Another class of application problems that can be formalised as DisCSP are the time-tabling tasks. For instance, Solotorevsky and Gudes tried to solve a real-life time tabling and transportation problem using distributed CSP techniques [SG96]. The real-life time tabling problem included the appropriate allocation of nurses in a hospital with their corresponding shifts in each department. Many other applications can be formalised as a DisCSP problem such as distributed scheduling.

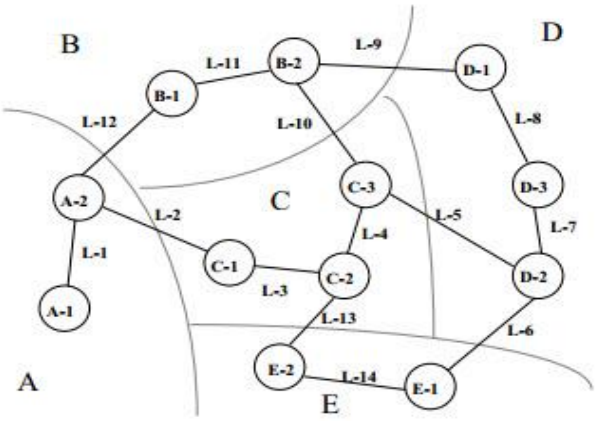


Figure 3.2: Resource allocation problem formalised as DisCSP [LS96]

3.5 Algorithms for solving DisCSP's

In this section, three different algorithms will be presented and analysed in order to identify the appropriate algorithm for solving specific DisCSP's problems. These algorithms are: a) Asynchronous backtracking algorithm b) Asynchronous weak-commitment search algorithm and c) Distributed Breakout algorithm. All these are algorithms are able to solve distributed satisfaction problems where each agent has only one variable. Furthermore, additional algorithms for handling multiple variables will be presented and analysed.

3.5.1 Asynchronous backtracking algorithm

The Asynchronous backtracking algorithm is a distributed and asynchronous algorithm that was first introduced by Yokoo et al. in 1992 [YDI$^+$92], [YDI$^+$98]. Agents communicate between them by sending messages with their current value. These type of messages are called *ok?* messages. However, there are some other messages that contain a new constraint and they are called *nogood* messages. The algorithm starts when an agent receives an *ok?* message and a *nogood* message. Every agent is communicating with its neighbouring agent via *ok?* messages and sending a tentative value assignment. Each agent receives the messages from the other agents and holds the current value assignment of each agent in its viewpoint which is called *agent_view*. The priority order of the agents is determined by alphabetical order of the variable identifiers. If the value assignment of an agent is not consistent with the value assignments of the higher priority agents, then the value assignment needs to be changed by the agent. However, if there is no other value assignment for the agent that is consistent with the higher priority agents then the agent sends a new *nogood* message to the higher priority agents, hence the higher priority agents need to change their values. According to [YDI$^+$98], if a solution exists then the algorithm is guaranteed to find that solution and if not then it will terminate without providing any solution to the problem. *Figure 3.3* presents the pseudo-code of the algorithm.

when received (**ok?**, (x_j, d_j)) **do**

add (x_j, d_j) to *agent_view*;
check_agent_view;**end_do**;

when received (**Nogood**, x_j, *nogood*) **do**

add *nogood* to *nogood_list*;
when *nogood* contains an agent x_k that is not its neighbour **do**
request x_k to add x_i as a neighbour,
and add (x_k, d_k) to a*gent_view*; **end_do**;
old_value ← *current_value*; **check_agent_view**;
when *old_value* = *current_value* **do**
send (**ok?**, (x_i, *current_value*)) to x_j; **end_do**; **end_do**;

procedure **check_agent_view**

when *agent_view* and *current_value* are not consistent **do**
if no value in D_i is consistent with *agent_view*
then backtrack;
else select $d \in D_i$ where *agent_view* and d are consistent;
current_value ← d;
send (**ok?**, (x_i, d)) to *low_priority_neighbours*; **end_if**;**end_do**;

procedure **backtrack**

nogood ← *inconsistent_subset*;
when *nogood* is an empty set **do**
broadcast to other agents that there is no solution;
terminate this algorithm; **end_do**;
select(x_j, d_j) where x_i has the lowest priority in **Nogood**;
send (**Nogood**, x_i, *nogood*) to x_j;
remove (x_j, d_j) from *agent_view*; **end_do**;
check_agent_view;

Figure 3.3: Pseudo-code of asynchronous backtrack algorithm [YDI+92]

An analytical example of how the Asynchronous backtracking algorithm is working is presented in *Figure 3.4* [YDI+98]. Firstly, the agent x_3 receives the *ok?* messages from x_1 and x_2. The *agent_view* of x_3 will be $\{(x_1, 1), (x_2, 2)\}$. A nogood message $\{(x_1, 1), (x_2, 2)\}$

is generated because there is no possible consistent value for x_3's *agent_view*. Therefore, x_3 chooses x_2 to send the message since x_2 is the lowest priority agent. In addition, x_2 receives the *nogood* message from x_3 and store it to its *agent_view*. This message contains also value for the x_1 but there is no connection between x_1 and x_2. Since they are neighbours a new link is created between these two and x_2 request x_1 to send x_1's value to x_2. Furthermore, x_2 receives the value of x_1 and store to its *agent_view*. The *agent_view* $(x_1, 1)$ and the value assignment $(x_2, 2)$ violate the *nogood* message $\{(x_1, 1), (x_2, 2)\}$ but there is no other value assignment for x_2, so x_2 generates a new *nogood* $(x_1, 1)$ and sends a new *nogood* message to x_1. *Figure 3.4* depicts the Asynchronous backtracking algorithm example.

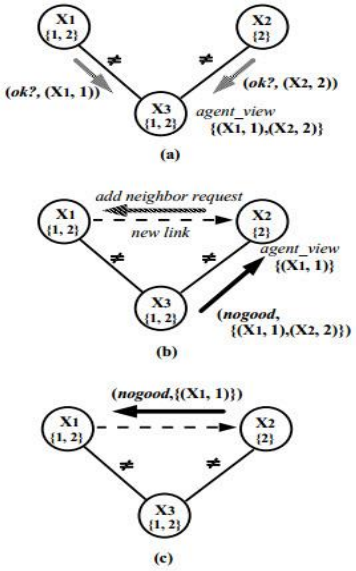


Figure 3.4: Asynchronous backtracking algorithm example [YDI+92]

3.5.2 Asynchronous weak-commitment search algorithm

The Asynchronous weak-commitment search algorithm was introduced by Makoto Yokoo in 1994 [Yok95]. *Figure 3.5* presents the Pseudo-code of Asynchronous weak-commitment search algorithm. The difference between Asynchronous backtracking algorithm and Asynchronous weak-commitment search algorithm is that in the latter the priority order of the agents is not determined. According to Asynchronous backtracking

algorithm, the high priority agents select the value ordering and if the selection is not adequate then the lower

priority agents need to perform a search in order to avoid the bad selection. In Asynchronous weak-commitment search algorithm, this procedure is changed in order to avoid bad decisions and the priority order of the agents is dynamically changed without executing an exhaustive search. A priority value is determined for each variable, and the priority value is communicated through the *ok?* message. If the current value is not

when received (**ok?**, (x_j, d_j, *priority*)) **do**

add (x_j, d_j, *prioriy*) to *agent_view*;
check_agent_view;**end_do**;

when received (**Nogood**, x_j, *nogood*) **do**

add *nogood* to *nogood_list*;
when (x_k, d_k, *priority*) where x_k is not in *neighbours*
is contained in *nogood* **do**
add x_k to *neighbours*, add (x_k, d_k, *priority*) to *agent_view*; **end do;**
check_agent_view;end do;
procedure check_agent_view
when agent_view and current_value are not consistent do
if no value in D_i is consistent with *agent_view* **then backtrack**;
else select $d \in D_i$ where *agent_view* and d are consistent and d minimizes the number of constraint violations with lower priority agents;
current_value ← d;
send (**ok?**, (x_i, d, *current_priority*)) to *neighbours*; **end_if**;**end_do**;

procedure **backtrack**

nogoods ← V = *inconsistent_subset*;
when *nogoods* is an empty set **do**
broadcast to other agents that there is no solution;
terminate this algorithm; **end_do**;
when no element of *nogoods* is included in *nogood_sent* do
for each V ∈ nogoods do;
add V to *nogood_sent*
for each(x_j,d_j,p_j) in V **do**;
send (*nogood*, x_i, V) to x_j;**end_do**; **end_do**;
$p_{max} \leftarrow max_{(x_j,d_j,p_j) \in agent\_view}(p_j)$;
current_priority ← 1+ p_{max};
select $d \in D_i$ where d minimizes the number of constraint violations with lower priority agents;
current_value← d;
*send (**ok?**, (x_i, d, current_priority))* to *neighbours;* **end_do**;

Figure 3.5: Pseudo-code of Asynchronous weak-commitment search algorithm [Yok95]

consistent with the *agent_view*, the agent changes its value using the min-conflict heuristic. If an agent cannot find a consistent value in its *agent_view*, then it sends *nogood* messages to the other agents and increases its priority value. If the agent cannot generate a new *nogood*, it will not change its priority value but will wait for the next message. The completeness of the algorithm is guaranteed by the fact that the agents record all *nogood* messages found. *Figure 3.5* illustrates the pseudo-code of Asynchronous weak-commitment search algorithm. [Yok95]

An example of the Asynchronous weak-commitment search algorithm process is illustrated in *Figure 3.6*. The example is based on the 4 queens' problem. Firstly, the position of the queens is presented in *Figure 3.6 (a)*. All the agents communicate with each other and passing their corresponding values. At the beginning of the process, all the priority values are equal to 0. For that reason, the priority order is chosen by the alphabetical order of the identifiers. Therefore, x_4 sends nogood messages to the other agents and increases its priority value. In this case, the number of constraint violations is 3, since it conflicts with x_3 only. Additionally, x_4 selects 3 position and sends *ok?* messages to the other agents. x_3, tries to change its value and sends *nogood* messages to the other agents and at the same time increases its priority value. Furthermore, x_3 selects 1 position and sends *ok?* messages to the other agents. Finally x_1 needs to change its value to position two and x_2 remains in the same position in order to reach a solution.

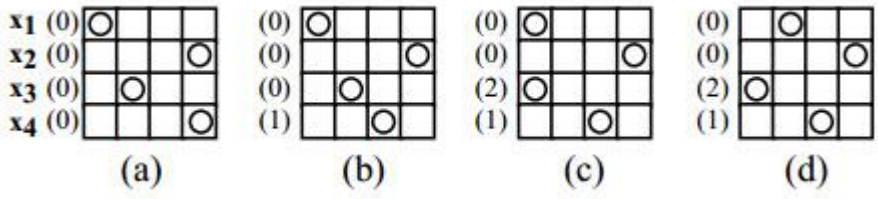


Figure 3.6: Asynchronous weak-commitment search algorithm example [Yok95]

3.5.3 Distributed breakout algorithm (DBA)

The breakout algorithm was initially introduced by Paul Morris in 1993 [Mor93]. It was mainly introduced in order to avoid the local-minima in search algorithms. Local-minima are states that violate some constraints and Morris introduced weights in order to eliminate

this situation. The initial value of weight is 1 and when a search reaches a local-minima then the breakout algorithm increases the weights of violated constraints in the current state by 1 so that the evaluation value of the current state becomes larger than those of the neighbouring states. The distributed breakout algorithm was introduced by Yokoo and Hirayama in 1996 [YH96]. In the distributed breakout algorithm each agent detects that it is in a quasi- local minimum state which weaker that a local-minimum state and requires only local communications instead of global across the agents. There are two type of messages in the communication between the agents. The *ok?* messages like asynchronous backtracking algorithm and the *improve* messages. The *improve* message is used to communicate the possible improvement of the evaluation value. The algorithm starts when an agent receives *ok?* and *improve* messages from all the neighbourhood agents.

An illustrative example of the distributed breakout algorithm is presented in *Figure 3.7*. *Figure 3.7 (a)*, shows the initial values of each agent. Every agent sends its initial value through ok? messages to the neighbour's agents. Therefore, every agent calculates the *current_eval* and *my_improve* and exchanges *improve* messages. At the first stage, the weight is 1 and the improvements of all the agents are equal to 0 hence the weights of the constraints (*nogoods*), {(x_1,white), (x_6,white)}, {(x_2, black), (x_5, black)}, and {(x_3, white), (x_4, white)}, are increased by 1.Then, the improvements of x_1, x_3, x_4, and x_6 are 1, and the improvements of x_2 and x_5 are 0.

Agents x_1 and x_3 change their values from white to black. Furthermore, the improvement of x_2 is 4 while the improvements of all the other agents are 0. Finally x_2 changes its value from black to white and all constraints are satisfied.

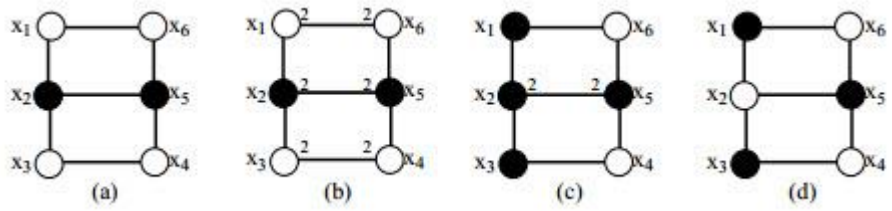


Figure 3.7: Distributed breakout algorithm example [YH96]

3.5.4 Agent-ordering asynchronous weak-commitment search

This algorithm was introduced by Armstrong and Durfee in 1997 as an updated version of asynchronous weak-commitment search [AD97]. The algorithm was developed in order to handle multiple variables by each agent. Firstly, each agent finds all solution to its local problem and then the constraint satisfaction problem is re-formalized where each agent controls only one variable which contains all the local solutions. After this step, all the previous algorithms can be used in order to find the solution. However, if the local solutions are large then it is difficult for the algorithm to find a solution to the problem.

3.5.5 Asynchronous Aggregation Search

The Asynchronous aggregation search algorithm was introduced by Silaghi et al. in 2000 [SHF00]. In the previous algorithms for distributed constrained satisfaction, each variable is only known from a specific agent and the constraints are widely known among all agents. In the asynchronous aggregation search algorithm, the variables can be handled by each agent but the constraints are private and can be restricted to specific agents. The algorithm's process is the same as asynchronous backtracking algorithm but instead of assigning a single value to a single variable, the algorithm aggregates multiple values to multiple variables. The authors investigated three different versions of this algorithm where the difference based on the number of *nogoods* that they record.

3.5.6 Asynchronous Forward Checking

The Asynchronous Forward Checking algorithm was introduced by Meisels and Zivan in 2007 [MZ07]. The algorithm performs a synchronous search instead of asynchronous but the forward checking computation is asynchronous. The algorithm uses a data structure for the search process which is called Current Partial Assignment (CPA). At the beginning of the process, the CPA is empty. Each agent records its assignments on CPA and passes it to the next agent. The next agent adds its assignment to CPA if a consistent assignment can be found. However, if the assignment is not consistent then the algorithm backtracks and sends back the CPA to the previous agent in order to change its assignment. At the

same time, even if the algorithm backtracks, the agent sends forward a copy of the CPA in order to perform a forward checking. If an agent cannot provide a value from its domain then it sends back a NOT_OK message that informs about the inconsistency. The agent that receives the message updates the CPA with a consistent assignment by its variable.

3.5.7 Asynchronous Partial Overlay

The Asynchronous Partial Overlay algorithm was introduced by Mailler ad Lesser in 2006 [ML06]. In this algorithm, a new methodology is introduced which is called cooperative mediation. This approach uses dynamically constructed, partial centralization and its advantage lie in the speed of the centralization process. Each agent provides a solution to a small portion of the problem and recommends value changes to the agents involved in the mediation process. If an agent provides a solution that conflicts with the solutions of other agents outside of the mediation process, a link is created between them in order to avoid future conflicts. The significant benefit of this process is the range of autonomy that each agent has, allowing each agent to act as the mediator if an undesirable solution conflicts with the overall solution of the problem. A later updated version of this algorithm was presented by Grinshpoun and Meisels in 2007 in order to fix and improve the termination and completeness of the algorithm [GM07].

3.5.8 Concurrent Dynamic Backtracking

The Concurrent Dynamic Backtracking algorithm was introduced by Meisels and Zivan in 2004 [ZM04]. This algorithm belongs to the framework of search algorithms for solving distributed constrained problems. The framework lies on the simultaneous search processes that are generated dynamically by the first agent and by any number of agents during the search. The Concurrent Dynamic Backtracking performs dynamic backtracking hence improves the efficiency of the algorithm. According to [ZM04], experimental results, under the Concurrent Dynamic Backtracking, illustrate that the algorithm is faster than asynchronous backtracking.

3.5.9 Secure Multiparty Computation

Secure multiparty computation algorithm was introduced by Silaghi in 2005 [Sil05]. This algorithm applies secure multiparty computation to solve the problem and assumes that the variables are public and the constraints are private for each agent. The algorithm can provide a solution with stronger privacy compared to the previous distributed constrained satisfaction algorithms.

3.6 Appropriate selection of algorithms

After the analysis of CSP and DisCSP algorithms in the previous section, it is crucial to identify the appropriate selection of each of the algorithms and compare them regarding their solutions for solving distributed CSP problems. From the above algorithms, only Asynchronous backtracking, Asynchronous weak-commitment, and Distributed breakout algorithm are used in Distributed Constraint Optimisation in an updated version. For this reason, the most appropriate algorithm needs to be picked in order to be applied to the power system network.

In order to compare the search algorithms, we have to evaluate their efficiency by an event simulation, where each agent maintains its own simulated clock. An agent's clock means that time is incremented by one simulated time unit whenever it performs one cycle of computation. One cycle means the reading of all messages, incoming and outgoing, and the appropriate local computation. If a message occurs at time t, then the recipient receives the message at time t + 1. To analyse the algorithms, the number of cycles is considered. The evaluation results were obtained by using a distributed graph-colouring problem, where the number of agents, n, are 60, 90 and 120 and the number of possible colours is 3. Graph colouring considers the problem of assigning colours to the nodes of a graph such that adjacent nodes do not share the same colour. The optimisation version of the problem concerns the minimization of the number of colours used. The Asynchronous backtracking algorithm managed to reach a solution only for 60 agents and needed 910 cycles to obtain the solution. On the other hand, the Asynchronous weak-commitment algorithm managed to reach solutions for any number of agents with 70, 80 and 90 cycles respectively. That means that the weak-commitment algorithm outperforms the

backtracking algorithm and it can be considered the most eligible to be used in power system networks. It is clear that asynchronous backtracking algorithm cannot handle a large number of agents and this is a limitation where it cannot be used in power systems. On the other hand, Asynchronous weak-commitment can be considered as an eligible algorithm for power systems however, the number of cycles are increasing as it handles more agents. The explanation of this result can be analysed, as these algorithms are using trees as a representation of their variable and the number of messages are increasing exponentially. The difference between these algorithms lies on the fact that on the asynchronous weak-commitment search, a mistake can be revised without conducting an exhaustive search. As for the distributed breakout algorithm, the number of cycles are 120,150 and 180 for 60, 90 and 120 agents, however, it is still considered slower than weak-commitment. However, the distributed breakout algorithm managed to converge all of the time and can be considered more efficient than the Asynchronous backtracking algorithm. Finally, it is crucial to notice that when a problem is critically difficult, that means controlling hundreds of agents and constraints, such as in power system networks, then Asynchronous weak-commitment is considered the best algorithm among the DisCSP community.

3.7 Conclusions

A distributed CSP is a CSP in which the variables and constraints are distributed among automated agents. The most important fact in CSP and DisCSP is to provide consistent solutions. Therefore, distributed constraint satisfaction techniques can be considered as an important infrastructure for cooperation. The main difference between them is that CSP problems can be modelled only as centralised problems while DisCSP considers constraint problems, where knowledge is distributed among communicating agents and cannot be centralised. For power system case studies, DisCSP algorithms are considered to be adequate to be integrated rather than CSP algorithms, because the agents share their constraints and information locally hence privacy is considered among the agents. Also, CSP algorithms cannot be considered for power systems as their agents are able to handle only one variable while agents of DisCSP algorithms are able to handle multiple variables.

CSP algorithms cannot converge if they are applied in power system networks because each agent is not only responsible for its own variable but has a relation with other local variables.

An analytical description of algorithms was presented in order to understand the difference between the algorithms and how they can be applied in different applications. Also, a knowledge of the above algorithms was necessary as most of them (Asynchronous backtracking algorithm, Distributed breakout algorithm, and Asynchronous weak-commitment search algorithm) are used in Distributed Constraint Optimisation to fit in distributed and optimised environments.

The comparison of the algorithms proved that only a few of them are able to find solutions to problems with many agents and variables hence not all of them can be considered for applications in power system networks. Also, it has to be clarified that DisCSP algorithms are able to find a solution to a significant problem but that doesn't mean that the solution is optimised. Some of the algorithms stop processing when they reach a solution while others provide many solutions to a problem. For that reason, Distributed Constraint Optimisation can be characterised as the updated version of DisCSP by providing optimisation.

All the algorithms that are based on CSP and DisCSP can find solutions when the solution exists. Finally, the solutions can be defined either as satisfactory or unsatisfactory. For this reason, Distributed Constraint Optimisation can be considered the proper framework for finding solutions, in which the constraints have costs (e.g. power systems). Each constraint of the problem is defined as an optimisation function (cost function) in order to find a set of assignments for all variables to optimise the global objective function.

Chapter 4

Distributed Constraint Optimisation (DCOP)

4.1 Introduction

In this chapter, the Distributed Constraint Optimisation (DCOP) framework is analysed and algorithms and examples are presented in order to understand how the framework can be used in power systems. The difference between DCOP and CSP and DisCSP lies in the fact that the solution of any problem needs not only to satisfy the constraints and provide values for the variables but also provide an optimal solution for the given problem.

4.2 Distributed Constraint Optimisation

DCOP is a framework particularly suitably for arbitration and negotiation within decentralised and distributed multi-agent control systems where conflicting control decisions may arise. Under this framework each variable is assigned to an agent who has control of its value and every variable has a domain of corresponding values. The agents coordinate between each other in order to optimise the objective function. Moreover, there is a set of constraints that have to be satisfied and every agent knows the constraints of its variables. Formally, a DCOP is a tuple $< A, V, D, F >$ such that:

- $A = \{A_1, \dots, A_m\}$ is a set of agents.
- $V = \{V_1, \dots, V_n\}$ is a set of decision variables such that each variable is controlled by an agent.
- $D = \{d_1, \dots, d_n\}$ is a set of domains of the variables.
- $F = \{f_1, \dots, f_n\}$ is a set of functions (normally hard constraints) where f_i is any function $d_{i1} \times \dots \times d_{im} \rightarrow \Re$ which denotes how much cost is assigned to each possible combination of values of the variables.

The aim of the DCOP is to find an assignment V^* for the variables $V_i , i =, 1..n,$ such that the combined cost or utility of all the constraints, defined as the summation of all constraints $\Sigma_i f_i$ is minimized or maximized.

Under the DCOP framework, algorithms can be categorised according to: a) the search strategy b) the synchronisation among agents, and c) the communication topology. However, since the vast majority of the algorithms are search algorithms, they can be categorised as complete or incomplete algorithms. The difference between these two categories is based on the problem that needs to be solved. In cases where a relatively smaller number of messages are exchanged between the agents, the complete algorithms provide an optimal solution in less computation time compared to incomplete algorithms. On the other hand, incomplete algorithms provide suboptimal solutions but they are more efficient when the number of messages is expanding.

4.3 Application problems of Distributed Constraint Optimisation

Various problems can be modelled in the DCOP framework. Petcu and Faltings presented two different applications regarding power networks [PF07]. The first application includes a distributed scheduling for maintenance of generation units in a distributed fashion and the second application considers the distributed reconfiguration of power networks. Additionally, the DCOP framework has been applied to sensor networks. These applications include one sensor to track the target [ZWX+05] or several sensors to track the target [BDF+05]. Farinelli et al. presented a decentralised coordination of low-power embedded devices under the DCOP framework [FRPN+08]. Furthermore, Maheswaran et al. provide different examples for distributed multi-event scheduling [MTB+04]. Further applications include the generation of coalition structures [UIW10] and the coordination of first responders in disasters [LKS+08]. The vast majority of the above applications will be presented and analysed in the following section.

4.3.1 Distributed scheduling for maintenance of generation units

The maintenance scheduling problem for power networks was introduced in 1998 by Frost and Dechter [FD98] regarding the use of constraint satisfaction framework, however, Petcu and Faltings used DCOP framework to address the problem [PF07]. In their model, each generation unit has an operating cost, a maintenance cost, a generation output and also a maintenance time duration. According to their assumptions, during the maintenance, a number of units need to generate electricity in order to cover the demand and also there are few maintenance people, so proper relocation of the crew is needed. The main objective of the model is to provide a valid maintenance schedule that minimises maintenance costs and maximises the generation output. Also, an updated model which considers multiple power plants is analysed.

4.3.2 Decentralised coordination of low-power embedded devices

Farinelli et al. presented an example of low-power embedded devices under the DCOP framework in 2008 [FRP$^+$08]. The main objective of the research was the coordination of the activities of different distributed devices under the satisfaction of the constraints in order to achieve good system-wide performance. Each device is able to communicate with a neighbouring device via low power antenna and share information about constraints and variables. The novelty of the application lies in the fact that the computation required is significantly smaller than using a central authority thus avoiding communication bottlenecks.

4.3.3 Optimal Decentralised Dispatch of Embedded Generation in the Smart Grid

Another example of the DCOP framework was presented by Miller et al. solving the economic dispatch in power system networks [MRR12]. A simple electricity network of 6 generation units was analysed and decomposed under the framework. According to the optimal economic dispatch, the main objective is to coordinate all the generators in order to provide maximum generation output with the minimum cost without the violation of network constraints. In addition, their objective goal is to minimise the CO_2 emissions of the entire network. Their approach considers networks physical constraints such as

thermal line ratings, satisfaction of loads but they did not decompose the entire network physical constraints such as voltage control and frequency control. Even though, their approach was the first attempt to decompose the power network into DCOP, the lack of the most significant constraints was important.

4.4 Algorithms for solving DCOP's

In this section, various different algorithms will be presented and analysed in order to identify the appropriate algorithm for solving specific DCOP problems. These algorithms are mainly divided in two categories: a) Complete algorithms such as: Asynchronous Distributed Optimisation (ADOPT), Synchronous Branch and Bound (SynchBB), Optimisation by Asynchronous Partial Overlay (optAPO), Dynamic Programming Optimisation Protocol (DPOP), Asynchronous Forward Bounding (AFB), No-Commitment Branch and Bound (NCBB), Concurrent Forward Bounding (ConcFB) and b) Incomplete algorithms such as: Approximate DPOP(A-DPOP), The Max-Sum Algorithm, and Distributed Asynchronous Local Optimisation (DALO). All these algorithms are able to solve distributed constraint optimisation problems and either find an optimal solution or a solution close to optimality. Furthermore, additional algorithms from both categories will be presented and analysed.

4.4.1 ADOPT

ADOPT was initially presented by Modi et al. in 2005 as the first asynchronous complete algorithm for DCOP [MST+05]. ADOPT, guarantees to find the optimal solution while agents run asynchronously. The difference between ADOPT and algorithms for DisCSP is that it uses pseudo-tree for passing variables instead of a total ordering of variables. In order to understand the construction of the pseudo-tree, an analytical presentation of this process is needed in order to clarify how the variable values are passing between agents.

Pseudo-tree is a hierarchical common structure which is used in search based algorithms in order to allow the parallel processing of the variables. In a pseudo-tree, decision variables are connected via a node which signifies the relation between one or more

variables [Fre85]. There are four types of relationships between nodes in a pseudo-tree such as a) parent (P) of a node which is a node higher in the pseudo-tree and they are connected by edges b) children (C) of a node, which is a set of nodes lower in the pseudo-tree and they are connected by edges c) pseudo-parents (PP) of node which is a set of nodes higher in the pseudo-tree and they are connected by back-edges and d) pseudo-children (PC) of a node, which is a set of nodes in the pseudo-tree and they are connected by back-edges. The main advantage of pseudo-trees is the appropriate allocation of constraints between the agents. According to this procedure, each agent will only have to deal with its own constraints and also with its parent's constraints that is connected. This approach is used to the vast majority of search based algorithms in DCOP. *Figure 4.1*, illustrates an example of a pseudo-tree based on a constraint graph and the connection between parents and children.

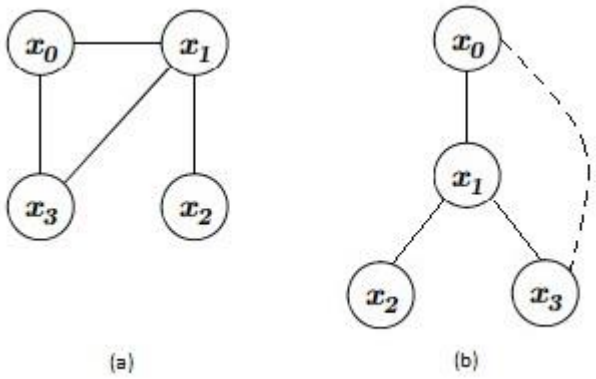


Figure 4.1: (a) Constraint graph (b) Pseudo-tree [Fre85]

In ADOPT, the constraint graph must be decomposed in a Depth- First Search (DFS) tree in order to define the relations between the agents. The DFS tree was presented by Collin et al. as a new version of the pseudo-tree [CDK91]. The DFS tree is a decomposition of the constraint graph with the same nodes and edges as the constraint graph while the pseudo-tree is a decomposition of the constraint graph but only with the same nodes. For any constraint graph, there are many possible representations of the DFS tree. In DCOP literature, the vast majority of the search based algorithms are depending on DFS trees as a presentation of the constraint graph.

The main idea behind ADOPT is that it uses local communication between agents. Each agent communicates only with the neighbour agents, thus all agents run asynchronously in parallel without waiting for the decisions from higher or lower agents. In ADOPT, each agent runs asynchronously and whenever it detects that there is a possibility of a better solution then automatically changes its variable value. This process decreases the computation time as there is no need of global information in order to provide a solution. ADOPT belongs to the backtrack search algorithms category, and after the decomposition of constraint graph to DFS tree, it performs a backtrack search. At any time during the process, each agent chooses the value with the smallest lower bound (lb). *VALUE* messages are sent from the top to bottom. These messages contain variable value assignments. *COST* messages are flowing from children to parents and they are actually working as *nogood* messages in DisCSP. Also, there are *THRESHOLD* messages which are sent only from parents to children and are used for reducing redundant search. Each agent receives the message for its child which contains the COST and adds it to its current value. If in its domain there is a value that has smaller lower bound, then automatically the agent chooses the new value and repeats the process. The novelty lies in the asynchronous nature of the agents where their values can change any time during the process if they found better values. *Figure 4.2*, illustrates the DFS tree and the flowing of messages between parents and children.

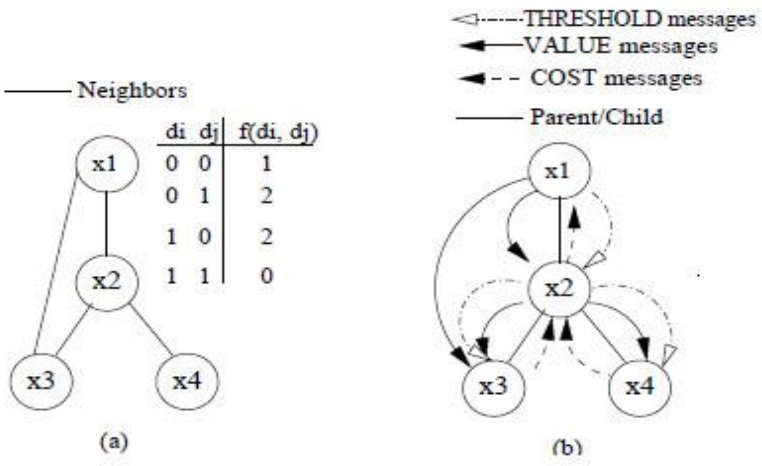


Figure 4.2: (a) DFS tree (b) Flowing of messages in DFS tree [MST+05]

```
Initialize
threshold ← 0; CurrentContext ← {};
forall d ∈ D_i, x_i ∈ Children do
  lb(d, x_i) ← 0; t (d, x_i) ← 0;
  ub(d, x_i) ← Inf; context (d, x_i) ← {}; enddo;
 d_i ← d that minimizes LB(d);
 backTrack;

when received (THRESHOLD, t, context)
  if context compatible with CurrentContext:
    threshold ← t;
    maintainThresholdInvariant;
    backTrack; endif;

when received (TERMINATE, context)
    record TERMINATE received from parent;
    CurrentContext ← context;
    backTrack;

when received (VALUE, (x_j, d_j))
    if TERMINATE not received from parent:
      add (x_j, d_j) to CurrentContext;
      forall d ∈ D_I, x_i ∈ Children do
        if context(d, x_i) incompatible with CurrentContext:
          lb(d, x_i) ← 0; t (d, x_i) ← 0;
          ub(d, x_i) ← Inf; context (d, x_i) ← {}; endif; enddo;
      maintainThresholdInvariant;
      backTrack; endif;
when received (COST, x_k, context, lb, ub)
   d ← value of x_i in context;
   remove (x_i, d) from context;
   if TERMINATE not received from parent:
     forall (x_j, d_j) ∈ context and x_j is not my neighbour do
       add (x_j, d_j) to CurrentContext;enddo;
     forall d' ∈ D_i, x_l ∈ Children do
       if context(d', x_l) incompatible with CurrentContext:
         lb(d', x_l) ← 0; t (d', x_l) ← 0;
         ub(d', x_l) ) ← Inf; context (d', x_l)) ← {};endif;enddo;
   if context compatible with CurrentContext:
     lb(d, x_k) ←lb;
     ub(d, x_k) ←ub;
     context (d, x_k)← context;
  maintainChildThresholdInvariant;
  maintainThresholdInvariant; endif;
 backTrack;

procedure backTrack
      if threshold== UB:
        d_i ← d that minimizes UB(d);
      else if LB(d_i) > threshold:
    d_i ← d that minimizes LB(d);endif;
      SEND (VALUE,( x_i, d_i))
          to each lower priority neighbour;
      maintainAllocationInvariant;

if threshold== UB:
      if TERMINATE received from parent
        or x_i is root:
    SEND(TERMINATE,
          CurrentContext ∪ {(x_i, d_i)})
      to each child;
          Terminate execution;endif;endif;
  SEND (COST, x_i, CurrentContext, LB, UB)
    to parent;
```

Figure 4.3: Pseudo-code of ADOPT algorithm for receiving messages [MST+05].

The pseudo-code of ADOPT algorithm example is presented in *Figure 4.3*. This example depicts the flows of *VALUE* and *COST* messages. At the beginning of the process, its agent picks a value for its variable from its domain. x_2 will receive x_1's VALUE message

and will record this value in the *CurrentContext*. *CurrentContext* contains the higher neighbours. After that process, the backtrack procedure starts. The upper bound (ub) will be defined and it will be compared with the threshold. Furthermore, the x_2 sends a COST message to x_1. At the same time, x_3 evaluate its constraints and sends a COST message to its parent x_2. The same procedure lies for x_4. When x_1 receives the COST message from x_2 then the received costs will be added to the x_1's domain. Then x_1 will pick the appropriate value that minimizes the lower bound and will change its value of this if it is necessary. Furthermore, x_2 receives the COST messages from x_3 and x_4, updates its *CurrentContext* and stores all the value costs. However, this occurs after x_2 informed x_3 and x_4 about its new values through VALUE messages. For this example the THRESHOLD was equal to 0 all the time. For the termination of the algorithm the upper bound has to be equal to THRESHOLD in order for x_1 to terminate. Additionally, the x_1 sends termination message to x_2 and x_2 to its children x_3 and x_4 in order to terminate the process. One of the main advantages of using ADOPT is that it contains an in built bounded error approximation mechanism. As the algorithm operates, the upper and lower bounds converge towards a solution.

The three key ideas in ADOPT are "a) to perform distributed backtrack search using a novel search strategy where agents are able to locally explore partial solutions asynchronously, b) backtrack thresholds for more efficient search and c) built-in termination detection" [MST+05].

The ADOPT algorithm was initially presented with the assumption that each agent is assigned to a single variable and the constraints can handle only two variables. These assumptions can be problematic in complex examples. Furthermore, the number of messages that are exchanged are exponential in the depth of the pseudo-tree. Updated versions of ADOPT such as ADOPT-ng [SY06] and BnB-ADOPT [YFK08] solve these problems and will be presented and analysed later in this chapter.

4.4.2 Asynchronous Branch-and-Bound (BnB-ADOPT)

The BnB-ADOPT algorithm was introduced by Yeoh et al. in 2010 [YFK08]. The difference between this algorithm and ADOPT lies on the search strategy that each algorithm performs. ADOPT, performs a best-first search strategy while BnB-ADOPT performs depth-first branch and bound search strategy. In BnB-ADOPT each agent communicates only with the neighbouring agents and agents run simultaneously and asynchronously. Depth-first branch and bound strategy finds a complete solution and stores it to the upper bound. Furthermore, if it finds a solution that is less costly than the upper bound then it updates the upper bound with the new cost value and continues the search process. The search process stops until it finds a cost solution that is less than the upper bound. *Figure 4.4* illustrates the pseudo-code of BnB-ADOPT.

At the beginning of the algorithm each agent stores its values of its domain in lower and upper bounds and also the values of all the ancestors' children, however, it stores only one context of each agent which is a set of value assignments. This context contains the variable or variables of the given agent and is not visible to other agents. Each agent sends VALUE messages to all the children with the context of the variable that belongs to the agent's domain. Therefore, the children send COST messages with lower and upper bounds. The agent receives the COST messages and updates its lower and upper bounds. In addition, the agent sends COST messages to its parent. If the lower bound is smaller than the upper bound the process is repeated until the lower bound is equal to the upper bound. If the bounds become equal which means that there is no further improvement, the agent receives the new value of the lower bound and repeats the process until its context changes. The process is repeated because the agent receives a different context from its parent. The above process described the agent's operation, however, the most important and valuable point lies in the search strategy. Each agent holds a threshold which has an infinity value at the beginning of the process. The threshold is used in order to reduce the number of values that each agent holds. There are many similarities with the ADOPT algorithm as both algorithms use the same VALUE, COST and TERMINATE messages and also the same process for the lower and upper bounds. The difference lies in the use of the threshold

procedure Start()
X^a := {$(a', \boldsymbol{ValInit}(a'), 0) \mid a' \in SCP(a)$}; SCP: set of ancestors
ID^a := 0;
forall $a' \in C(a), d \in Dom(a)$ Dom : Domain of agent
  InitChild (a', d);
InitSelf ();
Backtrack ();
loop forever
  if (message queue is not empty)
    while (message queue is not empty)
      pop msg off message queue;
      When Received (msg); **Backtrack** ();
procedure InitChild(a', d)
$lb^{a,a'}$:= $h^{a,a'}(d)$;
$ub^{a,a'}(d)$:= ∞;
procedure InitSelf ()
d^a:= $argmin_{d \in Dom(a)}\{\delta^a(d) + \sum_{a' \in C(a)} lb^{a,a'}(d)\}$;
ID^a := ID^a + 1;
TH^a := ∞;
procedure Backtrack()
  forall $d \in Dom(a)$
  $LB^a(d)$:= $\delta^a(d) + \sum_{a' \in C(a)} lb^{a,a'}(d)$;
  $UB^a(d)$:= $\delta^a(d) + \sum_{a' \in C(a)} ub^{a,a'}(d)$;
  LB^a := $min_{d \in Dom(a)}\{LB^a(d)\}$;
  UB^a := $min_{d \in Dom(a)}\{UB^a(d)\}$;
  if ($LB^a(d^a) \geq \min\{TH^a, UB^a\}$)
    d^a := $argmin_{d \in Dom(a)}\{LB^a(d)\}$ (choose the previous d^a if possible);
    if a new d^a has been chosen
      $ID^a := ID^a + 1$;
if ((a is root and $UB^a \leq LB^a$) or termination message received)
  Send(TERMINATE) to each $c \in C(a)$; terminate execution;
Send(VALUE, a, d^a, ID^a, $\min\{TH^a, UB^a\} - \delta^a(d^a) - \sum_{c' \in C(a) \setminus c} lb^{a,c'(d^a)}$)
Send(VALUE, a, d^a, ID^a, ∞) to each $c \in CD(a) \setminus C(a)$;
Send(COST, a, X^a, LB^a, UB^a) to $pa(a)$ if a is not root;
procedure When Received(VALUE, p, d^P, ID^P, TH^P)
  $X' := X^a$
  PriorityMerge($(p, d^P, ID^P), X^a$);
  if (!**Compatible**(X', X^a))
    forall $c \in C(a), d \in Dom(a)$
      if ($p \in SCP(c)$)
        InitChild(c, d);
    InitSelf ();
  if ($p = pa(a)$) pa: parent of the agent
    $TH^a := TH^P$;
procedure When Received($TERMINATE$)
  record termination message received;

Figure 4.4: Pseudo-code of BnB-ADOPT [YFK08]

where ADOPT uses it only to reconstruct a partial solution in comparison to BnB-ADOPT where the threshold is used in order to reduce the number of the values. This process provides a faster solution than ADOPT, especially for large DCOP problems.

4.4.3 Optimisation Asynchronous Partial Overlay (OptAPO)

OptAPO was initially presented by Mailler and Lesser in 2004 as an optimised version of APO algorithm [ML04]. The algorithm uses dynamic and partial centralization where each agent acts as a mediator and provides a partial solution to the problem. Furthermore, it stores and centralises all these solutions in order to provide a solution from the centralised solver. According to the authors, OptAPO is faster than ADOPT and also the message complexity is smaller than ADOPT. *Figure 4.5* illustrates an example of how the algorithm works in the 2-colouring problem.

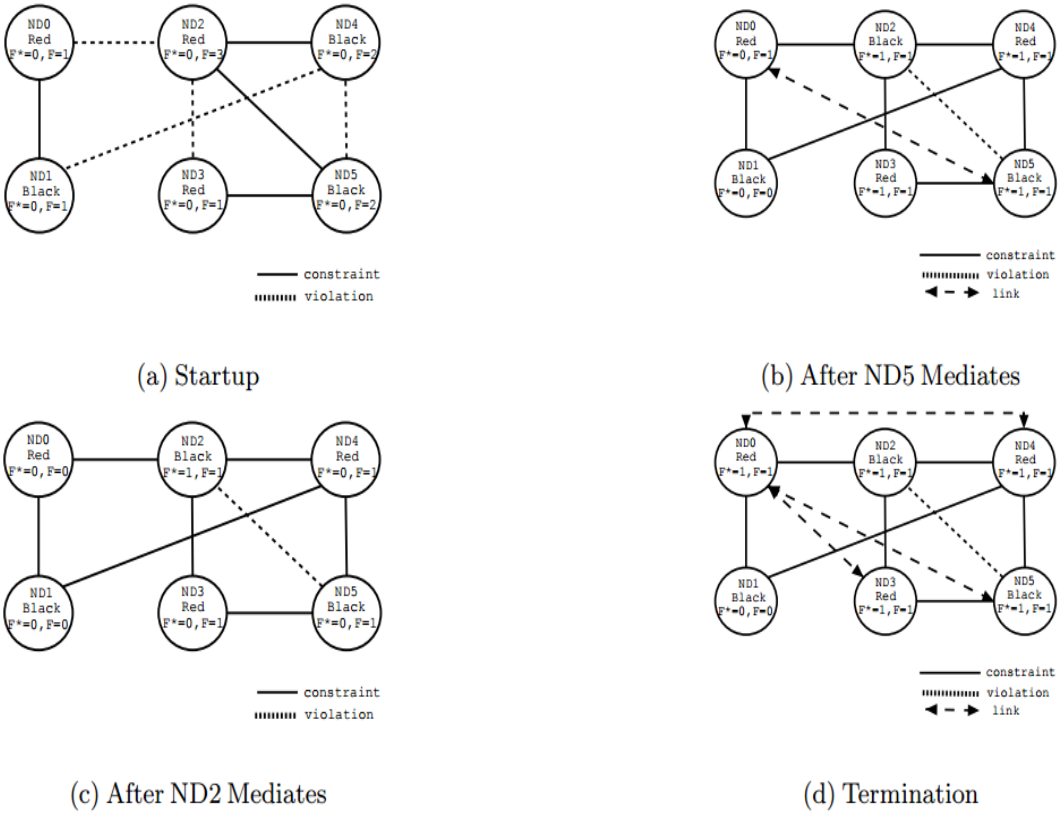


Figure 4.5: OptAPO 2-coloring problem example [ML04]

In this problem, each agent controls one variable and there are 8 relations between the variables. The main objective of the problem is to find the solution that minimises the total

cost. Each relation is associated with a cost of 1 if it's violated and with a cost of 0 if it's not. At the beginning of the process, each agent sends messages only to neighbour agents. For instance, ND4 sends messages to ND2, ND3, and ND5. When all the messages received by each agent, the agents check their *agent_view*. Furthermore, each agent is able to compute their internal optimal subsystem value which was 0 at the beginning of the process. For example, ND2 has an internal value of 3 because it knows the relations of ND0 and ND2, ND2 and ND3 ad finally ND4 and ND5. Additionally, ND2 is the priority agent and acts as a mediator in order to communicate with ND0, ND3, ND4, and ND5. Each of the neighbour agents labels their domain elements and send a message to ND2. Each agent sends a message that describes the conflicts with their neighbour agents. For example, ND5 sends a message that describes the black conflict with ND4 and red conflict with ND2 and ND3. The mediator receives the messages and performs a Branch and Bound search in order to find the suboptimal internal value that has 0 conflicts with the agents. After this process, it informs the appropriate agents to change their values. In the example, ND2 informs ND4 to change its value to Red and changes its own colour to Black. The process continues in the example after two mediation sessions until the agents change their colours in order to find the optimal cost.

An updated version of the algorithm, CompAPO, was presented by Grubshtein et al. in 2008 and proves the completeness of the algorithm [GM08]. Also, the experimental evaluation shows that it outperforms ADOPT in the total number of the messages as it is able to solve the problems in the centralised method.

4.4.4 No-Commitment Branch and Bound (NCBB)

NCBB algorithm was presented by Chechetka and Sycara in 2006 [CSY06]. The algorithm is based on the Branch and Bound search and performs a parallelization technique that improves the speed of the search and also reduces the amount of information that messages exchange between the agents. In addition, the algorithm is based on a pseudo-tree ordering of the variables and each agent shares information only with one parent and multiple children. Each agent shares information only with the agents that have the same constraint information. At the beginning of the process, each agent

selects its values and sends the information to the corresponding descendants. Therefore, when the agents receive the *search* messages from their parent in order to start the process of searching, they already know the values of the descendants. The search method is performed by branching, where each agent chooses the value to explore in the tree and for each value that is picked the total cost is stored and updated.

4.4.5 Concurrent Forward Bounding (ConcFB)

ConcFB was presented by Netzer et al in 2010 [NMG10]. The algorithm is performing multiple parallel searches on disjoint parts of the search space and each search is performed by forward bounding. It was proven that the algorithm is 4 times faster than BnB-ADOPT and sends 6 times fewer messages.

4.4.6 Dynamic Programming Optimisation Protocol (DPOP)

DPOP was initially introduced by Petcu and Faltings in 2005 [PF05]. This algorithm in comparison to the vast majority of DCOP algorithms is performing dynamic programming across the pseudo-tree. The algorithm can be modelled in three different phases. The first phase contains the construction of the pseudo-tree. The second phase contains the propagation of messages, from the leaves of the pseudo-tree to the root, through UTIL messages. Finally, the third phase contains the propagation of messages, from the root of the pseudo-tree to the leaves, through VALUE messages. One of the most important features of DPOP is that requires a linear number of messages in order to find the optimal solution hence decreasing the communication time.

Figure 4.6 illustrates the pseudo-code of DPOP and provides the analytical description of every phase. For a better understanding of the algorithm, each phase will be analysed and more importantly, an example of the algorithm will be presented.

DPOP (X, D, R)
Each agent X_i executes:
Phase 1: pseudo-tree creation
elect leader from all $X_J \in X$
elected leader initiates pseudo-tree creation
afterwards, X_i knows P(X_i), PP(X_i), C(X_i) and PC(X_i)

Phase 2: *UTIL* message propagation
if $|Children(X_i)| = = 0$ **then**
  $UTIL_{X_i}$(P(X_i)) ← Compute utils (P(X_i), PP(X_i))
  Send message (P(X_i), $UTIL_{X_i}$ (P(X_i)))
activate UTIL_Message_handler()
Phase 3: *VALUE* message propagation
activate VALUE_Message_handler()
END ALGORITHM
UTIL Message handler(X_k**,** $UTIL_{X_k}$ (X_i)**)**
store $UTIL_{X_k}$ (X_i)
if *UTIL* messages from all children arrived **then**
  if Parent(X_i)==null (*that means X_i is the root*) **then**
    $u_i^* \leftarrow Choose\_optimal(null)$
    Send $VALUE(X_i, u_i^*)$ to all C(X_i)
  else
    $UTIL_{X_i}$ (P(X_i)) ← Compute utils(P(X_i), PP(X_i))
    Send message(P(X_i), $UTIL_{X_i}$ (P(X_i)))
return
VALUE_Message_handler($VALUE_{P(X_i)}^{X_i}$**)**
add all $X_k \leftarrow u_k^* \in VALUE_{P(X_i)}^{X_i}$ to agent_view
$X_i \leftarrow u_i^* = Choose\_optimal(agent\ view)$
Send $VALUE_{X_i}^{X_l}$ to all $X_l \in$ C(X_i)
Choose_optimal (agent view)
$u_i^* \leftarrow argmax_{u_i} \sum_{X_l \in C(X_i)} UTIL_{X_l}(u_i, agent\_view)$
return u_i^*
Compute_utils(P(X_i**), PP(**X_i**))**
for all combinations of values of $X_k \in$ PP(X_i) **do**
  let X_j be Parent(X_i)
  similarly to DTREE, compute a vector $UTIL_{X_i}$ (X_j)
  of all $\{Util_{X_i}(u_i^*, (u_j), u_j) | u_j \in Dom(X_j)\}$
assemble a hypercube $UTIL_{X_i}$ (X_j) out of all these vectors (totaling |PP(X_i)| + 1 dimensions).
return $UTIL_{X_i}$ (X_j)

Figure 4.6: Pseudo-code of DPOP [PF05]

DPOP works on the representation of the problem as a constraint graph, in which the decision variables are nodes (which will behave like virtual agents exchanging messages), and in which two nodes are connected by an edge if and only if they are involved in a common constraint. This constraint graph is then ordered into a pseudo-tree, which can be constructed by starting from one variable and carrying out a depth-first traversal of the constraint graph. This produces a pseudo-tree, which is basically a tree in which nodes are also allowed to have edges with remote ancestors, but edges between nodes in different branches of the pseudo-tree are not allowed.

Once the pseudo-tree is constructed, it is important to understand which variable is responsible for enforcing which constraint. Given a constraint, the variable that is responsible for enforcing it is the variable in the scope of the constraint that is the lowest in the pseudo-tree. Furthermore, by definition of the pseudo-tree, edges between nodes/variables in different branches are not allowed, thus all the variables involved in a given constraint are in the same branch, and therefore the lowest one is properly defined.

The UTIL propagation phase is the first phase of the algorithm after the construction of the pseudo-tree. In this phase, messages are flowing from leaves to the root, propagating the optimal cost. At the beginning of the algorithm, each variable is computing the sum of the constraints including its constraints values. In order to do that, each variable waits for messages from all the children. After receiving the messages, it calculates the sum of all the constraints plus the relations with pseudo-parent if there is any linking connection. However, it is possible that a constraint may hold multiple variables. If this situation occurs, then the variable ignores all the constraints that involve at least one descendant of the variable. After the summation, the variable sends a message to the parent which contains the optimal cost of all the children, a function of variable's ancestors in the pseudo-tree. Each variable that is not a leaf waits until it has received one UTIL message from each of its children in the pseudo-tree, and then adds them up, adds the constraints it is responsible for enforcing, projects itself out, and sends the resulting cost function to its parent in a UTIL message. This UTIL message then basically contains a cost function that corresponds to the optimal cost achievable by the variable and all its subtree, as a function of all the variables higher in the pseudo-tree that are connected to a variable or any of variable's descendants. When the propagation of UTIL messages reaches the root

variable of the pseudo-tree, the root variable does the same and is then able to choose a value for itself that is optimal for the whole problem. *Figure 4.7*, illustrates the process of sending a message from leaves to the root.

DPOP phase 2: UTIL propagation
$JOIN_i^{P_i} = null$
forall $X_j \in C_i$*/* for all children of* X_i*; if* X_i *is a leaf, skip this */* **do**
wait for $UTIL_j^i$ message to arrive from X_j
$JOIN_i^{P_i} = JOIN_i^{P_i} \oplus UTIL_J^i$ *//we add to the join UTIL messages from children as they arrive*
$JOIN_i^{P_i} = JOIN_i^{P_i} \oplus R_i^{P_i} \oplus (\oplus_{X_j \in PP_i} R_i^j)$ *//also join all relations with parent/pseudo-parents*
$UTIL_i^{P_i} = JOIN_i^{P_i} \perp X_i$ *//use projection to eliminate self out of message to parent*
Send $UTIL_i^{P_i}$ message to P_i

Figure 4.7: UTIL message propagation [PF05]

When messages arrive at a parent the join (sum) utility is obtained by performing a sum of children's relations. The symbol $\oplus$ represents the join operator between the messages. Furthermore, a join needs to be obtained also with parent's relations or with any potential pseudo-parents. The final message uses projection to eliminate self out of message to parent. Symbol $\perp$ represents the projection operator.

DPOP phase 3: VALUE propagation
wait for $VALUE_i^{P_i}((Sep_i)*)$ msg from P_i *//* $(Sep_i)*$ *is the optimal assignment for all*

$X_i^* \leftarrow argmax i_{vi \in di}\ (JOIN_i^{P_i}\{(Sep_i)*\})$ *// slice* $JOIN_i^{P_i}$ *corresponding to* $(Sep_i)*$*; find best* v_i
forall $X_j \in C_i$ */* for all children of* X_i*; if* X_i *is a leaf, skip this */* **do**

send $VALUE((Sep_i)* \cap (Sep_j)*) \cup X_i^*)$ message to X_j

Figure 4.8: VALUE message propagation [PF05]

At the end of the UTIL propagation phase, the root variable has picked the optimal value and sends a VALUE message to the corresponding children. Each child that has received the message from the parent is performing a search to its corresponding domain. It picks the appropriate value that satisfies the constraints and sends a message to the corresponding children. The process terminates until all the children receive messages from their parents and calculate their own optimal values for their variables. *Figure 4.8*, illustrates the process of sending a message from root to the leaves.

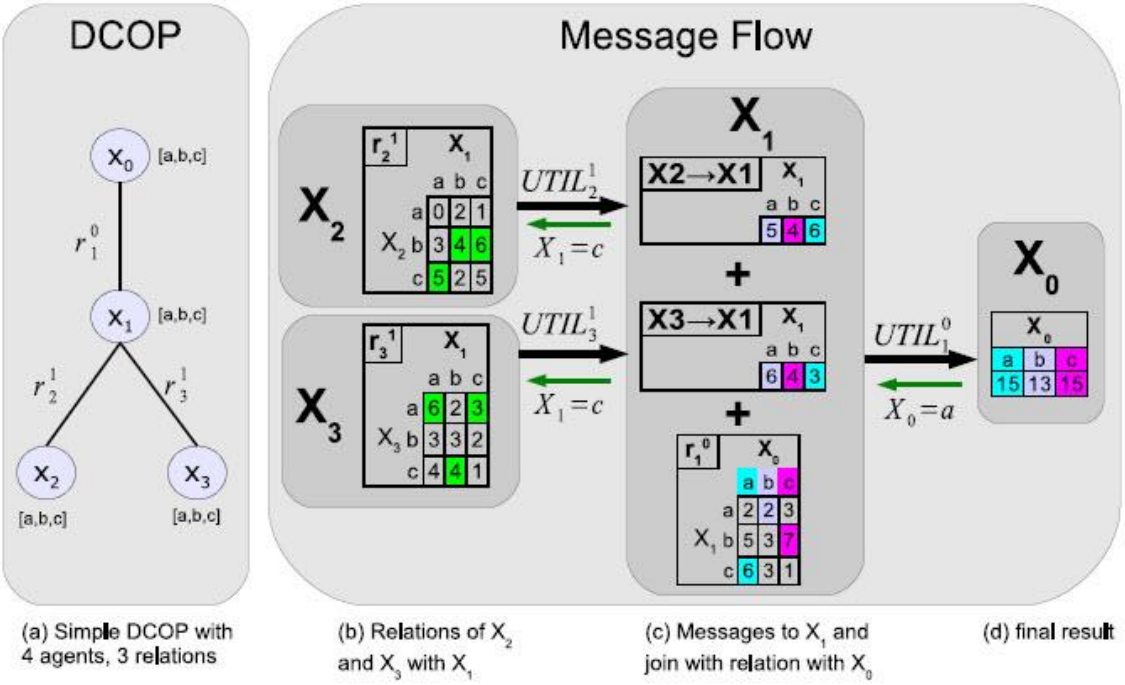


Figure 4.9: DCOP example using DPOP algorithm [PF05]

A simple example of DPOP algorithm is presented in *Figure 4.9*. The DCOP problem consists 4 agents (x_0, x_1, x_2 and x_3). Each agent contains a domain of a, b and c values and there are relations between the agents such as: r_2^1 is the relation between x_2 and x_1, r_3^1 is the relation between x_3 and x_1 and finally r_1^0 is the relation between x_1 and x_0. *Figure 4.9 (b)* illustrates the relations between the agents and the corresponding values of their domains. UTIL phase begins by sending UTIL messages from $x_2 \rightarrow x_1$ and from $x_3 \rightarrow x_1$.

The $UTIL_2^1$ message sends only the maximum values of every variable of its domain. For instance it sends the values of $a = 5$, $b = 4$ and $c = 6$. The same procedure occurs for $UTIL_3^1$ message where $a = 6$, $b = 4$ and $c = 3$. $x_{1,}$ receives the $UTIL_2^1$ and $UTIL_3^1$ message and join them with its relation r_1^0. In addition, it projects itself out and is ready to send a VALUE message. This message contains the values that x_0 picked regarding the utility cost of the whole problem. From the calculations of the join messages $a = 15$, $b = 13$ and $c = 15$. Both a and b are equal so either one can be chosen. For this example $x_0 = a$ is choosen to be sent as a VALUE message to x_1. x_1 receives the VALUE message from x_0, restores its value that was found optimal for x_0 and chooses as an optimal value $x_1 = c$. Moreover, it sends two VALUE messages to x_2 and x_3 in order to inform them about its optimal value. x_2 and x_3 restore their optimal values regarding to x_1 and choose $x_2 = b$ and $x_3 = a$. Finally, the maximum utility of the whole problem is 15, because $x_0 = a$, $x_1 = c$, $x_2 = b$ and $x_3 = a$.

4.4.7 Open Dynamic Programming Optimisation (O-DPOP)

O-DPOP was introduced by Petcu and Faltings in 2006 as an updated version of DPOP [PF06]. The advantages over DPOP based on the fact that size of messages is linear among the width of the pseudo-tree hence it reduces overall computation time. The algorithm is based on 3 phases like DPOP where phase 1 of construction the pseudo-tree and phase 3 of propagating the VALUE are almost identical. The difference lies on phase 2 regarding the propagation of UTIL messages. *Figure 4.10* provides the pseudo-code of O-DPOP and the difference in UTIL propagation compared to DPOP where this phase is called the ASK/GOOD phase. At the beginning of the algorithm, the parents do not assign any values to themselves and they ASK the children in order to provide them with the best values. Children receive the ASK messages from their parents and answer to these messages by proposing values for the parent's variables. These proposals are called GOOD messages and contain the associated cost of the children. The phase terminates when the parent receives the GOOD messages that would be ideal for finding the optimal value for their variables. In O-DPOP, a GOOD message corresponds to UTIL message from DPOP but the difference is that the valuations are sent on demand, and the parent sends ASK messages to children in order to find the best combination of values for the variables.

O-DPOP (X, D, R): each agent x_i does:
DFS arrangement
At completion, x_i knows $P_i, PP_i, C_i, PC_i, Sep_i$ Sep: separator of x_i
Main process
$sent\_goods \leftarrow \emptyset$
if x_i is $root$ **then**
    ASK until conditional sufficiency/GOOD
else
    while *!received VALUE message* **do**
        Process incoming ASK and GOOD messages
Process ASK
while *!conditional sufficiency* **do**
    select C_i^{ask} among C_i
    send ASK message to all C_i^{ask}
    wait for GOOD messages
find *best_good* ∈ Sep_i s.t. *best_good* /∈ *sent_goods*
add *best_good* to *sent_goods*, and send it to P_i
Process GOOD(g_d, x_k)
add *gd* to *goodstore*(x_k)
check for conditional sufficiency

Figure 4.10: Pseudo-code of O-DPOP [PF06]

4.4.8 Private Dynamic Programming Optimisation Protocol (P-DPOP)

P-DOP was introduced by Faltings et al in 2008 [FLP08]. This algorithm was presented as an updated version of DPOP dealing with the privacy guarantees among the agents. The vast majority of the DCOP and also DisCSP algorithms such as ADOPT, ABT and DPOP do not provide any privacy guarantee. Furthermore, it is crucial to provide the adequate privacy guarantees among the agents such that the coordination constraints will not be leaked. The authors defined 4 different types of privacy guarantees: “a) Agent privacy, where each agent knows only information of an agent that share a coordination constraint b) Topology privacy, where each agent does not know any information regarding the topology except if it involves a variable that has a constraint with c) Constraint privacy, where each agent does not know any other information about other constraints except the one that is involved in and d) Decision privacy, where each agent does not know any decision outcome from other agents in the solution” [FLP08].

The algorithm is based on DPOP, so the main process includes the same 3 phases of construction of the pseudo-tree, the UTIL and VALUE propagation of messages. The difference lies in the fact that DPOP did not provide any privacy guarantees. For instance, at the construction of the pseudo-tree, each agent knows the information of all the ancestors and also any links that are existed between them and pseudo-parents. The same situation occurs regarding UTIL and VALUE propagation. *Figure 4.11*, presents the pseudo-code of P-DPOP for a variable.

Elect a root variable

Construct a pseudo-tree rooted at the elected root
// Choose and exchange codenames for x and its domain D_x:
Wait for messages $(CODES, y_i^x, D_{y_i}^x, \sigma_{y_i}^x)$ from all $y_i \in \{parentx_x\} \cup pseudo\_parents_x$
for each $y_i \in children_x \cup pseudo\_children_x$ **do**
    $x^{y_i} \leftarrow$ large random number
    $D_x^{y_i} \leftarrow$ list of $|D_x|$ random, unique identifiers
    $\sigma_x^{y_i} \leftarrow$ random permutation of $[1, \dots, |D_x|]$
    Send message $(CODES, x^{y_i}, D_x^{y_i}, \sigma_x^{y_i})$ to y_i
// **Choose and exchange obfuscation key for x:**
Wait for and record a message $(KEY, key_{y_i}^x)$ from each $y_i \in pseudo\_parents_x$ (if any)
for each $y_i \in pseudo\_children_x$ **do**
    $key_x^{y_i} \leftarrow$ vector of large random numbers, indexed by D_x
    Send message $(KEY, key_x^{y_i})$ to y_i
Propagate cost values up the pseudo-tree
// Propagate decisions top-down along the pseudo-tree :
if x is not the root **then**
    Wait for message $(VALUE, p_x^*)$ from parent p_x
    $x^* \leftarrow x^* (p_x = p_x^*)$
for each $y_i \in children_x$ **do**
    Send message $(VALUE, sep_{y_i}^*)$ to y_i, with sep_{y_i}

Figure 4.11: Pseudo-code of P-DPOP [FLP08]

At the beginning of the algorithm, the pseudo-tree needs to be constructed. In order to achieve that, the root variable needs to be chosen. The root variable is the one that provides the topological privacy guarantee. The next phase is the UTIL propagation where the algorithm uses the same technique as DPOP by propagating cost messages from the leaves

to the root. The difference is that the technique which is used in P-DPOP protects the private information of UTIL messages by introducing codenames and random numbers. To understand the process let us consider the UTIL message x_2 to x_1 and let as assume that there is a link between x_2 and x_0. In DPOP, the message will be sent but without any agent and topology privacy guarantee because it is dependent on the pseudo-parent x_0. In P-DPOP, to guarantee the privacy, codenames need to be introduced. For example, the variable x_0 and its domain are replaced with random codenames. Moreover, these codenames are sending back to the pseudo-child x_2. Variable x_2, knows the domain of x_0 in terms of codenames before sending its UTIL message to x_1. The variables which are located between the leaf and the root passing the UTIL messages without knowing the domain of x_0 as its domain is encrypted by a codename. Using codenames, the algorithm is able to provide agent and topology privacy. As for the constraint privacy the authors introduced the obfuscation method. The obfuscation is performed by adding random numbers to cost that are linked to variable x_0. The numbers are passed through the UTIL messages from x_2 to x_1 until the message reach the root variable which is x_0 in our case. The phase terminates when the root variable receives the last UTIL message and subtracts the random numbers in order to calculate the cost.

The last phase of the algorithm takes into account the propagation of VALUE messages from the root variable to the leaves. The optimal cost that was calculated through the UTIL propagation from the root variable x_0 needs to be propagated to the leaves. Each variable receives the message from its parent and calculates the optimal cost regarding its own domain. Furthermore, the same procedure as UTIL propagation is used in order to guarantee the agent and topology privacy. As for the decision privacy, the algorithm cannot provide an absolute guarantee as the children variables know the variable values of parents but not the variable values of non-neighbouring variables.

4.4.9 Stochastic Uncertainty using Collaborative Sampling (EDPOP)

EDPOP algorithm was introduced by Léauté and Faltings in 2009 as an extension of DPOP algorithm [LF09]. The authors introduced the framework of DCOP under Stochastic Uncertainty (StochDCOP) in which random variables with known probability

distributions are used to model sources of uncertainty. The difference on this framework to the original DCOP lies on the fact that the variables are random and there is a set of probability distributions where each distribution defines the probability law for a random variable.

Each agent performs the procedure:
// **Bottom-up phase**
$R_0 \leftarrow \{r \in R \mid x\ is$ constrained with $r\}$
$S_0 \leftarrow \{sample(\pi_r) \mid r \in R_0\}$
wait for all $\{R_i, S_i\}_{i=1\ldots t}$ from all children
$R_x \leftarrow U_i \geq R_i$
$S_x \leftarrow U_i \geq S_i$
send $\{R_x, S_x\}$ to parent (if any)
if $R_x = \emptyset$ **then**
execute UTIL phase
// **Top-down phase**
$S \leftarrow \emptyset$ // all known sample sets
$R_x \leftarrow U_{1\leq i<j}(R_i \cap R_j) \cup R_0$
if $a(x)$ has a parent p **then**
wait for $\{R_p, S_p\}$ from parent
$S \leftarrow S_p$
$R_x \leftarrow R_x \cap R_p$
for all $r \in Rx$ **do**
$lca(r) \leftarrow a(x)$
$S_r \leftarrow sample(\cup_{S_r \in S_x} S_r)$
$S \leftarrow S \cup \{S_r\}$
for all children $i = 1 \ldots t$ such that $Ri \neq \emptyset$ **do**
send $\{(R_i \cap R_p) - R_x, \{S_r \in S \mid r \in R_i\}\}$ to child i
execute UTIL phase

Figure 4.12: Pseudo-code of the collaborative sampling [LF09]

The framework provides the same complexity as DCOP regarding the size and the number of the messages, however, the quality of solution can be inconsistent. To solve this issue, the authors introduced an algorithm, called collaborative sampling where for each random variable the agents propose sample sets for this variable and choose one agent in order to be responsible for picking a sample from all the proposed samples.

Figure 4.12, illustrates the pseudo-code of the collaborative sampling. The algorithms illustrate how agents compute *lca*, which is the lowest ancestor in pseudo-tree and is responsible for picking a sample for a random variable. There are two phases in the algorithm: “a) bottom-up phase, where the propagation of the messages that contain the random variables and the sample sets is starting from the leaves of the pseudo-tree to the root and b) top-down phase, where the propagation starts from the root of the pseudo-tree to the leaves after each agent knows if it is the *lca* of any random variable” [LF09].

4.4.10 Max-sum algorithm

DCOP problems can be solved by a wide range of algorithms, either complete or incomplete, however, the vast majority of them suffer from exponential computational cost with respect to the optimal solution and the number or size of the messages passed among the agents. However, the max-sum algorithm is an incomplete search algorithm which belongs to the Generalized Distributive Law [AM00] framework and was introduced by Farinelli et al. in 2008 [FRP+08]. The algorithm is able to provide good approximate solutions for mesh networks and also feasible results. Typically, the max-sum algorithm is applied in a factor graph which is a bipartite graph with function nodes and variable nodes. A factor graph is the graphical representation of the dependencies between variables and factors. *Figure 4.13*, illustrates a factor graph example where the variables are only connected to functions and the functions are only connected to variables.

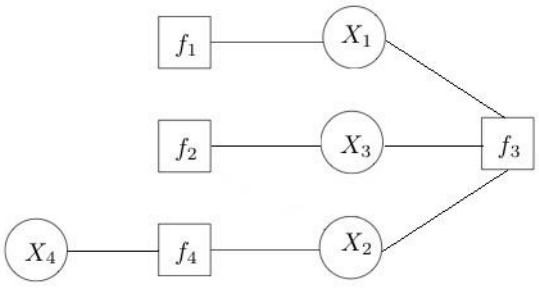


Figure 4.13: Factor Graph example

Max-sum is a message passing algorithm, which means it propagates the utilities of the variables through messages across the factor graph. The messages flow from variable node

x_i to function node f_j noted as $q_{i\to j}$ and also from function node f_j to variable node x_i noted as $r_{j\to i}$.

The messages can be defined as:

- From variable to function

$$q_{i\to j}(x_i) = a_{ij} + \sum_{j'\in M(i)\setminus j} r_{j'\to i}(x_i) \tag{4.1}$$

where $M(i)$ is a set of function indices, indicating which function nodes are connected to variable node i, and a_{ij} is a scalar such that

$$\sum_{x_i} q_{i\to j}(x_i) = 0 \tag{4.2}$$

The reason for adding a scalar is to prevent utility values from increasing endlessly in cyclic factor graphs.

- From function to variable

$$r_{j\to i}(x_i) = max_{x_{j\setminus i}}\left(f_j(x_j) + \sum_{i'\in N(j)\setminus i} q_{i'\to j}(x_{i'})\right) \tag{4.3}$$

where $N(j)$ is a set of variable indexes, indicating which variable nodes are connected to function node $j, f_j(x_j)$ is the utility function and $x_{j\setminus i} \equiv \{x_{i'}: i' \in N(j)\setminus i\}$. The local utility of each agent can be evaluated by the sum of all the messages flowing into the agent's variable and it can be defined as:

$$Z_i(x_i) = \sum_{j\in M(i)} q_{j\to i}(x_i) \tag{4.4}$$

The optimal value for the maximisation problem can be defined as $argmax_{x_i} Z_i(x_i)$but only in acyclic factor graphs while in cyclic factor graphs only an approximate solution can be found [WNR08].

The max-sum algorithm is able to provide an optimal solution if it is applied on constrained trees, however, it does not provide any guarantee if it is applied in networks that contain loops [RFS+11]. The difference of max-sum algorithm compared to complete DCOP algorithms lies in the fact that the number of the messages that are exchanged during the process are significantly less and also the computation required to find the solution is less costly. Even though complete DCOP algorithms guarantee an optimal solution in comparison to incomplete DCOP algorithms, their use in real application examples most of the time is limited due to above problems. In [RFS+11], the authors state that the vast majority of the DCOP complete algorithms such as OptAPO, ADOPT, BnB-ADOPT and DPOP suffer in terms of the number of the messages that are exchanged between the agents, which most of the times are exponential in the height of pseudo-tree. Furthermore, some latest versions of the above complete algorithms tried to reduce the number of messages, however, the overall time to reach the solution is still exponential.

The basic concept of incomplete algorithms is to provide a solution that may not be optimal, but with less computation and communication among the agents. Thus, the incomplete algorithms such as max-sum algorithm, most of the times are picked for large networks with complex computations and constraints. Even if the solution will be close to the optimal the algorithm is guaranteed that it will converge. *Figure 4.14*, illustrates the pseudo-code of max-sum algorithm and how agents send variable and function messages. Each variable node starts the algorithm by sending to its neighbouring function nodes a zero utility function of itself. Each function node sends to each of its neighbouring variable nodes a utility function that is equal to the maximum or minimum global utility achievable, based on the utility function it has received from each neighbouring variable nodes.

$\boldsymbol{Q} \leftarrow \emptyset$ {Initialize the set of received variable to function message}
$\boldsymbol{R} \leftarrow \emptyset$ {Initialize the set of received function to variable message}
while termination condition is not met **do**
  for $j \in N_i$ **do**
    $r_{i\to j}(x_j) = computeMessageToVariable(x_j, U_i, \boldsymbol{Q})$
    SendMsg ($r_{i\to j}(x_j)$, a_j)
  end for
  for $j \in M_i$ **do**
    $q_{i\to j}(x_i) = computeMessageToFunction(x_i, U_j, \boldsymbol{R})$
    SendMsg ($q_{i\to j}(x_i)$, a_j)
  end for
  $\boldsymbol{Q} \leftarrow get\ MessagesFrom\ Functions()$
  $\boldsymbol{R} \leftarrow get\ Messages\ From\ Variables()$
  $x_i^* = updateCurrent\ Value(x_i, R)$
  end while

Figure 4.14: Max-sum algorithm pseudo-code [RFS+11]

Figure 4.15 and *Figure 4.16* illustrate the computation messages to a variable and also to function. Each agent, with respect to its function and variable messages, computes its own variable assignment by finding the value which maximises or minimises its function. The value of the agent variable is not responsible for the message computation during the calculation of the messages.

Input: x_j : the receiver's variable,
U_i : the sender's function,
$\boldsymbol{Q}$: the current set of variable to function messages received by the sender.
Output: $r_{i\to j}(x_j)$ the function to variable message from function U_i to variable x_j.
$r_{i\to j}(x_j) = -\infty$
for $d_i \in D_i$ {all joint assignments of x_i} **do**
  $\sigma = U_i(d_i)$
  for $d_k \in d_j$, $(k \neq j)$ **do**
    $\sigma = \sigma + q_{k\to i}(d_k) \quad \{q_{k\to i} \in \boldsymbol{Q}\}$
  end for
  $r_{i\to j}(d_j) = max\ r_{i\to j}(d_j), \sigma\{d_j \in d_i\}$
end for
return $r_{i\to j}(x_j)$

Figure 4.15: Computation message to variable [RFS+11]

Input: x_i : the sender's variable,
U_j : the receiver's function,
$\boldsymbol{R}$: the current set of function to variable messages received by the sender.
Output: $q_{i\to j}(x_i)$ the variable to function message from variable x_i to function U_j .
$q_{i\to j}(x_i) = 0$
for $r_{k\to i} \in \boldsymbol{R}\ k \neq j$ **do**
$q_{i\to j}(x_i) = q_{i\to j}(x_i) + r_{k\to i}(x_i)$
end for
$$a_{ij} = -\frac{\sum d_i \in D_i\ q_{i\to j}(d_i)}{|D_i|}$$
$q_{i\to j}(x_i) = q_{i\to j}(x_i) + a_{ij}$
return $q_{i\to j}(x_i)$

Figure 4.16: Computation message to function [RFS+11]

For a better explanation of the algorithm, an example is provided in order to understand the factor graph and the messages flow from variable to functions and from functions to variable. *Figure 4.17*, shows a constrained network in factor graph presentation where the variables are represented by circles (x_1, x_2 and x_3) and the utility function are represented by squares (U_1, U_2 and U_3).

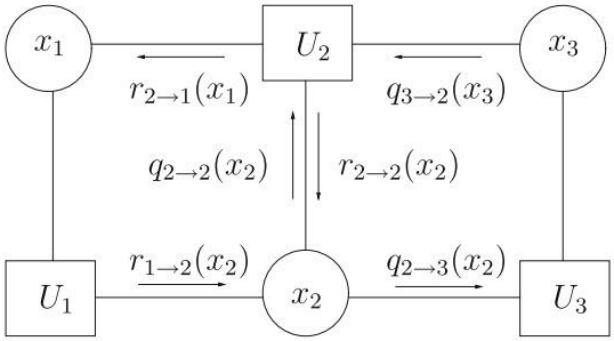


Figure 4.17: Factor graph of the constrained network [RFS+11]

According to *Figure 4.17*, each variable is receiving one or two utility function messages and its utility function is receiving one or two variable messages. An analytical calculation procedure will be explained for variable to utility function message $q_{2\to3}(x_2)$ and for

utility function to variable message $r_{2\to1}(x_1)$. In order to calculate the variable to function message Equation 1 is used but without the scalar which is considered to be equal to 0.

$$q_{2\to3}(x_2) = \sum_{k \in M(2)\setminus 3} r_{k\to2}(x_2) \quad => \quad q_{2\to3}(x_2) = r_{1\to2}(x_2) + r_{2\to2}(x_2) \tag{4.5}$$

As for the function to variable message, it can be calculated according to Equation 3 such that:

$$r_{2\to1}(x_1) = max_{x_{2\setminus1}}\left(U_2(x_2) + \sum_{k \in N(2)\setminus 1} q_{k\to2}(x_k)\right) \quad =>$$

$$r_{2\to1}(x_1) = max_{x_2,x_3}\left[U_2(x_1,x_2,x_3) + q_{2\to2}(x_2) + q_{3\to2}(x_3)\right] \tag{4.6}$$

Table 4.1, illustrates the calculation $r_{2\to1}(x_1)$ message where the variables x_1, x_2 and x_3 have a domain of [0, 1]. The same procedure applies to other messages until they reach a solution that satisfies all the variables.

x_1	x_2	x_3	$U(x_1,x_2,x_3)$	$q_{2\to2}(x_2)$	$q_{3\to2}(x_3)$	SUM	$r_{2\to1}(x_1)$
0	0	0	-3	-1	-1	-5	-1
0	1	0	-2	0	-1	-3	-1
1	0	0	-2	-1	-1	-4	0
1	1	0	-1	0	-1	-2	0
0	0	1	-2	-1	0	-3	-1
0	1	1	-1	0	0	-1	-1
1	0	1	-1	-1	0	-2	0
1	1	1	0	0	0	0	0

Table 4.1: Computation of $r_{2\to1}(x_1)$ message

4.4.11 Divide and Coordinate Subgradient Algorithm (DaCSA)

The DaCSA algorithm was introduced by Vinyals et al. in 2010 and belongs to the incomplete algorithms category [VGA+10]. There are two phases in the algorithm: "a) divide stage, where the agents divide the DCOP problem into local subproblems and solve them and b) coordination stage, where agents communicate between each other in order to exchange information and reach an agreement that provides the solution to the problem" [VGA+10]. *Figure 4.18*, illustrates the pseudo-code of the algorithm.

Each agent a_i runs:
$bound \leftarrow \infty$; $\{\lambda^0\} \leftarrow 0$; $solution \leftarrow \emptyset$;
$bestValue \leftarrow -\infty$; $C_i \leftarrow \emptyset$;
$\Phi_i \leftarrow$createSubproblem($< X^i, D^i, R^i >$);
repeat
  /* Divide stage */
  $\bar{\Phi}_i \leftarrow$modifySubproblem($\Phi_i, \{\lambda_i\}$);
  $(X^{*,i}, f_i^*) \leftarrow$solveSubproblem($\Phi i$);
  /* Coordinate stage */
  for $x_u \in N(x_i)$ **do**
    $\Psi_i^u \leftarrow$makeCoordInfo($x_i^{*,i}, x_u^{*,i}, f_i^*, C_i$);
    $\Psi_i^u \leftarrow$exchangeCoordInfo(Ψ_i^u);
  end for
  $\gamma_t \leftarrow$updateStepSize();
  $\{\lambda\} \leftarrow$updateCoordParams($\{\lambda\}, \gamma_t, X^{*,i}$);
  if betterBoundAvailable($\{\Psi\}, bound$) **then**
    Update *bound*.
  end if
  if betterSolAvailable($\{\Psi\}, bestValue$) **then**
    Update *solution* and *bestValue*.
  end if
  $C_i \leftarrow$selectCandidateSolutions(x_i, C_i);
until any termination condition satisfied
return $< solution, bestValue, bound >$

Figure 4.18: Pseudo-code of DaCSA algorithm [VGA+10]

At the starting point, each agent creates its local problem using local information. Moreover, in the divide phase, each agent modifies its local problem into subproblems in

order to reach an optimal local solution. For the optimisation of the problem two techniques are used: a) Lagrangian dual decompositions and b) subgradient dual methods. The second phase, coordination, contains the exchange of assignments between neighbour agents where the coordination parameters are updated in order to reduce the disagreements. Furthermore, each agent sends information of common variables to neighbour only agents.

In addition, each agent must calculate the bound, assessed as the sum of a solution of all subproblems and also the value of a candidate solution. To be able to do this, the algorithm is using external protocols to perform this task.

4.4.12 K-optima (KOPT)

KOPT is the first DCOP incomplete algorithm for arbitrary k-optima and was presented by Katagishi et al. in 2007 [KP07]. In k-optima algorithm, small groups of agents optimise based on their local constraints, resulting in a k-optimal assignment, in which no subset of k or fewer agents can improve the overall solution. A k-optimal solution is a solution that cannot be improved by any aberration by k or fewer agents. The algorithm contains 3 phases: a) phase 1, where each agent collects information from neighbour agents b) phase 2, where each agent determines the best value assignment regarding the collecting information of the agents and pass the results to agents that share the value assignment and c) phase 3, where each agent picks the value assignment of the highest utility and passes it to neighbour agents.

At the beginning of the algorithm, each agent forms a group that may contain one or more agents. It is clearly understandable that each agent may belong to more than one group and the groups may overlap. For every group, a mediator exists which collects the information of all the members of the group, obtains the best value assignment and passes the information to group members. At any time during the process, mediators are checking in order to find which one has the highest value assignment. At that time, the mediator passes the information to other groups in order to change their values.

4.4.13 Maximum Gain Message (MGM)

The MGM algorithm was introduced by Maheswaran et al. in 2004 [MPT04]. The algorithm is almost identical to DBA but focuses on gain message passing. In MGM algorithm, each agent sends a gain message to neighbour agents and informs them about the maximum change in its local utility. Furthermore, each agent who receives the gain messages from other agents performs a check with its own gain message and if its own message is larger then pass the message to the other agents. MGM difference between DBA lies on the fact that it is not used the change of constraint cost as DBA in order to break out the local minima. The authors also presented an updated version of the algorithm which is called MGM-2. In MGM-2, some of the agents are responsible for making *offers* to other agents. The agents that are the *offerers* are picked by randomization. In addition, the agent who is the *offerer* is able to pass messages to other agents and not receive any *offers* from other agents. The agents who receive the *offers* are called *receivers*. Every *offerer*, choose one of its neighbour agents to send an *offer* and this choice can be achieved by randomization. The *offer* message contains all the possible moves between the *offerer* and the *receiver* that are able to provide a local utility gain. Each *receiver* calculates the global utility gain by adding its own utility with *offerer's* local utility gain. If the larger global utility gain is positive among all the value pairs, the *receiver* sends an accept message to *offerer* with all the corresponding value pairs. If not, it sends a reject message to the *offerer*.

4.5 Appropriate selection of algorithms

After an analysis of complete and incomplete algorithms in the previous sections, it is crucial to identify which algorithms are appropriate for power system networks. The difference between the algorithms lies in their search process. The assignment space of a DCOP is exponential based on its number of variables, so search methods do not scale beyond very small numbers of agents. This can be solved either by using exhaustive search with clever heuristics or by performing a non-exhaustive local search. The basic concerns that DCOP algorithms must solve are: (a) how to determine the order of minimisations and maximisations to perform; (b) how to allocate the agents; and (c) how to obtain the

optimal solution from the DCOP framework. DPOP is the only complete algorithm that is able to overcome these issues by using a DFS tree (pseudotree). The algorithm guarantees that it uses only a linear number of messages. The complexity of the algorithm is exponential in the tree width of the pseudotree, not on the number of agents. This is a benefit for power systems with many agents and constraints, where the number of messages will be linear and the performance of the algorithm will not be affected by the number of the variables.

On the contrary, incomplete algorithms such as max-sum perform better in larger power system networks even if they are not guaranteed that they will provide the optimal solution. The difference lies in the fact that max-sum algorithms use a factor graph to obtain a solution which is assumed to be known before the algorithm's execution. Also, the biggest advantage of max-sum over the complete algorithms is that it allows the computation of the value of a discrete function. This has an impact on the performance of the max-sum algorithm and performs better and faster with a large number of variables and constraints. Max-sum uses a factor graph instead of DFS tree and that means that the execution of the algorithm could never terminate. Hence, max-sum terminates only when the messages are not changing. Furthermore, this also enables max-sum's nodes to operate in a fully parallel schedule (asynchronously), where each node of the factor graph computes and sends messages at the same time. For power system networks this is crucial because max-sum is able to perform parallel calculations of different control functions asynchronously while complete algorithms like DPOP need to wait to receive the messages in order to perform any calculation. Last but not least, the max-sum algorithm is one of the search algorithms that agents do not consider their own benefit only, but the benefit of the whole system. Analytically, agents don't try to maximise their own utility for their own benefits but they care about the optimisation of the global objective. On the other hand, complete algorithms are always trying to maximise their own utility. This is very crucial for power system networks, where each agent who is responsible for a generator want to maximise its utility.

Finally, max-sum has an important disadvantage as its cost is exponential on the number of neighbours of each agent. An update of the algorithm is needed in order to perform solutions without the use of systems memory requirements.

4.6 Conclusions

Between the several approaches to multi-agent coordination, the DCOP framework was picked for two main reasons. Firstly, because despite being one of the simplest approaches to multi-agent coordination, the DCOP model can represent a vast number of coordination situations. Secondly, because the framework supports the development of generic algorithms to solve those problems. There have been both a large number of proposed applications with varying requirements (meeting scheduling, wireless sensor networks,) and an equally vast array of solution algorithms of different characteristics (DPOP, O-DPOP, Max-Sum). Therefore, the DCOP framework was reviewed and analysed of how it can be used to model multi-agent coordination problems. As a result, three types of algorithms were identified: (a) optimal algorithms that guarantee the maximum solution quality but have exponential costs; (b) global approximate algorithms that have lower costs but they do not provide any guarantee for the solution; and (c) local approximate algorithms, that are able to provide solutions but they do not provide any guarantees for the solution.

A range of algorithms was presented in order to understand how they can be applied in various real world applications. However, there is a need to categorise which algorithm is adequate for each application. As mentioned before in the chapter, DCOP algorithms can be categorised as complete algorithms and incomplete algorithms. Additionally, the complete algorithms can be further categorised into partially centralised or fully decentralised algorithms. Depending on the problem, the appropriate algorithm needs to be picked in order to reach an optimal solution. Typically, complete algorithms (SynchBB, DPOP, O-DPOP) can be picked to solve small problems because they reach the solution faster than incomplete algorithms. Furthermore, incomplete algorithms like max-sum, are able to find the optimal solution for problems where there are hundreds of variables and constraints. However, depending on the problem both types of algorithms need to be tested in order to understand their effectiveness. For power systems, and especially for the case studies that will be presented in the following chapter, 3 complete algorithms (SynchBB, DPOP, O-DPOP) and one incomplete algorithm (max-sum) were picked for the simulations. The reason for picking these algorithms lies in the problem that needs to be solved. For instance, in power flow management case studies, there is a need to minimise

the cost of the generators. This is a minimization problem, and the costs are non-negative. For this reason, algorithms like SynchBB, DPOP and O-DPOP are able to find the optimal solution faster and terminate earlier. Furthermore, if the problem has a low tree width (depending on the network) then DPOP may not perform well because its complexity is exponential in the tree width. DPOP is performing differently in the case studies, due to its complexity because the case studies are different on the number of variables and constraints that are used. Max-sum was picked because according to the literature, incomplete algorithms are able to perform better than complete algorithms in larger networks.

Depending on the type of the constraints, different algorithms need to be picked. For instance, if the problem contains hard constraints, then search algorithms like SynchBB perform better than DPOP and O-DPOP.

The above 4 algorithms are considered to be the state of the art for solving DCOP problems. The following chapter evaluates the performance of these algorithms for power flow management.

Chapter 5

Case Studies

This Chapter will analyse the integration of DCOP algorithms to power systems with a simple example of a 3 bus network. Then, a more complicated network will be presented (11 buses) considering sensitivity factors. Additionally, a case study of an 11kV network with 40 buses will be analysed in order to evaluate the efficiency of DCOP algorithms and finally, an AC power flow management case study will be used to detail the limitations of using the Frodo software tool.

5.1 Application of DCOP to DC Power Flow Management

The management of distributed generation involves the satisfaction of network constraints without violating the constraints of the electricity network and provide the adequate demand at minimum cost. One of the network constraints is the satisfaction of thermal ratings of the network lines. A common practice in the UK is to manage DG connection curtailment through the use of "last in first off" (LIFO) access rights for non-firm connections [DTI04]. Under the LIFO scheme when a thermal violation is detected the most recently connected DG is curtailed first. Dolan et al. identified that the cost of DG curtailment is less than the network reinforcement [DDK+14].

To model the power system network we assume that a network has *n* nodes to which a set of n loads, $\left\{P_{d_{1,\dots,}} P_{d_n}\right\}$ and *m* generators $\left\{P_{g_1}, P_{g_2}, \dots, P_{g_m}\right\}$ are connected. Each load P_{d_i} has a power consumption $P_{di} \in R^-$, while each generator P_{g_j} has a power output $P_{gj} \in R^+$, as well as a cost of generation defined as $Cost_{P_{g_m}} * P_{gj}$ where $Cost_{P_{g_m}}$ is the cost of generating one unit of power. The nodes of the network described above determine a set of variables $V = \{V_1, \dots, V_n\}$ of the DCOP formulation. Furthermore, these *n* network nodes are connected with a number of branches (i.e. distribution cables). For each line *ij* (which connects nodes *i* and *j*) we can define the power which flows along the distribution cable $P_{f_{ij}} \in R$, as well as the maximum power capacity of the cable $P_{f_{ij}}^{max} \in R^+$.

The above definitions are then used to clearly define a set of constraints used by the DCOP.

For the PFM, the constraints that need to be considered are:

- $\left|P_{f_{ij}}\right| \leq P_{f_{ij}}^{max}$

which means that the power that flows across the distribution cable cannot exceed the thermal capacity of the network

- $P_{f_{ij}} = -P_{f_{ji}}$

which means that the net flows from u_i node to u_j node is equal to the opposite net flow (under the assumption that the network is lossless)

At each node, the summation of incoming power flows, loads and generators outputs must be equal to zero.

$$\sum_{ij \in I} P_{f_{ij}} + \sum_{j \in L} P_{d_j} + \sum_{gj \in M_j} P_{g_j} = 0 \tag{5.1}$$

where I is a set of buses, M_j is a set of generating units at bus j and L is the real power demand at bus j.

$$P_{f_{ij}} = \sum_{i=1}^{V} a_{i,ij}(P_{g_j} - P_{d_j}) \tag{5.2}$$

which means that the flow in each of the lines is defined using sensitivity factors and injection at each bus, i.e. the generator output and load at each bus. Sensitivity factors are used to provide a calculation of possible overloads in power flows. These factors show the approximate change in line flows for changes in generation on the network configuration. [DGS11]

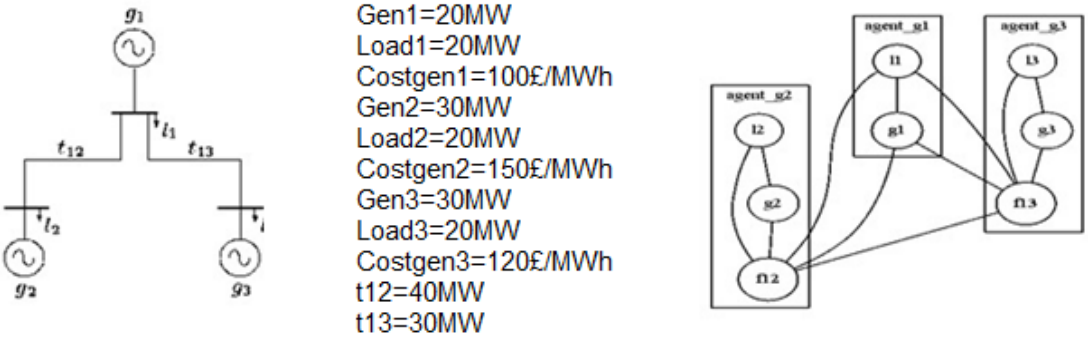
The main goal of the PFM scheme investigated here is to maximise DG outputs without violating operational and network constraints. Thus, its objective is to find the appropriate generation outputs that minimise the total cost without violating the constraints. The utility function of the agent A_i can be defined as:

$$U_i = \begin{cases} \sum_{g \in P_{g_m}(u_i)} Cost_{P_{g_m}} * P_{gj}, feasible\ flows \\ \infty, \qquad\qquad\qquad infeasible\ flows \end{cases} \tag{5.3}$$

where ∞ means that constraints are violated and infeasible flows occurred in the network. The optimisation problem is to find the assignment V^* for all of the variables that minimize the total cost i.e.:

$$argmin \sum_{i=0}^{n} U_i \tag{5.4}$$

Figure 5.1 (a) shows the electricity network model that includes the values of 3 generators, 3 loads, thermal line capacities and generator costs. *Figure 5.1 (b)* depicts its decomposition to a constraint graph. Each variable in DCOP framework becomes a variable node and each constraint in DCOP framework becomes a function node, whose neighbours are the variable nodes corresponding to the variables the constraint is expressed over.



(a) (b)

Figure 5.1: (a) Electricity network model (b) Decomposed constraint graph

Regarding the agents, it has been chosen that each agent should own the power output decision variable for its generator because it will be the responsibility of that agent to make this decision. Every agent owns a given variable that should have control of it and should be responsible for making that decision.

The agent that owns a constraint can be specified in the XCSP file using an "agent" attribute, in the same way that the owner of a variable is specified. By convention, if the owner of a constraint is not specified, then the algorithm will assign the constraint to the agent that owns the first variable in the constraint's scope.

The agent that is assigned a particular constraint (a particular function node) must be able to execute the algorithm for that function node. To be able to do so, the agent needs to know the domain of the constraint (and it must be able to communicate with the agents owning its neighbouring variable nodes). The whole philosophy behind this DCOP research is that knowledge should only be local: no agent knows everything about the problem (otherwise, that agent could just solve the whole problem and communicate the solution to the other agents). Instead, the knowledge of any given constraint is only available to the agents that are "local" to and directly affected by the physical limitation that the constraint expresses. It is in this way that a decentralised and fully flexible (in the future), the control system can be achieved.

The type of constraints implemented are power flow constraints and cost constraints. Flow conservation constraints (one per node in the power grid) impose that the values of the flow variables should be consistent with each other and with the values of the decision variables that model the generator power outputs. These are hard Weighted Sum constraints that make sure that the flows are feasible. The lower and upper bounds of the domains of the flow variables will also make sure that the flows are within limits. Cost constraints (one per generator) express the power generation costs. These are soft constraints. As for the loads, the variable nodes for the loads will be connected to the function nodes for the flow conservation constraints, which in turn will be connected to the variable nodes for the flows in the lines and for the generator power outputs. The ownership of a variable cannot be shared between several agents. This means that which generator should be assigned the exclusive ownership of each flow variable needs to be decided.

5.2 Frodo: An Open-Source Framework for DCOP

FRODO is a Java open-source framework for DCOP, initially developed at the Artificial Intelligence Laboratory (LIA) of École Polytechnique Fédérale de Lausanne (EPFL) by Adrian Petcu [Fro06].

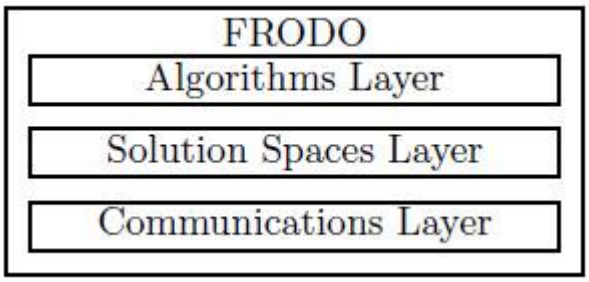


Figure 5.2: Frodo Architecture [Pet06]

Figure 5.2 presents the architecture of Frodo software which contains 3 different layers. The first layer is called communication layer and is responsible for passing messages through the agents. The second layer is called solution spaces layer and contains all the classes that can be used in order to model the optimisation problems. This layer can be presented as a combination of constraints that describes a subspace of solution to a problem. The third layer is called algorithms layer and contains the vast majority of DCOP algorithms that are able to solve the optimisation problems [Pet06].

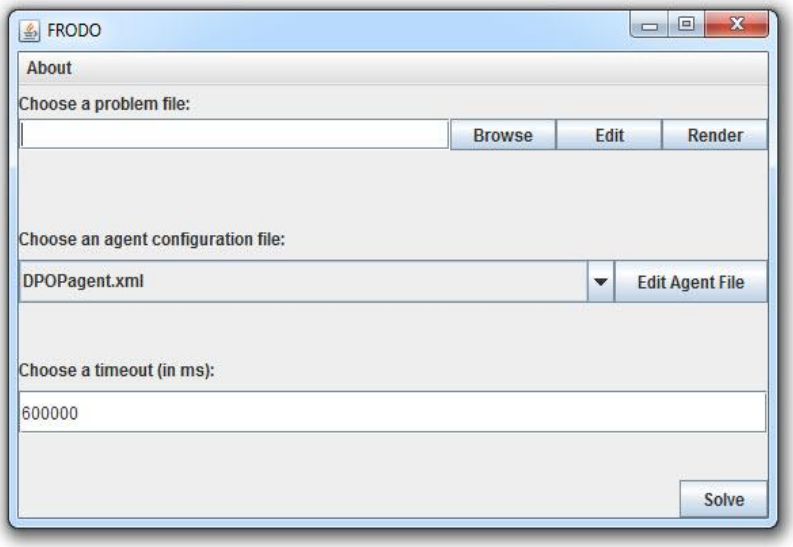


Figure 5.3: Frodo Software

Figure 5.3 illustrates the latest version of Frodo software. Firstly, the problem file must be chosen in order to apply any of the DCOP algorithms. The file format that is used in FRODO is based on the XCSP format [Oco08]. However, this format was initially developed for centralised optimisation problems and not for distributed optimisation problems. Additional extensions were imported to this format file in order to be able to handle the distributed problems and also to describe which agent own which variable and if the problem is a maximisation or minimization problem.

Figure 5.4 presents an XCSP file for DC Power Flow Management. Based on the DCOP framework the file format is divided in categories such as the number of agents, the number of the domains and the number of variables. Furthermore, the type of constraints is defined and the objective function is set in order to optimise the problem regarding the minimization or the maximisation of the problem.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="src/ch/epfl/lia/frodo/algorithms/XCSPschemaJaCoP.xsd">
    <presentation name="Athanasiadis13" maxConstraintArity="4" maximize="false" format="XCSP 2.1_FRODO" />
    <agents nbAgents="3">
        <agent name="agent_g1" />
        <agent name="agent_g2" />
        <agent name="agent_g3" />
    </agents>
    <domains nbDomains="5">
        <domain name="dom_g1" nbValues="5">-2..2</domain>
        <domain name="dom_g2_and_g3" nbValues="7">-3..3</domain>
        <domain name="dom_f12" nbValues="9">-4..4</domain>
        <domain name="dom_f13" nbValues="7">-3..3</domain>
        <domain name="dom_l" nbValues="1">-2</domain>
    </domains>
    <variables nbVariables="8">
        <variable name="g1" domain="dom_g1" agent="agent_g1" />
        <variable name="g2" domain="dom_g2_and_g3" agent="agent_g2" />
        <variable name="g3" domain="dom_g2_and_g3" agent="agent_g3" />
        <variable name="f12" domain="dom_f12" agent="agent_g2" />
        <variable name="f13" domain="dom_f13" agent="agent_g3" />
        <variable name="l1" domain="dom_l" agent="agent_g1" />
        <variable name="l2" domain="dom_l" agent="agent_g2" />
        <variable name="l3" domain="dom_l" agent="agent_g3" />
    </variables>
    <functions nbFunctions="1">
        <function name="times" return="int">
            <parameters>int p1 int p2</parameters>
            <expression>
                <functional>mul(p1, p2)</functional>
            </expression>
        </function>
    </functions>
    <constraints nbConstraints="6">
```

Figure 5.4: XCSP file for DC Power Flow Management

5.3 Matlab script from IPSA to FRODO

Frodo software is able to evaluate the vast majority of DCOP algorithms, however, the network has to be translated in an XML or XSCP file. In order to write the XML file, network data (generators, loads, buses, and branches) need to be obtained by a power systems simulation tool. For this reason, the IPSA software tool was picked in order to receive the data from any network.

The data are exported in a Matlab file, however, a Matlab script was created in order to translate these data to XML or XSCP format. *Figure 5.5* shows a part of the Matlab script defining agents and *Figure 5.6* illustrates the corresponding XSCP file from the Matlab script that will be used to apply the DCOP algorithms. The Matlab script contains all the appropriate transformations that need to be done in order to construct the file. FRODO cannot recognise the physical structure of the network because it was not built for power systems, hence the network topology has to be defined before applying the DCOP algorithms. Generators, Loads and Cost data need to be defined at the beginning of the Frodo software as well as agents that are responsible for every bus.

```
%% ------ Agents
% Agents = Nr of Generators
Nr_Agents=Bus_r;

outputStr3 = sprintf('\t<agents nbAgents="%d">\n',Nr_Agents); %
fprintf(fid, '%s', outputStr3);

for Agent_Index=1:1:Nr_Agents
   outputStr_ag= sprintf(char(strcat('\t\t<agent name="agent_g',num2str(Agent_Index),'"/>\n')));
    fprintf(fid, '%s', outputStr_ag);
end
outputStr4 = sprintf('\t</agents>\n'); %
fprintf(fid, '%s', outputStr4);
```

Figure 5.5: Part of Matlab script regarding agents

FRODO needs to understand the topology of the network and for that reason, the buses and the branches are defined as constraints. The objective function is defined and also all the laws for power flow management i.e. at each bus the summation of loads, generators and flows are equal to zero.

```
<constraints nbConstraints="35">
    <constraint name="conserv_bus1" arity="4" scope="g1 l1 f114 f351" reference="global:weightedSum">
        <parameters>[ { 1 g1 } { 1 l1 } {1  f114} {1  f351} ] <le/> 50</parameters>
    </constraint>
    <constraint name="conserv_bus2" arity="3" scope=" l2 f23 f142" reference="global:weightedSum">
        <parameters>[ { 1 l2 } {1  f23} {1  f142} ] <le/> 50</parameters>
    </constraint>
    <constraint name="conserv_bus3" arity="3" scope=" f34 f23 f403" reference="global:weightedSum">
        <parameters>[ {1  f34} {-1  f23} {1  f403} ] <le/> 50</parameters>
    </constraint>
    <constraint name="conserv_bus4" arity="3" scope=" f45 f442 f34" reference="global:weightedSum">
        <parameters>[ {1  f45} {1  f442} {-1  f34} ] <le/> 50</parameters>
    </constraint>
    <constraint name="conserv_bus5" arity="3" scope=" l5 f56 f45" reference="global:weightedSum">
        <parameters>[ { 1 l5 } {1  f56} {-1  f45} ] <le/> 50</parameters>
    </constraint>
    <constraint name="conserv_bus6" arity="3" scope=" l6 f67 f56" reference="global:weightedSum">
        <parameters>[ { 1 l6 } {1  f67} {-1  f56} ] <le/> 50</parameters>
    </constraint>
    <constraint name="conserv_bus7" arity="3" scope=" f78 f79 f67" reference="global:weightedSum">
        <parameters>[ {1  f78} {1  f79} {-1  f67} ] <le/> 50</parameters>
    </constraint>
    <constraint name="conserv_bus8" arity="2" scope=" f78 f418" reference="global:weightedSum">
        <parameters>[ {-1  f78} {1  f418} ] <le/> 50 </parameters>
    </constraint>
    <constraint name="conserv_bus9" arity="3" scope="g9 l9 f79" reference="global:weightedSum">
        <parameters>[ { 1 g9 } { 1 l9 } {-1  f79} ] <le/> 50</parameters>
    </constraint>
    <constraint name="conserv_bus10" arity="2" scope=" f1011 f1410" reference="global:weightedSum">
        <parameters>[ {1  f1011} {1  f1410} ] <le/> 50</parameters>
    </constraint>
```

Figure 5.6: Frodo's correspondent XSCP file script

The resulting XML file can then be applied in FRODO software, and every DCOP algorithm that belongs to the database of FRODO can be tested. *Figure 5.7* shows the results from FRODO software by applying a DCOP algorithm (DPOP) in 3 bus example. The results provide details for every generator, load, and flow as well as computation time for the process and total optimal cost. The Matlab script is able to transform any radial network to XML to XSCP, hence cannot be used for meshed networks. An updated version of the script will contain the proper configuration of any type of network.

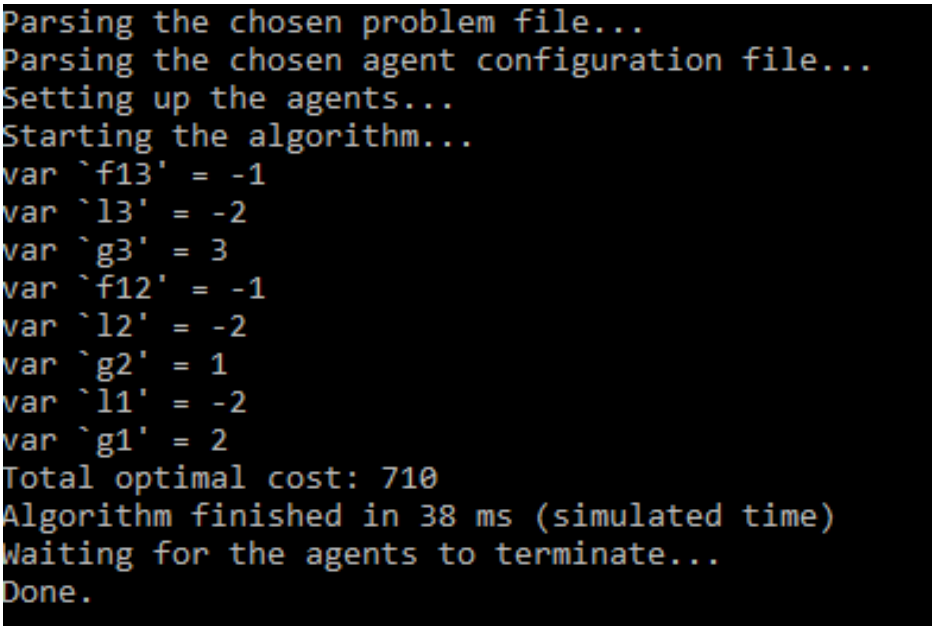


Figure 5.7: Results from FRODO software

5.4 Case Study of Max-Sum Algorithm under PFM

When the algorithm starts, messages flow from variables to functions. Each generator sends its own domain to the functions according to Figure 5.1 (a). This is the first step of the algorithm where each generator sends a zero domain to neighbourhood functions. The messages will be $Q_{1\to1} = \{0, \dots, 0\}$, $Q_{2\to2} = \{0, \dots, 0\}$ and $Q_{3\to3} = \{0, \dots, 0\}$ because according to equation 4.1 of chapter 4.4.10 each message from variable to function can be calculated regarding the sum of all the previous function messages. For the first step there is no function to variable message, thus the domains will be zero. The second step involves the messages from functions to variables and more precisely the messages $R_{2\to12}$ and $R_{3\to13}$. The first row of the messages presents the flows and second row of the messages shows the equivalent cost. Calculations of these messages are based on equation 4.2 of chapter 4.4.10, so that:

$$R_{2\to12} = \begin{Bmatrix} -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ \infty & \infty & 0 & 150 & 300p & 450p & \infty & \infty & \infty \end{Bmatrix}$$

$$R_{3\to13} = \begin{Bmatrix} -3 & -2 & -1 & 0 & 1 & 2 & 3 \\ \infty & 0 & 120 & 240 & 360 & \infty & \infty \end{Bmatrix}$$

To calculate the above domains the thermal ratings, the loads and the generator's output is considered in order to find the equivalent cost. For instance, in the $R_{2\to12}$ message it is clear that the feasible values are between $[-2,\dots,1]$ because the load that has to be satisfied is $-20MW$. In the same domain flow 0 means that Gen_2 generates $20MW$ so the cost will be the generator's output $20MW$ multiplied by the equivalent cost which is $150£/\text{MWh}$ for Gen_2 so the total cost will be $3000£$. The third step involves the messages from $Q_{12\to1}$ and $Q_{13\to1}$. According to equation 4.1 of chapter 4.4.10 the above messages will have the same domains as the messages $R_{2\to12}$ and $R_{3\to13}$ respectively. The domains will be:

$$Q_{12\to1} = \begin{Bmatrix} -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ \infty & \infty & 0 & 150 & 300 & 450 & \infty & \infty & \infty \end{Bmatrix}$$

$$Q_{13\to1} = \begin{Bmatrix} -3 & -2 & -1 & 0 & 1 & 2 & 3 \\ \infty & 0 & 120 & 240 & 360 & \infty & \infty \end{Bmatrix}$$

The fourth step involves the $R_{1\to1}$ message which will contain the sum of all the previous variables to functions messages. The domain of the message will be:

$$R_{1\to1} = \begin{Bmatrix} OMW & 10MW & 20MW \\ 270 & 540 & 810 \end{Bmatrix}$$

To understand the calculation of the $R_{1\to1}$ message the summation of $Q_{12\to1}$ and $Q_{13\to1}$ is needed. For instance, if there is $0MW$ generation output from Gen_1 to satisfy the constraint, which is in this case the load $-20MW$, this amount of generation needs to be covered by either one or two messages. Maximum values from either of the generators cannot be taken because infinity domains have occurred so the proper solution is to take $10MW$ from $Q_{12\to1}$ and $10MW$ from $Q_{13\to1}$. The total cost is equal to 270 and the same situation occurs for the other 2 values of the generator. At this point the algorithm needs to choose the appropriate value to send and it chose the value with the lowest cost. Finally, Gen_3 now knows that it has to generate $20MW$ and sends back the message. The messages are now flowing back in a different direction in order to finally send the last messages to the other two generators. For the $R_{3\to3}$ message the minimum cost is 3600 when Gen_3

generates $30MW$ and for $R_{2\to2}$ the lowest cost of 3600 is when Gen_2 produces $10MW$. So the optimal solution will be $Gen_1 = 20MW$ $Gen_2 = 10MW$ and $Gen_3 = 30MW$. The total cost is $(20 * 100) + (10 * 150) + (30 * 120) = 2000 + 1500 + 3600 = 7100$. According to max-sum algorithm theory, when message values converge each variable can choose its best state. This situation occurs in acyclic networks, like the case study, where the next iteration of messages will be the same in terms of the variable values. Acyclic networks are networks in AI that they do not have loops. In power systems these networks are called radial.

5.5 Case Study of DPOP, O-DPOP and SYNCHBB under PFM

In this section an analysis of DPOP, O-DPOP and also SynchBB algorithm will be presented under the same PFM case study, however, a comparison between them and the max-sum algorithm will be presented and evaluated.

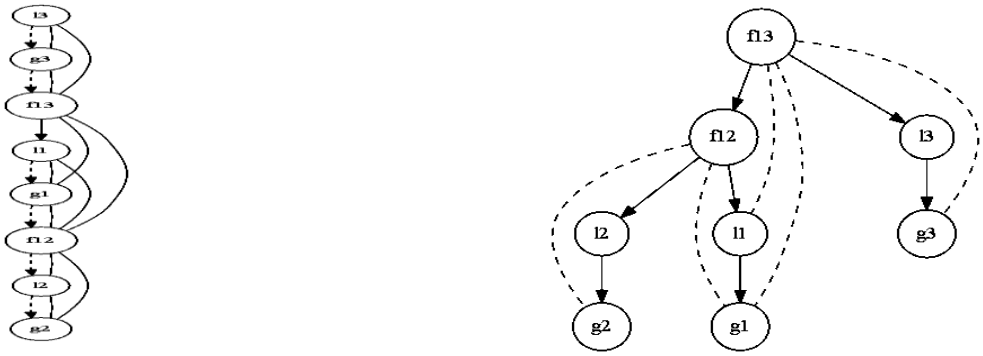
DPOP consists of three consecutive phases, which exactly correspond to the three types of messages:

1) DFS tree generation phase

2) UTIL propagation phase: UTIL messages;

3) VALUE propagation phase: VALUE messages.

The first phase is just a preliminary phase whose goal is to generate a DFS-tree ordering of the variables. This preliminary phase typically exchanges more messages than the two others (SynchBB and max-sum), but these messages are very small (163 bytes per message on average for this run) and very inexpensive to compute. The messages that are hard to compute (and can be very large) are the UTIL messages (694 bytes per message). *Figure 5.8* illustrates the constraint graphs of SynchBB, DPOP and O-DPOP.



(a) (b)

Figure 5.8: (a) Constraint graph of SynchBB (b) Constraint graph of DPOP and O-DPOP

This is where the bottleneck of the DPOP algorithm lies. Regarding the case study, DPOP only sends TWO such UTIL messages. This is why DPOP is so fast. In comparison, the VALUE messages are also very small (232 bytes per message) and inexpensive. *Figure 5.9* illustrates the computation time, the number of messages, the amount of information sent and the size of the largest message sent.

```
Algorithm finished in 85ms <simulated time>
Number of messages sent (by type):
ParallelDFSwrapper:    26
UTIL:    2
VALUE:                 2
- Total:  30
Amount of information sent <by type, in bytes>:
ParallelDFSwrapper:    4,227
UTIL:    1,388
VALUE:                 465
- Total:  6,080
Size of the largest message sent <by type, in bytes>:
ParallelDFSwrapper:    446
UTIL:    952
VALUE:                 242
- Overall maximum:     952
```

Figure 5.9: DPOP analysis

Regarding the O-DPOP the difference lies on UTIL propagation phase. According to *Figure 5.10*, UTIL messages are 32 compared to 2 from DPOP, however, the size of the messages is linear hence the total computation time to reach the solution is slightly faster than DPOP by 10ms.

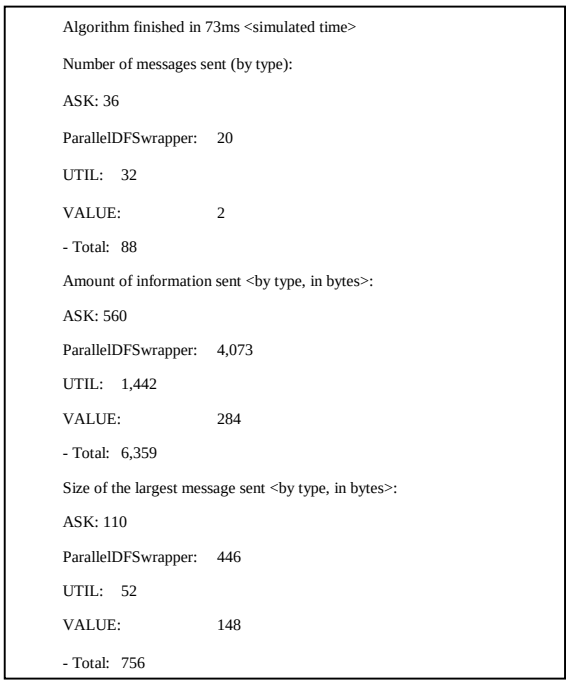


Figure 5.10: O-DPOP analysis

According to *Figure 5.11*, SynchBB exchanges much more various types of messages than DPOP, but it also consists of only three consecutive phases:

1) Root election: the agents exchange ELECT messages to elect one variable as the root of the linear variable ordering;

2) Variable ordering generation: the agents construct a linear ordering of the variables that start with the elected variable. This involves exchanging messages of type NextVarChosen, NextVarProposal, NextVarRequest and VarOrderNoSpace.

3) Synchronous branch-and-bound: the core of SynchBB's algorithm (68 messages of types Backtrack, Path and Solution).

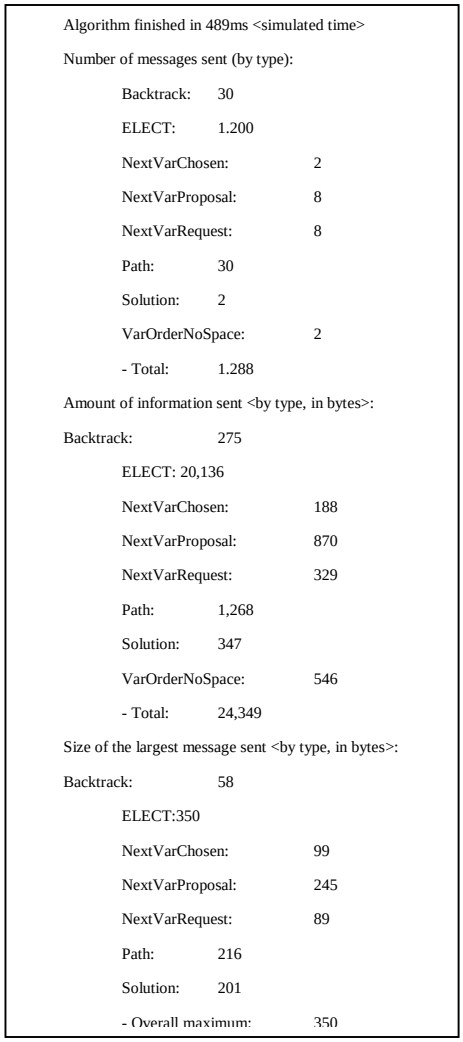


Figure 5.11: SynchBB analysis

The two first phases are actually preliminary phases whose sole purpose is to generate a linear variable ordering (like DPOP's first phase generates a DFS-tree variable ordering). These two phases are normally quite cheap compared to the 3rd phase, but in this case, because the problem instance is small, their part in the total computation time is actually quite important. In (SynchBB and max-sum), but these messages are very small (163 bytes per message on average for this run) and very particular, it is observed that of the 1288 message exchanges, 1200 are exchanged by the first election phase alone. The number of messages exchanged during this first, election phase is directly defined by a parameter of the algorithm, and it is called the "nbrSteps" parameter of the VariableElection module. The peculiarity of this VariableElection module is that it does not have a self-termination

property; one must define beforehand how many steps it must run. For it to work correctly, this nbrSteps parameter must be greater than the diameter of the constraint graph.

5.6 Comparison between the DCOP algorithms

The max-sum algorithm is a very simple algorithm that consists of a single phase, during which two types of messages are exchanged (*Figure 5.12*). If max-sum is compared to DPOP in terms of the amount of messages, it is clearly understandable that DPOP is faster than max-sum. This is because max-sum is a highly asynchronous algorithm. This means that all agents run in parallel, and the messages are exchanged in a relatively chaotic, non-deterministic fashion. This tends to result in inefficiencies, the agents sometimes having to compute multiple times almost the same computations again and again. In contrast, DPOP is a rather synchronous algorithm: some of the agents run in parallel, but each agent waits until it has received all UTIL messages from all its children in the DFS tree before it sends its UTIL message to its parent. This can sometimes result in inefficiencies because agents tend to be idle waiting for messages, but on the other hand, DPOP is very efficient in not performing the same computation twice. SynchBB is closer to DPOP than to max-sum in this respect: it is a fully synchronous algorithm (as its name suggests -only one single agent is performing computations at any single time), and it has the potential to be even more efficient than DPOP, because, contrary to DPOP and to max- sum, it is able to prune out infeasible solutions (i.e. it skips infeasible solutions).

When comparing the performance of max-sum to DPOP, O-DPOP and SynchBB, it was found that the complete algorithms DPOP, O-DPOP and SynchBB are between 5 and 10 times faster at solving the problem than max-sum for this particular case study. This is an interesting observation. It might means that FRODO's implementation of max-sum is not very efficient. To summarise, max-sum performs relatively poorly on this problem instance. In reality, where the max-sum algorithm becomes very useful is on much larger problem instances, that are too large to be solved by complete algorithms (because, for instance, DPOP would run out of memory, while SynchBB would time out). The case study example considered 3 bus power network with few variables. In such networks, DPOP is the most efficient algorithm if compared to the other algorithms. Max-Sum is

slow in terms of computation time due to its asynchronous nature, and the quality of the solution it produces is not guaranteed (it might not even find any feasible solution) due to its incompleteness. In contrast, on small enough problem instances, complete algorithms like DPOP and SynchBB will quickly output a solution that is guaranteed to be optimal. However, in power system networks of more than 3 variables, the efficiency of the max-sum algorithm can be depicted and the following sections will provide networks with more variables in order to understand the applicability of incomplete algorithms in larger networks.

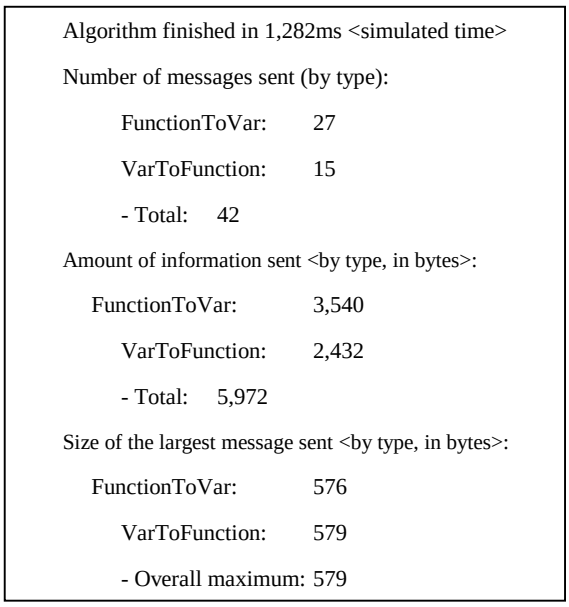


Figure 5.12: Max-sum analysis

Figure 5.13 illustrates graphically the comparison between the algorithms. The diagram contains the Optimal Power Flow (OPF) technique in order to compare the traditional technique for power flow management in power systems with the distributed optimisation algorithms. OPF is significantly fast and reaches the optimal solution in 26ms, however, DPOP and O-DPOP algorithms are considered to be almost fast algorithms with 85ms and 73ms respectively. The optimal solution for OPF is: $Gen_1 = 20MW$, $Gen_2 = 10MW$ and $Gen_3 = 30MW$. The total cost is $(20 * 100) + (10 * 150) + (30 * 120) =$

$2000 + 1500 + 3600 = 7100$. Both algorithms (OPF and DCOP algorithms) provide the same results, however DCOP algorithms are slightly slow compared to OPF.

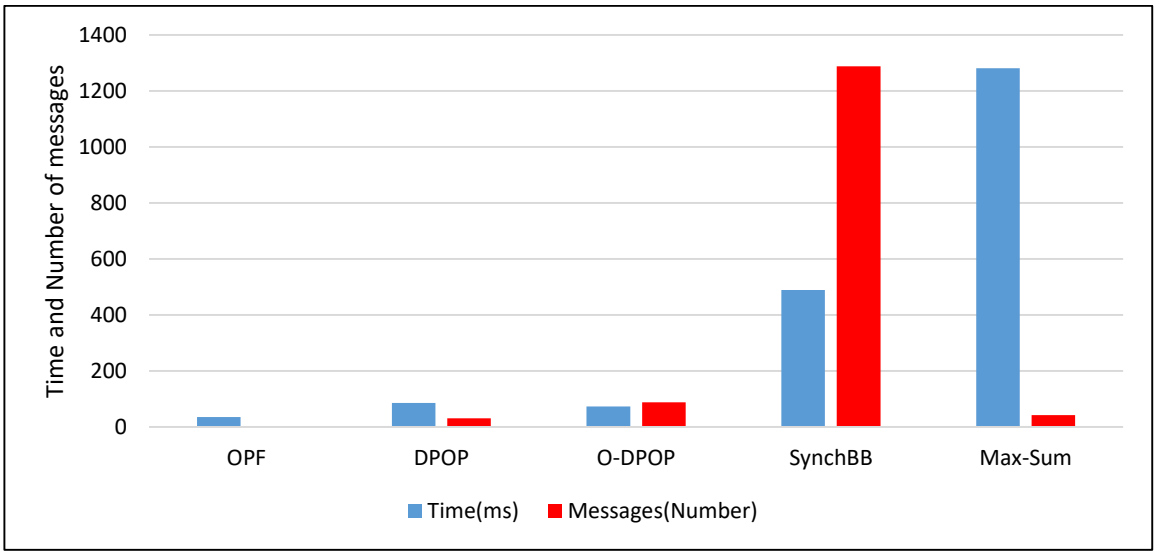


Figure 5.13: Algorithms Comparison

5.7 Case study of 11bus network with sensitivity factors

The network that is used for this case study is based on the IEEE 13 bus network model (*Figure 5.14a*). It has one substation voltage regulator, two shunt capacitor banks, one in-line transformer and several unbalanced spot loads. However, for the case study used here the regulator, the capacitors, the switch and distribution lines with zero length have been neglected. Consequently, the new network model is based on the IEEE 13 node network model but consists of 11 buses. *Figure 5.14b* depicts the 11 node test feeder which is used for applying DCOP algorithms.

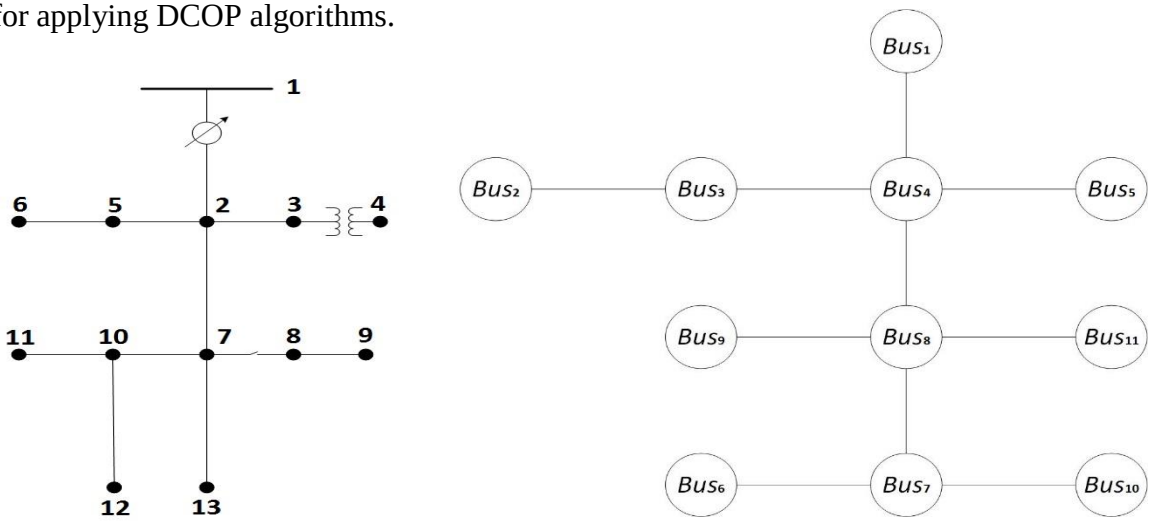


Figure 5.14: (a) IEEE 13 bus (b) 14 node test feeder

For the case study, sensitivity factors are considered and can be defined as $a_{ij} \in R^{+}$. Additionally, equation 5.2 is added:

$$PF_{ij} = \sum_{i=1}^{V} a_{i,ij}(PG_j - PL_j)$$

which means that each flow is depending by the sensitivity factor of each bus, and also the generator output and load at each bus. The sensitivity factors were obtained by MatPower and imported into Frodo software. In order to evaluate the results from the DCOP algorithms, a comparison between OPF technique from MatPower and DCOP algorithms will be accessed to identify the potential differences between the results. *Table 5.1* shows the sensitivity factors for all 11 buses of the network.

	Bus1	Bus2	Bus3	Bus4	Bus5	Bus6	Bus7	Bus8	Bus9	Bus10	Bus11
line1-4	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
line2-3	0	1	7.85E-16	5.80E-16	5.80E-16	5.80E-16	5.80E-16	5.80E-16	5.80E-16	5.80E-16	5.80E-16
line3-4	0	1	1	0	0	0	0	0	0	0	0
line4-5	0	0	0	0	-1	0	0	0	0	0	0
line4-8	0	4.35E-16	4.35E-16	4.35E-16	4.35E-16	-1	-1	-1	-1	-1	-1
line8-7	0	5.80E-16	5.80E-16	5.80E-16	5.80E-16	-1	-1	1.16E-15	1.16E-15	-1	1.16E-15
line7-6	0	5.80E-16	5.80E-16	5.80E-16	5.80E-16	-1	1.29E-15	1.16E-15	1.16E-15	1.29E-15	1.16E-15
line7-10	0	0	0	0	0	0	0	0	0	-1	0
line8-11	0	2.90E-16	2.90E-16	2.90E-16	2.90E-16	5.80E-16	5.80E-16	5.80E-16	5.80E-16	5.80E-16	-1
line8-9	0	-1.16E-15	-1.16E-15	-1.16E-15	-1.16E-15	-2.32E-15	-2.32E-15	-2.32E-15	-1	-2.32E-15	-2.32E-15

Table 5.1: Sensitivity factors for 11 Bus network

Each variable in the DCOP framework becomes a variable node and each constraint in the DCOP framework becomes a function node. A set of agents are defined, and each one controls a variable node and/or a function node. Neighbours are the variable nodes corresponding to the variables the constraint is expressed over (e.g. node 13 become agent A_{10} and contains a combination of F_{7-10} and L_{10}values).

Regarding the agents, the assumption taken was that each agent should own the power output decision variable for its generator because it will be the responsibility of that agent to make this decision (e.g. Agent 1 is responsible for G_1, L_1 and F_{14}). Every agent owns a given variable that it should have control over, and it should be responsible for making that decision. The agent to which a particular constraint is assigned to (a particular function node) must be able to apply the algorithm for that function node.

Under DCOP framework, 4 different algorithms were tested (DPOP, O-DPOP, Max-Sum and SynchBB) and the results are compared to OPF technique from MatPower. When the algorithm converges all DCOP algorithms managed to find the optimal solution. For this case study the results for the generators are: G_1=0.65MW, G_2=1.49MW, and G_5=1.15MW and for the flows are:$F_{14} = 0.65MW, F_{23} = 1.26MW, F_{34} = 1.09MW, F_{45} = -0.75MW, F_{48} = 2.4MW, F_{87} = 0.3MW, F_{76} = 0.17MW, F_{710} = 0.13MW, F_{811} = 0MW, F_{89} = 0.84MW$.

The minus symbol refers to a different flow direction. Furthermore, the total cost for the network is equal to 426.5. *Table 5.2* illustrates the number of messages sent, the amount of information sent and the size of the largest message sent for each of the DCOP Algorithms.

	DPOP	O-DPOP	SynchBB	Max-Sum
Number of messages sent (by type, in bytes)	UTIL: 25 VALUE: 25	UTIL: 90.649 VALUE: 25 ASK: 90.649	BACKTRACK: 1442 ELECT: 36600	FunctiontoVar: 350 Vartofunction: 225
Amount of information sent(by type, in bytes)	UTIL: 1.200.354 VALUE: 8.337	UTIL:6.601.782 VALUE: 4.960 ASK: 906.690	BACKTRACK: 15.869 ELECT: 588.784	FunctiontoVar: 64.918 Vartofunction: 48.154
Size of the largest message sent(by type, in bytes)	UTIL: 1.101.431 VALUE: 570	UTIL: 116 VALUE: 350 ASK:110	BACKTRACK: 59 ELECT: 350	FunctiontoVar: 598 Vartofunction: 601

Table 5.2: DCOP Algorithm Analysis

The Optimal Power Flow (OPF) technique using MatPower managed to reach the same solution with a total generation of 3.3MW by generators and total load consumption of 3.3MW in 30ms while G_1=0.65MW, G_2=1.49MW, and G_5=1.15MW. *Figure 5.15* presents the comparison of algorithms for the 11 bus example.

When comparing the performance of max-sum to DPOP, O-DPOP and SynchBB was found that the complete algorithms DPOP, O-DPOP and SynchBB are between 3 and 6 times faster at solving the problem than max-sum for this particular case study. This means that after the integration of more variables to the network the max-sum is performing better than before. In terms of time, it managed to find the solution in 0.950ms while previously needed 1.282ms. As for the other algorithms, they are performing slower than the previous case study of 3 buses but the difference is very small.

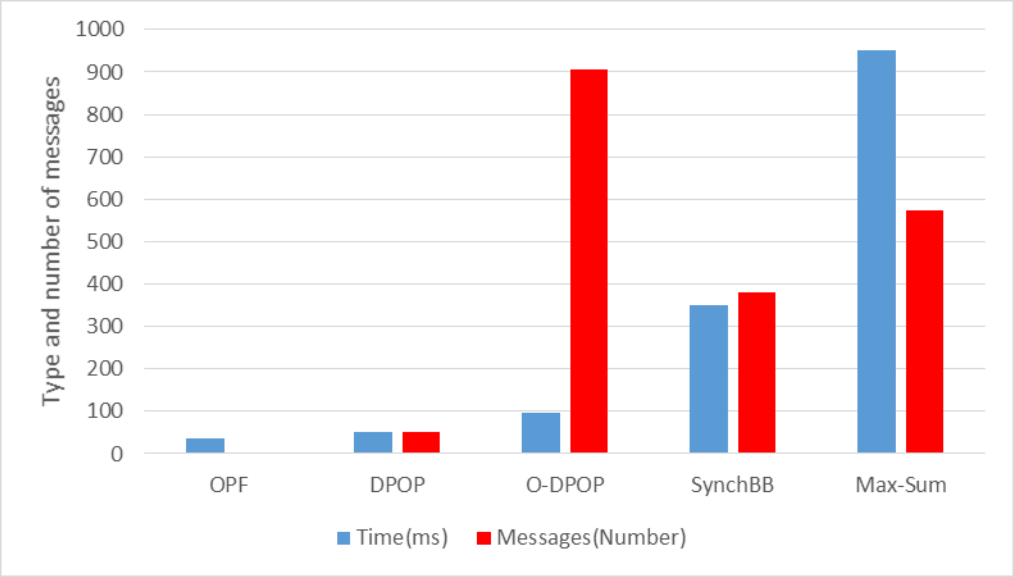


Figure 5.15: Algorithms Comparison

Finally, all the DCOP algorithms obtained the same results, in terms of the values of the variables, which means that the DCOP framework is able to provide feasible solutions in power systems. The case study proves that decentralised techniques, such as DCOP, can be used to solve the economic dispatch problem in electricity networks.

5.8 Case study of 11kV network under DCOP algorithms

The methodology of the previous section was used in order to test the vast majority of DCOP algorithms in a larger network. The network chosen was the real network used within the AuRA-NMS project and contains 40 buses and 3 generators. *Figure 5.16* illustrates the 11kV network.

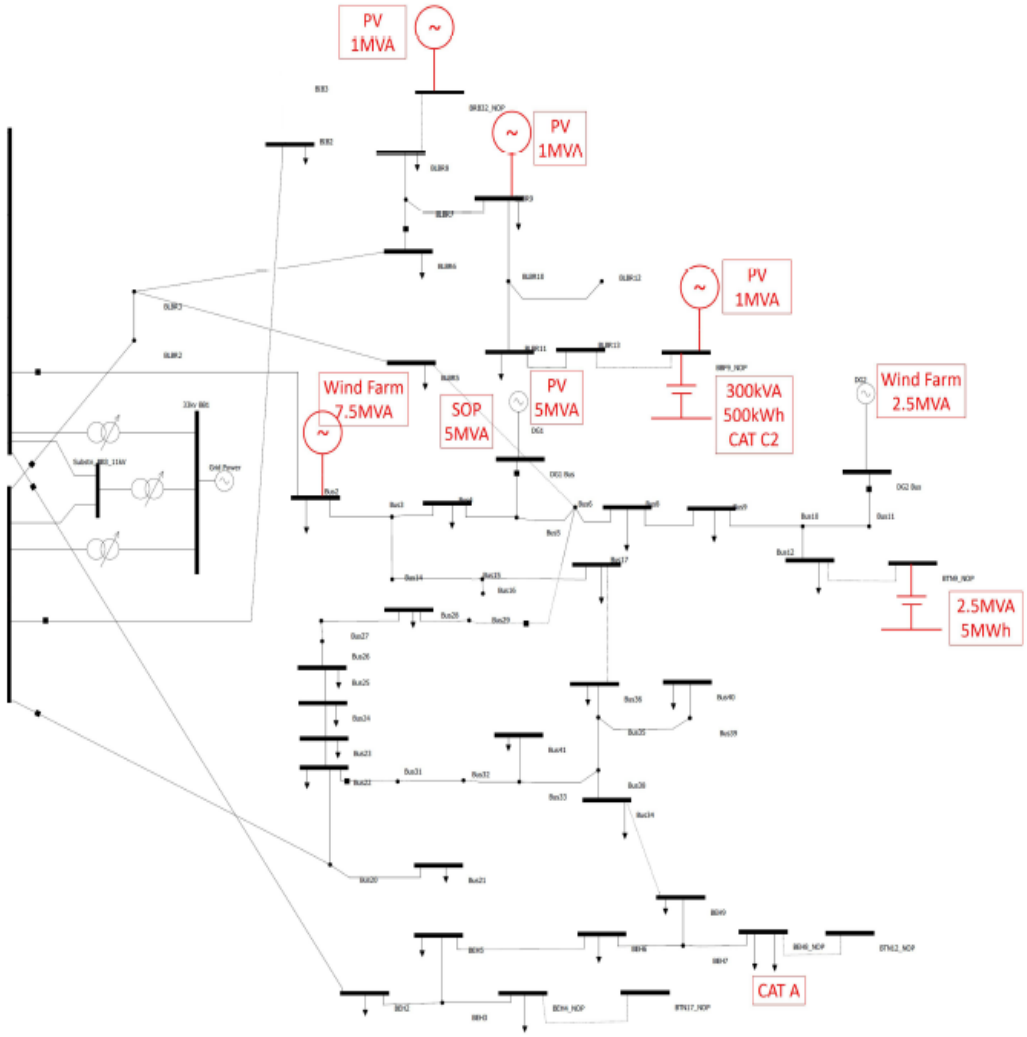


Figure 5.16: 11kV Network

In order to test DCOP algorithms in Frodo for a larger network, a Matlab script had to be written to automatically generate data from power systems software simulator to an XML file. Data obtained by IPSA power systems software simulator in Matlab format and contained all details in order to transform this file to XML. As is mentioned in the previous

section, Frodo software can handle only XCSP and XML files, hence Matlab files need to be translated in the appropriate format file. As the network expands, it's more difficult to write a code file in the Frodo environment so the appropriate Matlab script is necessary to generate XCSP and XML files.

The Matlab script is able to obtain any network file from IPSA software and transforms it to XML or XSCP file. In this situation any network can be tested in Frodo, however, it needs to be radial and not meshed. DCOP algorithms and Frodo's version cannot handle meshed networks according to experimental analysis.

The 11kV network consists of 40 buses, 16 loads, and 3 generators. This modified version of the 11kV network has been chosen in order to test DCOP algorithms in an environment with more variables. All the algorithms managed to find the optimal solution for the network. For all three DCOP algorithms, the solutions indicate that the total generation of the generators is 5.94MW while the total load consumption is 5.94MW. Moreover, $G_{37} = 2.34MW$, $G_{40} = 1.6MW$ and $G_{41} = 2MW$. G_{37} produces the maximum generation because it has the lowest cost for each MW produced ($100£/\text{MWh}$) while G_{40}'s cost is $120£/\text{MWh}$ and G_{41}'s cost is $150£/\text{MWh}$. The total cost should be 726. DPOP is able to find the solution in 15.1s, O-DPOP in 34.35s, SynchBB in 42.42s and max-sum algorithm in 54.63s. However, the Optimal Power Flow technique using MatPower managed to reach the solution with a total generation of 5.94MW and total load consumption of 5.94MW in 80ms while $G_{37} = 2.34MW$, $G_{40} = 1.6MW$ and $G_{41} = 2MW$. The total cost should be 726. The difference in results lies on the fact that in DCOP algorithms a granulated interval of 0.1 for the variables is used due to the limitations of FRODO software. According to Fig. 6, DPOP algorithm is able to find the solution with the minimum number of messages and based on the fact that is the quickest to find the solution hence it is the most efficient algorithm.

Figure 5.16 presents the comparison of algorithms for the 40 bus example. When comparing the performance of max-sum to DPOP, O-DPOP and SynchBB was found that the complete algorithms are between 2 and 4 times faster at solving the problem than max sum for this particular case study. This means that max-sum is performing even better in

	DPOP	O-DPOP	SynchBB	Max-Sum
Number of messages sent (by type, in bytes)	UTIL: 51 VALUE: 51	UTIL: 121 VALUE: 51 ASK: 196	BACKTRACK: 15 ELECT: 17400	FunctiontoVar: 350 Vartofunction: 225
Amount of information sent(by type, in bytes)	UTIL: 139.224 VALUE: 10.794	UTIL:5.641 VALUE: 3.514 ASK: 6960	BACKTRACK: 293 ELECT: 297.970	FunctiontoVar: 64.918 Vartofunction: 48.154
Size of the largest message sent(by type, in bytes)	UTIL: 33.316 VALUE: 288	UTIL: 52 VALUE: 70 ASK:110	BACKTRACK: 58 ELECT: 352	FunctiontoVar: 598 Vartofunction: 601

Table 5.3: DCOP Algorithm analysis

larger network. In terms of time, it managed to find the solution in 0.750ms while previously needed 0.950ms. As for the other algorithms, they are performing slower than the previous case study of 11 buses and the difference now can be depicted as they need more time to converge.

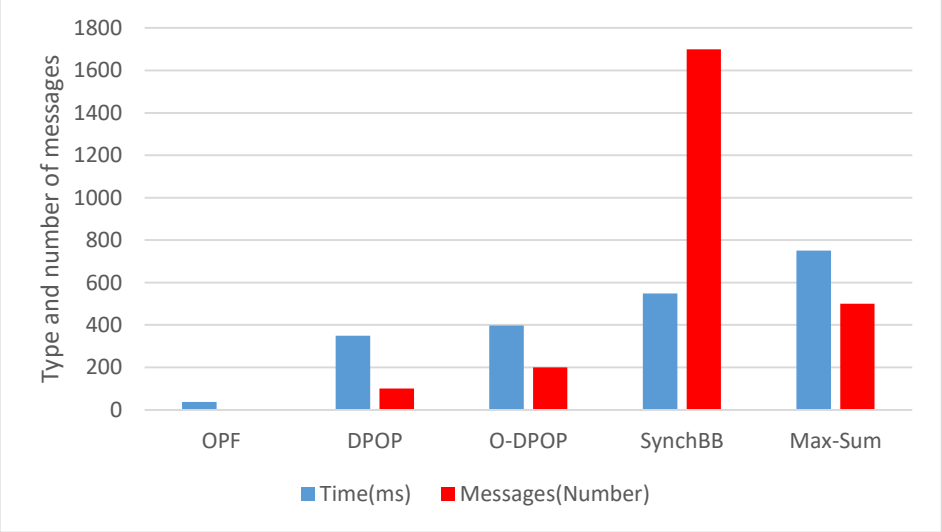


Figure 5.17: Algorithms comparison

5.9 Application of DCOP (Max-sum algorithm) to AC Power Flow Management

In this section, *Figure 5.1* will be used in order to perform the AC power flow management. The AC procedure lies on the model of voltages, angles and also losses in the network. The network contains three generator units with real power P_{g1}, P_{g2}, P_{g3} and reactive power $Q_{g1}, Q_{g2}\ and\ Q_{g3}$. Furthermore, the three loads are fixed such as that $P_{d1} = 10MW, P_{d2} = 10MW$ and $P_{d3} = 20MW$ for real power and $Q_{d1} = 5MW, Q_{d2} = 0MW\ and\ Q_{d3} = 10MW$ for reactive power. There are three voltages V_1 for bus_1, V_2 for bus_2 and V_3 for bus_3. $V_1 = 1$ because it is used as a slack bus and angles δ_1, δ_2 and δ_3 are unknown except from δ_1 which is equal to 0 as it is the reference angle. Also the line resistance in branch 1-3 is equal to 0.01pu and the same also occurs for branch 1-2. In order to solve the optimal power flow, the Ybus matrix needs to be defined. The following equations are used in order to be readable by Frodo software.

Y_{11}	Y_{12}	Y_{13}
100-j100	**-50+50j**	**-50+50j**
Y_{21}	Y_{22}	Y_{23}
-50+50j	**50-50j**	**0**
Y_{31}	Y_{32}	Y_{33}
-50+50j	**0**	**50-50j**

Y_{11}	Y_{12}	Y_{13}
141.42∠315°	**70.71∠135°**	**70.71∠135°**
Y_{21}	Y_{22}	Y_{23}
70.71∠135°	**70.71∠315°**	**0**
Y_{31}	Y_{32}	Y_{33}
70.71∠135°	**0**	**70.71∠315°**

Table 5.4 Ybus matrix

The equations of the power flows need to be defined in Frodo such that:

$$P_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \cos(\delta_k - \delta_n - \theta_{kn}) \tag{5.5}$$

$$Q_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \sin(\delta_k - \delta_n - \theta_{kn}) \tag{5.6}$$

In order to apply the max-sum algorithm in power systems, the network model has to be transformed into a factor graph. *Figure 5.18 (a)* presents the decomposition of the power network model into a factor graph where the circles represent the variables and the squares the utility functions. *Figure 5.18 (b)* illustrates how messages flow from variables to functions and the calculations needed at each step.

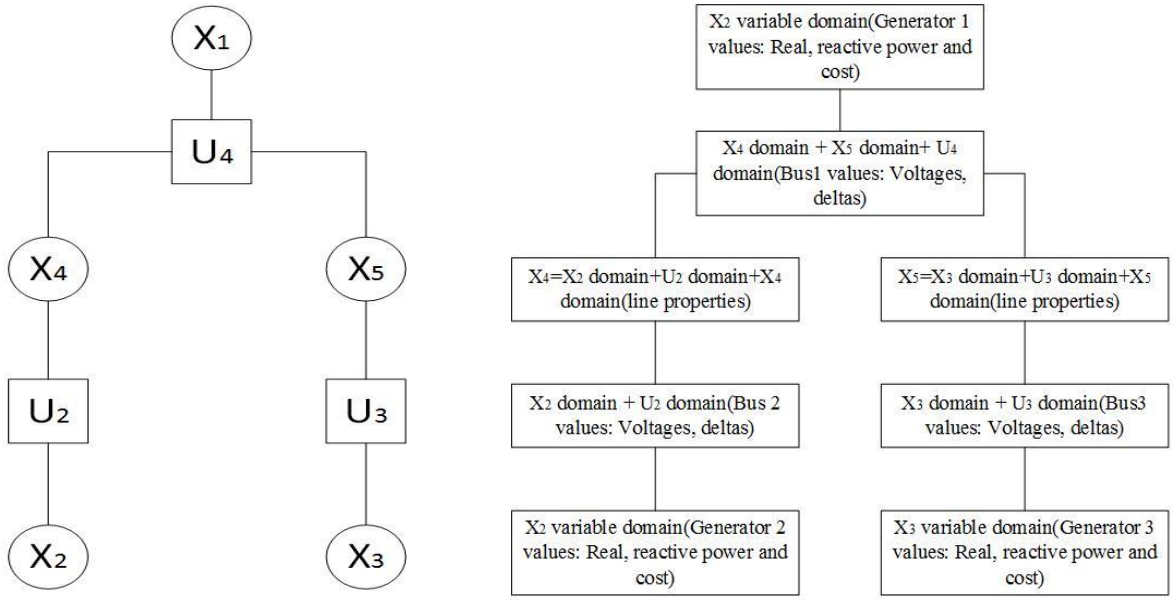


Figure 5.18: (a) Decomposed factor graph (b) Flow of messages considering domains

Figure 5.19, illustrates the procedure of getting an AC solution without using Frodo software by using a simple network of 3 buses.

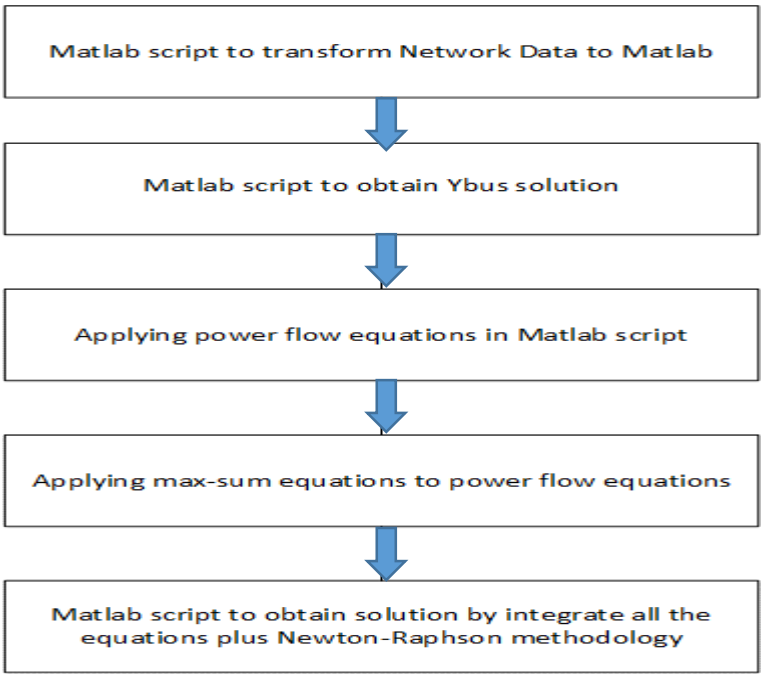


Figure 5.19: Procedure of obtaining AC solution

Equations 4.1 and 4.2 from Chapter 4 are used to calculate the messages. Messages are flowing from variables to functions and from functions to variables. At the beginning, the messages flow from x_1 , x_2 and x_3 variables. The following messages represent the domains of the variables:

$$Q_{1\to4} = 0..20MW$$

$$Q_{2\to2} = 0..20MW$$

$$Q_{3\to3} = 0..30MW$$

The above messages flow to the corresponding utility functions ($U_4, U_2\ and\ U_3$). $R_{2\to4}$ and $R_{3\to5}$ messages receive the domains of $Q_{2\to2}$ and $Q_{3\to3}$ in order to calculate the flows and the equivalent cost. However, $U_2$and U_3 functions contain the equations for power flow as well as input data for Bus_2 and Bus_3. Bus_2 data contain the loads $Pd_2 =$ 10MW and $Qd_2 =$ 10MW and Bus_3 data contain the loads $Pd_3 =$ 20MW and $Qd_3 =$ 10MW. $R_{2\to4}$ and $R_{3\to5}$ messages contain the equations 5.5-5.6 in order to calculate the flows, however they cannot perform any calculation at that time as some data are needed from the neighbour variable, such as the thermal capacity of the line and the resistances of the lines.

Since the data are missing the messages that contain the equations and the data flow to x_4 and x_5 variables. At that point, these variables contain the thermal limits values from the lines as well as line properties values. These variables contain the data for the resistances (0.01 for P_{12} and P_{13}) and also the thermal capacity of each line (30MW for both of them). All the above data are flowing into U_4 as two separate messages $Q_{4\to4}$ and $Q_{5\to4}$. In U_4, messages $Q_{1\to4}$, $Q_{4\to4}$ and $Q_{4\to5}$ are added in order to find the values for unknown data such as V's, deltas and power flows. A presentation of these messages is presented below where the first line illustrates the values of the generators and the second and third line depicts the equations.

$$Q_{4\to4} = \left\{ \begin{matrix} 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15\,16\,17\,18\,19\,20 \\ P_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \cos(\delta_k - \delta_n - \theta_{kn}) \\ Q_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \sin(\delta_k - \delta_n - \theta_{kn}) \end{matrix} \right\}$$

$$Q_{5\to4} = \left\{ \begin{matrix} 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15\,16\,17\,18\,19\,20\,21\,22\,23\,24\,25\,26\,27\,28\,29\,30 \\ P_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \cos(\delta_k - \delta_n - \theta_{kn}) \\ Q_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \sin(\delta_k - \delta_n - \theta_{kn}) \end{matrix} \right\}$$

$$Q_{1\to4} = \{1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15\,16\,17\,18\,19\,20\}$$

Each generator has a cost such as: $CostP_{g1} = 100£/\text{MWh}$, $CostP_{g2} = 150£/\text{MWh}$ and $CostP_{g3} = 120£/\text{MWh}$. For instance, to calculate flow P_{12}, all the possible variations of V_2(0.990-1.100, 110 values), δ_2(-90..90 , 181 values) and P_{g1}(21 values) need to be calculated , and 418,110 (110x181x21) possible values need to be examined. The same procedure occurs for flow P_{13} where all the possible variations of V_3(0.990-1.100, 110 values), δ_3(-90..90 , 181 values) and P_{g3}(31 values) need to be calculated , and 617,210 (110x181x31) possible values in total need to be examined. The Max-sum algorithm calculates even the infeasible solutions, and for that reason the possible variations are thousands meaning that the algorithm is slow when it needs to converge. From the

calculated solutions only the values that minimize the cost were taken into consideration. To analyse the procedure of max-sum algorithm the following calculations present equations regarding the flows.

For P_{12} flow:

$$P_{12} = P_{g_1} - P_{L_1} => P_{g_1} - 20 = V_1V_1Y_{11}cos(\delta_1 - \delta_1 - \theta_{11}) + V_1Y_{12}V_2cos(\delta_1 - \delta_2 - \theta_{12}) + V_1Y_{13}V_3cos(\delta_1 - \delta_3 - \theta_{13}) =>$$

$$P_{g_1} - 20 = 99.99 + 70.71V_2cos(-\delta_2 - 135°) + 70.71V_3cos(-\delta_3 - 135°)$$

For P_{21} flow:

$$P_{21} = P_{g_2} - P_{L_2} => P_{g_2} - 20 = V_2V_1Y_{21}cos(\delta_2 - \delta_1 - \theta_{21}) + V_2Y_{22}V_2cos(\delta_2 - \delta_2 - \theta_{22}) + V_2Y_{23}V_3cos(\delta_2 - \delta_3 - \theta_{23}) =>$$

$$P_{g_2} - 20 = 70.71V_2cos(\delta_2 - 135°) + 70.71V_2^2cos(-135°) =>$$

$$P_{g_2} - 20 = 70.71V_2cos(\delta_2 - 135°) - 49.99V_2^2$$

For P_{31} flow:

$$P_{31} = P_{g_3} - P_{L_3} => P_{g_3} - 20 = V_3V_1Y_{31}cos(\delta_3 - \delta_1 - \theta_{31}) + V_3Y_{32}V_2cos(\delta_3 - \delta_2 - \theta_{32}) + V_3Y_{33}V_3cos(\delta_3 - \delta_3 - \theta_{33}) =>$$

$$P_{g_3} - 20 = 70.71V_3cos(\delta_3 - 135°) + 70.71V_3^2cos(-315°) =>$$

$$P_{g_3} - 20 = 70.71V_3cos(\delta_3 - 135°) + 49.99V_3^2$$

From the above equations the unknowns are V_2, V_3, $\delta_2, \delta_3, P_{g_1}, P_{g_2}$ and P_{g_3}

For Q_{12} flow:

$$Q_{12} = Q_{g_1} - Q_{L_1} => Q_{g_1} - 20 = V_1V_1Y_{11}sin(\delta_1 - \delta_1 - \theta_{11}) + V_1Y_{12}V_2sin(\delta_1 - \delta_2 - \theta_{12}) + V_1Y_{13}V_3sin(\delta_1 - \delta_3 - \theta_{13}) =>$$

$$Q_{g_1} - 50 = -99.99 + 70.71V_2sin(-\delta_2 - 135°) + 70.71V_3sin(-\delta_3 - 135°)$$

For Q_{21} flow:

$$Q_{21} = Q_{g_2} - Q_{L_2} => Q_{g_2} = V_2V_1Y_{21}sin(\delta_2 - \delta_1 - \theta_{21}) + V_2Y_{22}V_2sin(\delta_2 - \delta_2 - \theta_{22}) + V_2Y_{23}V_3sin(\delta_2 - \delta_3 - \theta_{23}) =>$$

$$Q_{g_2} = 70.71V_2sin(\delta_2 - 135°) + 70.71V_2^2sin(-135°) =>$$

$$Q_{g_2} = 70.71V_2sin(\delta_2 - 135°) - 49.99V_2^2$$

For Q_{31} flow:

$$Q_{31} = Q_{g_3} - Q_{L_3} => Q_{g_3} - 100 = V_3V_1Y_{31}sin(\delta_3 - \delta_1 - \theta_{31}) + V_3Y_{32}V_2sin(\delta_3 - \delta_2 - \theta_{32}) + V_3Y_{33}V_3sin(\delta_3 - \delta_3 - \theta_{33}) =>$$

$$Q_{g_3} - 100 = 70.71V_3sin(\delta_3 - 135°) + 70.71V_3^2sin(-315°) =>$$

$$Q_{g_3} - 100 = 70.71V_3sin(\delta_3 - 135°) + 49.99V_3^2$$

From the above equations the unknowns are V_2, V_3, $\delta_2, \delta_3, Q_{g_1}, Q_{g_2}$ and Q_{g_3}

For V_2 and V_3 which have a domain of (0.990-1.100) the first value of 0.990 is picked and all the possible values of δ_2 and δ_3 are calculated (-90...90) such as:

For P_{12} flow:

$$P_{g_1} - 20 = 99.99 + 70.71V_2cos(-\delta_2 - 135°) + 70.71V_3cos(-\delta_3 - 135°) =>$$

$$P_{g_1} = 119.99 + 70.71V_2cos(-\delta_2 - 135°) + 70.71V_3cos(-\delta_3 - 135°) =>$$

$$P_{g_1} = 119.99 + 70.71 * 0.980 * cos(-90 - 135°) + 70.71 * 0.980 * cos(-90 - 135°)$$

$$P_{g_1} = 168.989 - 48.99 => P_{g_1} = 119.99$$

However, this is an infeasible solution because the domain of P_{g_1} is 0..20MW. The next step considers the next value of $V_2 = 0.981$ which provides also an infeasible solution.

The same procedure continues until all solutions are reached with respect to variables domains. The analytical results are presented in Appendix E. From the calculated solutions of the flows only the values that do not violate the constraints are picked. However, many solutions can be obtained that are inside the domains of the variables. The solution with the minimum cost is picked as it provides the optimal solution with respect to network constraints.

Frodo was unable to perform the calculations, and everything was calculated manually, however, the infeasible values were pruned from the beginning in order to be able to perform hundreds of calculations and not thousands.

According to the results V_1= 1(pu), V_2= 1.004(pu) and V_3= 1.007(pu). As for the flows, P_{12}=10MW, Q_{12} = -31.84MVar and Q_{13} =-32.46MVar and generators results are: $P_{G_1} = 20MW$, $P_{G_2} == 0MW$, $P_{G_3} = 20MW$ with the total cost of 4400. The results were compared and contrasted with Matpower's ACOPF and Powerworlds simulator. Both of them provide the same results compared to max-sum algorithm, however a comparison regarding the computation time cannot be performed as the max-sum process was performed manually.

As for the results in DCOP, Frodo was unable to perform the calculations, however, an analysis of AC in DCOP was needed in order to find the future potential of using this framework. In order to obtain the results, the max-sum algorithm was picked as it was the least complicated algorithm in order to perform a case study manually without using any software simulator. According to max-sum algorithm, the result is the same as OPF which makes the framework suitable for performing AC studies. However, without the use of a software simulator, it is unclear when the algorithm will converge and also it is unclear which of the DCOP community algorithms will be able to provide feasible solutions.

5.10 Conclusions

The case studies proved that DCOP algorithms can be applied in power systems in order to address complex distributed problems. However, a further investigation is needed in order to address more complex problems that arise from power systems area (AC case studies). For instance, DCOP algorithms are mainly used for radial networks and there is no evidence that they can be used in networks that contain loops. This is a limitation of DCOP algorithms and not of the DCOP framework, hence the algorithms need to be updated in order to be able to solve for meshed networks. Regarding the algorithms, complete algorithms such as DPOP, O-DPOP and SynchBB are able to provide optimal results faster than incomplete algorithms. However, incomplete algorithms such as max-sum algorithm are able to find a solution, while there is no guarantee that they will provide a solution.

Regarding power systems, all the algorithms can be applied in network models but the appropriate algorithm needs to be chosen for every case. Moreover, the case studies were evaluated under the Optimal Power Flow (OPF) technique in MatPower in order to compare the traditional technique for power flow management in power systems with the distributed optimisation algorithms.

The analytical description of the case studies shows that DCOP algorithms can provide the same results compared to OPF however, the simulation time is slower. To improve the simulation time of the algorithms better evaluation of the search process is needed and also a mechanism that will be able to prune messages with infeasible solutions. This mechanism will be able to send and receive messages that only contain solutions that are consistent with the domain of each agent.

Additionally, communication failure needs to be taken into account while performing simulations of DCOP algorithms. A solution for unreliable communication in networks is to implement an error correction layer in software that can ensure reliable message delivery even when the communication infrastructure itself is inherently unreliable. This can be achieved by an acknowledgement protocol where messages are used to verify that a message has been received. Nonetheless, simple reliance on a lower layer error-correction mechanism to ensure reliable delivery is an inefficient approach for dealing

with unreliable communication infrastructure in DCOP algorithms. Firstly, it will significantly increase the number of messages that must be communicated since every message must be acknowledged. Secondly, to enforce in-order-delivery between a given pair of agents, a sender cannot deliver any messages to a given agent until the message for a previously sent message is received from that agent. The time cost in waiting for that messages can degrade performance and reduce the efficiency of the algorithms.

However, DCOP algorithms can be more efficient and faster by using centralised techniques in order to solve agent sub-problems. The use of efficient centralised techniques will solve agent sub-problems and coordinate agents. These centralised solutions will allow DCOP algorithms to be faster, while the knowledge acquired from the DCOP model will allow reducing communication drawbacks.

Finally, there is no general language to formally specify a DCOP. Most of the algorithms specify DCOP problems in an ad-hoc manner and there are only two open source software's (FRODO and DisChoco) with such limitations that are difficult to be used for more complex power system examples. The appropriate software tool is needed in order to evaluate these algorithms and reduce the limitations that resulted after the case studies analysis.

DCOP framework is a promising area for power systems not only because it can represent a number of coordination situations but also because the framework supports the development of generic algorithms to solve those problems.

Despite the limitations of using FRODO software for DCOP algorithms in real world problems, the DCOP framework is a promising area for solving distributed and decentralised problems. In smart grids, due to the distributed and dynamic nature of loads and generators, agent-based decentralised autonomous control is a very promising solution. Also, DCOP algorithms can be applied to power supply restoration problems and microgrid islanding problems. Noncentralised intelligent techniques such as DCOP provide the required flexibility to support the evolving control functions within the future electricity grid. Local controls that use local information improve system flexibility, modularity, and reliability by removing centralised decision making and the

communications underpinning. However, more algorithms need to be created not only based in the Artificial Intelligent community but also based in areas like power systems.

Considering the efficiency of DCOP algorithms, the appropriate algorithm needs to be picked for any case regarding the computation cost. DCOP framework is NP-hard, which means that is hard to obtain a solution to the problem as the scale of application domain increases, hence the use of DCOP in practical applications, such as power systems, is limited. However, they can be integrated into small networks where the number of constraints and variables is not large. Therefore, incomplete algorithms such as max-sum provide scalability by decreasing the computation and communication cost. However, these algorithms do not guarantee the achievement of the global optimal solution. The case studies proved that max-sum is performing better in large networks compared to complete algorithms. The increase in the number of the variables plays a significant role in the convergence of the algorithms. It is shown that complete algorithms are becoming slower after the introduction of more variables. To sum up, each algorithm can be used for a different example, however, in power system networks which normally consist of a large number of variables and constraints, max-sum can be characterised as a scalable algorithm without even provide the quality guarantees.

Furthermore, regarding the domains, power system networks are usually considered as computationally constrained domains because they mainly operate over high-speed networks.

Finally, DCOP framework assumes that the environment is deterministic and fully-observable, meaning that agents have complete information about the utility of the outcomes of their possible decisions.

Last but not least, robustness becomes an important issue on a larger scale, practical applications, especially in power systems. Hence, the max-sum algorithm is a more suitable and promising algorithm for larger networks. Firstly, because it provides empirically good solutions while tolerating agent failures and communication losses and secondly because it is an algorithm where the agents do not try to maximise their own utility and care about the overall system.

Chapter 6

Conclusions and Further Work

6.1 Conclusions

In the move to a decarbonized energy network, the integration of renewable energy sources will be largely integrated into the electricity system. The grand challenge of electricity networks is to deliver the fundamental changes in the electrical power system that will support this transition and maintain the robustness of the network [MTA+12].

Chapter 2 presented the benefits and challenges of moving from centralised to decentralised energy system with the integration of distributed generation, energy storage and demand response. A challenging issue for system designers and network operators is to provide future control techniques in combination with the traditional practices.

Additionally, active network management was introduced and described its effectiveness in distribution networks. ANM involves the management of power generation and demand without violating network constraints and decreasing the costs of networks potential upgrade. ANM is able to monitor critical grid locations, communications status and the operation status of distributed energy sources. In the case of violations of thresholds, ANM operates autonomously in order to provide set points and signals to grid devices. The idea behind ANM lies on the balance of electricity flows and the maximisation of generators output. This thesis provides ANM schemes in Europe, USA and Canada and also the industrial use of ANM in the UK.

Chapter 3 presented the Constraint Satisfaction and Distributed Constraint Satisfaction process. This process was introduced and applied in Artificial Intelligence community in small examples such as: a) the n-Queen problem, where the aim of the problem is to find the appropriate positions of two queens in a chess game so that they do not attack each other (not on the same row, column or diagonal) b) crossword puzzle where the aim is to complete the puzzle.

Even though the process has been implemented in small examples, the idea behind the constraint satisfaction makes this process suitable to be applied in power system networks.

Problems in constraint satisfaction can be characterised as problems based on constraints on a finite domain. Power system networks problems can also be identified by the same characterization, so the implementation of this process provides a new technique of solving problems in power systems. Chapter 3 analysed the idea of constraint satisfaction and provided the vast majority of algorithms that have been used in order to tackle such problems. AuRA- NMS was the first project that integrated constraint satisfaction as a technique of power flow management regarding thermal constraints [DMM+09]. The constraint satisfaction approach offered a number of ranked solutions in the event of an overload. The algorithm was able to reduce the output of generators in order to satisfy the network constraints with the minimum cost. The process of constraint satisfaction was purely applied in power systems, however, the technique is not considered to be a decentralised technique hence a central controller needed to be informed from all the generators in order to provide a solution.

The integration of constraint satisfaction in power systems led to the importance of applying a distributed and decentralised technique. For this reason, this research focused on the importance of applying distributed constraint optimisation in power systems. This framework has a significant similarity with constraint satisfaction, however, it does not only satisfy its constraints but also provides and optimal solution in a distributed manner.

Chapter 4 illustrated the idea of DCOP and present the vast majority of algorithms that can be applied in different examples. The algorithms can be divided into two categories: a) complete algorithms that offer a guarantee of the optimal solution but have exponential costs due to a large number of messages. The implementation of large power networks is unfeasible due to a large number of constraints and consequently, they can only be applied in small networks and b) incomplete algorithms that offer no guarantees on the optimality, provide suboptimal solutions but with less communication and with a small number of constraints. Even though every algorithm cannot provide the optimal results to any problem, it does not mean that cannot be applied and tested in small networks in order to identify their effectiveness.

For this reason, Chapter 5 illustrated three different case studies regarding power flow management in order to test the effectiveness of the vast majority of DCOP algorithms. A

network example has been used for DC power flow management and AC power flow management. The algorithms were able to provide optimal solutions for this network, however, DPOP performed better than the others. In terms of computation time and number of exchanged messages, a comparison was presented in order to understand the differences between the algorithms. The results showed that in case studies all the algorithms are able to find the optimal solution. However, some of the algorithms (DPOP and O-DPOP) are performing faster computations than the others. This result based on the search process that any algorithm is using in order to find the optimal solution. Some algorithms are decomposed the system network in depth first tree (DPOP, O-DPOP and SynchBB) and others to factor graph (Max-sum).

The process of decomposing the system network in tree or factor graph results in increase of the computation time. The fact that all the algorithms provide optimal solutions based on the network that was used. For the case studies, the network that was used can be considered small if it is compared to typical power system networks examples. The reason of using a small network lies on the optimisation software FRODO that was picked to perform the algorithms' optimisation.

FRODO is the only optimisation software that is able to perform calculations among the vast majority of DCOP algorithms. The main reason for choosing this software was to be able to perform analysis of all the algorithms in order to compare and contrast them. Another option would be the implementation of only one algorithm without the use of FRODO software. FRODO is also able to decompose any power system network in factor graph or depth first tree. So, for simplicity reasons, FRODO was picked in order to optimise the problems. However, in AC power flow management problem, FRODO cannot be considered as the adequate software to obtain results. The reason of applying DCOP for AC and not only in DC is because it is more realistic to provide a technique that is able to solve the power flow management in power systems. For distribution networks, it is vital to include line losses and voltages hence the technique had to be tested in AC network.

As mentioned earlier, DCOP algorithms were used only in small examples like meeting scheduling. There was not any connection between power systems and DCOP algorithms.

Only one researcher tried to apply DCOP in power systems but did not take into account all physical aspects of the network [MRR12]. After many tests, it was understandable that FRODO cannot provide any solution to any problem regarding AC power flow management. Even though updates of software were provided, these could not fix the biggest challenges, where the issues still remain. FRODO is not able to perform complex calculations and even when the transformations were applied in order to avoid the issues, the software reached arithmetic overflow.

Various suggested methods were implemented in order to avoid these problems but all of them failed to reduce the arithmetic overflow. For instance, FRODO is not able to use real values but only integer values and the AC power flow management was difficult to be performed in these conditions. Voltages and magnitudes were transformed from real to integer values, however, all these multiplications had a significant rise of domains hence the calculations reached the arithmetic overflow. Another suggestion involved the use of smaller values for voltages and magnitudes but still, the problems remain the same. However, if FRODO is configured to handle real values in the future, then it can be considered as a powerful tool for solving DCOP problems. This will reduce the arithmetic overflow and be able to handle more complex calculations. In addition, it will contribute to the reduction of simulation time.

6.2 Novelty

The DCOP framework is a promising area for power systems control functions. Even though there is not yet significant contributions in the power systems domain, the framework can be characterised as a potential and different procedure for power systems control functions. This is the first time where physical constraints of the network were considered in order to apply DCOP to power flow management. The integration of sensitivity factors to the case studies was crucial by providing a calculation of possible overloads in power flows. These sensitivity factors show the approximate change in line flows for changes in generation on the network configuration. Hence, with the possibility of a line failure, the power flows of the other lines will be able to cover the demand of the network by satisfying the network constraints. The sensitivity factors were calculated and

applied in the DCOP framework as constraints in order to provide the lower and upper bounds of each power flow line.

The DCOP framework cannot be changed, as the agents, variables and constraints need to be defined at the beginning. However, the representation of the DCOP framework for power systems applications is the crucial part. The DCOP framework was not built for power systems, hence the appropriate transformation is needed in order to understand the physical topology of the network. The most important part of the transformation is the relations between the agents, variables, and constraints. The type of constraints (hard or soft) needs to be defined in order to avoid any possibility of violation among them. Also, the type of variables is crucial because they have an effect on the convergence of the solution. Finally, the number of variables and constraints need to be properly defined because they will affect the efficiency of the algorithm. For each case study the agents, variables and constraints were properly defined. This part is crucial because, in order to compare and contrast all the algorithms, the framework has to be the same for each of them.

Additionally, it is the first time where the vast majority of the algorithms were applied to a real problem. Most of the previous research applied only one algorithm into a specific application and didn't compare and contrast the efficiency of other algorithms. After the analysis of the algorithms for the specific case studies, it is clearer which algorithm is the most appropriate for each case study and what are the advantages and disadvantages of each one.

More importantly, the case studies provided results for power flow management by using the vast majority of complete and incomplete algorithms. However, the framework is also capable of integrating more technologies such as energy storage and renewable technologies. It can be considered as a plug and play framework because any additional technology to the framework can be applied without any significant change of the predefined network.

6.3 Further Work

The research of this thesis results in an integration of artificial intelligence techniques into power systems. Further work in a number of areas could provide more benefits to the existing research. DPOP is more efficient regarding the number of messages exchanged even though the other algorithms are able to find the optimal solution. However, in a more complex network, max-sum algorithm will provide better results close to optimal. In order to evaluate max-sum algorithm, a more complex network has to be tested. To be able to move to a more complex network, different optimisation software is needed. FRODO will not be updated in order to cover all the limitations that mentioned earlier so it will not be adequate to extend the work by using the same tool.

6.3.1 Complex Network

In order to evaluate properly the efficiency of the algorithm, a more complex network is needed. A complex network that will involve more generators and more loads but still has to be a radial network. All the DCOP algorithms cannot be applied in meshed networks because they were never developed in order to be implemented in a network with loops. For radial networks, the message values converge and each variable can choose its best state. However, in some circumstances, the variables can choose their best states but the message values keep changing their values subtly. In meshed networks, there is the potential for Lagrangian techniques and the Auxiliary Problem Principle [Coh80] to be used to express cyclic electricity problems as decoupled subproblems that can be solved in parallel. There is no evidence in academia of using DCOP algorithms in cyclic networks however an attempt to use the Auxiliary Problem Principle will be performed in order to evaluate the possible outcomes of using this method.

6.3.2 Demand-side Management

Future work will include demand-side management and the integration of consumers. Some initial case studies already have examined the possibility of adding customers and forecast demand for 24 hours. The network will contain energy storage devices and photovoltaic energy sources. During the day only the photovoltaic panels will be able to

provide energy to the network and in the case of excess energy (while satisfying the customer's demand), energy storage devices will be able to maintain the rest of the energy. Another possible scenario will examine the residual energy to be transferred across the network.

6.3.3 New optimisation software

FRODO was initially picked for solving optimisation problems regarding the algorithm selection. Even though it contributes significantly to the research, allowing compare and contrast studies of the majority of algorithms, a different optimisation software tool is needed to avoid the limitation arising from FRODO. Another optimisation software such as DisChoco may be used to solve optimisation problems [EBB+07]. DisChoco is a Java open source platform for solving distributed constraint reasoning. An updated version of the software is able to solve distributed problems however the software has the same limitations as FRODO. A possible way to overcome these problems is to pick only one algorithm from the DCOP framework and apply it in a Java environment without the use of FRODO or DisChoco. In that case, any potential radial network can be used by importing from a power systems simulator programme. This process was abandoned at the beginning of the research because a comparison of DCOP algorithms was needed in order to able to understand and evaluate their effectiveness. Following the research reported in this thesis, this process can now be used in order to examine only one algorithm and its benefits in cases such as AC power flow management.

6.3.4 Future work regarding FRODO software

Even though DCOP can be considered as a promising framework for power systems, the appropriate approach is needed to overcome the limitations that arose during the case studies. An updated version of FRODO will be able to handle real numbers in order to apply it to AC power flow management. Also, there is a need to include more libraries in FRODO to be able to use more complex calculations. Although this procedure sounds promising, it is difficult to tackle these updates because FRODO was created to tackle simple examples and not those from the power systems community. A new tool has to be

designed taking into account all the limitations. For this reason, the best approach is to pick one of the discussed algorithms (especially DPOP which is the most efficient algorithm) and try to apply power flow management without the use of FRODO. In this case, the DPOP algorithm can be designed in a Python or Java environment where the above limitations of complex calculations and real numbers can be tackled. This approach was considered at the beginning of this research, however, the FRODO software was picked in order to evaluate, compare and contrast the vast majority of DCOP algorithms. This research has explored and evaluated the algorithms, and the most efficient algorithm can now be used to tackle the limitations of the FRODO software tool. Furthermore, with this approach, the AC power flow management can be performed and the results can be evaluated and further analysed. Besides, this approach will provide flexibility and scalability as more control functions can be integrated into the network. Future simulation analysis will include energy storage, PV's and demand side management in order to prove that the DCOP framework is able to provide flexibility between the control functions but also scalability. Finally, meshed networks will be also assessed in order to test the applicability of DCOP algorithms in cyclic power system networks. Lagrangian techniques and the Auxiliary Problem Principle [Coh80] could be used to express cyclic electricity problems as decoupled subproblems that can be solved in parallel.

Appendix A

XCSP file code for DC Power Flow Management

```
<?xml version="1.0" encoding="UTF-8"?>
<instancexmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="src/ch/epfl/lia/frodo/algorithms/XCSPschemaJaCoP.xsd">
<presentation name="Athanasiadis13" maxConstraintArity="4" maximize="false" format="XCSP
2.1_FRODO" />
<agents nbAgents="3">
                <agent name="agent_g1" />
                <agent name="agent_g2" />
                <agent name="agent_g3" />
        </agents>
<domains nbDomains="5">
                <domain name="dom_g1" nbValues="5">-2..2</domain>
                <domain name="dom_g2_and_g3" nbValues="7">-3..3</domain>
                <domain name="dom_f12" nbValues="9">-4..4</domain>
                <domain name="dom_f13" nbValues="7">-3..3</domain>
                <domain name="dom_l" nbValues="1">-2</domain>
        </domains>
<variables nbVariables="8">
                <variable name="g1" domain="dom_g1" agent="agent_g1" />
                <variable name="g2" domain="dom_g2_and_g3" agent="agent_g2" />
                <variable name="g3" domain="dom_g2_and_g3" agent="agent_g3" />
                <variable name="f12" domain="dom_f12" agent="agent_g2" />
                <variable name="f13" domain="dom_f13" agent="agent_g3" />
                <variable name="l1" domain="dom_l" agent="agent_g1" />
                <variable name="l2" domain="dom_l" agent="agent_g2" />
                <variable name="l3" domain="dom_l" agent="agent_g3" />
        </variables>
<functions nbFunctions="1">
                <function name="times" return="int">
```

```
                        <parameters>int p1 int p2</parameters>
                        <expression>
                                <functional>mul(p1, p2)</functional>
                        </expression>
                </function>
        </functions>
<constraints nbConstraints="6">
                <constraint   name="conserv_g1"   arity="4"   scope="g1   l1   f12   f13"
reference="global:weightedSum">
                <parameters>[ { 1 g1 } { 1 l1 } { 1 f12 } { -1 f13 } ] <eq/> 0</parameters>
                </constraint>
                <constraint    name="conserv_g2"    arity="3"    scope="g2    l2    f12"
reference="global:weightedSum">
                <parameters>[ { 1 g2 } { 1 l2 } { -1 f12 } ] <eq/> 0</parameters>
                </constraint>
                <constraint    name="conserv_g3"    arity="3"    scope="g3    l3    f13"
reference="global:weightedSum">
                <parameters>[ { 1 g3 } { 1 l3 } { 1 f13 } ] <eq/> 0</parameters>
                </constraint>
                <constraint name="cost_g1" arity="1" scope="g1" reference="times">
                        <parameters>5 g1</parameters>
                </constraint>
                <constraint name="cost_g2" arity="1" scope="g2" reference="times">
                        <parameters>10 g2</parameters>
                </constraint>
                <constraint name="cost_g3" arity="1" scope="g3" reference="times">
                        <parameters>8 g3</parameters>
                </constraint>
        </constraints>
</instance
```

Appendix B

XCSP file code for AC Power Flow Management

```
<?xml version="1.0" encoding="UTF-8"?>

<instance xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="src/ch/epfl/lia/frodo/algorithms/XCSPschemaJaCoP.xsd">

        <presentation name="Athanasiadis14new" maxConstraintArity="4" maximize="false"
format="XCSP 2.1_FRODO" />

        <agents nbAgents="3">

                <agent name="agent_g1" />

                <agent name="agent_g2" />

                <agent name="agent_g3" />

        </agents>

        <domains nbDomains="34">

<domain name="dom_Pg1" nbValues="201">0..200</domain>

<domain name="dom_Pg2" nbValues="201">0..200</domain>

<domain name="dom_Pg3" nbValues="201">0..200</domain>

<domain name="dom_Pf13" nbValues="1">-998992012 -994011992 994011992 998992012</domain>

<domain name="dom_Pf23" nbValues="1">-1003996982 -998992012 998992012 1003996982</domain>

<domain name="dom_Qf13" nbValues="4">-998992012 -994011992 994011992 998992012</domain>

<domain name="dom_Qf23" nbValues="4">-1003996982 -998992012 998992012 1003996982</domain>

<domain name="dom_Pd1" nbValues="1">100</domain>

<domain name="dom_Pd2" nbValues="1">100</domain>

<domain name="dom_Pd3" nbValues="1">200</domain>

<domain name="dom_Qg3" nbValues="101">0..100</domain>

<domain name="dom_Qg2" nbValues="101">0..100</domain>

<domain name="dom_Qg1" nbValues="101">0..100</domain>

<domain name="dom_Qd1" nbValues="1">50</domain>

<domain name="dom_Qd2" nbValues="1">0</domain>

<domain name="dom_Qd3" nbValues="1">100</domain>

<domain name="angle3" nbValues="1">0</domain>

<domain name="angle2" nbValues="41">550..590</domain>

<domain name="angle1" nbValues="41">550..590</domain>
```

```
<domain name="angMinAng" nbValues="4">-575 -572 572 575</domain>
<domain name="dom_M31" nbValues="4">-998992012 -994011992 994011992 998992012</domain>
<domain name="dom_M32" nbValues="4">-1003996982 -998992012 998992012 1003996982</domain>
<domain name="dom_M33" nbValues="4">-998992012 -994011992 994011992 998992012</domain>
<domain name="dom_V1" nbValues="1">998</domain>
<domain name="dom_V2" nbValues="1">1003</domain>
<domain name="dom_V3" nbValues="1">998</domain>
<domain name="dom_X13" nbValues="4">-998 -1003 998 1003</domain>
<domain name="dom_X23" nbValues="4">-998 -1003 998 1003</domain>
<domain name="dom_X31" nbValues="4">-998 -1003 998 1003</domain>
<domain name="dom_X32" nbValues="4">-998 -1003 998 1003</domain>
<domain name="dom_X13s" nbValues="4">-998 -1003 998 1003</domain>
<domain name="dom_X23s" nbValues="4">-998 -1003 998 1003</domain>
<domain name="dom_X31s" nbValues="4">-998 -1003 998 1003</domain>
<domain name="dom_X32s" nbValues="4">-998 -1003 998 1003</domain>
        </domains>
        <variables nbVariables="35">
                <variable name="Pg1" domain="dom_Pg1" agent="agent_g1" />
                <variable name="Pg2" domain="dom_Pg2" agent="agent_g2" />
                <variable name="Pg3" domain="dom_Pg3" agent="agent_g3" />
                <variable name="Pf23" domain="dom_Pf23" agent="agent_g2" />
                <variable name="Pf13" domain="dom_Pf13" agent="agent_g1" />
                <variable name="Qf23" domain="dom_Qf23" agent="agent_g2" />
                <variable name="Qf13" domain="dom_Qf13" agent="agent_g1" />
                <variable name="Pd1" domain="dom_Pd1" agent="agent_g1" />
                <variable name="Pd2" domain="dom_Pd2" agent="agent_g2" />
                <variable name="Pd3" domain="dom_Pd3" agent="agent_g3" />
                <variable name="Qg3" domain="dom_Qg3" agent="agent_g3" />
                <variable name="Qg2" domain="dom_Qg2" agent="agent_g2" />
                <variable name="Qg1" domain="dom_Qg1" agent="agent_g1" />
                <variable name="Qd1" domain="dom_Qd1" agent="agent_g1" />
                <variable name="Qd2" domain="dom_Qd2" agent="agent_g2" />
                <variable name="Qd3" domain="dom_Qd3" agent="agent_g3" />
```

```
        <variable name="d3" domain="angle3" agent="agent_g3" />
        <variable name="d2" domain="angle2" agent="agent_g2" />
        <variable name="d1" domain="angle1" agent="agent_g1" />
        <variable name="D13" domain="angMinAng" agent="agent_g1" />
        <variable name="D23" domain="angMinAng" agent="agent_g2" />
        <variable name="M31" domain="dom_M31" agent="agent_g3" />
        <variable name="M32" domain="dom_M32" agent="agent_g3" />
        <variable name="M33" domain="dom_M33" agent="agent_g3" />
        <variable name="V1" domain="dom_V1" agent="agent_g1" />
        <variable name="V2" domain="dom_V2" agent="agent_g2" />
        <variable name="V3" domain="dom_V3" agent="agent_g3" />
        <variable name="X13" domain="dom_X13" agent="agent_g1" />
        <variable name="X23" domain="dom_X23" agent="agent_g2" />
        <variable name="X31" domain="dom_X31" agent="agent_g3" />
        <variable name="X32" domain="dom_X32" agent="agent_g3" />
        <variable name="X13s" domain="dom_X13s" agent="agent_g1" />
        <variable name="X23s" domain="dom_X23s" agent="agent_g2" />
        <variable name="X31s" domain="dom_X31s" agent="agent_g3" />
        <variable name="X32s" domain="dom_X32s" agent="agent_g3" />
</variables>
<relations nbRelations="8">
<relation name="cos13rel" arity="2" nbTuples="4" semantics="supports">
        -998 -572|-1003 -575|998 572|1003 575
        </relation>
<relation name="cos23rel" arity="2" nbTuples="4" semantics="supports">
        -998 -572|-1003 -575|998 572|1003 575
        </relation>
<relation name="cos3_1rel" arity="2" nbTuples="4" semantics="supports">
        998 -572|1003 -575|-998 572|-1003 575
        </relation>
<relation name="cos3_2rel" arity="2" nbTuples="4" semantics="supports">
        -998 -572|-1003 -575|998 572|1003 575
        </relation>
```

```
<relation name="sins13rel" arity="2" nbTuples="4" semantics="supports">
        -99995 -572|-99994 -575|-99995 572|-99994 575
        </relation>
<relation name="sins23rel" arity="2" nbTuples="4" semantics="supports">
        -99995 -572|-99994 -575|-99995 572|-99994 575
        </relation>
<relation name="sins3_1rel" arity="2" nbTuples="4" semantics="supports">
        99995 -572|99994 -575|99995 572|99994 575
        </relation>
<relation name="sins3_2rel" arity="2" nbTuples="4" semantics="supports">
        -99995 -572|-99994 -575|-99995 572|-99994 575
        </relation>
</relations>
<predicates nbPredicates="2">
<predicate name="xPred">
        <parameters>int X int Vx int Vy int M</parameters>
        <expression>
                <functional>eq(mul(mul(X,Vx),Vy),M)</functional>
        </expression>
        </predicate>
<predicate name="DPred">
        <parameters>int dx int dy int Dxy</parameters>
        <expression>
                <functional>eq(add(dy,Dxy),dx)</functional>
        </expression>
        </predicate>
</predicates>
<functions nbFunctions="1">
        <function name="times" return="int">
                <parameters>int p1 int p2</parameters>
                <expression>
                        <functional>mul(p1, p2)</functional>
                </expression>
```

```
        </function>
</functions>
<constraints nbConstraints="33">
<constraint name="cos13cons" arity="2" scope="X13 D13" reference="cos13rel">
        </constraint>
<constraint name="x13cons" arity="4" scope="X13 V1 V3 Pf13" reference="xPred">
        <parameters>X13 V1 V3 Pf13</parameters>
        </constraint>
<constraint name="D13cons" arity="3" scope="d1 d3 D13" reference="DPred">
        <parameters>d1 d3 D13</parameters>
        </constraint>
<constraint name="cos23cons" arity="2" scope="X23 D23" reference="cos23rel">
        </constraint>
<constraint name="x23cons" arity="4" scope="X23 V2 V3 Pf23" reference="xPred">
        <parameters>X23 V2 V3 Pf23</parameters>
        </constraint>
<constraint name="D23cons" arity="3" scope="d2 d3 D23" reference="DPred">
        <parameters>d2 d3 D23</parameters>
        </constraint>
<constraint name="eq3cons" arity="4" scope="Pf23 Pf13 M31 M32"
reference="global:weightedSum">
        <parameters>[ { 1 Pf23 } { 1 Pf13 } { 1 M31 } { 1 M32 }
        ]<eq/>0</parameters>
        </constraint>
<constraint name="cos3_1cons" arity="2" scope="X31 D13" reference="cos3_1rel">
        </constraint>
<constraint name="x3_1cons" arity="4" scope="X31 V3 V1 M31" reference="xPred">
        <parameters>X31 V3 V1 M31</parameters>
        </constraint>
<constraint name="cos3_2cons" arity="2" scope="X32 D23" reference="cos3_2rel">
        </constraint>
<constraint name="x3_2cons" arity="4" scope="X32 V3 V2 M32" reference="xPred">
        <parameters>X32 V3 V2 M32</parameters>
        </constraint>
```

```
<constraint name="sins13sins" arity="2" scope="X13s D13" reference="sins13rel">
        </constraint>
<constraint name="x13sins" arity="4" scope="X13s V1 V3 Qf13" reference="xPred">
        <parameters>X13s V1 V3 Qf13</parameters>
        </constraint>
<constraint name="sins23sins" arity="2" scope="X23s D23" reference="sins23rel">
        </constraint>
<constraint name="x23sins" arity="4" scope="X23s V2 V3 Qf23" reference="xPred">
        <parameters>X23s V2 V3 Qf23</parameters>
        </constraint>
<constraint name="eq3sins" arity="5" scope="Qf23 Qf13 M31 M32 M33"
reference="global:weightedSum">
        <parameters>[ { 1 Qf23 } { 1 Qf13 } { 1 M31 } { 1 M32 } { 1 M33 }
        ]<eq/>0</parameters>
        </constraint>
<constraint name="sins3_1sins" arity="2" scope="X31s D13" reference="sins3_1rel">
        </constraint>
<constraint name="x3_1sins" arity="4" scope="X31s V3 V1 M31" reference="xPred">
        <parameters>X31s V3 V1 M31</parameters>
        </constraint>
<constraint name="sins3_2sins" arity="2" scope="X32s D23" reference="sins3_2rel">
        </constraint>
<constraint name="x3_2sins" arity="4" scope="X32s V3 V2 M32" reference="xPred">
        <parameters>X32s V3 V2 M32</parameters>
        </constraint>
<constraint name="x3_3sins" arity="3" scope="V3 V3 M33" reference="xPred">
        <parameters>-200 V3 V3 M33</parameters>
        </constraint>
<constraint name="eq1" arity="3" scope="Pg1 Pd1 Pf13"
reference="global:weightedSum">
        <parameters>[ { 1 Pg1 } { -1 Pd1} { -1 Pf13 } ]<lt/>11000</parameters>
        </constraint>
```

```
<constraint name="eq2" arity="3" scope="Pg2 Pd2 Pf23"
reference="global:weightedSum">

        <parameters>[ { 1 Pg2 } { -1 Pd2} { -1 Pf23 } ]<lt/>11000</parameters>

        </constraint>

<constraint name="eq3" arity="4" scope="Pg3 Pd3 Pf13 Pf23"
reference="global:weightedSum">

        <parameters>[ { 1 Pg3 } { -1 Pd3} { 1 Pf13 } { 1 Pf23 }
        ]<lt/>11000</parameters>

        </constraint>

<constraint name="eq1s" arity="3" scope="Qg1 Qd1 Qf13"
reference="global:weightedSum">

        <parameters>[ { 1 Qg1 } { -1 Qd1} { -1 Qf13 } ]<lt/>11000</parameters>

        </constraint>

<constraint name="eq2s" arity="3" scope="Qg2 Qd2 Qf23"
reference="global:weightedSum">

        <parameters>[ { 1 Qg2 } { -1 Qd2} { -1 Qf23 } ]<lt/>11000</parameters>

        </constraint>

<constraint name="eq3s" arity="4" scope="Qg3 Qd3 Qf13 Qf23"
reference="global:weightedSum">

        <parameters>[ { 1 Qg3 } { -1 Qd3} { 1 Qf13 } { 1 Qf23 }
        ]<lt/>11000</parameters>

        </constraint>

<constraint name="eqPf13"  arity="3" scope="Pf13 Pg1 Qg1"
reference="global:weightedSum">

        <parameters>[ { 1 Pf13 } { -1 Pg1 } { -1 Qg1 }  ]<eq/>150000</parameters>

        </constraint>

<constraint name="eqPf23" arity="3" scope="Pf23 Pg2 Qg2"
reference="global:weightedSum">

        <parameters>[ { 1 Pf23 } { -1 Pg2 } { -1 Qg2 } ]<eq/>100000</parameters>

        </constraint>

<constraint name="eqPf3231" arity="4" scope="Pf23 Pf13 Pg3 Qg3"
reference="global:weightedSum">

        <parameters>[ { 1 Pf23 } { 1 Pf13 }{ 1 Pg3 } { 1 Qg3 }
        ]<eq/>300000</parameters>

        </constraint>
```

```
        <constraint name="cost_Pg1" arity="1" scope="Pg1" reference="times">
                        <parameters>5 Pg1</parameters>
                </constraint>
        <constraint name="cost_Pg2" arity="1" scope="Pg2" reference="times">
                        <parameters>8 Pg2</parameters>
                </constraint>
        <constraint name="cost_Pg3" arity="1" scope="Pg3" reference="times">
                        <parameters>100 Pg3</parameters>
                </constraint>
        </constraints>
</instance>
```

Appendix C

Matlab script from IPSA file to XSCP file

```
%%
clc
clear all
close all
close all hidden


%% Call File

run radial.m % Call the file


[Gen_r,Gen_c]=size(mpcgen);
[Bus_r,Bus_c]=size(mpcbus);
[Flow_r,Flow_c]=size(mpcbranch);

Line_Limit_Up=4;
Line_Limit_Down=-4;
Line_Range=Line_Limit_Down:1:Line_Limit_Up;


%% ------ File Preparations

outputFileName = 'Frodo_Out'; % Set the file name
xmlFileName = sprintf('%s.xml',outputFileName); % Set the extension of the file
fid = fopen(xmlFileName, 'w'); % Open the file with writing permissions


%% ------ Parameters from m file

maxConstraintArity=15;
name_header='Dimitrios_15';
run radial.m

header_string_1='<?xml version="1.0" encoding="UTF-8"?>';
header_string_2='<instance xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="src/ch/epfl/lia/frodo/algorithms/XCSPschemaJaCoP.xsd">';


%% Header
outputStr1 = sprintf('%s\n %s\n',header_string_1,header_string_2); %
fprintf(fid, '%s', outputStr1);

outputStr2 = sprintf('\t<presentation name="%s" maxConstraintArity="%d" maximize="false"
format="XCSP 2.1_FRODO" />\n',name_header,maxConstraintArity);
fprintf(fid, '%s', outputStr2);


%% ------ Agents
% Agents = Nr of Generators
Nr_Agents=Gen_r;

outputStr3 = sprintf('\t<agents nbAgents="%d">\n',Nr_Agents); %
fprintf(fid, '%s', outputStr3);

for Agent_Index=1:1:Nr_Agents
   outputStr_ag= sprintf(char(strcat('\t\t<agent
name="agent_g',num2str(Agent_Index),'"/>\n')));
    fprintf(fid, '%s', outputStr_ag);
end
outputStr4 = sprintf('\t</agents>\n'); %
fprintf(fid, '%s', outputStr4);
```

```
%% Domains

Nr_Domains=Gen_r+Bus_r+Flow_r;
Nr_Gen=Gen_r;
Nr_Flows=Flow_r;
Nr_Loads=Bus_r;

% Open Domain Tab
outputStr5 = sprintf('\t<domains nbDomains="%d">\n',Nr_Domains); %
fprintf(fid, '%s', outputStr5);

%---------------------- Generators---------------------
for Domain_Index_G=1:1:Nr_Gen;

    Domain_Value_G=mpcgen(Domain_Index_G,4)/10;
    Domain_Value_G_Margin=0:1:Domain_Value_G;
    Domain_Value_Nr=length(Domain_Value_G_Margin);

    outputStr_D_G=sprintf(char(strcat('\t\t<domain
name="dom_g',num2str(Domain_Index_G),'"
nbValues="',num2str(Domain_Value_Nr),'">',num2str(min(Domain_Value_G_Margin)),'..',num2st
r(Domain_Value_G),'</domain>\n')));
    fprintf(fid, '%s',outputStr_D_G);

end

% ---------------------- Flows-------------------------------
for Domain_Index_F=1:1:Nr_Flows;

    outputStr_D_F=sprintf(char(strcat('\t\t<domain
name="dom_f',num2str(mpcbranch(Domain_Index_F,1)),num2str(mpcbranch(Domain_Index_F,2)),'"
nbValues="',num2str(length(Line_Range)),'">',num2str(Line_Limit_Down),'..',num2str(Line_L
imit_Up),'</domain>\n')));
    fprintf(fid, '%s',outputStr_D_F);

end

%----------------------Loads----------------------
for Domain_Index_L=1:1:Nr_Loads;

    Domain_Value_L=mpcbus(Domain_Index_L,3)/10;
    Domain_Value_Nr=length(Domain_Value_L);

    outputStr_D_L=sprintf(char(strcat('\t\t<domain
name="dom_l',num2str(Domain_Index_L),'"
nbValues="',num2str(Domain_Value_Nr),'">',num2str(Domain_Value_L),'</domain>\n')));
    fprintf(fid, '%s',outputStr_D_L);

end

% Close domain tab
outputStr6 = sprintf('\t</domains>\n'); %
fprintf(fid, '%s', outputStr6);


%% Variables

%Nr_Domains=Gen_r+Bus_r+Flow_r;
Nr_Variables=Nr_Domains;
 outputStr_Var=sprintf(char(strcat('\t<variables
nbVariables="',num2str(Nr_Variables),'">\n')));
 fprintf(fid, '%s',outputStr_Var);

%---------------------- Generators---------------------
 for Var_Index_G=1:1:Nr_Gen;
      outputStr_Var_G=sprintf(char(strcat('\t\t<variable name="g',num2str(Var_Index_G),'"
domain="dom_g',num2str(Var_Index_G),'" agent="agent_g',num2str(Var_Index_G),'"/>\n')));
      fprintf(fid, '%s',outputStr_Var_G);
 end
```

```
% --------------------- Flows-------------------------------
for Var_Index_F=1:1:Nr_Flows;
     outputStr_Var_F=sprintf(char(strcat('\t\t<variable
name="f',num2str(mpcbranch(Var_Index_F,1)),num2str(mpcbranch(Var_Index_F,2)),'"
domain="dom_f',num2str(mpcbranch(Var_Index_F,1)),num2str(mpcbranch(Var_Index_F,2)),...
         '" agent="agent_g',num2str(mpcbranch(Var_Index_F,2)),'"/>\n')));
        fprintf(fid, '%s',outputStr_Var_F);
end

%---------------------Loads---------------------
for Var_Index_L=1:1:Nr_Loads;
     outputStr_Var_L=sprintf(char(strcat('\t\t<variable name="l',num2str(Var_Index_L),'"
domain="dom_l',num2str(Var_Index_L),'" agent="agent_g',num2str(Var_Index_L),'"/>\n')));
      fprintf(fid, '%s',outputStr_Var_L);

end

% Close Variable Tab
 outputStr_Var_End=sprintf(char(strcat('\t<variables>\n')));
 fprintf(fid, '%s',outputStr_Var_End);

 %% Functions
  outputStr_Function=sprintf('\t<functions nbFunctions="1">\n\t\t<function name="times"
return="int">\n\t\t\t<parameters>int p1 int
p2</parameters>\n\t\t\t<expression>\n\t\t\t\t<functional>mul(p1,
p2)</functional>\n\t\t\t</expression>\n\t\t</function>\n\t</functions>\n');
    fprintf(fid, '%s',outputStr_Function);

    %%


 Factor_Table=ones(Flow_r,1)



%% Constraints
Nr_Constraints=6;
 outputStr_Con=sprintf(char(strcat('\t<constraints
nbConstraints="',num2str(Nr_Constraints),'">\n')));
 fprintf(fid, '%s',outputStr_Con);

 %---------------------Conservation Constraints - ?F=0 ----------------

 Nr_Buses=Bus_r; % Set number of buses

 for Nr_Buses_Index=1:1:Nr_Buses% Bus


     if mpcbus(Nr_Buses_Index,3)~=0
            Nr_Load=1;
     else
         Nr_Load=0;
     end


Nr_Gen_Bus=length(find(mpcgen(:,1)==Nr_Buses_Index))+length(find(mpcgen(:,2)==Nr_Buses_In
dex));

Nr_Flows_Bus=length(find(mpcbranch(:,1)==Nr_Buses_Index))+length(find(mpcbranch(:,2)==Nr_
Buses_Index));
     Nr_Arity=Nr_Load+Nr_Gen_Bus+Nr_Flows_Bus;

     outputStr_Con_Bus=sprintf(char(strcat('\t\t<constraint
name="conserv_bus',num2str(Nr_Buses_Index),'" arity="',num2str(Nr_Arity),'" scope="')));
     fprintf(fid, '%s',outputStr_Con_Bus);


     if  find(mpcgen(:,1)==Nr_Buses_Index)>0 % Check for zero Generator Output  - den
ekei
```

```
        Gen_Index_Touch=Nr_Buses_Index;
        outputStr_Con_Bus_Name=sprintf(char(strcat('g',num2str(Gen_Index_Touch))));
        fprintf(fid, '%s',outputStr_Con_Bus_Name);

     end



     if mpcbus(Nr_Buses_Index,3)~=0 % Check Loads

      Load_Index_Touch=Nr_Buses_Index;
      outputStr_Con_Load_Name=sprintf(char(strcat(' l',num2str(Load_Index_Touch))));
      fprintf(fid, '%s',outputStr_Con_Load_Name);

     end



 Vres_1=[find(mpcbranch(:,1)==Nr_Buses_Index)];
 Vres_2=[find(mpcbranch(:,2)==Nr_Buses_Index)];
 mpcbranch(Vres_1,1);
 mpcbranch(Vres_1,2);
 mpcbranch(Vres_2,1);
 mpcbranch(Vres_2,2);
 Concatet_Branch=[mpcbranch(Vres_1,1) mpcbranch(Vres_1,2); mpcbranch(Vres_2,1)
mpcbranch(Vres_2,2)];

     for Con_Index_Branch=1:1:((length(Vres_1))+(length(Vres_2)))

     outputStr_Con_Bran_Name=sprintf(char(strcat('
f',num2str(Concatet_Branch(Con_Index_Branch,1)),num2str(Concatet_Branch(Con_Index_Branch,
2)))));
     fprintf(fid, '%s',outputStr_Con_Bran_Name);

     end

     outputStr_telos=sprintf(char(strcat('" reference="global:weightedSum"> \n')));
     fprintf(fid, '%s', outputStr_telos);



     %------------------Second Line--------------------------------------

      outputStr_Parameters_Start=sprintf(char(strcat('\t\t\t <parameters>[')));
      fprintf(fid, '%s', outputStr_Parameters_Start);


       if  find(mpcgen(:,1)==Nr_Buses_Index)>0 % Check for zero Generator Output  - den
ekei

        Gen_Index_Param=Nr_Buses_Index;
        outputStr_Con_Gen_Param=sprintf(char(strcat(' { 1 g',num2str(Gen_Index_Param),'
}')));
        fprintf(fid, '%s',outputStr_Con_Gen_Param);

       end


      if mpcbus(Nr_Buses_Index,3)~=0 % Check Loads

      Load_Index_Param=Nr_Buses_Index;
      outputStr_Con_Load_Param=sprintf(char(strcat(' { 1 l',num2str(Load_Index_Param),'
}')));
      fprintf(fid, '%s',outputStr_Con_Load_Param);

      end



Vres_1=[find(mpcbranch(:,1)==Nr_Buses_Index)];
```

```
 Vres_2=[find(mpcbranch(:,2)==Nr_Buses_Index)];
 f11=mpcbranch(Vres_1,1);
 f12=mpcbranch(Vres_1,2);
 f21=mpcbranch(Vres_2,1);
 f22=mpcbranch(Vres_2,2);
 Factor_Indices=[Vres_1; Vres_2];


 Concatet_Branch=[f11 f12; f21 f22];


     for Con_Index_Branch=1:1:((length(Vres_1))+(length(Vres_2)))

     Flow_Factor=  Factor_Table((Factor_Indices(Con_Index_Branch)))
     outputStr_Con_Bran_Name=sprintf(char(strcat(' { ',num2str(Flow_Factor),'
f',num2str(Concatet_Branch(Con_Index_Branch,1)),num2str(Concatet_Branch(Con_Index_Branch,
2)),'}')));
     fprintf(fid, '%s',outputStr_Con_Bran_Name);
     Factor_Table((Factor_Indices(Con_Index_Branch)))=-1;



     end

     outputStr_telos_Param=sprintf(char(strcat(' ] <eq/> 0</parameters> \n')));
     fprintf(fid, '%s', outputStr_telos_Param);



 end




 %---------------------Cost Constraints ----------------



 for Cost_Index_G=1:1:Nr_Gen;
 outputStr_Con_C=sprintf(char(strcat('\t\t<constraint
name="cost_g',num2str(Cost_Index_G),'" arity="1" scope="g',num2str(Cost_Index_G),'"
reference="times">\n')));
 fprintf(fid, '%s',outputStr_Con_C);

outputStr_Con_C=sprintf(char(strcat('\t\t\t<parameters>',num2str(mpcgencost(Cost_Index_G,
5)),' g',num2str(Cost_Index_G),'</parameters>\n')));
 fprintf(fid, '%s',outputStr_Con_C);
 outputStr_Con_C=sprintf('\t\t</constraint>\n');
 fprintf(fid, '%s',outputStr_Con_C);

 end

 % CLose Contraint tab
outputStr_Con_End=sprintf('\t</constraints>\n');
 fprintf(fid, '%s',outputStr_Con_End);


%% Close Document
 outputStr_Close=sprintf('</instance>\n');
 fprintf(fid, '%s',outputStr_Close);
 fclose(fid); % Close the document

%%

fprintf('Process Finished\n')
```

Appendix D

XCSP file code with sensitivity factors included

```
<?xml version="1.0" encoding="UTF-8"?>

<instancexmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="src/ch/epfl/lia/frodo/algorithms/XCSPschemaJaCoP.xsd">


	<presentation name="Athanasiadis15" maxConstraintArity="21" maximize="false"
format="XCSP 2.1_FRODO" />

	<agents nbAgents="11">

		<agent name="agent_g1" />

		<agent name="agent_g2" />

		<agent name="agent_g3" />

		<agent name="agent_g4" />

		<agent name="agent_g5" />

		<agent name="agent_g6" />

		<agent name="agent_g7" />

		<agent name="agent_g8" />

		<agent name="agent_g9" />

		<agent name="agent_g10" />

		<agent name="agent_g11" />

	</agents>

	<domains nbDomains="32">

		<domain name="dom_g1" nbValues="21">50..70</domain>

		<domain name="dom_g2" nbValues="12">149..160</domain>

		<domain name="dom_g3" nbValues="1">0</domain>

		<domain name="dom_g4" nbValues="1">0</domain>

		<domain name="dom_g5" nbValues="31">100..130</domain>

		<domain name="dom_g6" nbValues="1">0</domain>

		<domain name="dom_g7" nbValues="1">0</domain>

		<domain name="dom_g8" nbValues="1">0</domain>

		<domain name="dom_g9" nbValues="1">0</domain>

		<domain name="dom_g10" nbValues="1">0</domain>
```

```
        <domain name="dom_g11" nbValues="1">0</domain>
        <domain name="dom_f14" nbValues="16">65..80</domain>
        <domain name="dom_f23" nbValues="21">120..140</domain>
        <domain name="dom_f34" nbValues="21">100..120</domain>
        <domain name="dom_f45" nbValues="21">-80..-60</domain>
        <domain name="dom_f48" nbValues="21">230..250</domain>
        <domain name="dom_f87" nbValues="31">20..50</domain>
        <domain name="dom_f76" nbValues="21">10..30</domain>
        <domain name="dom_f710" nbValues="21">10..30</domain>
        <domain name="dom_f811" nbValues="1">0</domain>
        <domain name="dom_f89" nbValues="21">70..90</domain>
        <domain name="dom_l1" nbValues="1">0</domain>
        <domain name="dom_l2" nbValues="1">23</domain>
        <domain name="dom_l3" nbValues="1">17</domain>
        <domain name="dom_l4" nbValues="1">10</domain>
        <domain name="dom_l5" nbValues="1">40</domain>
        <domain name="dom_l6" nbValues="1">17</domain>
        <domain name="dom_l7" nbValues="1">0</domain>
        <domain name="dom_l8" nbValues="1">125</domain>
        <domain name="dom_l9" nbValues="1">84</domain>
        <domain name="dom_l10" nbValues="1">13</domain>
        <domain name="dom_l11" nbValues="1">0</domain>
</domains>
<variables nbVariables="32">
        <variable name="g1" domain="dom_g1" agent="agent_g1" />
        <variable name="g2" domain="dom_g2" agent="agent_g2" />
        <variable name="g3" domain="dom_g3" agent="agent_g3" />
        <variable name="g4" domain="dom_g4" agent="agent_g4" />
        <variable name="g5" domain="dom_g5" agent="agent_g5" />
        <variable name="g6" domain="dom_g6" agent="agent_g6" />
        <variable name="g7" domain="dom_g7" agent="agent_g7" />
        <variable name="g8" domain="dom_g8" agent="agent_g8" />
        <variable name="g9" domain="dom_g9" agent="agent_g9" />
```

```
        <variable name="g10" domain="dom_g10" agent="agent_g10" />
        <variable name="g11" domain="dom_g11" agent="agent_g11" />
        <variable name="f14" domain="dom_f14" agent="agent_g1" />
        <variable name="f23" domain="dom_f23" agent="agent_g2" />
        <variable name="f34" domain="dom_f34" agent="agent_g3" />
        <variable name="f45" domain="dom_f45" agent="agent_g5" />
        <variable name="f48" domain="dom_f48" agent="agent_g8" />
        <variable name="f87" domain="dom_f87" agent="agent_g7" />
        <variable name="f76" domain="dom_f76" agent="agent_g6" />
        <variable name="f89" domain="dom_f89" agent="agent_g9" />
        <variable name="f710" domain="dom_f710" agent="agent_g10" />
        <variable name="f811" domain="dom_f811" agent="agent_g11" />
        <variable name="l1" domain="dom_l1" agent="agent_g1" />
        <variable name="l2" domain="dom_l2" agent="agent_g2" />
        <variable name="l3" domain="dom_l3" agent="agent_g3" />
        <variable name="l4" domain="dom_l4" agent="agent_g4" />
        <variable name="l5" domain="dom_l5" agent="agent_g5" />
        <variable name="l6" domain="dom_l6" agent="agent_g6" />
        <variable name="l7" domain="dom_l7" agent="agent_g7" />
        <variable name="l8" domain="dom_l8" agent="agent_g8" />
        <variable name="l9" domain="dom_l9" agent="agent_g9" />
        <variable name="l10" domain="dom_l10" agent="agent_g10" />
        <variable name="l11" domain="dom_l11" agent="agent_g11" />
</variables>
<functions nbFunctions="1">
        <function name="times" return="int">
                <parameters>int p1 int p2</parameters>
                <expression>
                        <functional>mul(p1, p2)</functional>
                </expression>
        </function>
</functions>
<constraints nbConstraints="15">
```

<constraint name="conserv_g1a" arity="21" scope="f14 g2 l2 g3 l3 g4 l4 g5 l5 g6 l6 g7 l7 g8 l8 g9 l9 g10 l10 g11 l11 " reference="global:weightedSum">

<parameters>[{ 1 f14 } {1 g2} { -1 l2 } {1 g3} { -1 l3 }{1 g4} { -1 l4 }{1 g5} { -1 l5 }{1 g6} { -1 l6 }{1 g7} { -1 l7 }{1 g8} { -1 l8 }{1 g9} { -1 l9 }{1 g10} { -1 l10 }{1 g11} { -1 l11 }] <eq/> 0</parameters>

</constraint>

<constraint name="conserv_g2a" arity="3" scope="f23 g2 l2" reference="global:weightedSum">

<parameters>[{ 1 f23 }{ -1 g2} { 1 l2 }] <eq/> 0</parameters>

</constraint>

<constraint name="conserv_g3a" arity="5" scope="f34 g2 l2 g3 l3" reference="global:weightedSum">

<parameters>[{ 1 f34 }{ -1 g2} { 1 l2 } { -1 g3} { 1 l3 }] <eq/> 0</parameters>

</constraint>

<constraint name="conserv_g4a" arity="3" scope="f45 g5 l5" reference="global:weightedSum">

<parameters>[{ 1 f45 }{ 1 g5} { -1 l5 }] <eq/> 0</parameters>

</constraint>

<constraint name="conserv_g5a" arity="13" scope="f48 g6 l6 g7 l7 g8 l8 g9 l9 g10 l10 g11 l11" reference="global:weightedSum">

<parameters>[{ 1 f48 }{ 1 g6} { -1 l6 }{ 1 g7} { -1 l7 }{ 1 g8} { -1 l8 }{ 1 g9} { -1 l9 }{ 1 g10} { -1 l10 }{ 1 g11} { -1 l11 }] <eq/> 0</parameters>

</constraint>

<constraint name="conserv_g6a" arity="7" scope="f87 g6 l6 g7 l7 g10 l10" reference="global:weightedSum">

<parameters>[{ 1 f87 }{ 1 g6} { -1 l6 }{ 1 g7} { -1 l7 }{ 1 g10} { -1 l10 }}] <eq/> 0</parameters>

```
		</constraint>

		<constraint name="conserv_g7a" arity="3" scope="f76 g6 l6 "
reference="global:weightedSum">
			<parameters>[{ 1 f76 }{ 1 g6} { -1 l6 } ]  <eq/> 0</parameters>

		</constraint>

		<constraint name="conserv_g8a" arity="3" scope="f710 g10 l10 "
reference="global:weightedSum">
			<parameters>[{ 1 f710 }{ 1 g10} { -1 l10 } ]  <eq/> 0</parameters>

		</constraint>

		<constraint name="conserv_g9a" arity="3" scope="f811 g11 l11 "
reference="global:weightedSum">
			<parameters>[{ 1 f811 }{ 1 g11} { -1 l11 } ]  <eq/> 0</parameters>

		</constraint>

		<constraint name="conserv_g10a" arity="3" scope="f89 g9 l9 "
reference="global:weightedSum">
			<parameters>[{ 1 f89 }{ 1 g9} { -1 l9 } ]  <eq/> 0</parameters>

		</constraint>

		<constraint name="conserv_g1" arity="3" scope="g1 l1 f14"
reference="global:weightedSum">
			<parameters>[ { 1 g1 } {-1 l1} { -1 f14 } ]  <eq/> 0</parameters>

		</constraint>

		<constraint name="conserv_g3" arity="4" scope=" g3 l3 f23 f34"
reference="global:weightedSum">
			<parameters>[  { 1 g3 }{ -1 l3 } { -1 f23 } { -1 f34 } ]  <le/>
120</parameters>
		</constraint>
```

```
		<constraint name="conserv_g4" arity="6" scope=" g4 l4 f34 f45 f14 f48" reference="global:weightedSum">
			<parameters>[  { 1 g4 }{ -1 l4 } { -1 f34 } { -1 f45 } { -1 f14 } { -1 f48 } ]  <le/> 120 </parameters>
		</constraint>

		<constraint name="conserv_g7" arity="5" scope="g7 l7 f76 f87 f710" reference="global:weightedSum">
			<parameters>[  { 1 g7 }{ -1 l7 }{ -1 f76 } { -1 f87 } { -1 f710 } ] <le/> 120</parameters>
		</constraint>
		<constraint name="conserv_g8" arity="6" scope="g8 l8 f87 f811 f89 f48" reference="global:weightedSum">
			<parameters>[  { 1 g8 }{ -1 l8 } { -1 f87 } { -1 f811 } { -1 f89 } { -1 f48 } ]  <le/> 120</parameters>
		</constraint>
		<constraint name="cost_g1" arity="1" scope="g1" reference="times">
			<parameters>200 g1</parameters>
		</constraint>
		<constraint name="cost_g2" arity="1" scope="g2" reference="times">
			<parameters>200 g2</parameters>
		</constraint>
		<constraint name="cost_g5" arity="1" scope="g5" reference="times">
			<parameters>200 g5</parameters>
		</constraint>
	</constraints>
</instance>
```

Appendix E

AC Power Flow Management calculations

Matlab calculations code:

```
clc
clear all

V2_Matrix=[0.990:0.001:1.10];
V3_Matrix=[0.990:0.001:1.10];
delta2_Matrix=[-90:0.1:90];
delta3_Matrix=[-90:0.1:90];


PG1_Table=[];
PG2_Table=[];
PG3_Table=[];

PG1_Table_Counter=1;
PG2_Table_Counter=1;
PG3_Table_Counter=1;

QG1_Table=[];
QG2_Table=[];
QG3_Table=[];

QPG1_Table_Counter=1;
QG2_Table_Counter=1;
QG3_Table_Counter=1;




for V2_Counter=1:1:length(V2_Matrix)% start V2 loop
     V2=V2_Matrix(V2_Counter);

    for V3_Counter=1:1:length(V3_Matrix) % start V3 loop
        V3=V3_Matrix(V3_Counter);

        for delta2_Counter=1:1:length(delta2_Matrix) %
start delta 2 loop
            delta2=delta2_Matrix(delta2_Counter);

            for delta3_Counter=1:1:length(delta3_Matrix) %
start delta 31 loop
```

```
                delta3=delta3_Matrix(delta3_Counter);


PG_1=119.99+(70.71*V2*cosd(-delta2-135))+(70.71*V3*cosd(-
delta3-135));
PG_2=20+(70.71*V2*cosd(-delta2-135)-49.99V2*V2);
PG_3=20+(70.71*V3*cosd(-delta3-135)+49.99V3*V3);


QG_1=-49.99+(70.71*V2*sind(-delta2-135))+(70.71*V3*sind(-
delta3-135));
QG_2=70.71*V2*sind(delta2-135)-49.99V2*V2;
QG_3=100+(70.71*V3*sind(delta3-135)+49.99V3*V3);


if PG_1>=0 & PG_1<=20
   PG1_Table(PG1_Table_Counter,1)=PG_1;
   PG1_Table(PG1_Table_Counter,2)=V2;
   PG1_Table(PG1_Table_Counter,3)=V3;
   PG1_Table(PG1_Table_Counter,4)=delta2;
   PG1_Table(PG1_Table_Counter,5)=delta3;

   PG1_Table_Counter=PG1_Table_Counter+1;
end

if PG_2>=0 & PG_2<=20
   PG2_Table(PG2_Table_Counter,1)=PG_2;
   PG2_Table(PG2_Table_Counter,2)=V2;
   PG2_Table(PG2_Table_Counter,3)=V3;
   PG2_Table(PG2_Table_Counter,4)=delta2;
   PG2_Table(PG2_Table_Counter,5)=delta3;

   PG2_Table_Counter=PG2_Table_Counter+1;
end

if PG3_1>=0 & PG3_1<=20
   PG3_Table(PG3_Table_Counter,1)=PG_3;
   PG3_Table(PG3_Table_Counter,2)=V2;
   PG3_Table(PG3_Table_Counter,3)=V3;
   PG3_Table(PG3_Table_Counter,4)=delta2;
   PG3_Table(PG3_Table_Counter,5)=delta3;

   PG3_Table_Counter=PG3_Table_Counter+1;
end

if QG_1>=0 & QG_1<=6
   QG1_Table(QG1_Table_Counter,1)=QG_1;
```

```
    QG1_Table(QG1_Table_Counter,2)=V2;
    QG1_Table(QG1_Table_Counter,3)=V3;
    QG1_Table(QG1_Table_Counter,4)=delta2;
    QG1_Table(QG1_Table_Counter,5)=delta3;

    QG1_Table_Counter=QG1_Table_Counter+1;
end

if QG_2>=0 & QG_2<=6
    QG2_Table(QG2_Table_Counter,1)=QG_2;
    QG2_Table(QG2_Table_Counter,2)=V2;
    QG2_Table(QG2_Table_Counter,3)=V3;
    QG2_Table(QG2_Table_Counter,4)=delta2;
    QG2_Table(QG2_Table_Counter,5)=delta3;

    QG2_Table_Counter=QG2_Table_Counter+1;
end

if QG_3>=0 & QG_3<=6
    QG3_Table(QG3_Table_Counter,1)=QG_3;
    QG3_Table(QG3_Table_Counter,2)=V2;
    QG3_Table(QG3_Table_Counter,3)=V3;
    QG3_Table(QG3_Table_Counter,4)=delta2;
    QG3_Table(QG3_Table_Counter,5)=delta3;

    QG3_Table_Counter=QG3_Table_Counter+1;
end


            end % finish delta 2 loop

        end % finish delta 3 loop

    end % finish V3 loop

end % finish V2 loop

PG1_Table=sortrows(PG1_Table);
PG2_Table=sortrows(PG2_Table);
PG3_Table=sortrows(PG3_Table);
QG1_Table=sortrows(QPG1_Table);
QG2_Table=sortrows(QG2_Table);
QG3_Table=sortrows(QG3_Table);
```

Results from P_g's calculations (only 100 values inside domains range):

Pg1	V2	V3	delta2	delta3	Pg2	V2	delta2	Pg3	V3	delta3
19.99528	1.093	1.093	9	-16	0.011835	1.094	76	16.41149	1.093	-10
19.99528	1.093	1.093	81	-16	0.084745	1.093	76	16.4629	1.094	-80
19.99542	1.093	1.093	-27	35	0.157554	1.092	76	16.4629	1.094	-10
19.99542	1.093	1.093	-27	55	0.230264	1.091	76	16.51441	1.095	-80
19.99542	1.093	1.093	35	-27	0.302874	1.09	76	16.51441	1.095	-10
19.99542	1.093	1.093	55	-27	0.72975	1.1	77	16.56601	1.096	-80
19.99545	1.093	1.099	44	-28	0.802208	1.099	77	16.56601	1.096	-10
19.99545	1.093	1.099	46	-28	0.874565	1.098	77	16.61772	1.097	-80
19.99545	1.099	1.093	-28	44	0.946823	1.097	77	16.61772	1.097	-10
19.99545	1.099	1.093	-28	46	1.01898	1.096	77	16.66952	1.098	-80
19.99549	1.09	1.098	9	-16	1.091038	1.095	77	16.66952	1.098	-10
19.99549	1.09	1.098	81	-16	1.162995	1.094	77	16.72143	1.099	-80
19.99549	1.098	1.09	-16	9	1.234853	1.093	77	16.72143	1.099	-10
19.99549	1.098	1.09	-16	81	1.30661	1.092	77	16.77343	1.1	-80
19.99566	1.094	1.095	45	-28	1.378268	1.091	77	16.77343	1.1	-10
19.99566	1.095	1.094	-28	45	1.449825	1.09	77	17.03902	1.09	-81
19.99601	1.097	1.099	40	-28	1.874669	1.1	78	17.03902	1.09	-9
19.99601	1.097	1.099	50	-28	1.946085	1.099	78	17.09085	1.091	-81
19.99601	1.099	1.097	-28	40	2.017402	1.098	78	17.09085	1.091	-9
19.99601	1.099	1.097	-28	50	2.088619	1.097	78	17.14277	1.092	-81
19.99606	1.091	1.093	-9	0	2.159735	1.096	78	17.14277	1.092	-9
19.99606	1.091	1.093	-9	90	2.230752	1.095	78	17.19479	1.093	-81
19.99606	1.093	1.091	0	-9	2.301669	1.094	78	17.19479	1.093	-9
19.99606	1.093	1.091	90	-9	2.372485	1.093	78	17.24691	1.094	-81
19.99667	1.1	1.1	-5	-5	2.443202	1.092	78	17.24691	1.094	-9
19.99669	1.09	1.092	-13	5	2.513819	1.091	78	17.29914	1.095	-81
19.99669	1.09	1.092	-13	85	2.584336	1.09	78	17.29914	1.095	-9
19.99669	1.092	1.09	5	-13	3.006683	1.1	79	17.35146	1.096	-81
19.99669	1.092	1.09	85	-13	3.077071	1.099	79	17.35146	1.096	-9
19.99707	1.092	1.1	-20	15	3.147358	1.098	79	17.40388	1.097	-81
19.99707	1.092	1.1	-20	75	3.217546	1.097	79	17.40388	1.097	-9
19.99707	1.1	1.092	15	-20	3.287633	1.096	79	17.4564	1.098	-81
19.99707	1.1	1.092	75	-20	3.357621	1.095	79	17.4564	1.098	-9
19.9975	1.095	1.1	-28	39	3.427508	1.094	79	17.50903	1.099	-81
19.9975	1.095	1.1	-28	51	3.497296	1.093	79	17.50903	1.099	-9
19.9975	1.1	1.095	39	-28	3.566984	1.092	79	17.56175	1.1	-81
19.9975	1.1	1.095	51	-28	3.636571	1.091	79	17.56175	1.1	-9
19.99761	1.095	1.09	-26	31	3.706059	1.09	79	17.83917	1.09	-82
19.99761	1.095	1.09	-26	59	4.125449	1.1	80	17.83917	1.09	-8

19.99761	1.09	1.095	31	-26	4.194819	1.099	80	17.89172	1.091	-82
19.99761	1.09	1.095	59	-26	4.26409	1.098	80	17.89172	1.091	-8
19.99807	1.091	1.092	31	-26	4.33326	1.097	80	17.94438	1.092	-82
19.99807	1.091	1.092	59	-26	4.402331	1.096	80	17.94438	1.092	-8
19.99807	1.092	1.091	-26	31	4.471301	1.095	80	17.99714	1.093	-82
19.99807	1.092	1.091	-26	59	4.540172	1.094	80	17.99714	1.093	-8
19.99831	1.1	1.094	-28	42	4.608942	1.093	80	18.04999	1.094	-82
19.99831	1.1	1.094	-28	48	4.677613	1.092	80	18.04999	1.094	-8
19.99831	1.094	1.1	42	-28	4.746183	1.091	80	18.10295	1.095	-82
19.99831	1.094	1.1	48	-28	4.814654	1.09	80	18.10295	1.095	-8
19.99835	1.099	1.091	17	-21	5.230625	1.1	81	18.15601	1.096	-82
19.99835	1.099	1.091	73	-21	5.29899	1.099	81	18.15601	1.096	-8
19.99835	1.091	1.099	-21	17	5.367256	1.098	81	18.20916	1.097	-82
19.99835	1.091	1.099	-21	73	5.435422	1.097	81	18.20916	1.097	-8
19.9984	1.096	1.094	-24	24	5.503488	1.096	81	18.26242	1.098	-82
19.9984	1.096	1.094	-24	66	5.571453	1.095	81	18.26242	1.098	-8
19.9984	1.094	1.096	24	-24	5.639319	1.094	81	18.31577	1.099	-82
19.9984	1.094	1.096	66	-24	5.707085	1.093	81	18.31577	1.099	-8
19.9985	1.1	1.1	23	-24	5.774751	1.092	81	18.36923	1.1	-82
19.9985	1.1	1.1	67	-24	5.842317	1.091	81	18.36923	1.1	-8
19.9985	1.1	1.1	-24	23	5.909783	1.09	81	18.65806	1.09	-83
19.9985	1.1	1.1	-24	67	6.321874	1.1	82	18.65806	1.09	-7
19.99854	1.094	1.099	-20	15	6.389248	1.099	82	18.71136	1.091	-83
19.99854	1.094	1.099	-20	75	6.456522	1.098	82	18.71136	1.091	-7
19.99854	1.099	1.094	15	-20	6.523695	1.097	82	18.76477	1.092	-83
19.99854	1.099	1.094	75	-20	6.590769	1.096	82	18.76477	1.092	-7
19.99877	1.095	1.092	44	-28	6.657743	1.095	82	18.81828	1.093	-83
19.99877	1.095	1.092	46	-28	6.724617	1.094	82	18.81828	1.093	-7
19.99877	1.092	1.095	-28	44	6.79139	1.093	82	18.87189	1.094	-83
19.99877	1.092	1.095	-28	46	6.858064	1.092	82	18.87189	1.094	-7
19.99885	1.092	1.095	-18	12	6.924638	1.091	82	18.9256	1.095	-83
19.99885	1.092	1.095	-18	78	6.991112	1.09	82	18.9256	1.095	-7
19.99885	1.095	1.092	12	-18	7.398865	1.1	83	18.9794	1.096	-83
19.99885	1.095	1.092	78	-18	7.46526	1.099	83	18.9794	1.096	-7
19.99891	1.098	1.096	-22	19	7.531554	1.098	83	19.03331	1.097	-83
19.99891	1.098	1.096	-22	71	7.597749	1.097	83	19.03331	1.097	-7
19.99891	1.096	1.098	19	-22	7.663844	1.096	83	19.08732	1.098	-83
19.99891	1.096	1.098	71	-22	7.729838	1.095	83	19.08732	1.098	-7
19.99936	1.098	1.094	-11	2	7.795733	1.094	83	19.14143	1.099	-83
19.99936	1.098	1.094	-11	88	7.861528	1.093	83	19.14143	1.099	-7
19.99936	1.094	1.098	2	-11	7.927222	1.092	83	19.19564	1.1	-83

19.99936	1.094	1.098	88	-11	7.992817	1.091	83	19.19564	1.1	-7
19.99951	1.096	1.092	-27	35	8.058312	1.09	83	19.49545	1.09	-84
19.99951	1.096	1.092	-27	55	8.461269	1.1	84	19.49545	1.09	-6
19.99951	1.092	1.096	35	-27	8.526698	1.099	84	19.54953	1.091	-84
19.99951	1.092	1.096	55	-27	8.592027	1.098	84	19.54953	1.091	-6
19.99957	1.092	1.098	27	-25	8.657256	1.097	84	19.6037	1.092	-84
19.99957	1.092	1.098	63	-25	8.722385	1.096	84	19.6037	1.092	-6
19.99957	1.098	1.092	-25	27	8.787413	1.095	84	19.65798	1.093	-84
19.99957	1.098	1.092	-25	63	8.852342	1.094	84	19.65798	1.093	-6
19.99967	1.094	1.091	-23	22	8.917171	1.093	84	19.71235	1.094	-84
19.99967	1.094	1.091	-23	68	8.9819	1.092	84	19.71235	1.094	-6
19.99967	1.091	1.094	22	-23	9.046529	1.091	84	19.76683	1.095	-84
19.99967	1.091	1.094	68	-23	9.111058	1.09	84	19.76683	1.095	-6
19.99973	1.097	1.098	10	-17	9.508763	1.1	85	19.82141	1.096	-84
19.99973	1.097	1.098	80	-17	9.57324	1.099	85	19.82141	1.096	-6
19.99973	1.098	1.097	-17	10	9.637616	1.098	85	19.87608	1.097	-84
19.99973	1.098	1.097	-17	80	9.701893	1.097	85	19.87608	1.097	-6
19.99985	1.099	1.092	40	-28	9.766069	1.096	85	19.93086	1.098	-84
19.99985	1.099	1.092	50	-28	9.830146	1.095	85	19.93086	1.098	-6
19.99985	1.092	1.099	-28	40	9.894122	1.094	85	19.98573	1.099	-84
19.99985	1.092	1.099	-28	50	9.957999	1.093	85	19.98573	1.099	-6

Appendix F

AC Power Flow Management in Frodo

The main problem with the Frodo software tool is that it has not been designed to solve trigonometric functions. In that situation, all the complex equations need to be reformed in order to be able to be read by the software. For instance, if we consider the following equation:

$$\sum P_k = \sum V_i V_j Y_{ij} \cos(\delta_i - \delta_j - \theta_{ij}) \quad \text{(A.1)}$$

which is a generalisation of the power equations, then it can be formulated by putting all the terms involving variables on the left of the equal sign and all the constants on the right side such as:

$$\sum P_k - \sum V_i V_j Y_{ij} \cos(\delta_i - \delta_j - \theta_{ij}) = C \quad \text{(A.2)}$$

By introducing the auxiliary variables M_{ij} , Equation A.2 can be reformulated into a simple weighted sum over variables:

$$M_{ij} = V_i V_j Y_{ij} \cos(\delta_i - \delta_j - \theta_{ij}) \quad \text{(A.3)}$$

$$\sum P_k - \sum M_{ij} = \text{C} \quad \text{(A.4)}$$

Equation A.4 can then be expressed in XCSP using a weighted Sum constraint. For instance, assuming there are two variables P_1 and P_2, and two variables M_{12} and M_{13}, and assuming that C = 0:

```
<constraint name="sumCons" arity="4" scope="P1 P2 M12 M13"
reference="global:weightedSum">
```

```
<parameters>
[ { 1 P1 } { 1 P2 } { -1 M12 } { -1 M13} ] <eq/> 0
</parameters>
</constraint>
```

Equation A.3 involves a trigonometric function that is not supported by the XCSP format and must, therefore, be expressed as an extensional constraint. As such extensional constraints are expensive, and their complexity and computational cost increase with the number of variables they involve, it is recommended to reformulate Equation A.3 so that it is expressed over only two auxiliary variables K_{ij} and Δ_{ij}.

$$K_{ij} = Y_{ij} cos(\Delta_{ij} - \theta_{ij}) \quad \text{(A.5)}$$

$$K_{ij} = \frac{M_{ij}}{V_i V_j} \quad \text{(A.6)}$$

$$\Delta_{ij} = \delta_i - \delta_j \quad \text{(A.7)}$$

If it is assumed that the domain of the angle variables δ_i and δ_j is approximated by {0, 90, 180, 270} (in degrees); based on Equation A.7, the variable Δ_{ij} therefore has a domain {-270, -180, -90, 0, 90, 180, 270}, which corresponds to the following XCSP formulation for the domains:

```
<domain name="angle" nbValues="4">0 90 180 270</domain>
<domain name="angMinAng" nbValues="7">-270 -180 -90 0 90
180 270</domain>
```

and for the variables:

```
<variable name="di" domain="angle" agent="agent_gi"/>
<variable name="dj" domain="angle" agent="agent_gj"/>
<variable name="Dij" domain="angMinAng" agent="agent_gi"/>
```

assuming that the ownership of the auxiliary variable Δ_{ij} has been arbitrarily assigned to agent agent_g_i. Based on these domains, Equation A.5 can be expressed as the following binary, extensional constraint:

```
<constraint name="cosIJcons" arity="2" scope="Kij Dij"
reference="cosIJrel"/>
```

referencing the following relation, assuming for simplification that Y_{ij} = θ_{ij}= 0:

```
<relation name="cosIJrel" arity="2" nbTuples="7"
semantics="supports">
0 -270 | -1 -180 | 0 -90 | 1 0 | 0 90 | -1 180 | 0 270
</relation>
```

Equation A.6 can be reformulated as follows:

$$K_{ij}V_iV_j = M_{ij} \tag{A.8}$$

which can be expressed in XCSP as an intentional constraint:

```
<constraint name="xijCons" arity="4" scope="Kij Vi Vj Mij"
reference="xPred"/>
```

referencing the following predicate:

```
<predicate name="xPred">
<parameters> int K int Vx int Vy int M </parameters>
<expression>
<functional>
eq(mul(mul(K, Vx), Vy), M)
</functional>
</expression>
</predicate>
```

Finally, Equation A.7 can also be expressed as an intentional constraint:

```
<constraint name="DijCons" arity="3" scope="Dij di dj"
reference="Dpred"/>
```

referencing the following predicate:

```
<predicate name="Dpred">
<parameters>int dx int dy int D</parameters>
<expression>
<functional>
eq(add(dy, D), dx)
</functional>
</expression>
</predicate>
```

It is assumed that the domain of the angle variables δ_i and δ_j is approximated by {0, 90, 180, 270} (in degrees). In power systems, the domain of the angle is between -30 to 30. However, due to FRODO limitations, specific values were chosen in order to be able to reach the solution. FRODO cannot handle multiple calculations of large integers and the values were selected to prove the concept. FRODO is only able to deal with integer values and not real values. This means that all real values must be transformed to integers with the appropriate multiplications or divisions regarding the domains of the values. The above transformations are needed in order to create the XSCP file. However, all these transformations have a significant impact to the domains of the variables. The domains are becoming huge and FRODO is not able to perform any calculations due to the arithmetic overflow. Even if the domain values are divided by 10 or 100 in order to have smaller values the software is not able to perform any optimisation. FRODO was initially created in order to optimise smaller problems such as schedule meeting hence the AC power flow management is not able to be performed by the software. The biggest issue regards the use of integer values. While for DC power flow management only integer values are considered, for AC power flow the use of voltages and angles cannot be

considered by using integer values. For this reason, FRODO is not the appropriate tool for DCOP algorithms regarding the AC power flow management.

References

[AD97] A. Armstrong and E. Durfee. Dynamic Prioritization of Complex Agents in Distributed Constraint Satisfaction Problems. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 620-625, 1997.

[AEP14] AEP OHIO. gridSMART Demonstration Project: A Community-Based Approach to leading the Nation in Smart Energy use, 2014. URL: https://www.smartgrid.gov/sites/default/files/doc/files/AEP%20Ohio_DE-OE-0000193_Final%20Technical%20Report_06-23-2014.pdf

[AFG13] S. Anand, B. G. Fernandes, and J. Guerrero. Distributed control to ensure proportional load sharing and improve voltage regulation in low-voltage DC microgrids. In *IEEE Trans. Power Electron*, vol. 28, no. 4, pp. 1900–1913, Apr. 2013.

[AM00] S. M. Aji and R. J. McEliece. Generalized Distributive Law. In *IEEE Transactions on Information Theory*, 46(2):325 – 343, 2000.

[Arn11] G. W. Arnold. Challenges and opportunities in smart grid: A position article. In *Proc. IEEE*, vol. 99, no. 6, pp. 922–927, Jun. 2011.

[BAV+11] R. Belhomme, R. Cerero Real de Asua, G. Valtorta and P. Eyrolles. The ADDRESS project: Developing Active Demand in smart power systems integrating renewables. In *IEEE Power and Energy Society General Meeting*, Detroit, 2011.

[BBE01] U. Borup, F. Blaabjerg, and P. Enjeti. Sharing of nonlinear load in parallel-connected three-phase converters. In *IEEE Trans. Ind. Appl.*, vol. 37, no. 6, pp. 1817–1823, Nov. 2001.

[BBV+07] K. D. Brabandere, B. Bolsens, J. Van den Keybus, A. Woyte, J. Driesen, R. Belmans. A voltage and frequency droop control method for parallel inverters. In *IEEE Trans. Power Electron.*, vol. 22, no. 4, pp. 1107–1115, Jul. 2007.

[BCF08] F. Bastiao, P. Cruz, and R. Fiteiro. Impact of distributed generation on distribution networks. In *Proc. 5th Int. Conf. Eur. Electr. Market (EEM)*, pp. 1–6, May 2008.

[BCP+03] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. JADE—A white paper. . In *search of innovation*, 2003. URL: http://jade.tilab.com/papers/2003/-WhitePaperJADEEXP.pdf

[BDA+10] A. Beccuti, T. Demiray, G. Andersson, and M. Morari. A Lagrangian decomposition algorithm for optimal emergency voltage control. In *IEEE Trans. Power Syst.*, vol. 25, no. 4, pp. 1769–1779, Nov. 2010.

[BDF+05] R. Béjar, C. Domshlak, C. Fernández, C. Gomes, B. Krishnamachari, B. Selman and M. Valls. Sensor networks and distributed CSP: communication, computation and complexity. In *Artificial Intelligence*, 161(1–2):117–147, 2005.

[Bia96] J. Bialek. Tracing the Flow of Electricity. In *IEEE Proceedings, Generation, Transmission and Distribution*, Vol. 143, No. 4, P313-320, July 1996.

[BM07] M. E. Baran and I. M. El-Markabi. A Multiagent-Based Dispatching Scheme for Distributed Generators for Voltage Support on Distribution Feeders. In *IEEE Transactions on Power Systems*, vol. 22, no. 1, pp. 52-59, Feb. 2007. doi: 10.1109/TPWRS.2006.889140.

[CAF+10] R. A. F. Currie, G. W. Ault, C. E. T. Foote, N. M. McNeill and A. K. Gooding. Smarter Ways to Provide Grid Connections for Renewable Generators. In *IEEE Power and Energy Society General Meeting*, 2010.

[CAT05] R. A. F. Currie, G. W. Ault and D. Telford. Facilitate Generation Connections on Orkney by Automatic Distribution Network Management. In *DTI Project Final Rep.*, Contract: K/EL/00311/00/00, URN: 05/514, 2005.

[CAV+08] R. Belhomme, R. Cerero Real de Asua, G. Valtorta, A. Paice, F. Bouffard, R. Rooth, A. Losi. ADDRESS – Active demand for the smart grids of the future. In *Proceedings CIRED Seminar 2008: Smart Grids for Distribution*, Paper No. 0080, Frankfurt, Germany, 2008.

[CC14] P. T. Cheng and Y. H. Chen. Design and Implementation of Solid-State Transfer Switches for Power Quality Enhancement. In *Proceedings Power Electronics Specialist Conference*, pp.1108-1114, 2004

[CCM+13] A. J. Conejo, E. Castillo, R. Mínguez, and R. García-Bertrand. Decomposition Techniques in Mathematical Programming: Engineering and Science Applications. New York, NY, USA: Springer, 2006.

[CDK91] Z. Collin, R. Dechter, and S. Katz. On the Feasibility of Distributed Constraint Satisfaction. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, IJCAI91*, pages 318–324, 1991.

[CNM12] N. Cai, N. T. T. Nga, and J. Mitra. Economic dispatch in microgrids using multi-agent system. In *Proc. North Amer. Power Symp. (NAPS)*, Champaign, IL, USA, pp. 1–5, 2012.

[Coh80] G. Cohen. Auxiliary problem principle and decomposition of optimisation problems. In Journal of Optimisation Theory and Applications. Volume 32, Issue 3, pp 277-305, 1980.

[CS06] Y. J. Cheng and E. Sng. A novel communication strategy for decentralised control of paralleled multi-inverter systems. In *IEEE Trans. Power Electron.*, vol. 21, no. 1, pp. 148–156, Jan. 2006.

[CSY06] A. Chechetka and K. Sycara. No-commitment branch and bound search for distributed constraint optimisation. In *Proceedings of Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1427—1429, 2006.

[DAF+12] M. J. Dolan, G. W. Ault, D. F. Frame, S. Gill, I. Kockar, O. AnayaLara, S. Galloway, B. O'Neill, C. Foote, A. Svalovs, and K. Jackson. Northern isles

new energy solutions: Active network management stability limits. In *ISGT Europe*, Berlin, 2012.

[DDA+09] M. Dolan, E. M. Davidson, G. W. Ault and J. R. McDonald. Techniques for managing power flows in active distribution networks within thermal constraints. In *CIRED*, 2009.

[DDA+10] E. Davidson, M. Dolan, G. Ault, and S. McArthur. Aura-nms: An autonomous regional active network management system for edf energy and sp energy networks. In *IEEE Power and Energy Society General Meeting*, Minneapolis, MN, pp. 1–6, 2010.

[DDC09] M. J. Dolan, E. M. Davidson, F. Coffele, G. W. Ault, I. Kockar and J. R. McDonald. Using optimal power flow for management of power flows with active distribution networks within thermal constraints. In *University Power Engineering Conference*, Glasgow, 2009.

[DDK+14] M. J. Dolan, E. M. Davidson, E.M., Kockar, I., Ault, G.W., McArthur, S.D.J., "Reducing Distributed Generator Curtailment Through Active Power Flow Management. In *IEEE Transactions on Smart Grid*, pp. 149-157, 2014.

[DDM+09] E. M. Davidson, M. J. Dolan, S. D. J. McArthur, G. W. Ault. The use of constraint programming for the autonomous management of power flows. In *Intelligent Systems Applications to Power Systems*, Curitiba, 2009.

[DGS11] J. Duncan Glover, M.S.Sarma & Thomas J. Overbye. Power system analysis and design. 5th Edition, 2011.

[DH04] A. Dimeas and N.D Hatziargyriou. A multi-agent system for microgrids. In *IEEE Power Engineering Society General Meeting*, Vol. 1, pp. 55-58, 2004.

[DL13] C.-X. Dou and B. Liu. Multi-agent based hierarchical hybrid control for smart microgrid. In *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 771–778, Jan. 2013.

[DM07] E. M. Davidson and S. D. J. McArthur. Exploiting Multi-Agent System Technology within an Autonomous Regional Active Network Management System. In *Intelligent Systems Applications to Power Systems* (ISAP07), 2007.

[DMD+09] E. M. Davidson, S. D. J. McArthur, M. J. Dolan and J. R. McDonald. Exploiting Intelligent Systems Techniques within an Autonomous Regional Active Network Management System. In *Proceedings IEEE Power and Energy Society General Meeting*, Calgary, Canada, July 26- 30, 2009.

[DMM+09] E. M. Davidson, S. D. J. McArthur, J. R. McDonald, P. C. Taylor. An architecture for flexible and autonomous network management systems. In *CIRED*, June 2009.

[DPS+12] J. Dai, Y. Phulpin, A. Sarlette, and D. Ernst. Coordinated primary frequency control among non-synchronous systems connected by a multi-terminal high-voltage direct current grid. In *IEEE Gen. Transmiss. Distrib.*, vol. 6, no. 2, pp. 99–108, Feb. 2012.

[DTI04] Department of Trade and Industry, "Network management systems for active distribution networks—A feasibility study," DTI Distributed Generation Programme (Contractor: SP PowerSystems LTD), Contract Number: K/EL/00310/00/00, URN Number: 04/1361, 2004 URL: http://webarchive.nationalarchives.gov.uk/+/http://www.berr.gov.uk/files/file15177.pdf

[EBB+07] Redouane Ezzahir, Christian Bessiere, Mustapha Belaissaoui, and El Houssine Bouyakhf. DisChoco: A platform for distributed constraint programming. In 8th International Workshop on Distributed Constraint Reasoning, India, 8 January, 2007.

[EC12] European Commission. Grid4EU: A large-scale demonstration project of advanced smart grids solutions with wide replication and scalability potential for Europe, 2012. URL: http://www.grid4eu.com

[EC13] European Commission. IDE4L: Ideal grid for all, 2013. URL: http://www.ide4l.eu/

[ECO15] European Commission. S3C. Final summary report of S3C project, 2015. URL: http://s3c-project.eu/Down.asp?Name={BAPTMTOQSO-3162016131028-ZCIXPVUTIU}.pdf

[ECO13] European Commission. DISCERN: Distributed Intelligence for Cost-effective and Reliable Solutions, 2013. URL: http://www.discern.eu-/index.html

[ES10] G. Ezhilarasi and K. Swarup. Distributed load flow using partitioning and equivalencing of power networks. In *Proc. 16th National Power Syst. Conf.*, Hyderabad, India, Dec. 15–17, 2010, pp. 335–340

[EU09] European Union. DIRECTIVE 2009/72/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 13 July 2009 concerning common rules for the internal market in electricity and repealing Directive 2003/54/EC. 2009. URL: http://eurlex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:211:0055:003:EN:PDF

[EN10] European Network of Transmission System Operators for Electricity. European Grid: Towards 2020 Challenges and Beyond. URL: https://www.entsoe.eu/publications/research-and-development reports/Documents/121209_R_D_Plan_2011.pdf

[Far08] Hassan Farhangi. Intelligent Micro Grid Research at BCIT, 2008. URL: http://www.bcit.ca/files/appliedresearch/pdf/intelligentpowergridbcit_epec08.pdf

[FBE13] M. Falahi, K. Butler-Purry, and M. Ehsani. Dynamic reactive power control of islanded microgrids. In *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 3649–3657, Nov. 2013.

[FCC+11] C. E. T. Foote, R. A. F. Currie, F. Clifton, S. Reid. The Shetland Islands Smart Grid. In *CIRED 21st International Conference on Electricity Distribution,* Frankfurt, 2011.

[FD98] Daniel Frost and Rina Dechter. Optimizing with constraints: A case study in scheduling maintenance of electric power units. In *Fifth International Symposium on Artificial Intelligence and Mathematics,* 1998.

[FIPA02] Foundation for Intelligent Physical Agents (FIPA). Agent Management Specification. 2002. URL: http://www.fipa.org/specs/fipa00023/-SC00023J.html

[FLP08] B. Faltings, T. Léauté and A. Petcu. Privacy Guarantees through Distributed Constraint Satisfaction. In *International Conference on Web Intelligence and Intelligent Agent Technology*, pages 350–358, 2008.

[Fre85] E. C. Freuder. A sufficient condition for backtrack-bounded search. In *Journal of the ACM*, 32(14):755–761, 1985.

[FRP+08] A. Farinelli, A. Rogers, A. Petcu and N.R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In, *Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, Estoril, Portugal, 639-646, 2008.

[FRPN+08] A. Farinelli, A. Rogers, A. Petcu, and N. Jennings. Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In *Proc. of AAMAS*, pages 639–646, 2008.

[Fro06] FRODO: An open-source framework for DCOP. 2006. URL: http://liawww.epfl.ch/frodo/

[GA12] R. Grunbaum and P. Andersson. FACTS-Intelligent Solutions for Meeting Challenges in Power Transmission. In *IEEE PES PowerAfrica*, July 2012.

[GA14] S. Georgiopoulos, G. W. Ault. Flexible Plug and Play Project: Key considerations for network wide roll out of active network management for distributed generation connections. In *CIRED 23nd International Conference on Electricity Distribution*, Rome, 2014.

[Gal07] Galvin Electricity Initiative. Master controller requirements specification for perfect power systems (as outlined in the Galvin electricity initiative). White Paper, EPRI, Palo Alto, CA, USA, and The Galvin Project, Inc., Chicago, IL, USA, Feb. 2007.

[GCL+13] J. M. Guerrero, M. Chandorkar, T.-L. Lee, and P. C. Loh. Advanced control architectures for intelligent microgrids—Part I: Decentralised and hierarchical control. In *IEEE Trans. Ind. Electron.*, vol. 60, no. 4, pp. 1254– 1262, Apr. 2013.

[GE14] General Electric. The rise of distributed power. 2014. URL: https://www.ge.com/sites/default/files/2014%2002%20Rise%20of%20Distributed%20Power.pdf .

[GM07] Tal Grinshpoun and Ammon Meisels. CompAPO: A complete version of the APO algorithm. In *IEEE Computer Society*, pp. 370-376, 2007.

[GM08] T. Grinshpoun and A. Meisels. Completeness and performance of the APO algorithm. In *Journal of Artificial Intelligence Research (JAIR)*, 33, 223–258, 2008.

[Gre09] T. Green. AuRA-NMS: A substation automation project for a potential smart grid. Substation Technology, Birmingham, 2009.

[GV92] K. Ghedira and G. Verfaillie. A Multi-Agent Model for the Resource Allocation Problem: A Reactive Approach. In *Proceedings of the Tenth European Conference on Artificial Intelligence*, pp. 252-254, 1992.

[GVM+11] J. M. Guerrero, J. C. Vasquez, J. Matas, L. G. de Vicuña and M. Castilla. Hierarchical control of droop-controlled AC and DC microgrids—A general approach towards standardization. In *IEEE Trans. Ind. Electron.*, vol. 58, no. 1, pp. 158–172, Jan. 2011.

[HAI+07] N. Hatziargyriou, H. Asona, R. Iravani, and C. Marnay. Microgrids. In *IEEE Power Energy Mag.*, vol. 5, no. 4, pp. 78–94, Jul./Aug. 2007.

[Hir89] R.F. Hirsh. Technology and Transformation in the American Electric Utility Industry, Cambridge University Press: New York, 1989.

[HSH10] B. Hussain, S. Sharkh, S. Hussain, "Impact studies of distributed generation on power quality and protection setup of an existing distribution network", *Proc. Int. SPEEDAM*, pp. 1243-1246, 2010.

[IEA02] International Energy Agency. Distributed Generation in a liberalized energy market. Jouve: France, 2002.

[IEA03] International Energy Agency. Distributed Generation and Renewables Outlook 2030. 2003. URL: http://www.iea.org/textbase/work/2004/distgen /Birol.pdf.

[IEEE00] IEEE, The authoritative dictionary of IEEE standards terms. New York: Standards Information Network, 2000.

[JAK+00] N. Jenkins, R. Allan, D. Kirschen, G. Strbac. *Embedded Generation.* Institution of Electrical Engineers (IEE), London, 2000.

[KA13] Laura Kane, Graham Ault. The cost of active network management schemes at distribution level. In *EWEA Annual Wind Energy*, 2013.

[KDI10] H. Karimi, E. J. Davison, and R. Iravani. Multivariable servomechanism controller for autonomous operation of a distributed generation unit: Design and performance evaluation. In *IEEE Trans. Power Systems*, vol. 25, no. 2, pp. 853–865, May 2010.

[KEM2012] KEMA. Smart Grid Strategic Review: The Orkney Islands Active Network Management Scheme. 8th March 2012

[KK98] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. In *SIAM J. Sci. Comput.*, vol. 20, pp. 359–392, Dec. 1998.

[KP07] H. Katagishi and J. P. Pearce. KOPT: Distributed DCOP algorithm for arbitrary k-optima with monotonically increasing utility. In *Ninth DCR Workshop*, 2007.

[Las02] R. H. Lasseter. Microgrids. In *Proc. IEEE Power Eng. Soc. Winter Meet*, pp. 305–308, 2002.

[La11] R. Lasseter, "Smart distribution: Coupled microgrids," *Proc. IEEE*, vol. 99, no. 6, pp. 1074–1082, June 2011.

[LF09] T. Léauté, B. Faltings. E[DPOP]: Distributed constraint optimisation under stochastic uncertainty using collaborative sampling. In *Distributed Constraint Reasoning Workshop*, 2009.

[LKS+08] R. Lass, J. Kopena, E. Sultanik, D. Nguyen, C. Dugan, P. Modi, and W. Regli. Coordination of first responders under communication and resource constraints. In *Proc. of AAMAS*, pages 1409–1413, 2008.

[LL08] C.-H. Lin and S.-Y. Lin. Distributed optimal power flow with discrete control variables of large distributed power systems. In *IEEE Trans. Power Syst.*, vol. 23, no. 2, pp. 1383–1392, Aug. 2008.

[LMM05] J. A. P. Lopes, C. L. Moreira, and A. G. Madureira. Defining control strategies for analysing microgrids islanded operation. In *Proc. IEEE Russia Power Tech*, pp. 1–7, 2005.

[LS96] J.-S. Liu and K. P. Sycara. Multiagent Coordination in Tightly Coupled Task Scheduling. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pp. 181-188, 1996.

[LSK11] T. Logenthiran, D. Srinivasan and A. M. Khambadkone. Multi-agent system for energy resource scheduling of integrated microgrids in a distributed system. In *Elect. Power Syst. Res.*, vol. 81, no. 1, pp. 138–148, 2011. URL: http://www.sciencedirect.com/science/article/pii/S0378779-610001823

[LVL04] Y. Li, D. M. Vilathgamuwa, and P. C. Loh. Design, analysis, and realtime testing of a controller for multibus microgrid system. In *IEEE Trans. Power Electron.*, vol. 19, no. 5, pp. 1195–1204, Sep. 2004.

[Mac92] K. Mackworth. Constraint Satisfaction. In S. C. Shapiro (ed.): Encyclopedia of Artificial Intelligence. New York: Wiley-Interscience Publication, pp. 285-293, 1992.

[Mac02] J. Maciejowski. Predictive Control with Constraints. Harlow, England: Prentice-Hall, 2002.

[MDC12] R. Mudumbai, S. Dasgupta, and B. Cho. Distributed control for optimal economic dispatch of a network of heterogeneous power generators. In *IEEE Trans. Power Syst.*, vol. 27, no. 4, pp. 1750–1760, Nov. 2012.

[MDC+07] S. D. J. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziaryriou, F. Ponci, T. Funabashi. Multi-Agent Systems for Power Engineering Applications – Part I: Concepts, Approaches, and Technical

Challenges. In *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1743-1752, Nov. 2007. doi: 10.1109/TPWRS.2007.908471.

[MDCD+07] S. D. J. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziaryriou, F. Ponci, T. Funabashi. Multi-Agent Systems for Power Engineering Applications – Part II: Technologies, Standards, and Tools for Building Multi-Agent Systems. In *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1753-1759, Nov. 2007. doi: 10.1109/TPWRS.2007.908472.

[MKF+14] R. Macdonald, Y.H. Kwon, C. Foote, G. Ault, R. Johnston and A. Gooding. Challenges of Active Network Management Business as usual rollout. In *CIRED Workshop*, Rome, 2014.

[ML04] Roger Mailler and Victor Lesser. Solving Distributed Constraint Optimisation Problems Using Cooperative Mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, IEEE Computer Society, pp. 438-445, 2004.

[ML06] R. Mailler and V. Lesser. Asynchronous Partial Overlay: A new algorithm for solving distributed constraint satisfaction problems. In *Journal of Artificial Intelligence Research*, vol. 25, pp. 529–576, 2006.

[MM14] Y. Mehrdad and A. Mehrizi-Sani. Distributed Control Techniques in Microgrids. In *IEEE Transactions on Smart Grid*, pp. 2901-2909, Nov. 2014.

[Mor93] P. Morris. The Breakout Method for Escaping from Local Minima. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 40-45, 1993.

[MPT04] R.T. Maheswaran, J.P. Pearce and M. Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In *Proceedings of the ISCA Seventeenth International Conference on Parallel and Distributed Computing Systems (ISCA PDCS'04)*, pages 432–439, USA, 2004.

[MRG+02] V.H. Mendez, J. Rivier, T. Gomez, J. Arceluz, J. Marin. *Impact of Distributed Generation on Distribution Network.* Universidad Pontificia Comillas, Madrid, 2002.

[MRR12] S. Miller, S. D. Ramchurn and A. Rogers. Optimal Decentralised Dispatch of Embedded Generation in the Smart Grid. In *The Eleventh Int. Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Valencia, Spain, 2012.

[MS08] Y. A. R. I. Mohamed and E. El-Saadany. Adaptive decentralised droop controller to preserve power sharing stability of paralleled inverters in distributed generation microgrids. In *IEEE Trans. Power Electron.*, vol. 23, no. 6, pp. 2806–2816, Nov. 2008.

[MSI10] A. Mehrizi-Sani and R. Iravani. Potential-function based control of a microgrid in islanded and grid-connected modes. In *IEEE Trans. Power Syst.*, vol. 25, no. 4, pp. 1883–1891, Nov. 2010.

[MST+05] P. J. Modi, W. Shen, M. Tambe and M. Yokoo. ADOPT: Asynchronous distributed constraint optimisation with quality guarantees. In *Artificial Intelligence*, 161:149–180, 2005.

[MTA+12] S. McArthur, P. Taylor, G. Ault, J. King, D. Athanasiadis, V. Alimisis, and M. Czaplewski. The Autonomic Power System-Network operation and control beyond smart grids. In *3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, Berlin, 2012.

[MTB+04] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *AAMAS-04*, 2004.

[MZ07] A. Meisels and R. Zivan. Asynchronous forward-checking for DisCSPs. Constraints, 12:131–150, 2007.

[NIST10] NIST Special Publication 1108, NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 1.0, Office of the National Coordinator for Smart Grid Interoperability, January 2010.

[NLS+09] R.R. Negenborn, S. Leirens, B. De Schutter, and J. Hellendoorn. Supervisory nonlinear MPC for emergency voltage control using pattern search. Control Eng. Practice, vol. 17, no. 7, pp. 841–848, Jul. 2009.

[NMG10] A. Netzer, A. Meisels and A. Grubshtein. Concurrent forward bounding for DCOPs. In *Proceedings of the Twelfth International Workshop on*

Distributed Constraint Reasoning (DCR'10), pages 65–79, Toronto, Canada, 2010.

[NNN+04] John Godsk Nielsen, Michael Newman, Hans Nielsen and Frede Blaabjerg. Control and Testing of a Dynamic Voltage Restorer (DVR) at Medium Voltage Level. In *IEEE Transactions on Power Electronics*, vol. 19, no. 3, May 2004.

[Oco08] Organising Committee of the Third International Competition of CSP Solvers. *XML Representation of Constraint Networks – Format XCSP 2.1*, 2008. URL: http://www.cril.univ-artois.fr/~lecoutre/research/benchmarks/-benchmarks.html.

[OFM07] R. Olfati-Saber, A. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. In *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[OFM+13] B. O'Neill, C. E.T. Foote, J. R. McDonald, M. J. Dolan, S. Gill, G. Milne, S. Reid. Active network management on islanded systems: The Shetland experience. In *ISGT Europe*, Lyngby, 2013.

[PCF+06] M. Pehnt, M. Cames, M. Fischer, C. Praetorius, B. Schneider, L. Schneider, K. Vob. Embedding Micro Cogeneration in the Energy Supply System. *Micro cogeneration towards decentralised energy systems*, Berlin: Springer, pp. 197-218, 2006.

[PDH+05] G. Pepermans, J. Driesen, D. Haeseldonckx, R. Belmans and W. D'haeseleer. Distributed Generation: definition, benefits and issues. *Energy Policy*, 33, pp. 787-798, 2005.

[Pet06] Petcu, A.: FRODO: A FRamework for Open/Distributed constraint Optimisation. Technical Report 2006/001, EPFL, Lausanne, Switzerland, 2006.

[PF05] A. Petcu and B. Faltings. DPOP: A Scalable Method for Multiagent Constraint Optimisation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 266–271, 2005.

[PF06] A. Petcu and B. Faltings. O-DPOP: An algorithm for Open/Distributed Constrained Optimisation. In *Proceedings of the National Conference on Artificial Intelligence, AAAI-06*, 703-708, 2006.

[PF07] A. Petcu and B. Faltings. Distributed Constraint Optimisation Applications in Power Networks. In *International Journal of Innovations in Energy Systems & Power (IJESP)*, 2007.

[POW15] Power Systems Engineering Research Center. Technology Challenges in Designing the Future Grid to Enable Sustainable Energy Systems Future. Grid Thrust Area Synthesis White Paper. June 2012.

[RFS+11] A. Rogers, A. Farinelli, R. Stranders and N.R. Jennings. Bounded Approximate Decentralised Coordination via the Max-Sum Algorithm. In *Artificial Intelligence*, 175, (2), 730-759, 2011. urau

[RHG13] B. A. Robbins, C. N. Hadjicostis, and A. D. Dominguez-Garcia. A two-stage distributed architecture for voltage control in power distribution systems. In *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1470–1482, May 2013.

[RMJ+08] S. Repo, K. Maki, P. Jarventausta, and O. Samuelsson. ADINE-EU demonstration project of active distribution network. In *IET CIRED Seminar Smart Grids for Distribution*, pp. 1–5, 2008.

[RN03] S. Rusell and P. Norvig. Artificial Intelligence: A Modern Approach. Upper Saddle River, New Jersey: Prentice Hall, pp. 111-114, 2003.

[Ros03] J. Rossiter. Model Based Predictive Control- A Practical Approach. Boca Raton, FL: CRC Press, 2003.

[SG96] Solotorevsky, G. and E. Gudes. Solving a Real-life Time Tabling and Transportation Problem Using Distributed CSP Techniques. In *Proceedings of CP '96 Workshop on Constraint Programming Applications*, pp. 123-131, 1996.

[SGM13] Smart Grids Model Region Salzburg. Results and Findings from the smart grids model region Salzburg, 2013. URL: http://www.geode-eu.org/uploads/Walter%20Schaffer.pdf

[SHF00] Silaghi M.-C., D. Sam-Haroud, and B. Faltings. Asynchronous search with aggregations. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications*

of Artificial Intelligence (AAAI/IAAI'00), pages 917–922, Austin, TX, U.S.A. AAAI Press/The MIT Press, 2000.

[Sil05] Marius C. Silaghi. A suite of secure multi-party computation algorithms for solving distributed constraint satisfaction and optimisation problems. Technical Report, 2005.

[SJG11] M. Shahraeini, M. Javidi, and M. Ghazizadeh. Comparison between communication infrastructures of centralized and decentralised wide area measurement systems. In *IEEE Trans. Smart Grid,* vol. 2, no. 1, pp. 206–211, Mar. 2011.

[SJV+12] M. Savaghebi, A. Jalilian, J. Vasquez, and J. Guerrero. Secondary control scheme for voltage unbalance compensation in an islanded droop-controlled microgrid. In *IEEE Trans. Smart Grid*, vol. 3, no. 2, pp. 797–807, Jun. 2012.

[SL05] C. Sao and P. Lehn. Autonomous load sharing of voltage source converters. In *IEEE Trans. Power Del.*, vol. 20, no. 2, pp. 1009–1016, Apr. 2005.

[SRJ+10] O. Samuelsson, S. Repo, R. Jessler, J. Aho, M. Karenlampi, and A. Malmquist. Active distribution network—Demonstration project ADINE. In *Proc. IEEE PES Innovative Smart Grid Technologies Conf. Europe (ISGT Europe)*, Oct. 2010.

[Sou07] OA Soukhanov and Yvo Sharov. Hierarchical models in power systems analysis and control. Moscow: MPI Publishing House: 2007.

[SVG12] Q. Shafiee, J. Vasquez, and J. Guerrero. Distributed secondary control for islanded microgrids—A networked control systems approach. In *Proc. 38th Annual. Conf. IEEE Ind. Electron. Soc. (IECON)*, 2012, pp. 5637–5642.

[SWL+06] X. Sun, L.-K. Wong, Y. S. Lee, and D. Xu. Design and analysis of an optimal controller for parallel multi-inverter systems. In *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 1, pp. 56–61, Jan. 2006.

[SY06] M. –C. Silaghi and M. Yokoo. Nogood-based asynchronous distributed optimisation (ADOPT-ng). In *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1389–1396, 2006.

[TMA+08] P. Taylor, T. Xu, S. McArthur, G. Ault, E. Davidson, M. Dolan, C. Yuen, M. Larsson, D. Botting, D. Roberts, and P. Lang. Integrating voltage control and power flow management in AuRA-NMS. In *Proc. CIRED Smart Grids*, 2008.

[TSC+13] K. Tan, P. So, Y. Chu, and M. Chen. Coordinated control and energy management of distributed generation inverters in a microgrid. In *IEEE Trans. Power Del.*, vol. 28, no. 2, pp. 704–713, Apr. 2013.

[TXM+08] P. C. Taylor, T. Xu, S. D. J. McArthur, G. Ault, E. M. Davidson, M. Dolan, D. Botting, C. Yuen, M. Larsson, D. Roberts and P. Lang. Integrating voltage control and power flow management in AuRA-NMS. In *CIRED Smartgirds*, Frankfurt, 2008.

[UIW10] S. Ueda, A. Iwasaki, and M. Yokoo. Coalition structure generation based on distributed constraint optimisation. In *Proc. of AAAI*, pages 197–203, 2010.

[UKP14] UK Power Networks (2014). Implementation of active voltage and active power flow management within FPP Trial area. 2014. URL: http://innovation.ukpowernetworks.co.uk/innovation/en/Projects/tier-2 projects/FlexiblePlugandPlay(FPP)/ProjectDocuments/FPP%20SDRC%2 09.6%20%20Implementation%20of%20active%20voltage%20and%20ac tive%20power%20flow%20management.pdf

[VGA+10] M. Vinyals, M. Pujol-Gonzalez, J.A. Rodríguez-Aguilar and J. Cerquides, J. Divide-and-coordinate: DCOPs by agreement. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 149–156, 2010.

[VGL+09] J. C. Vasquez, J. M. Guerrero, A. Luna, P. Rodriguez, and R. Teodorescu. Adaptive droop control applied to voltage-source inverters operating in grid-connected and islanded modes. In *IEEE Trans. Ind. Electron*., vol. 56, no. 10, pp. 4088–4096, Oct. 2009.

[VKW+13] P. Vovos, A. Kiprakis, R. Wallace, and G. Harrison. Centralized and distributed voltage control: Impact on distributed generation penetration. In *IEEE Trans. Power Syst.*, vol. 22, no. 1, pp. 476–483, Feb. 2007.

[VVZ11] A. Vaccaro, G. Velotto, and A. Zobaa. A decentralised and cooperative architecture for optimal voltage regulation in smart grids. In *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4593–4602, Oct. 2011.

[WNR08] A. Waldock, D. Nicholson, A. Rogers. Cooperative Control using the Max-Sum Algorithm. In *Second International Workshop on Agent Technology for Sensor Networks*, Estoril, Portugal, 2008.

[Woo09] M. Wooldridge. An Introduction to Multi-Agent Systems - *Second Edition*. John Wiley & Sons, May 2009.

[WW99] M. Wooldridge, G. Weiss. Intelligent Agents. In *Multi-Agent Systems*. Cambridge, MA: MIT Press, pp. 3–51, Apr. 1999.

[WWT+04] J. Y. Wen, Q. H. Wu, D. R. Turner, S. J. Cheng and J. Fitch. Optimal coordinated voltage control for power system voltage stability. In *IEEE Trans. Power Syst.*, vol. 19, no. 2, pp. 1115–1122, May 2004.

[XL11] Y. Xu and W. Liu. Novel multiagent based load restoration algorithm for microgrids. In *IEEE Trans. Smart Grid*, vol. 2, no. 1, pp. 152–161, Mar. 2011.

[XPG+09] T. Xu, *M. Prodanovic*, T. Green, E. M. Davidson and S. D. J. McArthur. Case-based reasoning for distributed voltage control. In *CIRED*, 2009.

[YDI+92] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabar. Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving. In *12th International Conference on Distributed Computing Systems (ICDCS-92)*, pp.614-621, 1992.

[YDI+98] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabar. The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. In *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 5, 1998.

[YFK08] W. Yeoh, A. Felner and S. Koenig. BnB-ADOPT: An asynchronous branch-and bound DCOP algorithm. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 591–598, pages 591–598, 2008.

[YH96] M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pp. 401-408, 1996.

[YH00] M. Yokoo and K. Hirayama. Algorithms for Distributed Constraint Satisfaction: A Review. In *Autonomous Agents and Multi-Agent Systems*, Vol.3, No.2, pp.198-212, 2000.

[Yok94] Makoto Yokoo. Weak-commitment Search for Solving Constraint Satisfaction Problems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 313-318, 1994.

[Yok95] Makoto Yokoo. Asynchronous Weak-commitment Search for Solving Distributed Constraint Satisfaction Problems. In *Proceedings of the First International Conference on Principles and Practice of Constraint Programming (CP-95)*, pp. 88-102, 1995.

[ZB07] Q. Zhou and J. W. Bialek. Generation curtailment to manage voltage constraints in distribution networks. In *IET Proceedings, Generation, Transmission and Distribution*, pp.492-498, 2007.

[ZC11] Z. Zhang and M.-Y. Chow. The leader election criterion for decentralised economic dispatch using incremental cost consensus algorithm. In *Proc. 37th Annual Conf. IEEE Ind. Electron. Soc. (IECON)*, Melbourne, VIC, pp. 2730–2735, Australia, 2011.

[ZHQ12] J. Zhou, R. Hu, and Y. Qian. Scalable distributed communication architectures to support advanced metering infrastructure in smart grid. In *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 9, pp. 1632–1642, 2012.

[ZMA+15] Owen Zinaman, Mackay Miller, Ali Adil, Douglas Arent, Jaquelin Cochran, Sonia Aggarwal, Minnesh Bipath, Carl Linvill, Ari David, Matthew Futch, Efraín Villanueva Arcos, José María Valenzuela, Eric Martinot, Daniel Noll, Morgan Bazilian, Reji Kumar Pillai. Power Systems of the Future. In The Electricity Journal, Volume 28, Issue 2, Pages 113-126, March 2015.

[ZM04] R. Zivan and A. Meisels. Concurrent Dynamic Backtracking for Distributed CSP's. In *Proceeding Constraint Programming*, pp. 782-787, 2004.

[ZR09] S. Zhao and D. Raychaudhuri. Scalability and performance evaluation of hierarchical hybrid wireless networks. In *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1536–1549, Oct. 2009.

[ZWX+05] W. Zhang, G. Wang, Z. Xing and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimisation problems in sensor networks. In *Artificial Intelligence*, 161(1–2):55–87, 2005.