UNIVERSITY OF SOUTHAMPTON

Decentralised Coordination of Smart Distribution Networks using Message Passing

by

Samuel John Odell Miller

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Physical Sciences and Engineering
Electronics and Computer Science

February 2014

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Samuel John Odell Miller

Over the coming years, distribution network operators (DNOs) face the challenge of incorporating an increased number of electrical distributed generators (DGs) into their already capacity-constrained distribution networks. To overcome this challenge will require the DNOs to use active network management techniques, which are already prevalent in the transmission network, in order to constantly monitor and coordinate these generators, whilst ensuring that the bidirectional flows they engender on the network are safe. Therefore, this thesis presents novel decentralised message passing algorithms that coordinate generators in acyclic electricity distribution networks, such that the costs (in terms of carbon dioxide (CO_2) emissions) of the entire network are minimised; a technique commonly referred to as *optimal dispatch*. In more detail, we cast the optimal dispatch problem as a decentralised agent-based coordination problem and formalise it as a distributed constraint optimisation problem (DCOP). We show how this DCOP can be decomposed as a factor graph and solved in a decentralised manner using algorithms based on the generalised distributive law; in particular the max-sum algorithm. We go on to show that max-sum applied naïvely in this setting performs a large number of redundant computations. To address this issue, we present both a discrete and a continuous novel decentralised message passing algorithm that outperforms max-sum by pruning much of the search space. Our discrete version is applicable to network settings that are entirely composed of discrete generators (such as wind turbines or solar panels), and when the constraints of the electricity network have been discretised. Our continuous version can be applied to a wider range of network settings containing multiple types of generators, without the need to discretise the electricity distribution network constraints. We empirically evaluate our algorithms, using two large real electricity distribution network topologies, and show that they outperform max-sum (in terms of computational time and total size of messages sent).

Contents

List of Figures

Declaration of Authorship

I, Samuel John Odell Miller, declare that the thesis entitled *Decentralised Coordination of Smart Distribution Networks using Message Passing* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published in Miller et al. (2012) and Miller et al. (2013).

Signed:.....

Date:.....

Acknowledgements

I'd like to thank both my supervisors, Alex Rogers and Sarvapali (Gopal) Ramchurn, for their constant guidance, encouragement, and support throughout my entire PhD. Particularly, their ability to find errors in my writing that even the most wizened veteran of proof readers would of missed. I'd also like to thank my colleagues and friends in the Agents, Interaction, and Complexity research group (specifically the agents part of the group). It has been a pleasure to collaborate and socialise with such a great bunch of people.

I'd like to thank my internal examiner Jason Noble and my external examiner *Νικόλαο Βασιλειάδη* (Nick Bassiliades) for not giving me too hard a time in the viva.

I would like to give a special mention to my parents, Michael and Helen. They have, without a doubt, been amazingly supportive not just during my PhD, but throughout my entire education. They never put pressure on me to follow a particular avenue of study or hobby, and encouraged me to achieve what I wanted to achieve. For that I am truly thankful.

Finally, I would like to thank computers, the internet, and chocolate. Without these things, I don't think I could of made it through my PhD.

Nomenclature

$\mathcal{X}$	Set of h variables in the DCOP representation of an electricity network
x_i	Variable i
$\mathcal{D}$	Set of h domains of $\mathcal{X}$
$\mathbf{d}_i$	Domain of x_i
$\mathcal{F}$	Set of k relations between the variables in $\mathcal{D}$
F_i	Cost of variables having certain values. Utility of A_i
$\mathcal{A}$	Set of k agents
A_i	Agent i
$\mathcal{X}^*$	An assignment of the variables that minimises the sum of $\mathcal{F}$
$Q_{b\to a}(x_b)$	Max-sum message sent from variable x_b to function a
$R_{a\to b}(x_b)$	Max-sum message sent from function a to variable x_b
$B(a)$	Set of variables connected to function a
$A(b)$	Set of functions connected to variable x_b
$\mathcal{X}_a \backslash b$	Set of variables connected to function a not containing variable x_b
$Z_b(x_b)$	Sum of max-sum R messages flowing into x_b
P	Power
E	Electric potential difference (Voltage)
I	Current
Z	Impedance
R	Resistance
X	Reactance
P_{loss}	The power lost due to heat caused by impedance
v_i	Node i
$\mathbf{P}$	Vector of real resultant power for k nodes
p_i	Real resultant power for v_i
$\mathbf{B}$	$k \times k$ symmetrical matrix of susceptances for distribution cables between k nodes
b_{ij}	Negative susceptance of t_{ij}, for $i \neq j$
b_{ii}	$\sum_{j=0}^{k} \lvert b_{ij} \rvert$, sum of the susceptances for row i
x_{ij}	Reactance of t_{ij}

$\boldsymbol{\Theta}$	Vector of voltage phase angles for k nodes
θ_i	Voltage phase angle at v_i
f_{ij}	Power flowing from v_i and v_j
$\mathbf{G}$	Set of n generators
g_i	Generator i
$\boldsymbol{\alpha}$	Set of power output variables for the generators in $\mathbf{G}$
α_i	Power output value for g_i
α_i^{min}	Minimum output of g_i
α_i^{max}	Maximum output of g_i
S_i	Set of discrete power output levels for g_i
s_j^i	The jth power output level of g_i
q_i	Number of power output levels for g_i
e_i	CO_2 emissions of g_i
CI_i	Carbon intensity of g_i
$\mathbf{L}$	Set of m loads
l_i	Load i
$\boldsymbol{\beta}$	Set of power consumption variables for the loads in $\mathbf{L}$
β_i	Power consumption of l_i
$\mathbf{V}$	Set of k nodes
v_r	Root node of a tree network
$\hat{v_i}$	Parent of v_i
v_d	A node that contains a discrete generator
$chi(v_i)$	Immediate children of v_i
$adj(v_i)$	All nodes connected to v_i
$\mathbf{L}(v_i)$	Set of loads at v_i
$\mathbf{G}(v_i)$	Set of generators at v_i
$\mathbf{T}$	Set of s distribution cables
t_{ij}	Distribution cable between v_i and v_j
t_{ij}^c	Thermal capacity t_{ij}
$\mathbf{W}(\mathbf{V}, \mathbf{T})$	Finite undirected graph of nodes and distribution cables
$\mathbf{F}$	Set of power flows for s distribution cables
$\mathbf{F}(v_i)$	Set of power flows from the distribution cables connecting v_i to its children
$t_{i\hat{i}}$	Distribution cable between v_i and $\hat{v_i}$
$f_{i\hat{i}}$	Power flowing along $t_{i\hat{i}}$
$t_{i\hat{i}}^c$	Thermal capacity of $t_{i\hat{i}}$
$\boldsymbol{\gamma}(f_{i\hat{i}})$	CO_2 emissions of $f_{i\hat{i}}$
Φ_i	Worst case maximum number of messages v_i has to create and send to $\hat{v_i}$
M	Maximum number of *flowCO* elements a *PowerCost* message, received from the children of v_i, contains

$f_{i\hat{i}}^{min}$	Minimum power flow of a continuous *flowCO* element
$f_{i\hat{i}}^{max}$	Maximum power flow of a continuous *flowCO* element
δ	Gradient of a continuous *flowCO* element
$\boldsymbol{\gamma}(f_{i\hat{i}}^{min})$	Minimum CO_2 emissions of a continuous *flowCO* element
$\boldsymbol{\gamma}(f_{i\hat{i}}^{max})$	Maximum CO_2 emissions of a continuous *flowCO* element
var_i	Varying v_i for an *OPCState*
ω	Discretisation unit of the network
$\overline{\beta}_i$	Discretised power consumption of l_i
$\overline{\alpha}_i^{min}$	Discretised minimum output of g_i
$\overline{\alpha}_i^{max}$	Discretised maximum output of g_i
$\overline{t}_{ij}^{c}$	Discretised thermal capacity of t_{ij}
$\overline{\alpha}_i$	Power output of g_i for the discretised network
$\overline{f}_{ij}$	Power flowing along t_{ij} for the discretised network
$\tilde{\alpha}_i$	Power output of g_i after the generators in the discretised network have been adjusted to match the actual consumption of the network
$\tilde{f}_{ij}$	Power flowing along t_{ij} after the generators in the discretised network have been adjusted to match the actual consumption of the network
$[\varpi^{min}, \varpi^{max}]$	Experiment variables used in Section 3.4.4. Distribution cables in the network are assigned a uniformly distributed thermal capacity t_{ij}^{c} in the range of $[\varpi^{min}, \varpi^{max}]$
η	Experiment variable used in Sections 3.4.4, 4.4, and 5.4. Discrete generators are assigned a uniformly distributed power output level ηkW
k	Number of nodes in a network
n	Number of generators in a network of k nodes
m	Number of loads in a network of k nodes
s	Number of distribution cables in a network of k nodes
h	Number of variables in the DCOP representation of an electricity network

Abbreviations

ANM	active network management
AC	alternating current
ADMM	alternating direction method of multipliers
ADOPT	asynchronous distributed constraint optimisation
APO	asynchronous partial overlay
AuRA-NMS	autonomous regional active network management system
APP	auxiliary problem principle
CBR	case based reasoning
C–DYDOP	continuous dynamic programming decentralised optimal dispatch
CO_2	carbon dioxide
CHP	combined heat and power
CP	constraint programming
DC	direct current
DCOP	distributed constraint optimisation problem
D–DYDOP	discrete dynamic programming decentralised optimal dispatch
DG	distributed generator
DNO	distribution network operator
DPOP	distributed pseudotree optimisation procedure
GDL	generalised distributive law
IDAPS	intelligent distributed autonomous power system
MIP	mixed integer programming
OPF	optimal power flow
OptAPO	optimal asynchronous partial overlay
PF	power flow
SCADA/GMS	supervisory control and data acquisition/generation management system
ULCV	ultra-low carbon vehicles

In theory, there is no difference between theory and practice. But, in practice, there is.

Manfred Eigen

What can be asserted without proof can be dismissed without proof.

Christopher Hitchens

Don't Panic.

Douglas Adams, The Hitchhiker's Guide to the Galaxy

Chapter 1

Introduction

Over the coming years, distribution network operators (DNOs) face the challenge of incorporating an increased number of electrical distributed generators (DGs) into their already capacity-constrained distribution networks. Coupled with this is the desire to distribute and decentralise the management of these generators since traditional centralised techniques may present a number of issues.[1] To overcome these challenges will require the DNOs to use active network management techniques, which are already prevalent in the transmission network, in order to constantly monitor and coordinate these generators, whilst ensuring that the bidirectional flows they engender on the network are safe. This presents an interesting coordination problem which could be solved by using techniques from the artificial intelligence community. However, in order to fully understand this coordination problem, we first need to elaborate on the reasons why the distribution network will experience this increase in DGs.

Due to concerns about the effects climate change may have on regions of the world in the future, as a result of increasing levels of carbon dioxide (CO_2) emissions (US Department of Energy, 2003; UK Department of Energy and Climate Change, 2009a; Ramchurn et al., 2012; Intergovernmental Panel on Climate Change, 2013), agreements have been made by numerous governments to reduce global CO_2 emissions.[2] To help achieve this global reduction of CO_2 emissions, the UK has committed to transition to a low carbon economy via an 80% reduction in emissions by the year 2050. In the UK, the major contributors of CO_2 emissions are the residential and transport sectors. Thus, significantly reducing the CO_2 emissions from both these sectors will dramatically help the UK in achieving a low carbon economy.

In more detail, the residential sector accounts for 13% of all UK CO_2 emissions, with the majority as a consequence of heating living spaces and water (UK Department

[1] We discuss the issues associated with centralised control later in this section and also in more detail in Section 2.3.2.

[2] See (International Energy Agency, 2011) for a full list of countries.

of Energy and Climate Change, 2009b). Residential heating predominantly uses gas which produces high levels of CO_2 emissions. However, the transition to a low carbon economy will see an increasing demand for efficient electric heating technologies such as ground-source and air-source heat pumps (MacKay, 2008; UK Department of Energy and Climate Change, 2009a). A similarly high emitting sector is transport which accounts for 22% of all UK CO_2 emissions (UK Department of Energy and Climate Change, 2009b). Thus, a shift from conventional vehicles to ultra-low carbon vehicles (ULCV) will be needed in order to further reduce CO_2 emissions. Due to all of these factors, the demand for electricity will increase dramatically over the coming decades. However, if the electricity is only produced using conventional generators (i.e., coal or gas), the increasing demand for electricity will result in more CO_2 emissions. Therefore, a key aspect of achieving the 2050 targets will be to increase the amount of low CO_2 emitting electricity generators, which use renewable resources such as wind and solar, in electricity networks.

In particular, a large amount of this increased generation will be connected to distribution networks,[3] in the form of potentially thousands of DGs,[4] in order to help satisfy local demands (Roberts, 2004; Alarcon-Rodriguez et al., 2006; UK Department of Energy and Climate Change, 2009a). However, current electricity networks around the world are outdated, inefficient at transporting electricity, and unable to sustain this increase in generation (US Department of Energy, 2003; UK Department of Energy and Climate Change, 2009a). In particular, distribution networks are already highly capacity-constrained (or becoming increasingly so) with little automation for controlling the generators in the network (Roberts, 2004). Thus, incorporating potentially thousands of DGs, into already capacity-constrained distribution networks, is a major challenge that DNOs will face.

The current practice that UK DNOs use to connect additional generation to their networks involves adding network reinforcements to increase network capacity. Network reinforcements consist of additional circuitry and distribution cables to ensure that any additional generators added to the electricity network do not overload existing infrastructure. Whilst this is essential, installing network reinforcements can be time consuming and may have significant monetary and environmental costs (Roberts, 2004). In order to avoid these costly network reinforcements, Roberts (2004) and the US Department of Energy (2003) suggest that more dynamic and 'smart' electricity networks, capable of managing electricity more efficiently, are needed; a vision commonly referred to as the *smart grid*. The smart grid is a fully automated power delivery network that is capable

[3]National electricity networks consist of a high voltage transmission network, and lots of lower voltage distribution networks. See Section 2.2 for more details.

[4]These generators are considered to be distributed because they will be embedded throughout the distribution network instead of the transmission network.

of managing data from smart meters,[5] end users, micro-storage devices, and DGs using a two-way flow of electricity and information.

One key aspect of this smart grid is the ability to add additional DGs into increasingly capacity-constrained networks. Roberts (2004) and the Department for Business Enterprise and Regulatory Reform (2008) suggest the use of active network management (ANM) to incorporate additional DGs. ANM allows the electricity to be efficiently managed whilst satisfying demands and ensuring the maximum capacities of the distribution cables are not exceeded. Therefore, there is a clear incentive for DNOs to adopt ANM techniques in order to incorporate potentially thousands of DGs without using costly network reinforcements. In particular, ANM can be used to coordinate the power output of DGs such that the loads and the constraints of the network are satisfied. The coordination of generators in an electricity network is typically referred to as *optimal dispatch* (Wood and Wollenberg, 1995) and has been traditionally completed using centralised calculations.

For a central control system to work, large amounts of data must be transmitted to a central location, manipulated, and then sent back to each generator. However, as the electricity network size increases, the complete control loop that is necessary, in order to control the electricity generators centrally, may be too slow to respond to fluctuating changes in a timely fashion (Department for Business Enterprise and Regulatory Reform, 2008; Granada et al., 2008) (See Section 2.3.2 for a thorough discussion of centralised versus distributed and decentralised control in electricity networks). Hence, this thesis is concerned with how DNOs can incorporate distributed and decentralised optimal dispatch methods, using ANM, in order to be able to manage an increased number of DGs in the distribution networks, whilst reducing CO_2 emissions and ensuring electricity demand is satisfied.[6] The following section discusses the requirements of a system for controlling the power outputs of generators in electricity distribution networks.

1.1 Research Requirements

From the discussion of the problems in the previous section, a number of requirements can be identified for coordinating generators in an electricity distribution network, whilst reducing the cost of the network, and satisfying the loads and the network constraints:

I **Autonomy** The algorithms developed should require minimal human interaction in order to coordinate the power output of generators in electricity networks. This

[5] A smart meter is an electrical device that receives real-time electricity pricing and displays this information to customers.

[6] It should be noted that considering the change of electricity demand over *time* is beyond the scope of this thesis. We consider the demands of a network to be static. See Chapter 6 for future work concerning varying demand over time.

is because as the size of the network grows, it may become increasingly difficult for human operators to make optimal and efficient decisions due to the large amount of data and computation required.

II **Minimise CO_2 Emissions** The generators should coordinate such that they increase the efficiency of the network by minimising CO_2 emissions.

III **Scalability** The algorithms should be able to scale to large electricity networks that contain thousands of generators and distribution cables.

IV **Handle Different Types of Generation** The algorithms should be able to coordinate different types of generators with varying outputs and constraints. For instance, renewable generators, such as wind turbines, will only generate electricity when the wind blows. Thus, the algorithms need to be able to handle intermittent generators.

V **Distributed and Decentralised Control** The algorithms must decentralise and distribute the computation and information throughout the electricity distribution network. Each generator in the network will take on some of the computation to solve the optimal dispatch problem. See Section 2.3.2 for a discussion of the advantages of distributing and decentralising control in electricity networks.

VI **Enable Plug-and-Play** Since the electricity network is going to evolve and increase in size over the coming years, the algorithms will need to be able to adapt to different network topologies and electrical devices. Thus, they should allow operators to "plug-and-play" new generators, loads and distribution cables into (and out of) the network. This should be achievable without having to redesign the algorithms for the particular problem at hand.

VII **Graceful Degradation** In the event that one or more generators malfunction and are unable to communicate with their neighbours, the algorithms should still be able to compute a solution without failing in an unexpected way.

The following section discusses the existing work on coordinating generators in an electricity network, and the gap in the literature that this thesis addresses.

1.2 Research Challenges

From the problems and requirements identified in the previous section, there are some key challenges that must be addressed for coordinating generators within an electricity network:

1. When coordinating generators in an electricity network, the first key challenge is ensuring the algorithms minimise the global cost of the network (i.e., CO_2 emissions) whilst satisfying the local and global constraints of the electricity network. The local constraints of the electricity network consist of the maximum thermal capacities of the distribution cables, and the maximum and minimum power outputs of the generators. The global constraint of the electricity network requires that total production must equal total consumption of electricity.

2. The algorithms that coordinate generators in an electricity network must be robust to failure (i.e., if one of the generators is unable to communicate, due to a communication network failure, the algorithm should still be able to arrive at a solution). This implies that the system should be decentralised so that the computation does not rely on a single entity, and also that the system should be distributed so that the computation is distributed between generators, giving higher robustness than a centralised approach.[7] Thus, the second key challenge is ensuring that the coordination problem is split up in such a way so that the optimal decision for each generator can still be calculated without centralised control.

Some of these key challenges are already being addressed within the artificial intelligence and power systems communities using ANM. In particular a number of authors address the issues of coordinating generation from intermittent resources in the transmission network (where lines are less constrained than in the distribution network) (Kim and Baldick, 1997; Davidson et al., 2009). For example, Davidson et al. (2009) present an algorithm to manage the power output of the generators in the transmission network in order to reduce overloaded transmission lines.[8] However, their technique involves a central authority calculating the power output of each generator. As the size of the network grows, solving an optimisation problem in a centralised manner may become infeasible. This is due to the amount of data that must be held centrally about each generator and distribution cable.[9]

Kim and Baldick (1997) overcome the above issue by introducing a decentralised algorithm which uses Lagrangian techniques and the auxiliary problem principle (APP). The network is split into regions and each region communicates with its neighbouring regions to decide the optimal power outputs for their generators. However, this algorithm has only been tested on problems containing up to two regions. Thus, it is unclear whether it will scale well when applied to larger electricity networks. Recently, Kraning et al.

[7] See Section 2.3.2 for a discussion of the advantages of distributing and decentralising control within an electricity network.

[8] An overloaded transmission line (or distribution cable) occurs when the power travelling through it is higher than its normal operating thermal capacity. As a result, the transmission line protection will trip and the line will be unable to transmit power leading to network failure.

[9] For instance, data such as the minimum and maximum outputs, carbon intensity, and current output of the generators, topology of the network, and the amount of power flowing through each distribution cable. See Section 2.3.2 for a detailed discussion of centralised and decentralised control within electricity networks.

(2013) present a solution that decomposes an electricity network into subproblems using similar techniques to the APP. They claim they are able to coordinate networks containing up to 100,000 nodes and project that it will take 200ms to solve in a decentralised way. However, no concrete results are presented.

In contrast, Kumar et al. (2009) introduce a message passing technique which extends distributed pseudotree optimisation procedure (DPOP) to solve the related area of research for reconfiguring feeder trees within a distribution network. While this approach is decentralised, and was shown to work for real-world networks, it does not address the problems highlighted above of incorporating an increasing amount of generation in the distribution network, and the need to coordinate their output.

Against this background, there is a clear need for a decentralised algorithm that can coordinate an increased amount of generation in the distribution networks, such that CO_2 emissions of the entire network are minimised (Requirement II). The algorithm should scale well with the size of the network (Requirement III), and distribute the information and computation required to coordinate electricity generators (Requirement V). Due to the projected increase in renewable generators embedded in distribution networks, the algorithm will need to be able to handle a wide variety of different types of generator (Requirement IV).

Now, Requirement I means the system should run automatically and require no human intervention. This suggests the use of software agents (Wooldridge and Jennings, 1995)[10] that interact with each other as part of a multi-agent system. Using a multi-agent system also means that Requirements V, VI, and VII will be met. Multi-agent systems typically do not require a fixed number of agents to interact with each other at any one time. Thus, they can be inherently plug-and-play capable (Requirement VI) and can gracefully degrade (Requirement VII) since any agent can join the system (or leave) and the remaining agents are still capable of interacting. Moreover, multi-agent systems can be implemented in a decentralised and distributed fashion (Requirement V) so that each agent undertakes some of the computation to reach a global solution.

By distributing and decentralising the computation, the scalability issues of a system can be addressed (Requirement III). Centralised approaches to coordinate generators in an electricity network can be fast, as Davidson et al. (2009) show, but could become infeasible for large networks. This is due to the large amount of data that must be transmitted to a central location. Moreover, having a central authority provides a central point of failure for the entire system.[11] Thus, to address the scalability issues, there is a clear incentive to decompose and distribute the computation.

[10] An agent is an autonomous piece of software that has the ability to interact with its environment, exhibiting goal driven behaviour, in order to maximise the agent's utility function.

[11] The disadvantages of centralised control within an electricity network are discussed in more detail in Section 2.3.2.

The decentralised and distributed coordination of potentially tens of thousands of physically distributed entities has long been a focus of research for the distributed constraint optimisation problem (DCOP) community (Modi et al., 2005; Petcu and Faltings, 2005; Mailler and Lesser, 2006; Farinelli et al., 2008). As such, the DCOP community have an extensive set of algorithms to coordinate distributed systems using agent-based message passing. Examples include DPOP (Petcu and Faltings, 2005), asynchronous distributed constraint optimisation (ADOPT) (Modi et al., 2005), asynchronous partial overlay (APO) (Mailler and Lesser, 2006), optimal asynchronous partial overlay (OptAPO) (Mailler and Lesser, 2004), and max-sum (Farinelli et al., 2008). A subset of these algorithms (i.e., DPOP and max-sum) are from the generalised distributive law (GDL) family of algorithms (Aji and McEliece, 2000; Farinelli et al., 2013) which provides a message passing framework that has been proven analytically and empirically to converge to optimal solutions very quickly (Aji and McEliece, 2000; Kschischang et al., 2001; Farinelli et al., 2008). Thus, these existing algorithms provide a starting point for solving the optimal dispatch problem in a distributed and decentralised fashion.

The following section discusses the research contributions of this thesis.

1.3 Research Contributions

To address the challenges of distributing and decentralising the optimal dispatch problem, we present a number of novel message passing algorithms in Chapters 3, 4, and 5, which solve the problem of coordinating generators in acyclic electricity distribution networks (i.e., radial networks, or ring main and interconnected networks that have been configured into acyclic networks). In what follows, we elaborate on the individual contributions of this thesis.

In Chapter 3 we provide a new formalism of the optimal dispatch problem as a DCOP. We show how this can be decomposed as a factor graph and solved using algorithms based on the GDL family, such as max-sum. We go on to show that discrete max-sum applied naïvely in this setting performs a large number of redundant computations. To address this issue, in Chapters 4 and 5 we present both discrete and continuous novel message passing algorithms.

In order to apply a discrete algorithm to the optimal dispatch problem, the electricity distribution network constraints must be discretised as detailed in Section 3.4. Whilst discretising the electricity distribution network constraints using a small discretisation unit decreases the error between the discrete optimal dispatch solution and the actual optimal dispatch solution, the computation time increases. However, if a large discretisation unit is used, a discrete algorithm can outperform a continuous algorithm for

certain scenarios.[12] In contrast, a continuous algorithm does not require the electricity constraints to be discretised and does not suffer from accuracy issues as a result.

In more detail, in Chapter 4 we present a novel message passing algorithm, called D–DYDOP (Discrete–DYnamic programming Decentralised OPtimal dispatch) which discretises the power outputs of the generators, the loads, and the distribution cable flows. Agents communicate with their neighbours to determine the optimal power outputs for the entire network such that the global objective to minimise CO_2 emissions is achieved. In Section 4.4 we benchmark D–DYDOP against both an optimal centralised approach based on mixed integer programming (MIP), and our benchmark decentralised algorithm in Chapter 3. The contents of Chapters 3 and 4 appeared in Miller et al. (2012) and Miller et al. (2013).

In Chapter 5 we present C–DYDOP (Continuous–DYnamic programming Decentralised OPtimal dispatch) which extends D–DYDOP to use continuous variables for generator power outputs, loads, and distribution cable flows. Instead of calculating a cost for each discrete power output, agents use piecewise linear functions to represent the cost of generating power over a range of distribution cable flows. This allows for a much more compact representation of a message which agents communicate with their neighbours. The advantage of using continuous variables is that C–DYDOP does not suffer from the discretisation of the search space; unlike D–DYDOP which must iterate through every possible combination of discrete values when calculating messages. Furthermore, compared to D–DYDOP, C–DYDOP does not suffer from the branching factor of the network, or accuracy issues with regard to the optimal answer. We benchmark C–DYDOP against D–DYDOP, the centralised approach, and max-sum, to show that C–DYDOP outperforms D–DYDOP and max-sum in terms of total number of message elements sent and computation time. The contents of this chapter appeared in Miller et al. (2013). The following section details the structure of this thesis.

1.4 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2 A review of the literature relevant to this thesis is presented. We introduce the global target to reduce CO_2 emissions and how the UK is helping to achieve this target. The UK electricity network is described along with the current problems that exist and the need for a 'smarter' electricity network. Power flow coordination, the problems associated with electrical power flows, and active network management are discussed along with various approaches to coordinating

[12] In Section 5.4 we show that our discrete algorithm D–DYDOP can outperform, in terms of computation time, our continuous algorithm C–DYDOP when a large discretisation unit is used.

the power output of generators using a smarter grid. We present typical distribution network topologies including radial, ring main, and interconnected. Finally, we explain DCOPs and introduce the GDL family of algorithms including a formal definition of max-sum.

Chapter 3 We introduce the electricity network model that is used by Chapters 3, 4, and 5. Furthermore, we present our novel representation of an electricity distribution network as a DCOP and solve using our benchmark decentralised algorithm discrete max-sum. Note that the techniques presented in Section 3.4 are required to discretise the electricity distribution network constraints before applying max-sum. Thus, the accuracy of the solution calculated by max-sum is dependent on this discretisation. Moreover, max-sum applied naïvely in this setting performs a large number of redundant computations. The contents of this chapter appeared in Miller et al. (2012) and Miller et al. (2013).

Chapter 4 To address the redundant computation problems associated with applying a naïve implementation of max-sum, we present our novel message passing algorithm, called D–DYDOP, which uses techniques based on local consistency to prune much of the search space. Messages are propagated from leaf nodes, up to the root of the tree using a dynamic programming approach. Like with discrete max-sum, D–DYDOP requires the electricity distribution network constraints to be discretised. The contents of this chapter appeared in Miller et al. (2012) and Miller et al. (2013).

Chapter 5 In order to alleviate the accuracy issues associated with both discrete max-sum and D–DYDOP, we present an extension of D–DYDOP, called C–DYDOP, which does not require the electricity distribution network constraints to be discretised. C–DYDOP uses continuous values for the generator power outputs and distribution cable flows, enabling computational overheads to be reduced. The contents of this chapter appeared in Miller et al. (2013).

Chapter 6 Finally, we conclude by summarising this thesis and giving a detailed plan of the future work.

Chapter 2

Background

This chapter gives a detailed overview of the key background research related to this thesis and provides the necessary theory in developing the algorithms and techniques presented in Chapters 3, 4, and 5. Section 2.1 describes the global drivers and initiatives to reduce carbon dioxide (CO_2) emissions, and how the UK government is aiming to achieve its reduction targets. Section 2.2 introduces nationwide electricity networks, describes the physics of electricity, introduces the complexities that arise when coordinating power flows within a network, and presents typical electricity distribution network topologies. Section 2.3 presents the need for a smarter network and describes state of the art techniques for coordinating generators within a smart grid. Moreover, we detail active network management (ANM), which provides an alternative solution to network reinforcements, several techniques for coordinating generators in both a centralised and decentralised framework, and a discussion of the advantages of distributing and decentralising the control of generators in electricity networks. Section 2.4 introduces the distributed bucket elimination algorithm, the distributed constraint optimisation problem (DCOP) framework, and the generalised distributive law (GDL) family of algorithms. Finally, Section 2.5 summarises the key concepts described and justifies the techniques that are built upon within this thesis.

2.1 Reducing Global Carbon Dioxide Emissions

As discussed in Chapter 1, agreements have been made by numerous governments to reduce global CO_2 emissions (International Energy Agency, 2011). In particular, the UK has committed to transition to a low carbon economy with the specific claim of achieving an 80% reduction of CO_2 emissions by 2050 (UK Department of Energy and Climate Change, 2009b). This is set against a background where the UK residential sector accounts for 13% of all UK CO_2 emissions, (the majority of which is a consequence of

heating living spaces and water) while the UK transport sector contributes 22% of all UK CO_2 emissions (UK Department of Energy and Climate Change, 2009b).[1]

In more detail, the UK residential sector predominantly uses gas for heating which produces high levels of CO_2 emissions. However, the transition to a low carbon economy will see an increasing demand for more efficient heating technologies such as electric ground-source and electric air-source heat pumps (MacKay, 2008; UK Department of Energy and Climate Change, 2009a).[2] Within the transport sector, a shift from conventional vehicles to ultra-low carbon vehicles (ULCV) will be needed in order to further reduce CO_2 emissions.[3] Thus, this transition to a low carbon economy, from both the residential and transport sectors, will dramatically increase the demand for electricity over the coming decades.[4] However, if electricity is only generated using coal or gas, the increasing demand for electricity will result in more CO_2 emissions. Therefore, a key aspect of achieving the 2050 targets will be to increase the amount of low CO_2 emitting electricity generators, which use renewable resources such as wind and solar, into electricity networks.

In particular, a large amount of this increased generation will be connected to distribution networks, in the form of potentially thousands of distributed generators (DGs), in order to help satisfy local demands (Roberts, 2004; UK Department of Energy and Climate Change, 2009a).[5] However, current distribution networks are already highly capacity-constrained with little automation for controlling the generators in the network (Weedy and Cory, 2004; Roberts, 2004). Therefore, in order to incorporate potentially thousands of DGs, the UK national electricity network, and in particular the distribution network, will need to be updated (US Department of Energy, 2003). In order to understand the motivations for updating national electricity networks, the following section elaborates on current national electricity infrastructure and how power is transmitted.

2.2 Existing Transmission and Distribution of Electricity

Electric power grids are made up of a transmission and a distribution network, enabling the transportation of electricity from generators to consumers (Weedy and Cory, 2004;

[1] The majority of the remaining 65% of UK CO_2 emissions can be attributed to the energy supply and business sectors.

[2] Heat pumps use electricity to efficiently transfer thermal energy in order to heat (or cool) buildings.

[3] Typically, ULCVs use either solid state batteries or hydrogen fuel cells.

[4] The UK has already experienced an increase from 59GW of peak electricity demand in 2010 to 61GW in 2013, and will experience a predicted increase to 65GW by the year 2017 (National Grid, 2011).

[5] Examples of DGs include photovoltaic panels which convert sunlight into electricity, wind turbines which convert wind into electricity, biomass generators which convert biodegradable matter and specially grown crops into electricity, and combined heat and power (CHP) generators which attempt to capture the heat lost during electricity generation and use it to heat buildings.

Gönen, 2007; Grigsby, 2012). The transmission network consists of heavy duty lines capable of transmitting electricity at 400kV and above between a number of large power stations and the main substations. The large power stations generate electricity, using coal, gas, oil, and nuclear fuels, as three-phase alternating current (AC) at a voltage of 11–25kV. Transformers situated nearby are used to increase the voltage to 400kV for transportation on the transmission network. High voltages are used, when transporting electricity, in order to reduce power loss caused by resistance. Resistance is part of impedance which is the measure of the opposition that a circuit presents to AC when a voltage is applied. The following equation defines the impedance of a path Z where the real part R, is the resistance and the imaginary part X, is the reactance:

$$Z = R + jX \tag{2.1}$$

Thus, in more detail, the amount of power lost P_{loss}, due to heat, is proportional to the resistance and the current I:

$$P_{loss} = I^2 R \tag{2.2}$$

Now, if a high voltage E is used, a low current is required to produce the same amount of power P:

$$P = IE \tag{2.3}$$

Thus, for a particular transmission line, assuming the resistance remains constant, a lower current results in a smaller amount of power lost. Once the power reaches the main substations, step-down transformers are used to decrease the voltage from 400kV to either 33kV, 11kV, or 6.6kV which is fed onto the distribution network controlled by the distribution network operators (DNOs). In order to meet local demands, the distribution network also contains a high number of smaller generation units distributed throughout the network. Distribution cables transport electricity between further transformers to decrease the voltage to 400–415V three-phase, giving 230–240V per phase which is suitable for end consumers (i.e., households).

Having given an overview of electricity generation, transmission, and distribution, we now focus on distribution network topologies (since this thesis is concerned with controlling generators in distribution networks). Distribution networks differ in a number of ways from transmission networks. Typically, distribution networks contain a higher number of branches throughout the network,[6] and are much more capacity-constrained compared with the transmission network (Weedy and Cory, 2004; Roberts, 2004). This means that the projected introduction of possibly thousands of DGs in the distribution network will need to be managed efficiently in order to ensure the network does not become overloaded. In the following section we provide a detailed explanation of typical distribution network topologies.

[6]For example, the Indian distribution network topology that we use to test our algorithms, Figure 4.2(a), contains a substation which is connected by 4 distribution cables.

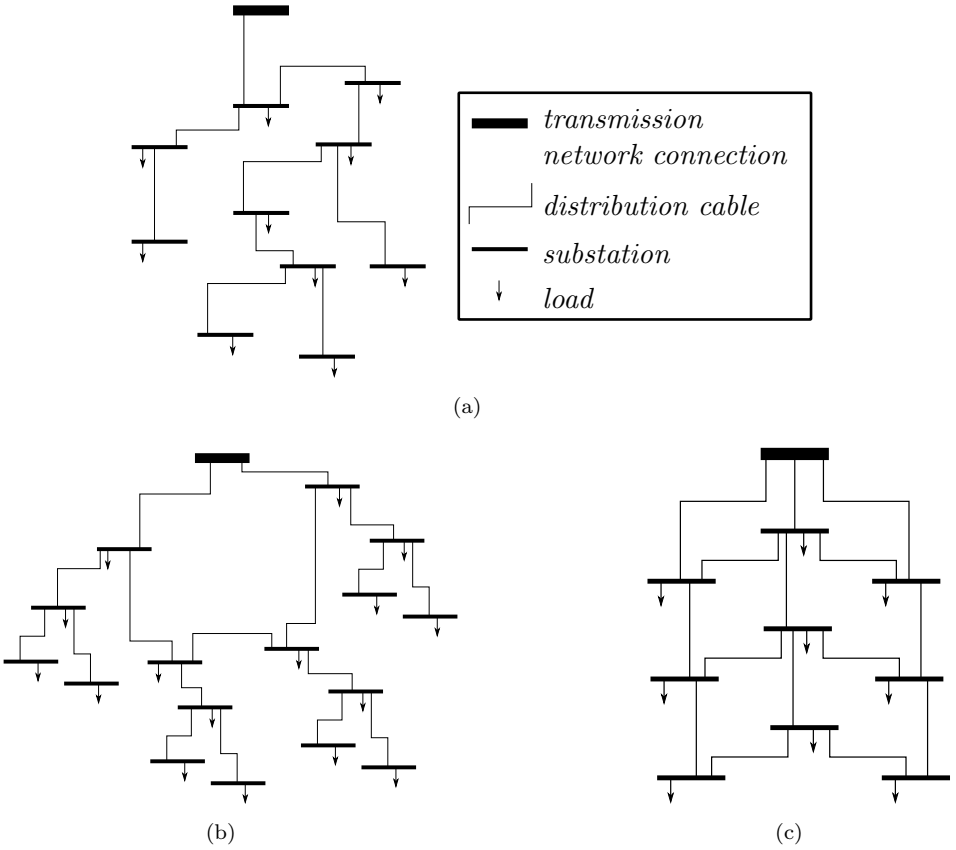


FIGURE 2.1: (a) Radial electricity distribution network. (b) Ring main electricity distribution network. (c) Interconnected electricity distribution network.

2.2.1 Distribution Network Topologies

There are three common types of distribution network topologies, exemplified on Figure 2.1, that are used around the world; radial, ring main, and interconnected (Weedy and Cory, 2004; Gönen, 2007; Grigsby, 2012). Substations are represented by the thick horizontal lines, each with an arrow indicating a load; the thickest substation at the top of each figure is connected to the transmission network. Substations are connected to each other by distribution cables carrying 11kV, represented by the thin vertical lines. Figure 2.1(a) shows a radial distribution network which is predominantly used in rural areas. There is only one path from the step-down transformer, connected to the transmission network, to each load via the connecting substations (i.e., an acyclic network). Loads are relatively small and widely dispersed (5–50kVA per group of houses).

Suburban distribution networks combine rural distribution network topologies into a ring main network, as shown in Figure 2.1(b). A ring main network contains a number of substations in a ring around the step-down transformer connected to the transmission network. The ring is sectionalised[7] so that all substations can still be supplied if a

[7] A sectionalised network contains a number of switches which can be opened to cut off power to a particular section of the network.

distribution cable were to fail, and allows it to be configured into an acyclic network. Loads are higher than in rural networks (2–10MW/mile^2) due to higher density housing.

Finally, Figure 2.1(c) shows an interconnected system used in urban towns and cities with very heavy loadings (up to 100MW/mile^2). Each substation is connected to a number of other substations resulting in a sectionalised interconnected network. Higher 33kV distribution cables are used as well as 11kV in order to supply power to the loads.

Throughout this thesis, we only consider acyclic electricity distribution networks.[8] This assumption can be justified as follows. Ring main and interconnected networks contain cycles but are configured into acyclic networks, using switches, in order to supply power (Weedy and Cory, 2004; Grigsby, 2012). The extra distribution cables are used to ensure uninterrupted supply to all loads in the event of a distribution cable being unable to transmit power.[9] To clarify, consider Figures 2.2(a) and 2.2(b) which are examples of the interconnected electricity distribution network in Figure 2.1(c) configured as acyclic networks. Figure 2.2(a) shows an example configuration where two paths from the transmission network connection to each load pass through separate substations. Figure 2.2(b) shows an example configuration where two paths from the transmission network connection to each load pass through some of the same substations. Switches inside each substation ensure the paths are not connected when both paths pass through the same substation.

Currently, distribution networks contain little automation for the transportation of power.[10] Whilst automation was not necessary in distribution networks when nationwide electricity networks came into existence, as electricity distribution networks grow and become more complex, it becomes increasingly hard for human operators to control all of the factors within the network without more intelligent techniques (Roberts, 2004; Department for Business Enterprise and Regulatory Reform, 2008). In Sections 2.3 and 2.4, and Chapters 3, 4, and 5, we explore the intelligent techniques to enable DNOs to manage their distribution networks efficiently. However, to fully understand the complexity of electricity transmission, distribution, and management, the following section discusses the physics of electricity.

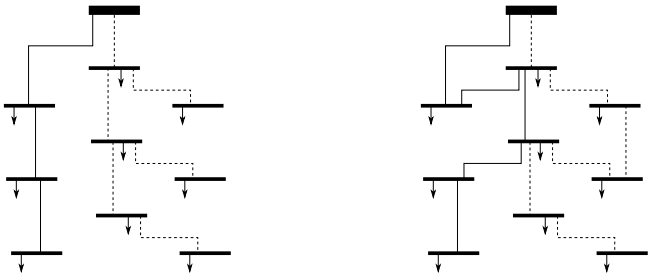
2.2.2 Physics of Electricity in a Network

In a network of generators and loads (represented by nodes) connected by distribution cables, the amount of power flowing through each distribution cable will vary according

[8] We leave cyclic distribution networks for future work, see Chapter 6.

[9] For instance, Kumar et al. (2009) address the problem of configuring cyclic distribution networks into acyclic topologies when a fault has occurred.

[10] It should be noted that the transmission network is already highly automated using supervisory control and data acquisition/generation management system (SCADA/GMS). SCADA/GMS supervises, controls, optimises, and manages large scale generation, and transmission systems. However, distribution networks still lack a lot of this automation (Roberts, 2004).



(a) (b)

FIGURE 2.2: Example of the interconnected electricity distribution network in Figure 2.1(c) configured as two different acyclic electricity distribution networks. 2.2(a) shows an example configuration where two paths from the transmission network connection to each load pass through separate substations. 2.2(b) shows an example configuration where two paths from the transmission network connection to each load pass through some of the same substations. Switches inside each substation ensure the paths are not connected when both paths pass through the same substation.

to Kirchoff's laws.[11] Thus, complications arise (such as overloaded distribution cables) when transporting power through networks. This is because power cannot be sent through a particular distribution cable and must be indirectly manipulated by varying the loads and the generation within the network. Power flows through each available path inversely proportional to the impedance of that path. The following equation links the voltage and the current flowing through the path, with the impedance of the path:

$$E = IZ \tag{2.4}$$

Assuming the voltage stays constant, if the impedance is increased, the current will decrease and vice versa. The consequence of Equations (2.1) and (2.4) is that in a network of generators and loads connected by distribution cables, if one of the distribution cables in the network is overloaded, the power cannot be easily redirected away from the overloaded distribution cable. Instead, in order to change the amount of power flowing through a distribution cable (i.e., the power flow (PF)), the power output of the generators across the network must be changed.

To calculate the PF through each distribution cable in a network of generators and loads, alternating current power flow (AC PF) equations can be used, which consider the nonlinear constraints: balance, branch flow, and generation for real and reactive power (Wood and Wollenberg, 1995). However, due to the nonlinear nature of these calculations, finding a solution is computationally intensive (Overbye et al., 2004). Therefore, in practice, AC PF equations are approximated by more feasible linear direct current power flow (DC PF) equations that use the real power constraints and make a number of simplifying assumptions (Kaye and Wu, 1984), specifically:

[11] The sum of the currents flowing into and out of a node in an electrical circuit is zero (Kirchoff's current law). The directed sum of the voltages in a network is zero (Kirchoff's voltage law).

1. Line losses are assumed to be zero.

2. The difference between two neighbouring nodes' voltage phase angles[12] θ_i and θ_j are assumed to be small, such that $\cos(\theta_i - \theta_j) \approx 1$ and $\sin(\theta_i - \theta_j) \approx 0$.

3. Voltages are assumed to be identically unity using the per-unit system.[13]

Although DC PF is an approximation, results show that it provides a good approximation to the actual AC PF with the advantage that it is much faster to compute (Overbye et al., 2004; Sun and Tesfatsion, 2007). The DC PF equations calculate the power travelling through distribution cables that connect k nodes within an electricity network. In what follows, we denote v_i as node i and t_{ij} as the distribution cable that connects v_i and v_j where $i, j \in \{1, \ldots, k\}$:

$$\mathbf{P} = \mathbf{B}\boldsymbol{\Theta} \tag{2.5}$$

where $\mathbf{P} = \{p_1, \ldots, p_k\}$ is a vector of real resultant power such that $p_i \in \mathbb{R}^+$ kW is the real resultant power for v_i. $\mathbf{B} = [b_{ij}]_{k \times k}$ is a $k \times k$ sparse symmetrical matrix of susceptances for the distribution cables, where $b_{ij} \in \mathbb{R}^-$ siemens is the negative susceptance of t_{ij} for $i \neq j$, $b_{ii} = \sum_{j=0}^{k} |b_{ij}|$ is the sum of the susceptances for row i, and the susceptance of t_{ij} is given by:

$$b_{ij} = \frac{1}{x_{ij}} \tag{2.6}$$

where $x_{ij} \in \mathbb{R}^+\Omega$ is the reactance of t_{ij}. For v_i and v_j that are not connected, $b_{ij} = 0$. Finally, $\boldsymbol{\Theta} = \{\theta_1, \ldots, \theta_k\}$ is a vector of voltage phase angles, where $\theta_i \in \mathbb{R}$ is the voltage phase angle at v_i. The following equation calculates the power flowing through a distribution cable:

$$f_{ij} = b_{ij}(\theta_i - \theta_j) \tag{2.7}$$

where $f_{ij} \in \mathbb{R}$ kW is the power flowing through t_{ij}.[14] $\boldsymbol{\Theta}$ can be calculated by rearranging Equation (2.5) to give:

$$\boldsymbol{\Theta} = \mathbf{P}\mathbf{B}^{-1} \tag{2.8}$$

If one node were to change the power output of its generator, one or more distribution cable power flows would change due to the coupled nature between the power outputs, the node voltage angles, and the susceptances of the distribution cables in Equation (2.5).

In more detail, changing p_i in Equation (2.5) will cause one or more θ to change in $\boldsymbol{\Theta}$ (since $\mathbf{B}$ is fixed for a particular network). Thus, changing θ_i will affect the flow of each

[12] When generating AC electricity, the voltage varies along a sinusoid curve between a positive and negative amount. The voltage phase angle corresponds to the angle of the sinusoid curve.

[13] The per unit system expresses actual values of quantities as fractions of referenced quantities (Weedy and Cory, 2004).

[14] It should be noted that the sign of f_{ij} gives the direction of flow. When $f_{ij} > 0$ the power is flowing from v_i to v_j, and when $f_{ij} < 0$ the power is flowing from v_j to v_i.

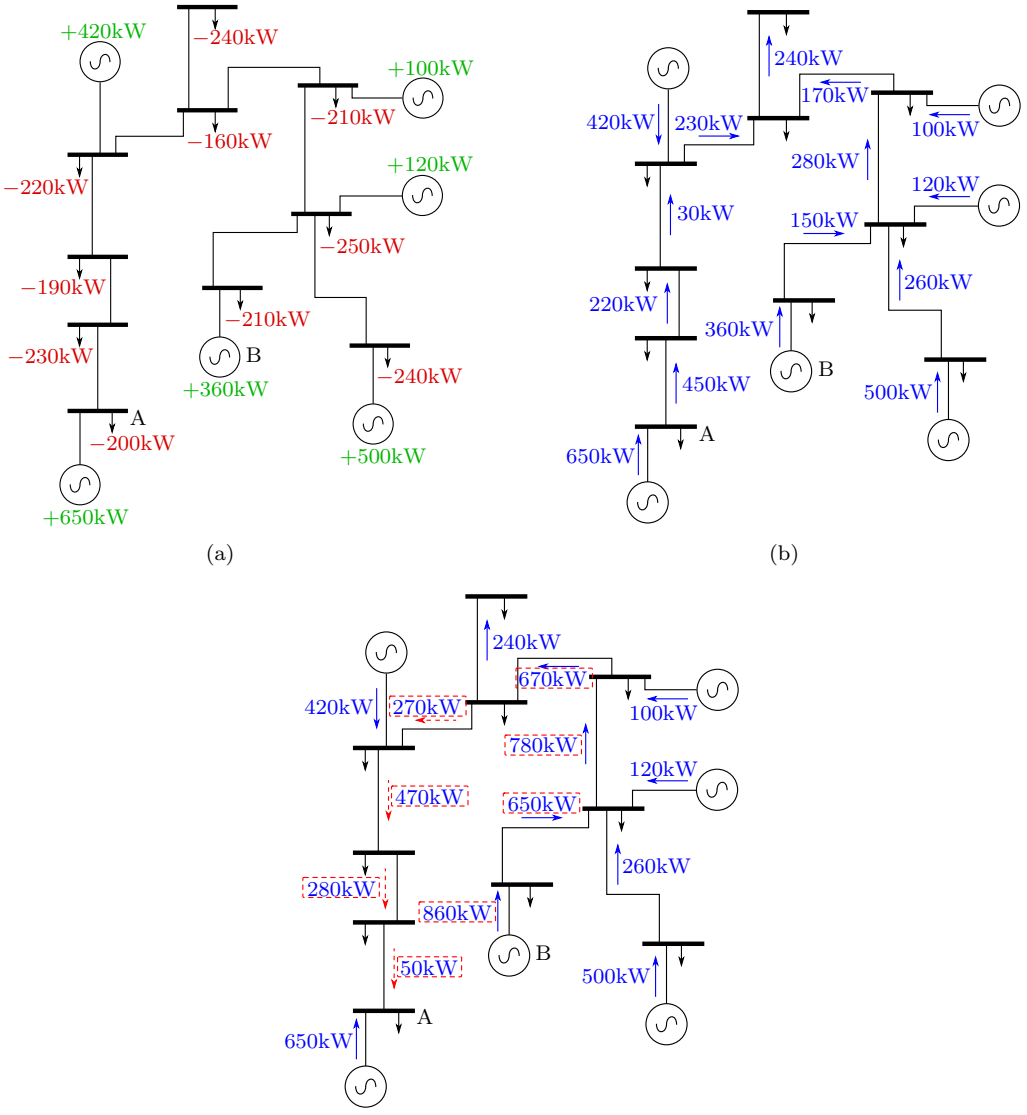


FIGURE 2.3: (a) Consumption and generation within an example electricity distribution network. (b) Resulting distribution cable power flows from (a). (c) Consumption at load A changes to –700kW, generator B increases its power output to compensate. The amount of power travelling along distribution cables that have changed are marked with dashed red boxes, and the power travelling along distribution cables that have changed direction are marked with dashed red arrows.

distribution cable connected to v_i due to Equation (2.7). Figure 2.3 shows an example of how changing the power output of a generator in a network affects the power flows of multiple distribution cables. Figure 2.3(a) shows consumption and generation within an electricity network, and Figure 2.3(b) shows the resulting power flows of the distribution cables from Figure 2.3(a). Figure 2.3(c) shows the new distribution cable power flows if load A changes its consumption to –700kW and generator B increases its power output to compensate. The amount of power travelling along distribution cables that have changed are marked with dashed red boxes, and the power travelling along distribution cables that have changed direction are marked with dashed red arrows. This illustrates the difficulty of coordinating power in an electricity network; changing the power output of one generator in one part of the network can potentially affect power flows in all other parts of the network. Having presented the physics of controlling electricity in a network, as well as the associated difficulties, the following section presents the vision of a more intelligent electricity network capable of managing an increased number of generators, end consumers, and electrical devices, in capacity-constrained networks more efficiently.

2.3 The Smart Grid

To transition to a low carbon economy, the outdated electricity networks will have to be modernised in order to incorporate increased generation capacity and more efficient management of electricity by creating a smart grid. The US Department of Energy (2003) describes a smart grid as:

A fully automated power delivery network that monitors and controls every customer and node, ensuring a two-way flow of electricity and information between the power plant and the appliance, and all points in between. Its distributed intelligence, coupled with broadband communications and automated control systems, enables real-time market transactions and seamless interfaces among people, buildings, industrial plants, generation facilities, and the electric network.

The UK Department of Energy and Climate Change (2009a) identifies the following principles of the smart grid which will need to be implemented in order to achieve a smarter electricity network:

Observable View a wide range of operational statistics including the location of losses in the system and the current condition of equipment in not only the transmission network but also the distribution network.

Controllable Manage and optimise the smart grid to a greater extent than current electricity networks. This will allow network operators to control some demand for electricity depending on the supply available and will facilitate the integration of intermittent renewable generation on a large scale.

Automated The smart grid will be able to 'intelligently' manage the electricity production and consumption in an automated manner. It will also be able to 'self-heal' in the event of a network failure (i.e., it will identify the fault and then reconfigure the network in an optimal manner such that the fault is resolved).

Fully integrated Components on the smart grid will need to be fully compatible with existing electricity networks.

The smart grid separates the electricity network into manageable micro-grids[15] that incorporate smart metering techniques for real time pricing of electricity, DG and micro-storage devices (US Department of Energy, 2003; UK Department of Energy and Climate Change, 2009a). A micro-grid may be connected to major electricity generators, such as nuclear or coal power stations, much like the current system. However, the difference is that each micro-grid can be disconnected and isolated from the main electricity network (i.e., working in island mode) and continue to balance the supply and demand by managing the electricity flow around the decentralised network.

The European smart grids technology platform (European Commission, 2006) and EPRI's IntelliGrid initiative (Chuang and Gellings, 2008) have both proposed smart grids as the key to meeting rising future demands for energy. Therefore, various research groups and consortia are investigating the technologies that would be needed for such a smart grid. A partnership in the UK between seven universities, EDF Energy, ScottishPower and ABB have developed a smart grid called the autonomous regional active network management system (AuRA-NMS). The main direction of research for the AuRA-NMS is voltage control and power flow management.

Voltage control involves managing the voltage of a network to ensure that all the devices and distribution cables that are part of the network operate within safe working limits (Taylor et al., 2008, 2010). Taylor et al. (2010) present a distributed way to maintain safe voltage levels within AuRA-NMS using case based reasoning (CBR) techniques. For each case, a number of precomputed solutions are produced using simulations offline, that describe what measures should be taken to return the network to safe voltage levels. When a voltage excursion is detected, a number of cases are selected which match the current situation. Each solution is verified using power flow calculations and then a single preferred solution is selected to be implemented. If more than one solution exists, then contract agreements between the network operators and the generators are taken into consideration.[16]

Rahman et al. (2007) propose another promising micro-grid management system called the intelligent distributed autonomous power system (IDAPS). This focuses on the

[15] A micro-grid is an electricity network that contains a number of electrical devices as well as end users.

[16] In the UK, these contract agreements are based on a last in first out approach (i.e., generators that were connected more recently will be switched off first).

integration of cleaner and more efficient small-scale generation sources and advanced IP-based communication technologies in order to build a resilient power system. The project conducted a number of experiments where they tested IDAPS in both grid mode (i.e., connected to the main electricity network) and island mode. Results showed that IDAPS could be used to incorporate renewable generation, voltage and frequency control, communication protocols, and fault resolution in an efficient manner (Rahman and Pipattanasomporn, 2010), such that demand within the network is satisfied when working in both grid mode and island mode. By combining these technologies, the IDAPS and the AuRA-NMS projects proved that it is possible to make an electricity network that is able to cope with the increased amount of generation and load that will be experienced in the future without costly network reinforcements.

The above approaches focus on a wide variety of problems associated with implementing a smarter electricity grid, such as communication between devices on a network, small-scale generation, renewable generation, voltage control, and power flow management. This thesis will focus on how an increased amount of generation, particularly from renewable sources, can be coordinated in distribution networks without the need to install additional infrastructure by using ANM. The following section introduces ANM along with the state of the art approaches to implement such a system.

2.3.1 Active Network Management

The current practice for adding generation to electricity networks generally involves constructing new circuits (network reinforcements) to increase network capacity. Whilst this is essential, installing network reinforcements can be time consuming and may have significant monetary and environmental costs (Roberts, 2004). To alleviate the need for network reinforcements, generators can be connected to the electricity network with additional constraints, such as voltage limits and thermal overloads, that can be applied when the system capacity is restricted.

For each generator that is added in this way, a number of predetermined 'hard-wired' intertrip schemes have been developed to decrease the power output of the generators in the event of a network failure. Whilst this is a solution for individual generators, as more generators are added, these predetermined schemes become increasingly complex and quickly become infeasible to implement (Roberts, 2004). Moreover, renewable generators that are added to electricity networks create further problems since they use resources that are intermittent. Thus, balancing supply with demand using renewable generators will be more complex than using higher CO_2 emitting coal or gas generators that are not intermittent in nature and more controllable. Therefore, to minimise the cost of network reinforcement and CO_2 emissions, Roberts (2004) suggest the use of ANM that incorporates dynamic schemes for coordinating generators in an electricity network. When calculating what power output each generator should have, there is often an

associated cost.[17] Therefore, finding the optimal configuration that minimises the cost of the network is often desired. This type of problem is referred to in the literature as optimal power flow (OPF) (Kaye and Wu, 1984; Sun and Tesfatsion, 2007).

As with PF, alternating current optimal power flow (AC OPF) can be approximated by a more feasible direct current optimal power flow (DC OPF) using Equations (2.5) – (2.8) (Kaye and Wu, 1984). Sun and Tesfatsion (2007) provide a comprehensive study of the accuracy of DC OPF compared with AC OPF and conclude that DC OPF is much faster to compute and is a good approximation for AC OPF. The following section discusses and compares the current techniques for coordinating generators in an electricity network, in order to minimise the cost of the network, using centralised, as well as decentralised and distributed, approaches.

2.3.2 Coordination of Generators

Using a central authority to calculate the power output of each generator requires the data from every device connected to the network to be transfered to the central authority. The advantage of having a complete view of the data is that the central authority is able to calculate the optimal solution. For example, Davidson et al. (2009) use constraint programming (CP) to change the power outputs of the generators in the transmission network in order to reduce the power travelling through an overloaded transmission line subject to a number of constraints; such as ensuring the capacity of each transmission line is not exceeded. For a given situation, there may be many different solutions for each generator that meets the constraints of the network. In this case, Davidson et al. (2009) use a number of preference constraints which attempt to maximise the use of DGs subject to contract agreements for each generator. CP has an advantage over OPF because it can offer a number of ranked solutions based on the contract constraints. If one solution does not reduce the overloaded line to within a safe working level, the next best solution can be used instead. Typically, OPF simply gives one solution to the problem and if that solution cannot satisfy all the constraints in the network then the resulting management of the generators may fail.

The central control of generators, however, presents a number of issues. For a central control system to work, large amounts of data must be transmitted to a central location, manipulated, and then sent back to each generator. As the electricity network size increases, the amount of data that must be transmitted will increase as well; eventually it may actually be infeasible to transmit that amount of data due to the capacity and quality of the communication channels (Granada et al., 2008). As such, a centralised approach may have significant scalability issues for large networks (Platt, 2007; Granada et al., 2008; Kumar et al., 2009). Moreover, a centralised control system creates a single point of failure (Roberts, 2004; Solanki et al., 2007), which could be detrimental to the

[17] For instance, cost in terms of CO_2 emissions, fuel consumption, or line losses.

security of electricity supply with respect to reliability (Platt, 2007; Kumar et al., 2009). If the centralised control system was to fail (in terms of communication error, processor malfunction, or power supply loss for instance), each generator would be unable to determine what it should efficiently output.[18] Furthermore, a large change in the total demand (i.e., larger than can be satisfied by just following the frequency signal), could result in significant problems if the generator power outputs are not changed accordingly; problems such as overloaded distribution cables which could lead to complete loss of power throughout the entire electricity network.

Local environmental conditions (such as fluctuating wind, cloud cover, and rain) can potentially create another problem to centralised techniques. Although considering the change of electricity demand over *time* is beyond the scope of this thesis, future systems will need to take full advantage of renewable generators in order to ensure the reduction of CO_2 emissions. Using a central system to predict the availability of the environmental resources that renewable generators harness, as well as controlling generators at remote locations, requires large amounts of data to be transmitted periodically to the central system. A centralised approach may be unable to respond to very localised changing conditions in a timely fashion (i.e., in a matter of seconds) due to the complete control loop that is necessary to control the generators centrally (Department for Business Enterprise and Regulatory Reform, 2008).

Thus, in order to address the reliability and scalability issues, and the need for localised control, a decentralised[19] and distributed[20] approach to electricity network control will be needed (Solanki et al., 2007; Granada et al., 2008; Kumar et al., 2009). In terms of electricity generator control, the decentralised and distributed coordination of generators involves dividing the computation and information required to calculate the optimal power output for each generator, among the nodes in the network.[21] The nodes communicate with their respective neighbours (i.e., the nodes that they are connected to via a distribution cable) in order to decide the level of output required to balance the loads while respecting distribution cable capacities. Doing so provides the following advantages over centralised techniques.

The decentralised and distributed control addresses the reliability issue associated with centralised control (Platt, 2007; Kumar et al., 2009). If the computing capabilities of one of the nodes fails, the rest of the network may still be able to compute an

[18] Most generators in electricity networks are synchronously connected and hence all produce power at the same frequency (Weedy and Cory, 2004). This means that generators can to some extent satisfy the demand in the network by just using this frequency signal. However, this process is not optimal when considering additional costs of the network (for instance CO_2 emissions).

[19] Decentralised computation does not use one central entity to compute a solution but uses multiple entities that can each compute part of a solution (often in parallel). Note, this computation could still be within one machine on multiple processors or cores.

[20] Distributed computation requires the processing entities to be physically distributed throughout a network.

[21] Each node would have some computing capability in order to control the generators it is connected to.

optimal solution (provided that the required information from the failed node can be computed by another neighbouring node) because the computation does not solely rely on a single entity (Granada et al., 2008). Even if a neighbouring node is unable to receive the required information from the failed node, then a suboptimal solution can still be calculated.[22] In contrast, if the computer of a centralised system were to fail, each generator would be unable to determine the optimal output with respect to its neighbouring generators. This could, in the worst case, lead to overloaded distribution cables.

Finally, by distributing the computation, each distributed node can monitor the local conditions of the network (for instance, environmental changes to the resources available), and update its neighbouring nodes via small amounts of data (Granada et al., 2008). Since nodes only send data to their neighbours, and perform the computation locally, the control loop (i.e., from a change in the local conditions to performing the required actions) is small. This is in contrast with a centralised approach which would require a node to transmit each local change to the central controller, wait for it to be processed, and then receive a control action back (Department for Business Enterprise and Regulatory Reform, 2008; Granada et al., 2008). Furthermore, a distributed approach allows local networks to grow without the need to notify the centre of additional nodes. Having discussed the advantages of distributing and decentralising generator control in an electricity network, the rest of this section details the current state of the art in the literature for doing so.

Kim and Baldick (1997) first presented the notion of decentralised generator coordination using Lagrangian techniques (Wu et al., 1994). The problem is decomposed into regions which contain a number of generators, loads, and distribution cables by using the auxiliary problem principle (APP) (Cohen, 1980). Each region iteratively communicates with its neighbours, via the distribution cables spanning the regions, in order to solve the global OPF. Two neighbouring regions construct a dummy bus[23] that holds a number of variables: real and reactive power, voltage magnitude, and phase angle. The dummy bus variables are duplicated between two regions and each region iteratively updates its values by exchanging messages until duplicate variables from both regions converge to within some tolerance. This process is performed in parallel for all regions and the global OPF is calculated for the entire network. To avoid the need to construct a dummy bus, thus removing the need to perform costly fine tuning of the dummy bus parameters, Bakirtzis and Biskas (2003) decouple the OPF problem by using the distribution cables to separate the OPF parameters between regions. However, both the above techniques have only been tested on problems containing up to six regions. Thus, it is unclear whether they will scale well when applied to larger electricity networks.

[22] In this case the node with the failed generator can be removed from the network topology (using switches), and two subnetworks can be created which can still calculate what each generator should output.

[23] A bus in this context can be thought of as a node.

Recently, Kraning et al. (2013) presented a novel approach to decomposing an electricity network into subproblems using similar techniques to the APP. They use alternating direction method of multipliers (ADMM) to achieve both separability and robustness for distributed optimisation. Moreover, using ADMM allows them to control the direction of convergence without external parameters; the parameters are implicitly embedded in ADMM. They mention that they can solve a network size of 100,000 nodes in 5 minutes on average using a centralised implementation of their algorithm. Although they have not created a working decentralised implementation of their algorithm, they project that it will take roughly 200ms to solve the network of 100,000 nodes. However, no concrete results are presented.

In contrast to using Lagrangian techniques, Kumar et al. (2009) introduce a message passing technique which extends distributed pseudotree optimisation procedure (DPOP) (Petcu and Faltings, 2005) for reconfiguring feeder trees within a distribution network (a related area of research). DPOP is a dynamic programming based algorithm which solves DCOP problems. Kumar et al. (2009) apply their algorithm to the distribution network and decompose the network into overlapping acyclic regions such that there is a single path from a generator to loads. Experiments show that their algorithm is able to overcome the limitations of other resource constrained algorithms applied to multiple constraint DCOP problems (Bowring et al., 2006), such as asynchronous distributed constraint optimisation (ADOPT) (Modi et al., 2005), because DPOP is able to exploit the structure and topology of the network. However, it does not address the problems highlighted above of incorporating an increasing amount of DGs in the distribution network, and the need to coordinate their output.

Having introduced the current literature on generator coordination in electricity networks, and discussed the advantages of distributing and decentralising control, the following section introduces the necessary theory for formulating problems as a distributed constraint optimisation problem.

2.4 Distributed Constraint Optimisation Problems

This section presents the formal definition of a DCOP (Yokoo and Durfee, 1991; Rossi et al., 2006). A DCOP is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \mathcal{A} \rangle$ consisting of a set of h variables $\mathcal{X} = \{x_1, \ldots, x_h\}$ which can be assigned discrete values in the set of finite domains $\mathcal{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_h\}$ respectively, and $\mathcal{F} = \{F_1, \ldots, F_k\}$ is the set of functions (also called constraints) where $F_i : \mathbf{d}_{i1} \times \ldots \times \mathbf{d}_{ij} \rightarrow \mathbb{R}^+$ denotes the cost of each possible combination of the involved variable values. We denote $\mathcal{A} = \{A_1, \ldots, A_k\}$ as the set of k agents where each variable x_i is assigned to an agent. In our representation of a DCOP, only the agent that is assigned the variable has knowledge of its domain and control over its

value. Moreover, the constraint F_i corresponds to the utility of agent i. For instance, in the case of an electricity network, each agent could have control of a generator.

The goal of the agents is to find an assignment $\mathcal{X}^*$ for the variables in $\mathcal{X}$ that minimises the sum of the functions:

$$\arg\min_{\mathcal{X}^*} \sum_{i=0}^{k} F_i \tag{2.9}$$

A DCOP can be represented by a factor graph, whose vertices correspond to variables and the edges denote the dependencies between the variables. The advantage of using a factor graph is that the GDL family of algorithms provide a message passing framework which can be applied to DCOPs using factor graphs. Therefore, in Section 3.2, we show how a typical electricity network can be transformed into a factor graph. The following section introduces the GDL family of algorithms, and specifically the max-sum message passing algorithm.

2.4.1 The Generalised Distributive Law

The generalised distributive law is based on the distributive law, which states that $ab + ac \equiv a(b+c)$. The left side of this equation involves one addition and two multiplications. However, the right side of the equation involves only one addition and one multiplication (since it has been factorised). Therefore, the distributive law gives a fast way (i.e., fewer arithmetic operations) of computing $ab + ac$. By generalising the distributive law (hence GDL), Aji and McEliece (2000) present a large family of fast algorithms that are able to exploit factorisable problems in order to solve them quickly.[24] A DCOP that has been decomposed into a factor graph can be solved by applying one of the algorithms from the GDL family; namely max-sum (or min-sum) (Farinelli et al., 2008; Rogers et al., 2011).[25]

Max-sum can be seen as a decentralised and distributed version of the bucket elimination algorithm (Dechter, 1996), which is based on the generalised forms of variable elimination and nonserial dynamic programming. Bucket elimination can be used to solve a wide variety of problems including adaptive consistency for constraint satisfaction (Dechter and Pearl, 1987), dynamic programming for combinatorial optimisation (Bertele and Brioschi, 1972), and directional resolution for propositional satisfiability (Davis and Putnam, 1960). Bucket elimination splits the problem up into a number of buckets, and processes these buckets in a given order. Each time a bucket is processed, the result is placed in an unprocessed bucket until all buckets have been processed.

[24] In order to generalise the distributive law, Aji and McEliece (2000) use commutative semirings. A commutative semiring is a set K with two binary operations called "+" and "." which can be substituted for a variety of operations. For instance, the set of real or complex numbers with ordinary addition and multiplication is a semiring (i.e., $(a.b) + (a.c) \equiv a.(b + c)$).

[25] We use the term max-sum to encompass both max-sum and min-sum as they are identical except for the maximisation or minimisation of the objective function respectively.

As an example, consider the context of constraint satisfaction, whereby the value of each variable must satisfy a number of constraints. Each bucket is a set of constraints for a particular variable. Processing a bucket involves eliminating the variable from the bucket's constraints, and adding the new constraints to the other buckets. Once all buckets have been processed, the variable values can be calculated. Bucket elimination was first proposed as a centralised algorithm. However, as well as max-sum, a number of other algorithms have also extended the bucket elimination framework to solve DCOPs; algorithms such as DPOP (Petcu and Faltings, 2005), D–DYDOP (Chapter 4), and C–DYDOP (Chapter 5). In the following section we formally describe the max-sum algorithm.

2.4.2 Max-sum

Farinelli et al. (2008) were the first to show how to apply max-sum to maximise the sum of utilities of all the agents in a network. Max-sum scales well with the size of the network, since the size and number of messages sent, is only dependent on a local neighbourhood. Max-sum works by factorising the network into a bipartite graph that consists of variables and functions (utilities) assigned to agents. Each variable has a number of discrete possible values with a certain utility that is affected by their neighbour's variable values. Each agent exchanges messages with its immediate neighbours by making local decisions to maximise their local utilities. In so doing, the global utility is maximised. In more detail, the agents communicate with each other by sending messages from variable to function, and from function to variable as follows:
From variable to function:

$$Q_{b\to a}(x_b) = \sum_{a'\in A(b)\backslash a} R_{a'\to b}(x_b) \tag{2.10}$$

From function to variable:

$$R_{a\to b}(x_b) = \min_{\mathcal{X}_a\backslash b}\left[F_a(\mathcal{X}_a) + \sum_{b'\in B(a)\backslash b} Q_{b'\to a}(x_{b'})\right] \tag{2.11}$$

Let $B(a)$ be the set of variables connected to the function a, $A(b)$ be the set of functions connected to the variable x_b, $F_a(\mathcal{X}_a)$ be the utility of agent a when each variable in $B(a)$ has a certain value, and finally $\mathcal{X}_a\backslash b \equiv \{x_{b'} : b' \in B(a)\backslash b\}$. To find the optimal value for each variable x_b, an agent uses the following equation:

$$\arg\max_{x_b} Z_b(x_b) \tag{2.12}$$

where $Z_b(x_b)$ is the sum of R messages flowing into x_b:

$$Z_b(x_b) = \sum_{a \in A(b)} R_{a \to b}(x_b) \tag{2.13}$$

Max-sum is provably optimal and guaranteed to converge when applied to acyclic graphs (Aji and McEliece, 2000; Farinelli et al., 2008; Vinyals et al., 2010). Whilst only limited theoretical results exist for applying max-sum to cyclic graphs, there exists extensive empirical evidence of its effectiveness on such graphs (Aji et al., 1998; Weiss, 2000; Farinelli et al., 2008; Vinyals et al., 2010; Winsper and Chli, 2012). Therefore, max-sum presents a compelling framework to apply to the electricity domain.[26] In Chapter 3, we show how max-sum can be used to solve the optimal dispatch problem, and in Chapters 4 and 5 we benchmark our novel message passing algorithms (D–DYDOP and C–DYDOP) against max-sum.

As an alternative to discretising the search space, Stranders et al. (2009) present an extension to max-sum which uses continuous variable values.[27] This is particularly applicable in areas where the variables involved cannot or should not be discretised (i.e., problems involving velocities, orientation, or location). The following section provides a summary of the current state of the art research related to this thesis.

2.5 Summary

A key challenge in the delivery of a more efficient electricity network is how additional generation can be added to the smart grid without using costly network reinforcements. Roberts (2004) suggests that ANM is the key to adding increased generation within electricity networks without using complex predetermined intertrip schemes or network reinforcements. There are a number of centralised and decentralised systems which coordinate the power outputs of the generators using ANM.

Using a centralised approach, Davidson et al. (2009) use constraint satisfaction techniques to calculate generator power outputs, which conform to flows within the network and contracted agreements. Whilst this calculates an optimal solution, it uses a centralised approach to coordinate generators for which we have discussed the disadvantages in Section 2.3.2. Thus, using this technique does not address Requirements III and V since the system may have scaling issues when applied to larger networks, and is not distributed or decentralised. Moreover, it assumes that complete information about each generator is known centrally which might not always be possible.

[26] However, applying max-sum to cyclic electricity networks does not work because the resulting optimisation problem cannot be decomposed into subproblems to be solved independently using traditional DCOP techniques. Dealing with cyclic networks remains a key challenge for the general DCOP research community and is beyond the scope of this thesis, see Chapter 6.

[27] Chapter 5 uses the notion of piecewise linear functions in (Stranders et al., 2009) and extends it to be used with C–DYDOP

In contrast, distributed constraint optimisation techniques have been developed to solve the related area of research for reconfiguring feeder trees within a distribution network using an extension of DPOP (Kumar et al., 2009). However, they do not address the need to incorporate an increased amount of generation into the distribution network whilst reducing CO_2 emissions. Thus, their techniques do not satisfy Requirement III, when applied to a large network, or Requirement II.

To address the challenge of producing decentralised and distributed algorithms that scale well with the size of the network, we present a novel extension of max-sum, called D–DYDOP, that coordinates the power outputs of generators whilst minimising CO_2 emissions, in Chapter 4.[28] We benchmark our approach on real distribution network topologies against both an optimal centralised approach, based on a mixed integer program solver, and a naïve implementation of max-sum (presented in Chapter 3). In Chapter 5 we extend D–DYDOP to the continuous domain, called C–DYDOP, which uses continuous variables for the generator power outputs and the distribution cable flows. The advantage is that C–DYDOP does not suffer from the discretisation of the search space, since D–DYDOP must iterate through every possible combination of discrete values when calculating messages. We benchmark C–DYDOP against D–DYDOP and the centralised approach on the same distribution network topologies.

[28] Our algorithm uses dynamic programming techniques that have been used in other DCOP algorithms such as DPOP.

Chapter 3

A Benchmark Algorithm for Decentralised Optimal Dispatch in Acyclic Electricity Networks Using Discrete Variables

Against the background highlighted in Chapter 2, this chapter addresses the challenge of coordinating large numbers of DGs, embedded in the distribution network, by providing a novel formalism of the optimal dispatch problem as a decentralised agent-based coordination problem, represented as a DCOP. We show how this DCOP can be decomposed as a factor graph and solved using algorithms based on the GDL framework (Aji and McEliece, 2000), such as max-sum (Farinelli et al., 2008). In more detail, each node in the network is represented by an agent that undertakes some of the computation required to solve the optimal dispatch problem; such that demands within the network are satisfied and CO_2 emissions of the entire network are minimised. In particular, we solve the optimal dispatch problem on the most common distribution network topologies, namely acyclic networks (see Section 2.2.1). We use the max-sum algorithm as a benchmark for testing our novel message passing algorithms in Chapters 4 and 5. Crucially, our algorithms handle the complexities of balancing flows within the network, *without needing central verification* of a particular solution.

The remainder of this chapter is organised as follows: Section 3.1 introduces the electricity network model that is used by this chapter, and Chapters 4 and 5, in order to solve the optimal dispatch problem. In Section 3.2 we detail our novel formalism of the optimal dispatch as a DCOP. We show how this DCOP can be decomposed as a factor graph and solved using algorithms based on the GDL family, such as the max-sum algorithm in Section 3.3. Section 3.4 presents the necessary techniques for discretising

the electricity distribution network constraints so that a discrete algorithm can actually be applied to the optimal dispatch problem. Finally, Section 3.5 concludes.

3.1 Electricity Network Model

In this section, we formally describe the model of an electricity network for which we need to solve the optimal dispatch problem. Hence, we consider an electricity distribution network to be a network of generators, loads, and distribution cables. In this network, we consider a set of n generators $\mathbf{G} = \{g_1, \ldots, g_n\}$. Each generator g_i has a certain power output variable $\alpha_i \in \mathbb{R}^+$ kW, which is bounded by $\alpha_i^{min} \in \mathbb{R}^+$ and $\alpha_i^{max} \in \mathbb{R}^+$ such that $\alpha_i^{min} \leq \alpha_i \leq \alpha_i^{max}$.

In this model we consider different classes of generators. Continuous generators capable of producing any amount of power between α_i^{min} and α_i^{max} (i.e., diesel or biomass generators), and discrete generators capable of producing power at set intervals between α_i^{min} and α_i^{max} (i.e., wind turbines or solar panels can either produce the maximum amount of power available given the current environment, or be switched off). Thus, each discrete generator has a set of power output levels which it can produce such that $\alpha_i \in S_i$ kW, where $S_i = \{s_1^i, \ldots, s_{q_i}^i\}$, $s_j^i \in \mathbb{R}^+$ and $q_i \in \mathbb{Z}^+$ is the number of power output levels for generator g_i. Let $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_n\}$ denote the set of power output variables for the generators in $\mathbf{G}$. Let $e_i = CI_i\alpha_i$ denote the CO_2 emissions that are produced when g_i, with carbon intensity $CI_i \in \mathbb{R}^+\text{kgCO}_2/\text{kWh}$, outputs α_i.

We consider a set of m loads $\mathbf{L} = \{l_1, \ldots, l_m\}$. Each load l_i has a certain power consumption $\beta_i \in \mathbb{R}^-$ kW, where $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_m\}$ is the set of power consumption variables for the loads in $\mathbf{L}$. We denote $\mathbf{V} = \{v_1, \ldots, v_k\}$ as the set of k nodes within the network. A node relays power to other nodes but can also contain a combination of generators and loads. Let $adj(v_i)$ denote all nodes that are connected to v_i via a distribution cable, let $\mathbf{L}(v_i)$ be the set of loads that are at v_i and $\mathbf{G}(v_i)$ be the set of generators that are at v_i.

$\mathbf{T}$ is the set of s distribution cables within the network, where $t_{ij} \in \mathbf{T}$. Each distribution cable has an associated thermal capacity $t_{ij}^c \in \mathbb{R}^+$ kW, which is the maximum power the cable can safely carry. It should be noted that we assume that all the distribution cables have the same reactance.

Finally, $\mathbf{W}(\mathbf{V}, \mathbf{T})$ is a finite undirected graph describing a network of nodes and distribution cables. $\mathbf{F}$ is the set of all power flows $f_{ij} \in \mathbb{R}$ kW along the distribution cables in the network. Given the above definitions, the optimal dispatch problem, of finding an allocation of power outputs $\boldsymbol{\alpha}$ whilst minimising CO_2 emissions,[1] is defined as per the

[1] In this thesis, our objective function minimises CO_2 emissions as part of the optimal dispatch problem. Typically, optimal dispatch (also referred to as economic dispatch) is concerned with minimising the monetary cost of running the generators in the network (Fink et al., 1969; Ross and Kim, 1980). We are simply using a different metric of cost in terms of CO_2 emissions instead of generator running costs.

following objective function:

$$\arg\min_{\boldsymbol{\alpha}} \sum_{i=0}^{n} CI_i\alpha_i \tag{3.1}$$

subject to the following constraints:

Constraint 1 The flow along a distribution cable cannot exceed its capacity:

$$|f_{ij}| \leq t_{ij}^{c} \tag{3.2}$$

Constraint 2 The net flow from v_i to v_j must be the opposite of the net flow from v_j to v_i:

$$f_{ij} = -f_{ji} \tag{3.3}$$

Constraint 3 The sum of the outputs from each generator at v_i, the sum of the loads at v_i and the net flow from each node w connected to v_i is zero:

$$\sum_{w \in adj(v_i)} f_{wi} + \sum_{l \in \mathbf{L}(v_i)} \beta_l + \sum_{g \in \mathbf{G}(v_i)} \alpha_g = 0 \tag{3.4}$$

The optimal dispatch problem presented here can be solved using a mixed integer programming (MIP) solver; in this thesis we use IBM's ILOG CPLEX 12.2. However, simply solving this MIP with CPLEX uses a centralised approach. As the complexity and size of the distribution networks grow, using a centralised MIP approach may suffer from a number of the issues highlighted in Section 2.3.2. Thus, in the remainder of this chapter, and Chapters 4 and 5, we present novel message passing techniques which can solve the optimal dispatch problem in a distributed and decentralised fashion. However, in order to solve the optimal dispatch problem using distributed and decentralised techniques, the following section decomposes the optimal dispatch problem into a DCOP.

3.2 DCOP Representation

Using the notation introduced in Section 2.4, this section shows how we decompose the optimal dispatch problem, as defined in Equations (3.1) – (3.4), into a DCOP represented as a factor graph. Crucially, we provide a mapping of the DCOP to a factor graph that preserves the acyclic topology of the electricity network. Moreover, this mapping allows the optimal dispatch problem to be calculated in a fully decentralised way, without needing centralised verification, whilst balancing all of the loads with generation, and satisfying the constraints of the distribution cables and generators.

Figure 3.1(a) shows an example electricity distribution network consisting of distribution cables, generators, and nodes. Example values for the power output range and carbon

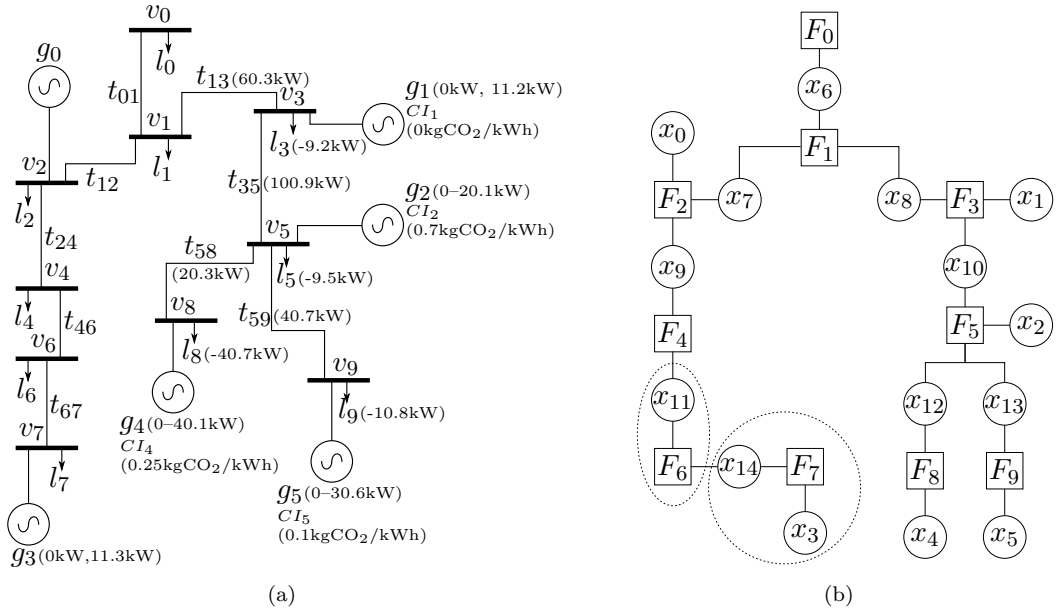


FIGURE 3.1: (a) An electricity distribution network. Showing example values for the power output range and carbon intensity of the generators, thermal capacity of the distribution cables, and power consumption at the loads. Node v_0 is connected to the rest of the electricity grid. (b) The corresponding factor graph representation of the electricity distribution network, showing variables x_i and the functions F_j between dependent variables connected by edges. The dashed circles give an example of the agents with the variables they control.

intensity of each generator, thermal capacity of the distribution cables, and power consumption at the loads are given. Node v_0 is connected to the rest of the electricity grid. In our representation of a DCOP, x_i corresponds to either a generator power output or a distribution cable flow, $\mathcal{X} = \boldsymbol{\alpha} \cup \mathbf{F}$ where $\mathcal{X}$ is the set of h variables. The corresponding domain $\mathbf{d}_i$ of x_i is:

$$\mathbf{d}_i = \begin{cases} S_i & \text{if } x_i \in \boldsymbol{\alpha} \\ \{-t^c_{ab}, \ldots, t^c_{ab}\} & \text{if } x_i \in \mathbf{F} \end{cases} \tag{3.5}$$

For clarification, if x_i corresponds to a distribution cable flow f_{ab}, the domain of x_i is $\{-t^c_{ab}, \ldots, t^c_{ab}\}$. The function $F_i \in \mathcal{F}$ corresponds to the utility of agent i. With regard to our formulation of an optimal dispatch problem, F_i maps to the CO_2 emissions of v_i with respect to the constraints of the network (i.e., a lower cost means lower CO_2 emissions):

$$F_i = \begin{cases} \sum\limits_{g \in \mathbf{G}(v_i)} CI_g \alpha_g & \text{if Equation (3.4) holds for } v_i \\ \infty & \text{otherwise} \end{cases} \tag{3.6}$$

where ∞ is used to penalise variable values that lead to inconsistent flows within the network (i.e., when Equation (3.4) is not satisfied). With this in mind, the objective function of the optimal dispatch problem, Equation (3.1), can be factorised in terms of the agent utility functions using Equation (3.6). The goal of the agents is to find an

assignment $\mathcal{X}^*$ for the variables in $\mathcal{X}$ that minimises the CO_2 emissions of the network using Equation (2.9).

Figure 3.1(b) shows the corresponding factor graph of the electricity network in Figure 3.1(a). Note that by using our decomposition of the optimal dispatch problem to a DCOP, an acyclic electricity distribution network has a corresponding acyclic factor graph. Once an electricity distribution network has been decomposed into a factor graph, the optimal dispatch problem can be solved using an algorithm from the GDL family, such as max-sum.

We choose max-sum to solve the DCOP because max-sum maps directly onto a factor graph, and directly works with n-ary constraints (i.e., functions connected to more than two variables, see F_5 on Figure 3.1(b) for an example) without any additional modifications. This property is particularly attractive because electricity distribution networks often contain a large number of nodes with high branching factors. As discussed in Section 1.2, other algorithms exist for solving DCOPs, such as ADOPT, or optimal asynchronous partial overlay (OptAPO), but they have a number of drawbacks (Farinelli et al., 2008). For instance, OptAPO uses mediator agents which may need to perform calculations that grow exponentially with the size of the subproblem for which they are responsible. Similarly, agents using ADOPT will exchange an exponential number of messages depending on the height of the problem tree. Max-sum, on the other hand, will only perform calculations that grow exponentially with the number of variables which each factor graph function depends on; incidentally, this is much smaller than the subproblems in OptAPO. Moreover, the number of messages exchanged when using max-sum grows linearly with the number of agents in the system (Farinelli et al., 2008, 2009). The following section explains how the max-sum algorithm, introduced in Section 2.4.2, can be applied to solve the optimal dispatch problem.

3.3 Max-sum Optimal Dispatch

As described in Section 2.4.2, the max-sum algorithm (or min-sum as is the case with minimising CO_2 emissions) uses message passing in order to propagate the utilities of the variables around the factor graph using Equations (2.10) – (2.13). In max-sum, functions and variables can be arbitrarily assigned to any agent. However, in our model each agent is assigned the computation of one function which is associated with a specific node within the network. Moreover, a natural assignment of variables to agents involves an agent controlling the generator variables at its designated node, and the distribution cable variables connected to its node. If two or more agents' functions share the same variable, the variable is arbitrarily assigned to one of them, as shown in Figure 3.1(b) by the dashed circles.

More importantly, since max-sum has been proven to converge to an optimal solution on acyclic factor graphs, and given that we provide a mapping from an acyclic electricity network to an acyclic factor graph, max-sum will be able to calculate the optimum solution to the optimal dispatch problem. The utility of each agent F_i (used in Equation (2.11) to calculate the value of a function to variable message and denoted $F_i(\mathcal{X}_i)$), is calculated using Equation (3.6). In order for an agent to choose the optimal output for each generator it controls, it sums all the messages it receives from neighbouring variables using Equation (2.13), and then using Equation (2.12), chooses the combination of generator power outputs which have minimum CO_2 emissions. It should be noted that the optimal solution taken by each agent is exactly the same as the optimal solution calculated using the MIP technique detailed in Section 3.1. In what follows, we describe what each message, from function to variable and from variable to function, means in terms of the optimal dispatch problem.

A max-sum message sent from function to distribution cable variable is a function of the flow in the cable with its domain bounded by the thermal capacity of the distribution cable. In order to apply a discrete algorithm to the electricity distribution network in Figure 3.1(a), the network must first be discretised. A full discussion on the need to discretise the electricity distribution network constraints, and the techniques for doing so, are presented in Section 3.4. For now, consider the following example from Figure 3.2, which is the discretised version of the electricity distribution network in Figure 3.1(a) (when the discretisation unit $\omega = 1$kW). Let the distribution cable t_{59} between v_5 and v_9 have a thermal capacity t^c_{59} of 40kW, the load l_9 at v_9 be –11kW, and the generator g_5 at v_9 have a power output range of 0–30kW. The message $R_{9\to13}(x_{13})$, sent from F_9 to x_{13} on the corresponding factor graph, Figure 3.1(b), will have domain $x_{13} \in \{-40, \ldots, 0, \ldots, 40\}$, having 81 utility values corresponding to the 81 variable values, when $\omega = 1$kW. A negative variable value indicates that the power is travelling from v_5 to v_9, and a positive variable value indicates that the power is travelling from v_9 to v_5.

A max-sum message sent from function to generator variable is bounded by the minimum and maximum output of the generator. Consider the following example. Let the generator g_5 at v_9 have a power output range of 0–30kW. The message $R_{9\to5}(x_5)$ will have domain $x_5 \in \{0, \ldots, 30\}$, having 31 utility values corresponding to the 31 variable values. Each possible value indicates the amount of power α_5 that g_5 is producing. Messages are propagated around the factor graph until the values of the messages converge. Messages are guaranteed to converge to the optimal solution on acyclic graphs, at which point each variable chooses its optimal value using Equation (2.12). However, simply applying the max-sum algorithm naïvely in this manner produces poor performance. This is because much of the search space is infeasible and does not need to be searched. For instance, consider the previous example for the message $R_{9\to13}(x_{13})$. The message has a total of 81 variable values. However, the maximum amount of power that

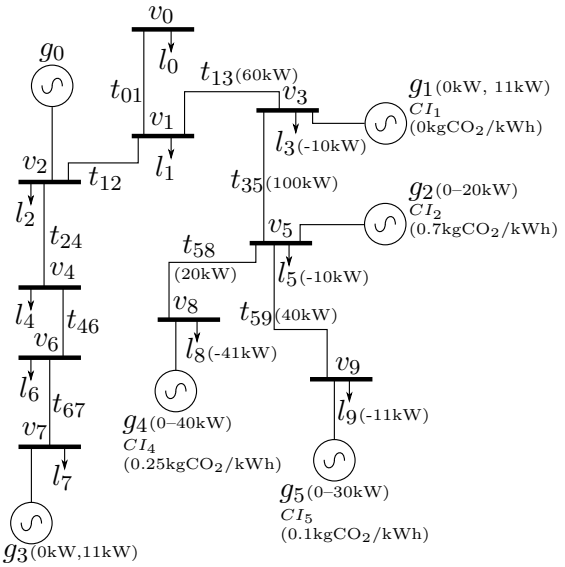


FIGURE 3.2: The discretised version of the electricity distribution network in Figure 3.1(a) when the discretisation unit $\omega = 1$kW.

could travel along t_{59} from v_5 to v_9, in order to satisfy l_9, is only 11kW. Moreover, the maximum output of g_5 means that the maximum amount of power that could travel along t_{59} from v_9 to v_5, after l_9 is satisfied, is 19kW. Therefore, the utilities calculated for variable values $\{-40, \ldots, -12\}$ and $\{20, \ldots, 40\}$ are all infeasible, even though it is within the thermal capacities of the distribution cable. This highlights the wasted computation that a naïve implementation of max-sum performs. The domain of the message is bounded by t^c_{59}. However, the actual feasible variable values are dependant on the load and the available generation at v_9, which is considerably less. As the network size grows, this wasted computation from calculating the utility of infeasible states becomes a major overhead (as we show in Section 4.1.1.2).

Thus, to address this wasted computation issue, in Chapter 4 we present a novel decentralised message passing algorithm, D–DYDOP, which uses techniques based on local consistency and dynamic programming. As we show later, doing so greatly reduces the computation time as it allows us to prune much of the search space. However, in order for a solution to be generated by D–DYDOP or max-sum, the electricity distribution network constraints must be discretised, as presented in the following section.

3.4 Converting the Electricity Network into a Discrete Optimal Dispatch Problem

To be able to use a discrete algorithm to coordinate generators in an electricity network (such as discrete max-sum or D–DYDOP), the electricity distribution network constraints (i.e., the thermal capacity of the distribution cables, loads, and the generator power outputs) must be discretised. We denote $\omega \in \mathbb{R}^+$ kW as the discrete unit of power

that will be used to discretise the electricity distribution network constraints. A seemingly sensible approach would be to discretise each variable in the electricity network to be multiples of ω. However, as we go on to show in this section, discretising the electricity distribution network constraints in order to apply a discrete algorithm only works within certain scenarios. We denote $\overline{\beta}_i \leq \beta_i$ as the discretised load, $\overline{\alpha}_i^{min} \geq \alpha_i^{min}$ as the discretised minimum output of a generator, $\overline{\alpha}_i^{max} \leq \alpha_i^{max}$ as the discretised maximum output of a generator, and $\overline{t}_{ij}^{c} \leq t_{ij}^{c}$ as the discretised thermal capacity of a distribution cable.

The discretised load must always overestimate the actual amount of power required since excess power can be wasted (although this is not favourable) but never generated to make up a shortfall at a load. Similarly, the discretised minimum output of the generators must always overestimate the actual minimum otherwise the constraints of the generators will be violated. Conversely, the discretised maximum output of the generators, and the discretised thermal capacity of the distribution cables must always underestimate the actual amount of power, again because the constraints of the generators and the distribution cables will otherwise be violated.

Therefore, the following equations are used to calculate the discretised variables such that they are multiples of ω:

$$\overline{\beta}_i = \left\lfloor \frac{\beta_i}{\omega} \right\rfloor \omega \qquad \overline{\alpha}_i^{min} = \left\lceil \frac{\alpha_i^{min}}{\omega} \right\rceil \omega \qquad \overline{\alpha}_i^{max} = \left\lfloor \frac{\alpha_i^{max}}{\omega} \right\rfloor \omega \qquad \overline{t}_{ij}^{c} = \left\lfloor \frac{t_{ij}^{c}}{\omega} \right\rfloor \omega \tag{3.7}$$

For example, consider the electricity network in Figure 3.1(a), Figure 3.2 shows the same electricity network discretised with $\omega = 1$kW. For instance, $\alpha_5^{max} = 30.6$kW becomes $\overline{\alpha}_5^{max} = 30$kW, $\beta_3 = -9.2$kW becomes $\overline{\beta}_3 = -10$kW, and $t_{35}^{c} = 100.9$kW becomes $\overline{t}_{35}^{c} = 100$kW. Thus, the set of power outputs of g_i become $S_i = \{\overline{\alpha}_i^{min}, (\overline{\alpha}_i^{min}+\omega), (\overline{\alpha}_i^{min}+2\omega), \ldots, (\overline{\alpha}_i^{min} + (q_i - 1)\omega), \overline{\alpha}_i^{max}\}$. For the continuous generator g_5, the power output range 0–30.6kW gets discretised to $S_5 = \{$0kW,1kW,2kW,...,29kW,30kW$\}$. For the discrete generator g_3, the power output values 0kW and 11.3kW get discretised to $S_3 = \{$0kW,11kW$\}$.

The discretised values can be computed in a decentralised way by each node before a discrete algorithm can be used to coordinate the power outputs of the generators. The solution calculated will correspond to the discretised version of the electricity network. Now, if the appropriate discretisation unit is used (i.e., one where each value is not over- or underestimated), then the solution calculated for the discretised version is guaranteed to work on the real electricity network. For instance, take the distribution network in Figure 3.1(a). If $\omega = 0.1$kW then $\alpha_5^{max} = 30.6\text{kW} = \overline{\alpha}_5^{max}$, $\beta_3 = -9.2\text{kW} = \overline{\beta}_3$, and $t_{35}^{c} = 100.9\text{kW} = \overline{t}_{35}^{c}$. Since the real network loads are expressed to 0.1kW, it is impossible for a solution to contain a generator that will output anything finer than this amount (such as 5.05kW). Therefore, a discrete algorithm applied to the discretised

network will produce a solution that can be applied to the real network. In other words, there is no error between the discretised network and the real network.

However, if the discretisation unit is not appropriate for the current network (i.e., some of the values have to be over- or underestimated), then the solution requires some manipulation in order for it to be applicable to the real electricity distribution network. This is primarily because of the overestimation of the loads. Furthermore, these techniques do not guarantee that the solution is even feasible as it may violate some of the constraints of the distribution cables, as we shall now demonstrate.

Consider the electricity distribution network in Figure 3.3(a). Nodes v_a, v_b, and v_c have loads of –1.1kW, –2.1kW, and –5.1kW respectively, with a total of –8.3kW. However, in the corresponding discretised network, Figure 3.3(b), using a discretisation unit of 1kW, nodes v_a, v_b, and v_c have loads of –2kW, –3kW, and –6kW respectively, with a total of –11kW. This means that any solution that the discrete algorithm calculates for the discrete network will be for a total load of –11kW. However, since the real electricity distribution network only has a total load of –8.3kW, there is a difference of 2.7kW which must be accounted for in some way.

We now detail three ways to apply such a solution calculated by a discrete algorithm to the real electricity distribution network; by the use of electricity storage batteries in Section 3.4.1, by reducing the power output of a number of generators and then rerunning the discrete algorithm in Section 3.4.2, and by modifying the distribution cable constraints before running the discrete algorithm in Section 3.4.3.

3.4.1 Battery Storage

We first consider that each node in the network has access to a battery that could store any of the remaining power on the network. Hence, any excess power at each node can be used to charge the battery, and ensures demand and supply are balanced. This would allow a solution to the discretised electricity distribution network to be directly applied to the real electricity distribution network. However, requiring each node to have a battery is an impractical assumption due to the current cost of battery technology; this cost is magnified when applied to large networks with thousands of nodes. Therefore, the following section describes a solution that does not require battery storage.

3.4.2 Reducing Generator Power Outputs

The second solution would be to reduce the power output from some of the generators by the difference between the sum of the actual consumption and discretised consumption (i.e., 2.7kW for the example in Figure 3.3). However, this introduces a new problem in that changing the power output of a generator in the network may change the power

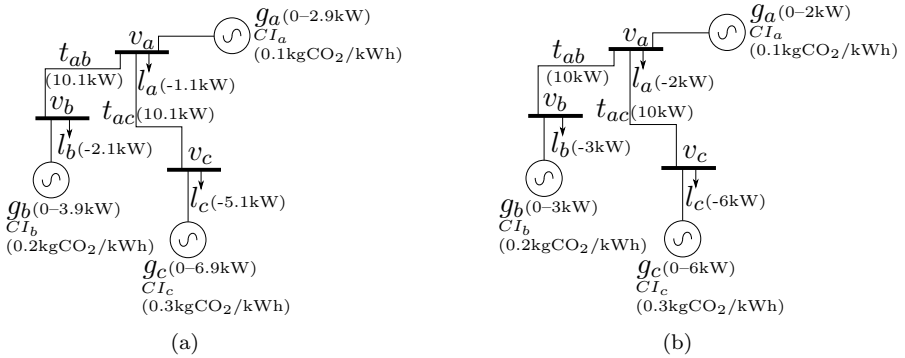


FIGURE 3.3: (a) An electricity distribution network, showing example values for the power output range and carbon intensity of the generators, thermal capacity of the distribution cables, and power consumption at the loads. (b) The discretised version of the same network when $\omega = 1$kW.

flowing along one or more distribution cables (see Figure 2.3). As a result, this could lead to distribution cables that have violated their thermal capacity constraints resulting in an infeasible solution to the problem. To overcome this, we would have to check that all the distribution cables in the network are within their thermal capacities, adjusting the power outputs of the generators if some distribution cables are not. Thus, in order for this technique to work, and allow us to apply a solution calculated by a discrete algorithm to the real electricity distribution network, we must do the following. Firstly, a number of generators must reduce their power output until the actual total load has been met. This can be done in a decentralised fashion by starting at an arbitrary node (i.e., the node with the smallest ID in the network) and reducing as much of its generator's power output as it can. Secondly, the chosen node chooses one of its neighbours with the highest carbon intensity to repeat the process until the required amount of power has been reduced. In order to ensure that each distribution cable is still within its capacity, each generator that reduced its power output fixes it at the current level. Finally, the discrete algorithm (be it D–DYDOP or max-sum) can then be run again on the modified network to produce a solution that can then be applied to the real electricity distribution network. However, there are a number of disadvantages to this technique. Firstly, it is obviously not optimal due to the arbitrary starting point to decide which generators must reduce their output. Secondly, once the generator that changed its output has been fixed, there may not even be a feasible solution that can be calculated. Finally, it requires the discrete algorithm to be run twice on the network problem and this is computationally expensive. Therefore, to avoid the need to run a discrete algorithm twice, the following section describes a solution that modifies the network constraints before applying a discrete algorithm.

3.4.3 Modifying the Distribution Cable Constraints

The third solution overcomes the need to run the discrete algorithm a second time on the network (after a number of generators have reduced their output to match the total actual consumption in the network), by constraining the distribution cables. The distribution cables are constrained, to specific values, to ensure that when reducing the generator power outputs to match the total amount of actual consumption, the maximum amount of power (flowing along a distribution cable) that can change, will never exceed the actual thermal capacity of any of the distribution cables.

Consider the example electricity distribution network in Figure 3.3(a) and the corresponding discretised electricity distribution network in Figure 3.3(b). The optimal solution for the discretised electricity distribution network is $\overline{\alpha}_a = 2\text{kW}$, $\overline{\alpha}_b = 3\text{kW}$, and $\overline{\alpha}_c = 6\text{kW}$ giving 2.7kW of excess power and $\overline{f}_{ab} = \overline{f}_{ac} = 0\text{kW}$ (i.e., no power is flowing through either distribution cable), where $\overline{f}_{ij}$ is the amount of power flowing through t_{ij} and $\overline{\alpha}_i$ is the power output of g_i, for the discretised electricity distribution network. In order to apply this solution to the real electricity distribution network, one or more of the generators must reduce its power output to a total of 2.7kW. For this example, we will choose the generator that can reduce its power output by the entire amount of power required (i.e., g_c). Therefore, the solution applied to the real electricity distribution network becomes $\tilde{\alpha}_a = 2\text{kW}$, $\tilde{\alpha}_b = 3\text{kW}$ and $\tilde{\alpha}_c = 6 - 2.7 = 3.3\text{kW}$, with $\tilde{f}_{ab} = -0.9\text{kW}$ and $\tilde{f}_{ac} = 1.8\text{kW}$; we use a tilde over a variable (i.e., $\tilde{\alpha}_a$) to denote the value of the variable after the electricity generators in the discretised electricity distribution network have been adjusted to match the actual consumption of the network. Therefore, the maximum amount the power flow of a distribution cable has varied by, between the initial discretised network and the network after the generators have been changed, is 1.8kW.

Now, consider the worst case where each load, in the real electricity distribution network, is very close to a discrete unit of power (i.e., multiples of $\omega = 1\text{kW}$ in this example) requiring an overestimation of ≈1kW. The optimal solution for the discretised electricity distribution network is still $\overline{\alpha}_a = 2\text{kW}$, $\overline{\alpha}_b = 3\text{kW}$, $\overline{\alpha}_c = 6\text{kW}$, and $\overline{f}_{ab} = \overline{f}_{ac} = 0\text{kW}$, but with an excess of ≈3kW. Again, we choose g_c to reduce its power output by ≈3kW. Thus, the solution applied to the real electricity distribution network becomes $\tilde{\alpha}_a = 2\text{kW}$, $\tilde{\alpha}_b = 3\text{kW}$ and $\tilde{\alpha}_c \approx 6 - 3 \approx 3\text{kW}$, with $\tilde{f}_{ab} \approx -1\text{kW}$ and $\tilde{f}_{ac} \approx 2\text{kW}$. As the loads get closer to a multiple of ω (requiring a large overestimate), the maximum amount by which the power flow in distribution cable t_{ij} can vary is bounded by:

$$||\overline{f}_{ij}| - |\tilde{f}_{ij}|| \leq s\omega \tag{3.8}$$

To clarify, $\overline{f}_{ij}$ is the amount of power flowing through t_{ij} *before* the generators in the network have been adjusted to match the actual consumption in the electricity distribution network, and $\tilde{f}_{ij}$ is the amount of power flowing through t_{ij} *after* the generators have been adjusted. Thus, we must do the following to apply a discrete algorithm to a real electricity distribution network and avoid having to run it twice. When discretising the network, we subtract $s\omega$ from the thermal capacity of each distribution cable by modifying the distribution cable discretisation from Equation (3.7). Thus, the maximum thermal capacity of each distribution cable becomes $\overline{t}^c_{ij} = \omega \left\lfloor \frac{t^c_{ij}}{\omega} \right\rfloor - s\omega = \omega \left(\left\lfloor \frac{t^c_{ij}}{\omega} \right\rfloor - s \right)$. Once a solution to the discretised electricity distribution network has been calculated using a discrete algorithm, the generators can safely reduce their power outputs (as described previously) and the maximum amount the power flow in each distribution cable will vary by is $s\omega$. Since the thermal capacity of each distribution cable has already been reduced by $s\omega$ for the discretised network, it is not possible for a solution to violate the constraints of the original electricity distribution network.

With this method of constraining the distribution cables when discretising the electricity distribution network, a discrete algorithm only needs to be run once. However, there are a number of problems with this technique. If $t^c_{ij} < s\omega$ then this technique cannot be applied because subtracting $s\omega$ would result in a negative thermal capacity constraint, which is of course infeasible. Furthermore, even if each distribution cable can be reduced by $s\omega$, there is no guarantee that there is a feasible solution to the discretised electricity distribution network. Finally, as with the technique in the previous section, the solution is not optimal. The following section tests the feasibility of solutions from a discrete algorithm applied to electricity networks.

3.4.4 Testing the Feasibility of Discrete Algorithms Applied to the Optimal Dispatch Problem

The experiment was run in Java on a 2.67GHz Intel Xeon quadcore with 12GB of RAM for max-sum. During each iteration, a random topology is generated with a maximum branching factor of 2 and the number of nodes fixed at 200. Nodes are assigned a uniformly distributed load value in the range of [1kW, 5kW], and either a continuous or discrete generator with a uniformly distributed carbon intensity. There is a 90% chance that the generator will be continuous.[2] If the generator is continuous, it is assigned a uniformly distributed minimum power output in the range of [0kW, 2kW], and a uniformly distributed maximum power output in the range of [3kW, 20kW]. If the generator is discrete, it is assigned a uniformly distributed power output level η in the range of [3kW, 20kW] (i.e., each discrete generator can either be off, or produce ηkW). Each distribution cable in the network is assigned a uniformly distributed thermal

[2]This is an arbitrary number chosen so that the majority of the network contains continuous generators.

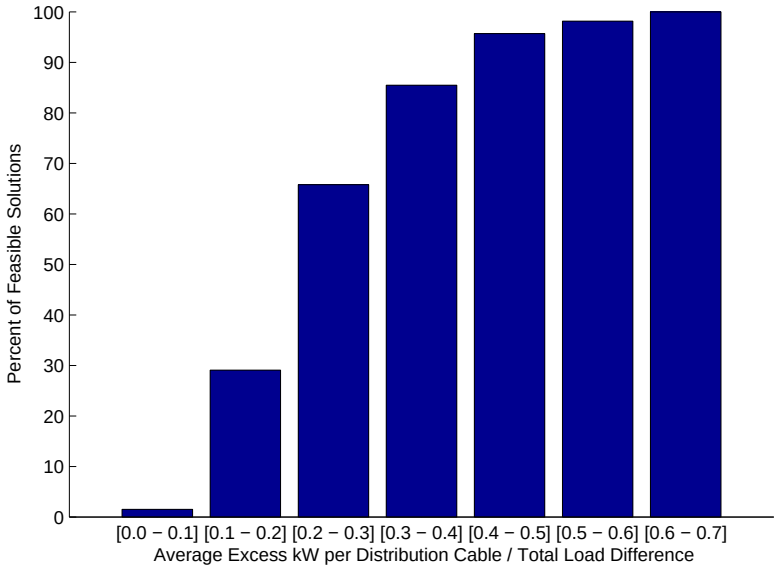


FIGURE 3.4: Experiment to show how increasing the amount of excess power per distribution cable affects the number of feasible solutions generated by max-sum. Using random acyclic electricity distribution network topologies with 200 nodes and a maximum branching factor of 2, ϖ^{max} is varied from 15kW to 60kW in 5kW steps.

capacity t^c_{ij} in the range of $[\varpi^{min}, \varpi^{max}]$, where ϖ^{max} is varied from 15kW to 60kW in 5kW steps with 200 iterations each, and $\varpi^{min} = \varpi^{max} - 5\text{kW}$. The real electricity distribution network is then discretised using the equations in (3.7) and max-sum is applied.

To determine whether each solution max-sum calculates is feasible when applied back to the real electricity distribution network, we apply the same technique as in Section 3.4.2. We then check, using direct current (DC) power flow analysis, that the discretised electricity distribution network is still a feasible solution given the thermal capacities of the distribution cables and the changed electricity generator power outputs.

Figure 3.4 shows how increasing the amount of excess power per distribution cable affects the number of feasible solutions generated by max-sum. By increasing the amount of excess each distribution cable has available (whilst the total load in the network remains constant), the number of feasible solutions increases. This is because if there is more power available per distribution cable, adjusting the power output of a number of generators is less likely to overload one or more distribution cables. Thus, in order to apply discrete algorithms to electricity distribution networks by adjusting the power output from a number of generators, one of two techniques can be used. The first technique is to increase the capacity of each distribution cable within the network. However, this process can be time consuming and has significant monetary and environmental costs (Roberts, 2004). The second technique is to use a smaller discretisation unit. Using a smaller discretisation unit decreases the difference between the total load in

the real electricity distribution network and the total load in the discretised electricity distribution network. Thus, a smaller total load difference requires fewer generators to reduce their power output, meaning the distribution cables are less likely to become overloaded, resulting in more feasible solutions. However, using a smaller discretisation unit increases the computation time of a discrete algorithm, as will be shown in Sections 4.4 and 5.4. The following section provides a discussion of discrete and continuous algorithms applied to the optimal dispatch problem.

3.4.5 Discussion of Discrete and Continuous Algorithms Applied to the Optimal Dispatch Problem

As discussed previously, in order for a solution to be generated by a discrete algorithm such as max-sum (or D–DYDOP presented in Chapter 4), the electricity distribution network constraints must be discretised. Unless an appropriate discretisation unit is used (which is often very small resulting in an increased amount of computation) the solution produced is not always guaranteed to be applicable to the real electricity distribution network. While discretising the electricity distribution network constraints can be completed in a distributed and decentralised way, a large improvement would be to use a continuous algorithm (such as C–DYDOP presented in Chapter 5) that can be applied directly to the optimal dispatch problem.

However, there are a number of advantages for using a discrete algorithm over a continuous algorithm to solve the optimal dispatch problem. Firstly, if the appropriate discretisation unit is large, a discrete algorithm can reduce its computation time significantly and may be able to outperform a continuous algorithm for certain scenarios.[3] By using a larger discretisation unit, the number of different power output values that must be evaluated for each generator is reduced. For example, consider the previous example of the continuous generator g_5 from Figure 3.3(a) with a power output range of 0–30.6kW. Using a discretisation unit $\omega = 5$kW, the power output range of g_5 gets discretised as $S_5 = \{0\text{kW},5\text{kW},10\text{kW},15\text{kW},20\text{kW},25\text{kW},30\text{kW}\}$. Instead of having to evaluate 31 different power outputs (i.e., when $\omega = 1$kW), only 7 different power outputs need to be evaluated. Secondly, if the network contains only discrete generators, then a continuous algorithm is not appropriate for finding a solution and a discrete algorithm must be used instead. The following section concludes this chapter.

3.5 Conclusions

In this chapter we addressed the optimal dispatch challenges faced by DNOs. Namely how an increasing amount of cleaner DGs can be added to already highly constrained

[3] In Section 5.4 we show that our discrete algorithm D–DYDOP can actually outperform our continuous algorithm C–DYDOP when a large discretisation unit is used.

distribution networks, and coordinated in an efficient fashion using optimal dispatch. We provided a novel DCOP formulation of the optimal dispatch problem; we showed how this can be decomposed as a factor graph and solved in a decentralised manner using algorithms based on GDL; in particular, the max-sum algorithm. Furthermore, we showed that max-sum applied naïvely in this setting performs a large number of redundant computations.

To address the problems associated with applying a naïve implementation of max-sum (i.e., calculating utilities for infeasible solutions), in the following chapter we present our novel message passing algorithm, called D–DYDOP, which uses techniques based on local consistency to prune much of the search space. We empirically evaluate D–DYDOP to test its performance on different network topologies and benchmark it against max-sum. Moreover, as discussed in Section 3.4, discrete algorithms can only be applied to the optimal dispatch problem in certain settings. Thus, to avoid the issues associated with discretising the optimal dispatch problem, in Chapter 5 we present C–DYDOP, which uses continuous variables for the power output ranges of the generators, the thermal capacities of the distribution cables, and the loads.

Chapter 4

A Novel Algorithm for Decentralised Optimal Dispatch in Acyclic Electricity Networks Using Discrete Variables

As discussed in the previous chapter, max-sum applied naïvely to the optimal dispatch problem performs a large number of redundant computations. Thus, to address this issue, we present a novel message passing algorithm, called D–DYDOP, to calculate an optimal solution in a decentralised and distributed fashion. Thus, this chapter makes the following contributions to the state of the art:

1. We present D–DYDOP, a novel decentralised message passing algorithm, that outperforms max-sum by only exploring the search space of feasible generator and distribution cable states.
2. We provide proof of the optimality of D–DYDOP and empirically evaluate it on a variety of large real electricity distribution network topologies, showing that it outperforms max-sum in terms of computational time and total size of messages sent.

The remainder of this chapter is organised as follows: Section 4.1 describes the two message passing phases of D–DYDOP including the construction and merging of discrete messages as they are propagated up the acyclic network. Section 4.2 provides a proof of the completeness and correctness of D–DYDOP, and Section 4.3 calculates the computational complexity with regard to the size and topology of the network. Section 4.4 provides an empirical evaluation against the benchmark algorithm max-sum, presented in Section 3.3, and a highly optimised centralised approach based on MIP. Finally, Section 4.5 concludes.

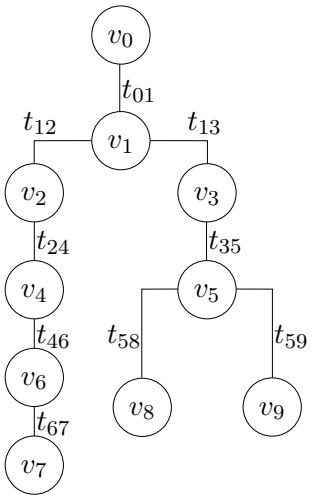


FIGURE 4.1: The tree representation of the electricity distribution network in Figure 3.1(a).

4.1 Message Passing Phases

D–DYDOP can be applied to acyclic electricity networks and uses a dynamic programming approach. Figure 4.1 gives a simplified tree representation of the electricity distribution network in Figure 3.1(a) using nodes. Each node is assumed to have a number of generators, and a number of loads. For instance v_3 contains generator g_1 and load l_3, where as v_4 contains just load l_4. Each node, which is controlled by an agent, has exactly one parent node and zero or more child nodes, apart from one node v_0 which is the root node and has no parent. Leaf nodes, (i.e., v_7, v_8, and v_9), have no children. D–DYDOP proceeds in two phases (which we describe in more detail in the following sections):

Phase 1 – Value Calculation *PowerCost* messages are sent from the leaf nodes to the root node. A node waits until it has received *PowerCost* messages from all of its children before computing its own *PowerCost* message which it sends to its parent. Each *PowerCost* message describes the CO_2 emissions of its own generation and the generation of its children.

Phase 2 – Value Propagation When the root node receives *PowerCost* messages from all of its children, it calculates the optimum power output for each of its own generators such that the demands of its children are satisfied and the CO_2 emissions are minimised. It then propagates power flow values to all its children which in turn propagate power flow values to their children.

The algorithm terminates when all leaf nodes receive a power flow value, at which point each generator knows the optimal amount of power it needs to output. It should be noted that we draw on the techniques used in the max-sum algorithm and in similar

dynamic programming algorithms such as DPOP. Max-sum, D–DYDOP, and DPOP are similar in that D–DYDOP and DPOP are essentially extended versions of the max-sum algorithm (specifically the GDL family of algorithms). The similarities between DPOP and D–DYDOP are that each algorithm contains two stages that are similar (called UTIL propagation and VALUE propagation in DPOP). However, D–DYDOP differs in the optimisation of the message values and the number of message values sent in the value calculation stage. D–DYDOP only sends the optimal message values at each stage after local consistency techniques have been applied allowing us to prune much of the search space. We elaborate on this pruning, along with the two phases of message propagation in the following section.

4.1.1 Phase 1: Value Calculation

In what follows we give a detailed overview of the value calculation phase of D–DYDOP. Section 4.1.1.1 introduces the structure of a discrete *PowerCost* message, Section 4.1.1.2 describes how a leaf node constructs its discrete *PowerCost* messages, and finally Section 4.1.1.3 details how a node merges discrete *PowerCost* messages from its children.

4.1.1.1 Discrete *PowerCost* Messages

A discrete *PowerCost* message sent from v_i to its parent $\hat{v_i}$, is an array of y *flowCO* elements:

$$PowerCost_{i \to \hat{i}} = [flowCO_1, \ldots, flowCO_y] \tag{4.1}$$

where a *flowCO* element describes the CO_2 emissions that occur, when v_i and all of its children output certain amounts of power, such that there is a specified flow of power between v_i and its parent $\hat{v_i}$ along the distribution cable $t_{i\hat{i}}$:

$$flowCO_j =< f_{i\hat{i}}, \boldsymbol{\gamma}(f_{i\hat{i}}) > \tag{4.2}$$

where $flowCO_j$ denotes the j^{th} *flowCO* element, $f_{i\hat{i}} \in \mathbb{R}$ kW is the resultant power flow travelling along $t_{i\hat{i}}$, and $|f_{i\hat{i}}| \leq t^c_{i\hat{i}}$ where $t^c_{i\hat{i}}$ is the thermal capacity of $t_{i\hat{i}}$. Note that $f_{i\hat{i}} > 0$ denotes the resulting power is flowing out of v_i to $\hat{v_i}$, $f_{i\hat{i}} < 0$ denotes the resulting power is flowing into v_i from $\hat{v_i}$, $f_{i\hat{i}} = 0$ denotes no power is flowing between v_i and $\hat{v_i}$. The function $\boldsymbol{\gamma} : \mathbb{R} \to \mathbb{R}^+$ $kgCO_2/h$ denotes the CO_2 emissions that result from v_i and all of its children generating certain amounts of power.[1] Each *flowCO* element that v_i calculates maps to an $OPCState_j$ which describes the power output at v_i along with the flows between v_i and its children that results in the CO_2 emission described by the function $\boldsymbol{\gamma}(f_{i\hat{i}})$:

$$OPCState_j = \langle [\alpha_i, \mathbf{F}(v_i)] \rangle \tag{4.3}$$

[1] Generator g_5 at node v_9 (Figure 3.2) with a carbon intensity of $0.1kgCO_2/kWh$ and a power output of 20kW, will have a resulting CO_2 emissions of $2kgCO_2/h$ and 9kW of resulting power travelling to v_5.

Algorithm 1 Constructing a leaf node *PowerCost* message.

```
  constructDiscreteLeafMessage() {
1.  FOREACH (α_i ∈ S_i) { //Iterate through generator power output values
2.    f_îi ← α_i + β̄_i; //Calculate resultant flow
3.    IF (|f_îi| ≤ t^c_îi) { //If the thermal capacity is not violated
4.      γ(f_îi) ← α_i CI_i; //Calculate resultant CO2 emissions
5.      flowCO ←createFlowCO(f_îi,γ(f_îi)); //Create flowCO element and store in PowerCost
                                            //message
6.      OPCState←linkToOPCState(flowCO); //Link flowCO element to OPCState
7.    }
8.  }
9.  sendPowerCostMessageToParent();
  }
```

where the array $[\alpha_i, \mathbf{F}(v_i)]$ contains the power output of a generator at v_i and the set of power flows $\mathbf{F}(v_i)$ from the distribution cables connecting v_i to its children.[2] This mapping represents the dynamic programming aspect of D–DYDOP because as power flow values are propagated down the tree, during the value propagation phase, the associated *OPCState* is used to find the power output of v_i given a particular power flow $f_{i\hat{i}}$. Having introduced the notation of a discrete *PowerCost* message, the following section describes how to construct discrete *PowerCost* messages at leaf nodes.

4.1.1.2 Constructing a Discrete *PowerCost* Message at a Leaf Node

Only the power output of the generators, and the loads at the leaf node need to be taken into consideration when a leaf node constructs a discrete *PowerCost* message. For each power output from each generator at v_i, it constructs a corresponding *flowCO* element with flow $f_{i\hat{i}}$ calculated as:

$$f_{i\hat{i}} = \sum_{l \in \mathbf{L}(v_i)} \beta_l + \sum_{g \in \mathbf{G}(v_i)} \alpha_g \tag{4.4}$$

giving the resultant power flowing between v_i and $\hat{v_i}$. The CO_2 emissions $\boldsymbol{\gamma}(f_{i\hat{i}})$ of the *flowCO* element, is calculated as:

$$\boldsymbol{\gamma}(f_{i\hat{i}}) = \sum_{g \in \mathbf{G}(v_i)} \alpha_g CI_g \tag{4.5}$$

where CI_g is the carbon intensity of generator g situated at v_i. See Algorithm 1 for a pseudocode representation of constructing a discrete *PowerCost* message at a leaf node.[3] We iterate through the power outputs of the generator at v_i (lines 1 – 8). For

[2]The definition assumes one generator at v_i. The generalisation to more than one generator at each node is trivial.

[3]For ease of reading, all pseudocode representations in this thesis assume that each node v_i in the network contains one generator g_i and one load l_i; the generalisation to more than one generator and more than one load at each node is trivial.

each power output the resultant flow is calculated (line 2). If the thermal capacity of the distribution cable to the parent node is not violated (line 3), the corresponding CO_2 emissions are calculated (line 4), and then a *flowCO* element is created (line 5) and linked to the *OPCState* using the generator's power output which resulted in the CO_2 emissions (line 6). All the *flowCO* elements created are added to a discrete *PowerCost* message and then sent to the parent node (line 9). Note that *OPCStates* that are linked to by each *flowCO* element are never sent on to the parent node and are instead kept for use during phase 2 of the algorithm.

Consider the following discrete $PowerCost_{9\to5}$ message, which v_9 sends to v_5, as shown in Figure 3.2. Let the distribution cable t_{59} have a thermal capacity t^c_{59} of 40kW, the load l_9 be –11kW, and the generator g_5 have a power output range of 0–30kW and a carbon intensity CI_5 of 0.1kgCO_2/kWh. The following is part of the discrete $PowerCost_{9\to5}$ message:

$$\begin{array}{lcll} flowCO_j & = & <0, 1.1> & \to \quad [+11kW] \\ flowCO_{j+1} & = & <1, 1.2> & \to \quad [+12kW] \\ flowCO_{j+2} & = & <2, 1.3> & \to \quad [+13kW] \end{array} \tag{4.6}$$

Now, $flowCO_{j+2}$ indicates that a flow of 2kW, from v_9 to v_5, will result in 1.3kgCO_2/h emission with g_5 outputting 13kW. The total number of *flowCO* elements in the discrete $PowerCost_{9\to5}$ message is 31. By contrast, compare with the example $R_{9\to13}(x_{13})$ message of max-sum in Section 3.3, which has 81 variable values instead. This further highlights the wasted computation that the naïve implementation of max-sum performs and the advantages of pruning the search space (i.e., a difference of 50 variable values).

The pruning of the search space is related to ensuring local consistency of the constraints and variable values (i.e., thermal capacities of the distribution cables and generator power outputs), via constraint propagation in a distributed manner (Dechter, 2003). More specifically we ensure node and arc consistency. Node consistency for v_i is enforced for each *flowCO* element by constricting α_i to conform to $t^c_{i\hat{i}}$ and β_i (i.e., the range of power g_i is capable of outputting will not be the actual power output range it could generate due to the thermal capacity of its parent's distribution cable and the consumption at v_i). As a result, arc consistency is also enforced because each *flowCO* element will only specify a flow of power between v_i and $v_{\hat{i}}$ that not only conforms to $t^c_{i\hat{i}}$, but also the actual amount of power that can flow from $v_{\hat{i}}$ to v_i;[4] this is known since messages are propagated from leaf nodes to the root of the tree. The following section describes how discrete *PowerCost* messages are merged.

[4]Whereas DPOP would send utility values (message values) based on the number of power outputs the generator at $v_{\hat{i}}$ has, which could contain solutions with infeasible power flows along $t_{i\hat{i}}$ (Petcu and Faltings, 2005).

Algorithm 2 Merging *PowerCost* messages.

```
   mergeDiscreteMessages() {
 1.  FOREACH (α_i ∈ S_i) { //Iterate through generator's power output values
 2.    FOREACH (flowCO ∈ getChildPowerCostMessages()) { //Iterate through each
                                                        //combination of flowCO
                                                        //elements from each child
 3.      f_ii^ ← α_i + β̄_i + sum(state(f_ci)); //Calculate resultant flow using a flowCO
                                               //element from each child
 4.      IF (|f_ii^| ≤ t^c_ii^) { //If the thermal capacity is not violated
 5.        γ(f_ii^) ← α_i CI_i + sum(state(γ(f_ci))); //Calculate resultant CO2 emissions
                                                      //using flowCO element from each child
 6.        IF (MIN(f_ii^, γ(f_ii^))) { //If the resultant CO2 emissions for the resultant
                                       //flow is the minumum calculated so far
 7.          flowCO ←createFlowCO(f_ii^, γ(f_ii^)); //Create flowCO element and store in
                                                    //PowerCost message
 8.          setNewMinimum(f_ii^, γ(f_ii^)); //Set resultant CO2 emissions as new minimum
                                             //for resultant flow
 9.          OPCState←linkToOPCState(flowCO); //Link flowCO element to OPCState
10.        }
11.      }
12.    }
13.  }
14.  sendPowerCostMessageToParent();
   }
```

4.1.1.3 Merging Discrete *PowerCost* messages

For each v_i that has at least one child, the discrete *PowerCost* messages that it receives must be processed in order to produce its own discrete *PowerCost* message that it sends to $\hat{v_i}$. The amount of power that can flow from v_i to $\hat{v_i}$, or from $\hat{v_i}$ to v_i, is bounded by $t^c_{i\hat{i}}$. With these bounds, v_i is able to calculate each valid flow that can travel into or out of it. For each valid flow, v_i calculates the minimum CO_2 emissions that result from the power output at v_i, and the power output from all of v_i's children. To calculate the *flowCO* element for each resultant flow with the lowest CO_2 emissions value, v_i iterates through every possible power output that it can produce and every *flowCO* element from each of its children's discrete *PowerCost* message. A state represents the combination of a *flowCO* element from each of v_i's children and the power output at v_i.[5] The flow $f_{i\hat{i}}$ of this state is calculated as:

$$f_{i\hat{i}} = \sum_{l \in \mathbf{L}(v_i)} \beta_l + \sum_{g \in \mathbf{G}(v_i)} \alpha_g + \sum_{c \in chi(v_i)} f_{ci} \tag{4.7}$$

where $\sum_{c \in chi(v_i)} f_{ci}$ is the sum of the chosen *flowCO* elements' flows from each of v_i's immediate children $chi(v_i)$. In order to choose the minimum state for each resultant

[5]Note, this state is different from an *OPCState* which contains only the power flow from each of v_i's children and the power output at v_i.

flow, the CO_2 emissions of the state must be calculated as follows:

$$\boldsymbol{\gamma}(f_{\hat{ii}}) = \sum_{g \in \mathbf{G}(v_i)} \alpha_g CI_g + \sum_{c \in chi(v_i)} \boldsymbol{\gamma}(f_{ci}) \tag{4.8}$$

where $\sum_{c \in chi(v_i)} \boldsymbol{\gamma}(f_{ci})$ is the sum of the chosen *flowCO* elements' CO_2 emissions from each of v_i's immediate children $chi(v_i)$. See Algorithm 2 for a pseudocode representation of merging *PowerCost* messages. We iterate through the power outputs of the generator at v_i (lines 1 – 13). For each power output, we iterate through every possible combination of the *flowCO* elements from the children of v_i's discrete *PowerCost* messages (lines 2 – 12). For a particular state (i.e., a combination of *flowCO* elements, one from each child, and the power output of the generator at v_i) the resultant flow is calculated by summing each flow of the *flowCO* elements, in the state, with the generator power output and the load (line 3). If the thermal capacity of the distribution cable to the parent node is not violated (line 4), the resultant CO_2 emissions are calculated by summing the CO_2 emissions of the *flowCO* elements, in the state, together with the product of the generator's power output and its carbon intensity (line 5). If the resultant CO_2 emissions are the minimum recorded for the particular resultant flow (line 6), then the *flowCO* element is created (line 7), set as the new minimum for that particular resultant flow (line 8), and linked to the *OPCState* (line 9). All the *flowCO* elements created are added to a discrete *PowerCost* message and then sent to the parent node (line 14).

As an example of merging discrete *PowerCost* messages, consider the following discrete $PowerCost_{5\rightarrow3}$ message, v_5 sends to v_3, as shown in Figure 3.2. Let t^c_{35} be 100kW, t^c_{58} be 20kW, t^c_{59} be 40kW, l_5 be –10kW, l_8 be –41kW, l_9 be –11kW, g_2 have a power output range of 0–20kW, CI_2 be 0.7kgCO_2/kWh, g_4 have a power output range of 0–40kW, CI_4 be 0.25kgCO_2/kWh, g_5 have a power output range of 0–30kW, and CI_5 be 0.1kgCO_2/kWh. The following is part of the discrete $PowerCost_{5\rightarrow3}$ message (after receiving messages from v_8 and v_9):

$$\begin{array}{lcll} flowCO_j & = & < -10, 8.5 > & \rightarrow \; [+0kW]\; v_8(-19)\; v_9(19) \\ flowCO_{j+1} & = & < -9, 8.75 > & \rightarrow \; [+0kW]\; v_8(-18)\; v_9(19) \\ flowCO_{j+2} & = & < -8, 9.00 > & \rightarrow \; [+0kW]\; v_8(-17)\; v_9(19) \end{array} \tag{4.9}$$

Now, $flowCO_{j+1}$ indicates that a flow of 9kW, from v_3 to v_5, will result in 8.75kgCO_2/h emission with g_2 outputting 0kW, a flow of 18kW from v_5 to v_8, and a flow of 19kW from v_9 to v_5. The following section describes the second phase of D–DYDOP whereby power output values are propagated from the root node to the leaf nodes.

4.1.2 Phase 2: Value Propagation

Once the root node has received discrete *PowerCost* messages from all of its children, it calculates how much power each of its generators should output in order to satisfy its immediate children's loads whilst minimising CO_2 emissions. It does this by iterating through every possible power output that it can produce and every *flowCO* element from each of its children's discrete *PowerCost* messages. Equation (4.7) is used to calculate the resultant flow of a state. If the flow is not equal to zero, then this particular state for the network is infeasible, since excess power means that supply and demand is imbalanced. For every state that has a flow equal to zero, the CO_2 emissions of the network are calculated by using Equation (4.8).

The root node's state with the minimum CO_2 emissions corresponds to the minimum CO_2 emissions for the entire network, and is selected as the optimum state.[6] Power flow values are then sent to each of the root node's children telling them which of their *flowCO* elements resulted in the minimum CO_2 emissions. The child retrieves the correct *flowCO* element by matching the power flow value sent to them with the flow from the *flowCO* element. The *OPCState* which is referenced by each child recipient's corresponding *flowCO* element tells the child exactly how much power to output. The child recipient can then send the power flow specified in its *OPCState* to each of its corresponding children. Power flow values are propagated in this manner to the leaf nodes, at which point each generator in the network knows its optimum power output that results in the minimum CO_2 emissions for the entire network. It should be noted that if there are no states with a flow equal to zero, this indicates that there is no solution possible which D–DYDOP will report. There are two possibilities for not being able to find a solution. The first possibility is that there is actually no solution given the real electricity distribution network constraints (i.e., CPLEX would also report a solution does not exist). The second possibility is that due to the discretisation unit used, there is no solution to the discretised electricity distribution network. In this case a smaller discretisation unit can be used to determine whether a solution does exist to the real electricity distribution network. Having introduced D–DYDOP, we now present completeness and correctness of the algorithm.

[6] If multiple states exist with the same minimum CO_2 emissions, this indicates there are multiple optimal configurations for the outputs of the generators and one of the states is chosen arbitrarily. This can happen when two or more generators have the same carbon intensity.

4.2 Completeness and Correctness

In what follows, we prove that D–DYDOP applied to acyclic networks is complete[7] and correct.[8]

Proposition 4.1. *D–DYDOP is complete.*[9]

Proof. To construct discrete *PowerCost* messages, v_i must iterate through all of its own generator's possible power outputs and every *flowCO* element from each of its children's discrete *PowerCost* messages. Each *flowCO* element contains the minimum CO_2 emissions that result from each $l \in \mathbf{L}(v_i)$, and all of its childrens' loads, being satisfied. The root node chooses a feasible state that results in the minimum CO_2 emissions. Therefore, at each node, all feasible states are evaluated and the root node chooses the optimal state which minimises CO_2. Hence, the algorithm is complete. □

Proposition 4.2. *D–DYDOP is correct.*

Proof. This proof follows on from Proposition 4.1. When constructing messages, v_i only evaluates feasible states; the states that conform to Equations (3.2) – (3.4) and the power outputs of each $g \in \mathbf{G}(v_i)$. Each message will contain the minimum CO_2 emissions that result from a feasible set of states. Therefore, any solution calculated by the algorithm will be valid as it has explicitly conformed to the local and global constraints of the entire network (since constraint checks are explicitly embedded in the algorithm). Hence, the algorithm is correct. □

Having presented the correctness and completeness of D–DYDOP, we now calculate the computational complexity.

4.3 Computational Complexity

Here, the worst-case complexity of D–DYDOP is calculated, with regard to the size of the network and the number of children of a node, in order to show its suitability for large optimal dispatch problems (Requirement III).

Proposition 4.3. *The size of discrete PowerCost messages (i.e., the total number of flowCO elements) sent by D–DYDOP grows linearly with the size of the network.*

[7] Complete in terms of finding the optimal solution calculated by CPLEX using Equations (3.1) – (3.4).

[8] Correct such that any solution returned by D–DYDOP is feasible given Equations (3.1) – (3.4).

[9] To clarify, if an optimal solution exists for the real electricity distribution network, D–DYDOP will find the optimal solution to the discretised electricity distribution network. Furthermore, as discussed in Section 3.4, if the appropriate discretisation unit is used (i.e., each discretised value does not over- or underestimate the real value), then the solution calculated for the discretised electricity distribution network is guaranteed to work on the real electricity distribution network.

Proof. In the worst case, the maximum size of the discrete *PowerCost* message v_i has to create and send to $\hat{v_i}$ is Φ_i:

$$\Phi_i = \frac{2t^c_{i\hat{i}}}{\omega} \tag{4.10}$$

This highlights how ω influences a node's message size; a small ω increases the size of the message, whereas a large ω decreases the size of the message, see Section 3.4 for a full description of the discretisation unit ω. In the worst case, the size of the messages D–DYDOP has to create and send in total is:

$$\sum_{v_i \in \mathbf{V} \setminus v_r} \Phi_i \tag{4.11}$$

where v_r is the root node. Therefore, the size of the messages D–DYDOP sends grows linearly in $O(|\mathbf{V}|)$. □

Proposition 4.4. *The number of states that v_i must iterate through is exponential in $|chi(v_i)|$.*

Proof. When merging discrete *PowerCost* messages, v_i must iterate through all states in the Cartesian product of all of its children's states and its own power output values. Therefore, the number of states a node must iterate through in the worst case grows exponentially in $O(M^{|chi(v_i)|})$, where M is the maximum number of *flowCO* elements a discrete *PowerCost* message received from the children of v_i contains. □

Even though the worst-case complexity of D–DYDOP is exponential in the number of children a node has, it is significantly less than the total number of nodes in the entire network. Thus, D–DYDOP may be able to exploit the structure of the network (unlike max-sum that does not explicitly take this structure into consideration) and compute an optimal solution with less computation. Having presented D–DYDOP and analysed its theoretical properties, the following section provides an empirical evaluation against max-sum and a highly optimised centralised approach based on MIP.

4.4 Empirical Evaluation

To highlight the improvements of D–DYDOP against the discrete max-sum algorithm presented in Section 3.3, we conducted two experiments on two large real electricity distribution network topologies (see Figure 4.2), and one experiment on large random acyclic electricity distribution network topologies.[10] We benchmark D–DYDOP against max-sum and a highly optimised centralised approach, which uses IBM's ILOG CPLEX

[10] We use random topologies in order to vary the branching factor of each node.

12.2. CPLEX simply solves a large MIP without having to use message passing or decentralised control. Thus, CPLEX is able to calculate a solution in under a second.[11] The reason we benchmark D–DYDOP against the large MIP solution to the optimal dispatch problem, calculated by CPLEX, is because this solution represents the optimal decision for each generator. As discussed previously in Section 3.2, we choose to benchmark D–DYDOP against max-sum, instead of benchmarking against other message passing algorithms such as ADOPT, OptAPO, or DPOP, because max-sum is the current state of the art message passing algorithm for solving DCOPs. The three experiments were conducted in order to test the following:

Experiment 1 Tests the effect of ω for CPLEX, D–DYDOP, and max-sum on the two large real electricity distribution network topologies in Figure 4.2.

Experiment 2 Tests the effect of the size of the network for CPLEX, D–DYDOP, and max-sum on the two large real electricity distribution network topologies in Figure 4.2.

Experiment 3 Tests the effect of the branching factor for CPLEX, D–DYDOP, and max-sum on large random acyclic electricity distribution network topologies.

Figure 4.2 shows the two real electricity distribution networks used for the experiments. Figure 4.2(a) is located in India[12] and contains 76 substations, each of which can further be connected to as many as 400 nodes. Figure 4.2(b) is a section of the electricity distribution network located in Southampton UK, and contains 27 substations, each of which can further be connected to hundreds of nodes. We have taken a small section of the Southampton electricity distribution network and indicated which substations connect to the larger distribution network.[13] We only use two real network topologies because the topologies of electricity distribution networks are similar throughout the world. The remainder of this section is organised as follows: Section 4.4.1 describes the setup of the electricity distribution networks. Section 4.4.2 details Experiment 1, Section 4.4.3 details Experiment 2, and Section 4.4.4 details Experiment 3. Finally, Section 4.4.5 draws conclusions from all three experiments.

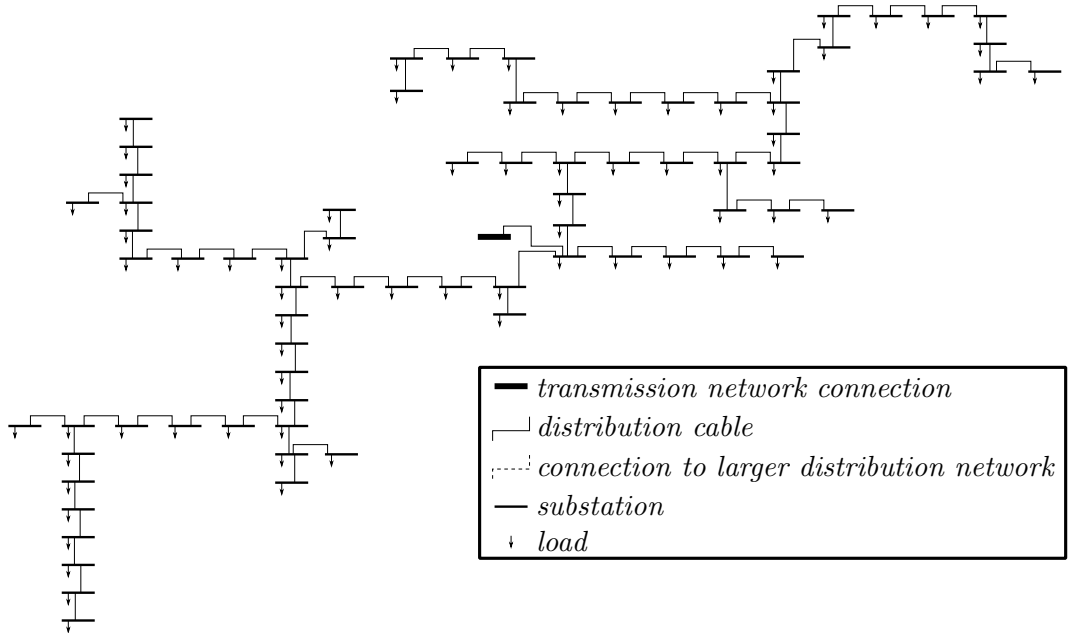
4.4.1 Experiment Setup

Each experiment was run in Java on a 2.67GHz Intel Xeon quadcore with 12GB of RAM. During each iteration, nodes are assigned a uniformly distributed load value in the range

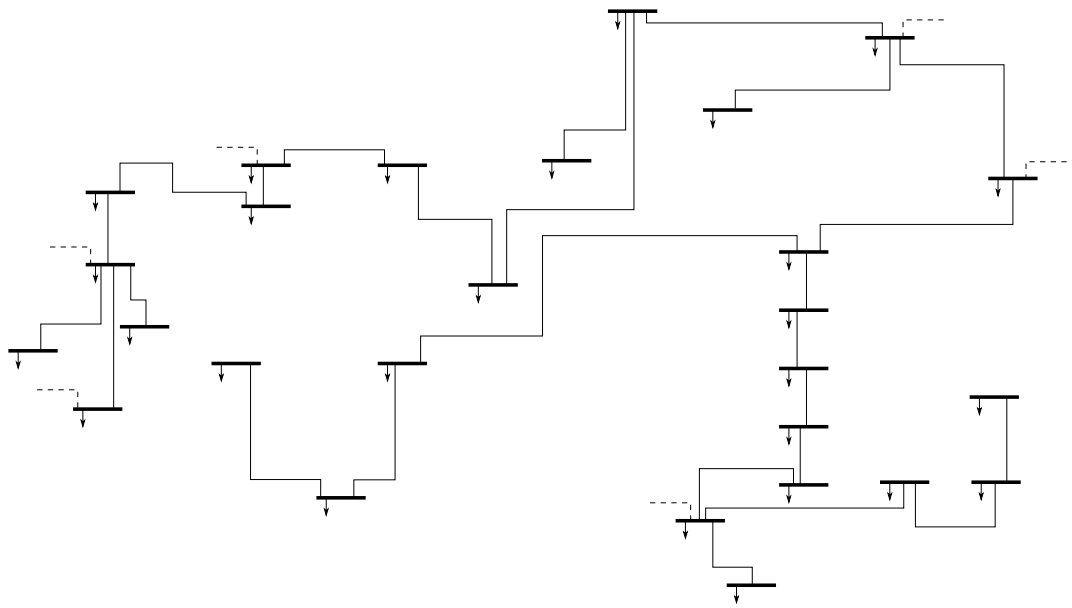
[11] Although centralised techniques in this context are fast, they suffer from a number of problems as described in Section 2.3.2. Thus, distributed and decentralised techniques may be the only feasible solution to coordinating the power output of generators in electricity networks.

[12] We cannot disclose the exact location due to commercial sensitivities.

[13] This section was chosen in order to contrast against the Indian electricity distribution network. The Southampton electricity distribution network is smaller but contains a higher number of high branching factor substations.



(a)



(b)

FIGURE 4.2: (a) Indian electricity distribution network topology containing 76 substations. (b) A section of Southampton UK electricity distribution network topology containing 27 substations.

of [1kW, 5kW], and either a continuous or discrete generator with a uniformly distributed carbon intensity. There is a 90% chance that the generator will be continuous.[14] If the generator is continuous, it is assigned a uniformly distributed minimum power output in the range of [0kW, 2kW], and a uniformly distributed maximum power output in the range of [3kW, 20kW]. If the generator is discrete, it is assigned a uniformly distributed

[14]This is an arbitrary number chosen so that the majority of the network contains continuous generators.

power output level η in the range of [3kW, 20kW] (i.e., each discrete generator can either be off, or produce ηkW). Each distribution cable in the network is assigned a uniformly distributed thermal capacity in the range of [10kW, 15kW]. The electricity distribution network constraints are then discretised with the equations in (3.7) (where ω is varied between 0.5kW and 3.0kW in Experiment 1, and $\omega = 1$kW in Experiments 2 and 3), in order to apply D–DYDOP and max-sum. Having described the setup for each experiment, the following section details the first experiement.

4.4.2 Experiment 1 : Impact of Varying Discretisation Unit

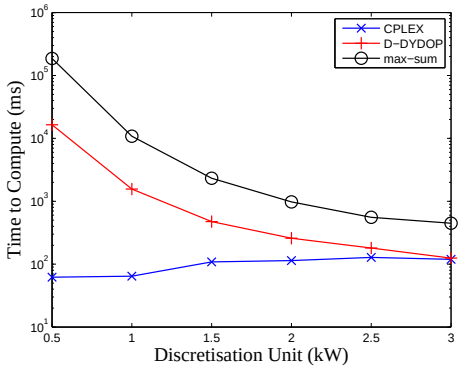
Experiment 1 was set up in order to test the effect of ω for CPLEX, D–DYDOP, and max-sum. Using both the Indian and UK electricity distribution networks, the number of additional nodes at each substation was fixed at 8 and arranged as a random acyclic network with a maximum branching factor of 2.[15] Thus, the total number of nodes in the whole network was 596 and 203 for the Indian and UK electricity distribution networks respectively. The discretisation unit ω, used to discretise the electricity distribution network constraints, was varied from 0.5kW to 3.0kW in 0.5kW steps, each with 50 iterations.[16] During each iteration, the nodes and distribution cables were initialised as in Section 4.4.1.

Figure 4.3 shows four plots of the results from the first experiment (error bars showing the standard error across the 50 iterations omitted due to being negligible). We use a logarithmic scale for the y-axis in all four plots. Figures 4.3(a) and 4.3(b) show how the computation time is affected by the discretisation unit ω for CPLEX, D–DYDOP, and max-sum on the Indian and UK electricity distribution networks respectively. For CPLEX, regardless of the discretisation unit, the computation times remain constant. This is because CPLEX does not require the electricity distribution network constraints to be turned into a discrete problem before it can calculate a solution. However, for D–DYDOP and max-sum, the time complexity is exponential in ω.

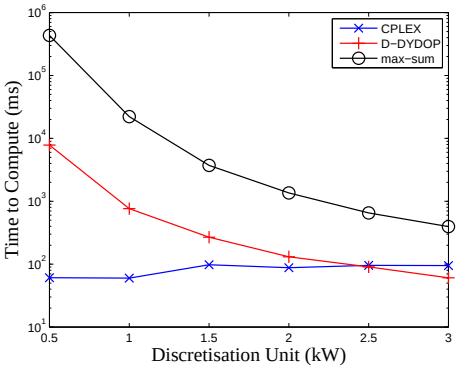
Thus, it can be seen that for small ω, the computation time for both D–DYDOP and max-sum suffers greatly. This is because a small discretisation unit results in both algorithms iterating through many generator power outputs (in increments of ω), and message elements, to calculate the utility of each possible resultant amount of power that can flow along a distribution cable, as shown by Pseudocode 1 and 2. Max-sum performs significantly worse than D–DYDOP because of the wasted computation that it undertakes to calculate the utility for infeasible amounts of distribution cable power (see Section 4.1.1.2 for an example of the wasted computation a naïve implementation of

[15] We choose a branching factor of 2 so that the discrete algorithms can calculate a solution within a reasonable time frame.

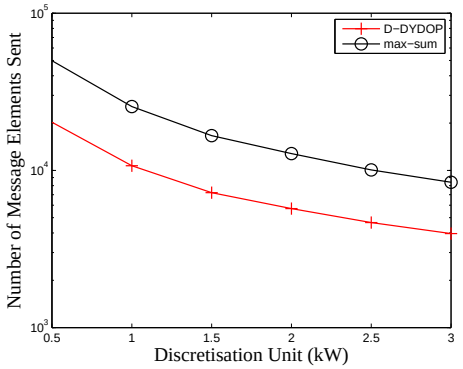
[16] We found 50 iterations to be an adequate amount since further iterations did not improve the statistical significance of the results.



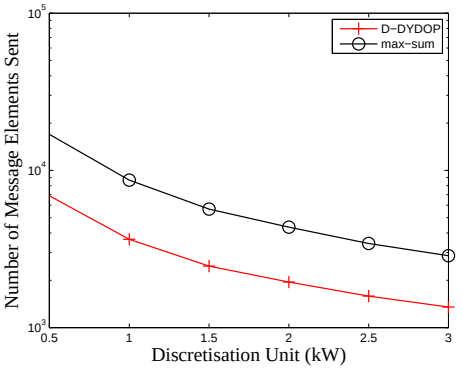
(a) Indian electricity distribution network.



(b) UK electricity distribution network.



(c) Indian electricity distribution network.



(d) UK electricity distribution network.

FIGURE 4.3: Experiment 1 tests the effect of ω for CPLEX, D–DYDOP, and max-sum. Using both electricity distribution network topologies in Figure 4.2, the number of additional nodes at each substation was fixed at 8 and arranged as a random acyclic network with a maximum branching factor of 2. The discretisation unit ω was varied from 0.5kW to 3.0kW in 0.5kW steps, each with 50 iterations. 4.3(a) and 4.3(b) show how ω affects computation time for CPLEX, D–DYDOP, and max-sum on the Indian and UK electricity distribution networks respectively. 4.3(c) and 4.3(d) show how ω affects the total number of message elements sent for D–DYDOP and max-sum on the Indian and UK electricity distribution networks respectively. We use a logarithmic scale for the y-axis in all four plots.

max-sum performs). As the disretisation unit is increased, both algorithms have to calculate fewer state utilities, resulting in decreased computation time. It can be seen that there is a difference of performance for D–DYDOP and max-sum on the two networks. D–DYDOP is faster at computing a solution for the UK electricity distribution network as apposed to the Indian electricity distribution network, whereas max-sum is faster at computing a solution for the Indian electricity distribution network as apposed to the UK electricity distribution network. This difference between Figures 4.3(a) and 4.3(b) for the two algorithms is because the section of UK electricity distribution network is smaller, but contains a number of nodes with higher branching factors. The higher number of high branching factor nodes affects max-sum more than D–DYDOP. Therefore, even though the UK electricity distribution network contains a smaller number of total

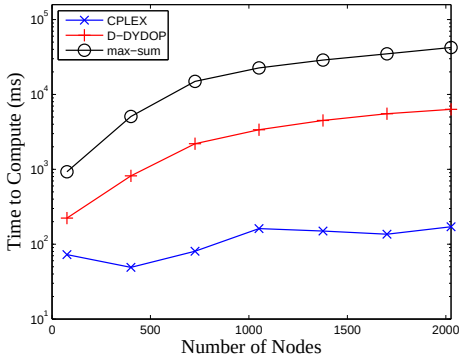
nodes, compared with the Indian electricity distribution network, max-sum is affected much more by the branching factor of a network as apposed to a larger number of total nodes in a network (see Section 4.4.4 for the branching factor experiment).

Figures 4.3(c) and 4.3(d) show how the total number of message elements sent (i.e., the sum of the total message sizes) is affected by the discretisation unit ω for D–DYDOP and max-sum. Note that CPLEX cannot be compared to the other algorithms using the discretisation unit because it does not use message passing to calculate a solution. The total number of message elements sent grows exponentially with regard to ω for D–DYDOP and max-sum. For small ω, D–DYDOP and max-sum send a large number of message elements. As explained previously in this section, this is because both algorithms must iterate through a large number of states in order to calculate the utility for each resultant flow along a distribution cable, and consequently send more message elements. It can be seen that for each discretisation unit, max-sum sends almost double the number of message elements compared to D–DYDOP. This is because max-sum calculates the utility for states that are infeasible, as shown in Section 3.3. Having presented the first experiment, the following section details the second experiment.

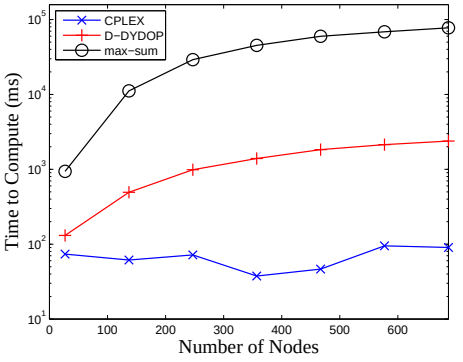
4.4.3 Experiment 2 : Impact of Varying Network Size

Experiment 2 was set up to demonstrate how the size of the network affects CPLEX, D–DYDOP, and max-sum. Using both the Indian and UK electricity distribution networks, the number of additional nodes at each substation was varied from 0 to 30 in steps of 5, each with 50 iterations. At 30 additional nodes per substation, the total number of nodes in the network was 2026 and 687 for the Indian and UK distribution networks respectively. During each iteration, the nodes and distribution cables are initialised as in Section 4.4.1. Figure 4.4 shows four plots of the results from the second experiment (error bars showing the standard error across the 50 iterations omitted due to being negligible). We use a logarithmic scale for the y-axis in all four plots.

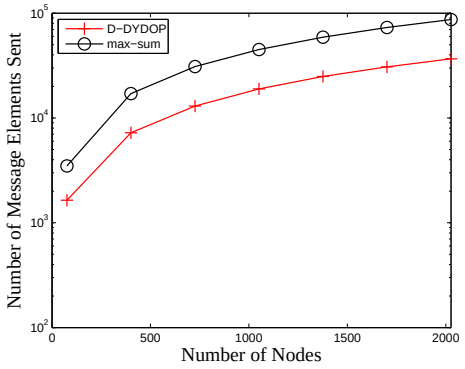
Figure 4.4(a) and 4.4(b) show how the computation time is affected by the number of nodes at each substation (and consequently the total number of nodes in the network) for CPLEX, D–DYDOP, and max-sum on the Indian and UK electricity distribution networks respectively. The time complexity of each algorithm is linear in the total number of nodes in the network. CPLEX has an almost constant computation time of 100ms on average (for very large networks, the effects of the network size on CPLEX would obviously be more apparent, but still linear). Max-sum has the worst computation time, quickly reaching over 5 seconds to calculate a solution when there are more than 300 nodes in the Indian electricity distribution network, and over 5 seconds when there are more than 60 nodes in the UK electricity distribution network. Again, this is because of the wasted computation that max-sum undertakes when calculating the utility for infeasible states. Moreover, the effect of the branching factor on max-sum can be seen



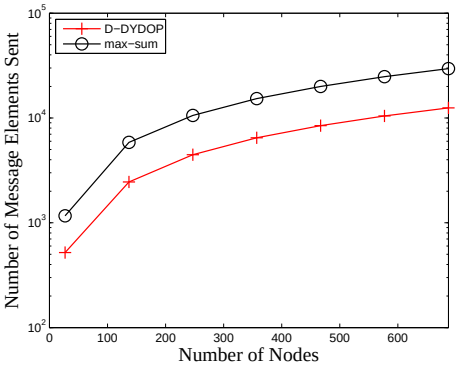
(a) Indian electricity distribution network.



(b) UK electricity distribution network.



(c) Indian electricity distribution network.

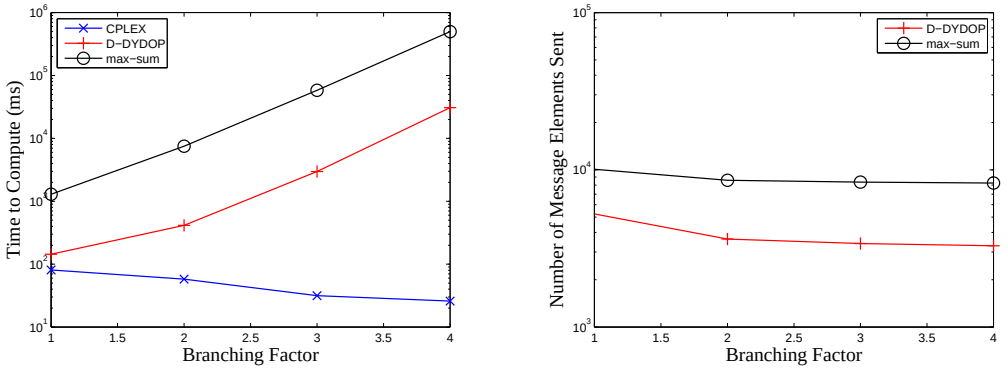


(d) UK electricity distribution network.

FIGURE 4.4: Experiment 2 tests how the size of the network affects CPLEX, D–DYDOP, and max-sum. Using both electricity distribution network topologies in Figure 4.2, the number of additional nodes at each substation was varied from 0 to 30 in steps of 5, each with 50 iterations. 4.4(a) and 4.4(b) show how the number of nodes in the network affects the computation time for CPLEX, D–DYDOP, and max-sum on the Indian and UK electricity distribution networks respectively. 4.4(c) and 4.4(d) show how the number of nodes in the network affects the total number of message elements sent for D–DYDOP and max-sum on the Indian and UK electricity distribution networks respectively. We use a logarithmic scale for the y-axis in all four plots.

between the two networks by comparing Figures 4.4(a) and 4.4(b). As discussed in the previous section, even with fewer nodes in the UK electricity distribution network, the higher number of high branching factor nodes greatly affects the computation time of max-sum.

However, the large reduction in computation time by using a dynamic programming approach, and propagating messages from leaf nodes up to the root of the tree, can be clearly seen from the results of D–DYDOP. For 2026 nodes in the Indian electricity distribution network, Figure 4.4(a), there is a reduction of computation time by a factor of 10 for D–DYDOP compared with max-sum. For 687 nodes in the UK electricity distribution network, Figure 4.4(b), there is a reduction of computation time by a factor of 65 for D–DYDOP compared with max-sum.



(a) (b)

FIGURE 4.5: Experiment 3 tests how the branching factor of the electricity distribution network affects CPLEX, D–DYDOP, and max-sum. Using random acyclic electricity distribution network topologies with 200 nodes, the branching factor of the network was varied from 1 to 4 with 50 iterations for each. 4.5(a) shows how the branching factor of the network affects the computation time for CPLEX, D–DYDOP, and max-sum. 4.5(b) shows how the branching factor of the network affects the total number of message elements sent for D–DYDOP and max-sum. We use a logarithmic scale for the y-axis in both plots.

Figures 4.4(c) and 4.4(d) show how the total number of message elements sent is affected by the number of nodes at each substation for D–DYDOP and max-sum on the Indian and UK electricity distribution networks respectively. The total number of message elements sent for D–DYDOP and max-sum grows linearly with regard to the total number of nodes in the network. Max-sum sends more than twice as many message elements compared with D–DYDOP for both the Indian and UK electricity distribution networks. Having presented the second experiment, the following section details the final experiment.

4.4.4 Experiment 3 : Impact of Varying Branching Factor

Finally, Experiment 3 was set up to demonstrate how the branching factor of the network affects CPLEX, D–DYDOP, and max-sum. Random acyclic electricity distribution network topologies were used to test this aspect of the algorithms since a variation of the branching factor was required. The number of nodes in the electricity distribution network was fixed at 200, and the branching factor of the network was varied from 1 to 4 with 50 iterations for each. During each iteration, a random acyclic electricity distribution network was generated with the nodes and distribution cables initialised as in Section 4.4.1. Figure 4.5 shows two plots of the results from the third experiment (error bars showing the standard error across the 50 iterations omitted due to being negligible). We use a logarithmic scale for the y-axis in both plots.

Figure 4.5(a) shows how the computation time is affected by the branching factor of the network for CPLEX, D–DYDOP, and max-sum. CPLEX has constant computation time with regard to the branching factor of the network because it does not use the structure of the network in order to solve the optimal dispatch problem; it simply solves a very large optimisation problem, that includes every node and distribution cable, in a centralised fashion. On the contrary, D–DYDOP and max-sum both have an exponential time complexity with regard to the branching factor of the network. This is because both D–DYDOP and max-sum must iterate through every possible combination of generator power outputs and message elements in order to calculate a utility for each state of a distribution cable. As the branching factor increases, the number of possible combinations grows exponentially, shown in Proposition 4.4. Max-sum performs significantly worse due to the wasted computation that it undertakes when calculating utilities for infeasible states of the distribution cables.

Figure 4.5(b) shows how the total number of message elements sent is affected by the branching factor of the network for D–DYDOP and max-sum. The total number of message elements sent for D–DYDOP and max-sum decreases exponentially in the branching factor of the network. This is because as the number of children for each node increases, the height of the network (i.e., from leaf node to root node) decreases meaning that fewer messages (and consequently fewer message elements) must be sent in order to calculate a solution (see Section 4.1.1 for how D–DYDOP constructs message elements). The reasons why D–DYDOP sends fewer message elements than max-sum have already been explained in Section 4.4.2. The following section discusses the results from all three experiments.

4.4.5 Discussion

Our results show that D–DYDOP significantly outperforms a naïve implementation of max-sum, for the optimal dispatch problem, in terms of the total number of message elements sent and the computation time by pruning the search space efficiently. When compared to the centralised CPLEX approach, both D–DYDOP and max-sum are significantly slower in terms of computation time. However, due to the disadvantages of a centralised approach highlighted in Section 2.3.2, a decentralised and distributed algorithm, which solves the optimal dispatch problem, may be the only solution. The following section concludes this chapter.

4.5 Conclusions

To address the redundant computation issues which result from a naïve implementation of max-sum (introduced in Section 3.3), in this chapter we presented D–DYDOP, a

novel decentralised message passing algorithm which uses dynamic programming, that outperforms max-sum by pruning the search space. It does this by propagating messages from leaf nodes to the root and only calculates the utility for feasible variable states using techniques based on local consistency. We empirically evaluated D–DYDOP using two real electricity distribution network topologies based in India and the UK, showing that it outperformed max-sum (in terms of computational time and total size of messages sent).

As we have shown in Sections 4.4.2, 4.4.3, and 4.4.4, however, discrete algorithms are greatly affected by the branching factor of the network. Moreover, in order for a solution to be generated by D–DYDOP or max-sum, the electricity distribution network constraints must be discretised, as presented in Section 3.4. While this process can be completed in a distributed and decentralised way, there are only certain scenarios where a discrete algorithm is a viable solution to the optimal dispatch problem (this is discussed in depth in Section 3.4). Therefore, to avoid the need to discretise the electricity distribution network constraints, and address the issues associated with the branching factor of a network when using discrete algorithms, the following chapter extends D–DYDOP to use continuous variables.

Chapter 5

A Novel Algorithm for Decentralised Optimal Dispatch in Acyclic Electricity Networks Using Continuous Variables

D–DYDOP suffers from a number of issues due to the discretisation of the electricity distribution network constraints (see Section 3.4). Therefore, in the following section we present C–DYDOP, which extends D–DYDOP, such that discretising the electricity distribution network constraints is not necessary. We do this by using continuous, instead of discrete, variables. This extension is non-trivial because it not only requires a new way of representing *PowerCost* messages, using piecewise linear functions, but also requires additional techniques for creating and merging *PowerCost* messages, with a number of special cases. Continuous variables have been used in max-sum using similar techniques (Stranders et al., 2009). We use the notion of piecewise linear functions and adapt it to be used with C–DYDOP.[1] Thus, this chapter makes the following contributions to the state of the art:

1. We present C–DYDOP which extends D–DYDOP by using continuous variables for the power outputs of the generators, the loads, and the thermal capacities of the distribution cables. The advantage of using continuous instead of discrete values, is that the real electricity distribution network constraints do not need to be discretised before C–DYDOP can be applied. Moreover, since continuous *PowerCost* messages can now be represented by continuous piecewise linear functions,

[1] To clarify, C–DYDOP differs from continuous max-sum in (Stranders et al., 2009) because, like D–DYDOP, C–DYDOP uses dynamic programming and local consistency to prune much of the search space.

fewer *flowCO* elements need to be created, which greatly reduces the amount of computation required.

2. We provide proof of the optimality of our algorithm and empirically evaluate it on a variety of large real electricity distribution network topologies, showing that it outperforms D–DYDOP in terms of computational time and total size of messages sent.

By presenting C–DYDOP, we address the drawbacks of D–DYDOP and increase the standard of benchmarks for the deployment of agent-based coordination algorithms to solve the optimal dispatch problem in the smart grid.

The remainder of this chapter is organised as follows: Section 5.1 describes the two message passing phases of C–DYDOP including the construction and merging of continuous messages as they are propagated up the acyclic network. Section 5.2 provides a proof of the completeness and correctness of C–DYDOP, and Section 5.3 calculates the computational complexity with regard to the size and topology of the network. Section 5.4 provides an empirical evaluation against D–DYDOP, presented in Chapter 4, and a highly optimised centralised approach based on MIP. Finally, Section 5.5 concludes.

5.1 Message Passing Phases

In order to extend D–DYDOP we use the continuous model of an electricity distribution network, presented in Section 3.1, in its entirety. The mechanism for propagating messages is exactly the same as D–DYDOP (i.e., a value calculation phase and a value propagation phase). We elaborate on the value calculation phase in the following section.

5.1.1 Phase 1: Value Calculation

In what follows, we present the continuous versions of constructing and merging *PowerCost* messages, and constructing *OPCStates*. Section 5.1.1.1 describes a new formulation for continuous *PowerCost* messages and *OPCStates*, Section 5.1.1.2 shows how a leaf node constructs a continuous *PowerCost* message, and Section 5.1.1.3 details how a node merges continuous *PowerCost* messages from its children.

5.1.1.1 Continuous *PowerCost* Messages

A continuous *PowerCost* message sent from v_i to its parent $\hat{v}_i$, is a piecewise linear function[2] which describes the CO_2 emissions that occur between a *range* of power flowing

[2]We use a piecewise linear function because it represents the underlying utility of carbon emissions from a generator against the generator's power output.

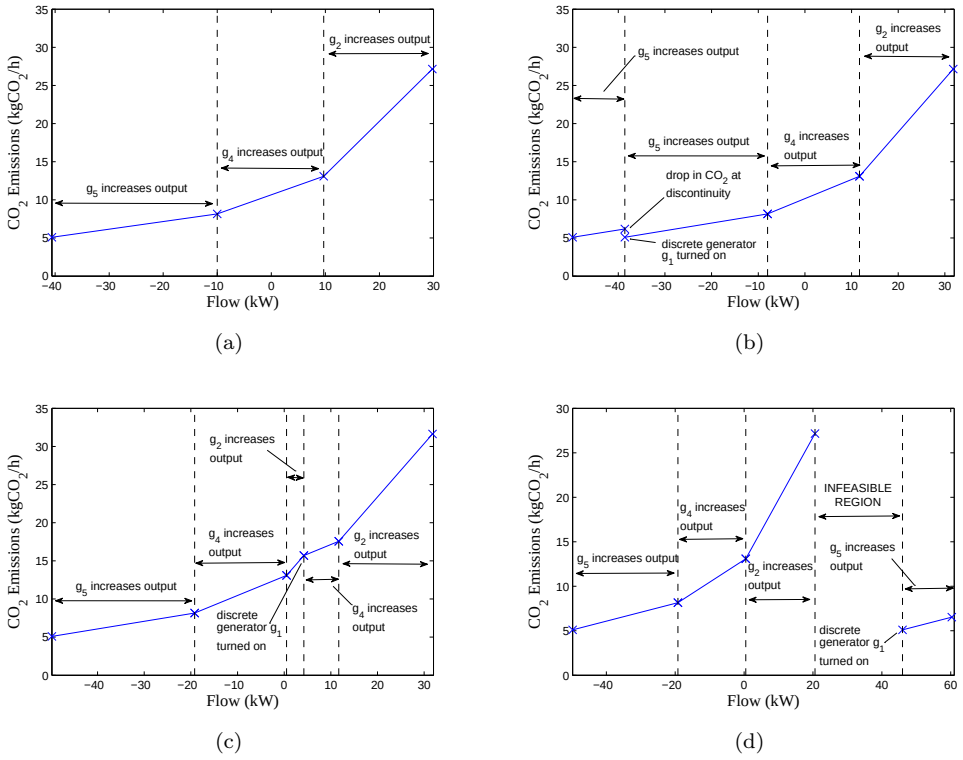


FIGURE 5.1: Continuous *PowerCost* messages, sent within the electricity distribution network in Figure 3.1(a), depicted as piecewise linear functions, where 5.1(a) is the continuous $PowerCost_{5\to3}$ message sent from v_5 to v_3, 5.1(b) is the continuous $PowerCost_{3\to1}$ message sent from v_3 to v_1, 5.1(c) is the continuous $PowerCost_{3\to1}$ message when g_1 has a carbon intensity of 0.4kgCO_2/kWh (instead of 0kgCO_2/kWh), and 5.1(d) is the continuous $PowerCost_{3\to1}$ when g_1 has discrete power values of 0kW and 95.8kW (instead of 0kW and 11.2kW).

along the distribution cable $t_{i\hat{i}}$. The power comes from generators at v_i and all generators in the subtree below it outputting certain amounts of power whilst satisfying their loads. The gradient of each line segment of the piecewise linear function corresponds to the carbon intensity of one of the generators either at v_i or in the subtree below it. If v_i and all nodes in the subtree below it contain generators that are continuous, then the message sent to $\hat{v_i}$ will be a monotonically increasing piecewise linear function, as described in Figure 5.1(a). However, if v_i or its subtree contains discrete generators, then the piecewise linear function may be discontinuous and won't necessarily be monotonically increasing, as described in Figures 5.1(b) – 5.1(d).

Figure 5.1 shows the continuous *PowerCost* messages, sent within the electricity distribution network in Figure 3.1(a), depicted as piecewise linear functions. Each figure shows a possible type of continuous *PowerCost* message that can be sent using C–DYDOP (i.e., monotonically increasing gradients, Figure 5.1(a), discontinuous, Figure 5.1(b), non-monotonically increasing gradients, Figure 5.1(c), and infeasible power ranges, Figure 5.1(d)). A continuous *PowerCost* message could be very complex and contain an amalgamation of more than one type.

In more detail, Figure 5.1(a) describes continuous $PowerCost_{5\to3}$ that v_5 sends to v_3. The x-axis corresponds to the power flowing along t_{35}, and the y-axis corresponds to the resulting CO_2 emissions when g_2, g_4, and g_5 output certain amounts of power. It should be noted that for each line segment of the piecewise linear function, the cleanest generator available (i.e., the generator with smallest CI_i that is not at its maximum output) increases its output. All other generators that this particular continuous *PowerCost* message represents stay at the same output. Thus, between –40.6kW and –10kW g_5 will increase its output, between –10kW and 9.7kW g_4 will increase its output, and finally between 9.7kW and 29.8kW g_2 will increase its output.

Figure 5.1(b) describes continuous $PowerCost_{3\to1}$ that v_3 sends to v_1 on the example electricity distribution network given in Figure 3.1(a). The x-axis corresponds to the power flowing along t_{13}, and the y-axis corresponds to the resulting CO_2 emissions when g_1, g_2, g_4, and g_5 output certain amounts of power. The discontinuity at –38.6kW indicates that the discrete generator g_1 has switched from producing 0kW of power to 11.2kW of power. The drop in CO_2 emissions at –38.6kW is because g_1 has a lower carbon intensity (i.e., 0kgCO_2/kWh) than g_2, g_4, and g_5. Thus, switching g_1 on allows the same amount of power that could have been produced by just g_2, g_4, and g_5 to be produced at a lower cost.

Figure 5.1(c) describes an alternative continuous $PowerCost_{3\to1}$ message which results from g_1 having a carbon intensity of 0.4kgCO_2/kWh instead of 0kgCO_2/kWh. The non-monotonicity of the gradients is because at 4.23kW, g_1 can switch on (i.e., producing 11.2kW) whilst g_2 (the generator with the highest carbon intensity in the current subtree) can be switched off, and g_4 can reduce its power output. Therefore, the line segment immediately following 4.23kW has a gradient of CI_4. This is because the additional power from g_1 means that g_4 now has some available power to produce before it reaches its maximum output. This results in less CO_2 emissions than if g_1 had remained off.

Finally, Figure 5.1(d) describes another alternative continuous $PowerCost_{3\to1}$ message which results from g_1 having discrete power values of 0kW and 95.8kW instead of 0kW and 11.2kW. The infeasible region indicates that between 20.6kW and 46kW, power cannot feasibly flow along t_{13} due to the configuration of the generators. This is because at 20.6kW, g_1 is off, and g_2, g_4, and g_5 are all at their maximum outputs. When g_1 is switched on it produces 95.8kW of additional power. Even if g_2, g_4, and g_5 are all switched off, 46kW is the minimum amount of power that can flow along t_{13}. Thus, an infeasible region of power is created in the continuous *PowerCost* message.

Each piecewise linear function is represented as an array of y *flowCO* elements:

$$PowerCost_{i\to\hat{i}} = \left[flowCO_1, \ldots, flowCO_y\right] \tag{5.1}$$

A *flowCO*$_j$ element describes the jth line segment of the piecewise linear function:

$$\mathit{flowCO}_j = \left\langle \left[f_{i\hat{i}}^{min}, f_{i\hat{i}}^{max} \right], \delta, \left[\boldsymbol{\gamma}(f_{i\hat{i}}^{min}), \boldsymbol{\gamma}(f_{i\hat{i}}^{max}) \right] \right\rangle \tag{5.2}$$

where $[f_{i\hat{i}}^{min}, f_{i\hat{i}}^{max}]$ denotes the range of power flowing along t_{ij}, δ denotes the gradient of the line segment, and $[\boldsymbol{\gamma}(f_{i\hat{i}}^{min}), \boldsymbol{\gamma}(f_{i\hat{i}}^{max})]$ denotes the range of corresponding CO_2 emissions. Each *flowCO*$_j$ element that v_i calculates, maps to an *OPCState*$_j$ describing the power output at v_i, along with the flows between v_i and its children, that results in the range of CO_2 emissions $\left[\boldsymbol{\gamma}(f_{i\hat{i}}^{min}), \boldsymbol{\gamma}(f_{i\hat{i}}^{max}) \right]$:

$$\mathit{OPCState}_j = \langle [\alpha_i, \mathbf{F}(v_i)], var_i \rangle \tag{5.3}$$

where the array $[\alpha_i, \mathbf{F}(v_i)]$ contains v_i's own power output and the set of power flows $\mathbf{F}(v_i)$ from the distribution cables connecting v_i to its children when $f_{i\hat{i}}^{min}$ is flowing along $t_{i\hat{i}}$, and var_i indicates the node that varies its output between $[f_{i\hat{i}}^{min}, f_{i\hat{i}}^{max}]$.[3] This mapping represents the dynamic programming aspect of C–DYDOP because as power flow values are propagated down the tree, during the value propagation phase, the associated *OPCState* is used to find node v_i's power output given a particular power flow $f_{i\hat{i}}$. Consider the continuous *PowerCost*$_{5\to3}$ in Figure 5.1(a) using the notation of Equations (5.1) – (5.3):

$$\mathit{PowerCost}_{5\to3} = \begin{array}{ll} \langle [-40.6, -10], 0.1, [5.1, 8.16] \rangle & \to \langle [\alpha_2 = 0, f_{58} = -20.3, f_{59} = -10.8], v_9 \rangle \\ \langle [-10, 9.7], 0.25, [8.16, 13.085] \rangle & \to \langle [\alpha_2 = 0, f_{58} = -20.3, f_{59} = 19.8], v_8 \rangle \\ \langle [9.7, 29.8], 0.7, [13.085, 27.155] \rangle & \to \langle [\alpha_2 = 0, f_{58} = -0.6, f_{59} = 19.8], v_5 \rangle \end{array} \tag{5.4}$$

which shows the continuous *PowerCost* message sent to v_3 and the mapping from each *flowCO* element to *OPCState* that v_5 stores for use during the propagation phase. For example, the *flowCO* element $\langle [-40.6, -10], 0.1, [5.1, 8.16] \rangle$ details that for power flowing along t_{35} in the range of $[-40.6, -10]$, the CO_2 emissions for nodes v_5, v_8, and v_9 will increase from 5.1kgCO_2/h to 8.16kgCO_2/h at a rate of 0.1 (i.e., CI_5). The corresponding *OPCState* $\langle [\alpha_2 = 0, f_{58} = -20.3, f_{59} = -10.8], v_9 \rangle$ denotes that when 40.6kW flows from v_3 to v_5 along t_{35}, g_2 will output 0kW, 20.3kW will flow from v_5 to v_8 along t_{58}, and 10.8kW will flow from v_5 to v_9 along t_{59}. The node that will increase its output with the flow range of the *flowCO* element is v_9. The resulting CO_2 emissions for nodes v_5, v_8, and v_9 will be 5.1kgCO_2/h.

Thus, if 20kW were to flow from v_3 to v_5, $\alpha_2 = 0$kW, $f_{58} = -20.3$kW and $f_{59} = f_{59} + |f_{35}^{min} - f_{35}| = -10.8 + |-40.6 - -20| = 9.8$kW with a corresponding CO_2 emissions of $\boldsymbol{\gamma}(f_{35}^{min}) + (|f_{35}^{min} - f_{35}| \times \delta) = 5.1 + (20.6 \times 0.1) = 7.16kgCO_2$/h.

In the following sections, in order to clearly explain the construction and merging of continuous *PowerCost* messages effectively, we assume that each node v_i in the network

[3] The definition assumes one generator at v_i. The generalisation to more than one generator at each node is trivial.

Algorithm 3 Constructing continuous *PowerCost* messages at leaf nodes.

```
  constructContinuousLeafMessage() {
1.  f_ii^min ← minFlow(); //Calculate the minimum flow
2.  f_ii^max ← maxFlow(); //Calculate the maximum flow
3.  α_i ← calculateOutput(f_ii^min); //Calculate the power output of the generator
4.  flowCO ←createFlowCO(f_ii^min, f_ii^max); //Create flowCO element and store in PowerCost
                                              //message
5.  OPCState←linkToOPCState(flowCO); //Link flowCO element to OPCState
6.  sendPowerCostMessageToParent();
  }
```

contains one generator g_i and one load l_i; the generalisation to more than one generator and more than one load at each node is trivial.

5.1.1.2 Constructing a Continuous *PowerCost* Message at a Leaf Node

A leaf node v_i with one continuous generator g_i and a corresponding carbon intensity CI_i will be a piecewise linear function with only one line segment (i.e., it will be a linear function and will be constructed with one *flowCO* element),[4] $\delta = CI_i$, and $f_{i\hat{i}}^{min}$, $f_{i\hat{i}}^{max}$, $\boldsymbol{\gamma}(f_{i\hat{i}}^{min})$, and $\boldsymbol{\gamma}(f_{i\hat{i}}^{max})$ calculated as follows:

$$f_{i\hat{i}}^{min} = \max\left(\alpha_i^{min} + \beta_i, -t_{i\hat{i}}^{c}\right) \qquad f_{i\hat{i}}^{max} = \min\left(\alpha_i^{max} + \beta_i, t_{i\hat{i}}^{c}\right) \tag{5.5}$$

$$\boldsymbol{\gamma}(f_{i\hat{i}}^{min}) = |\beta_i - f_{i\hat{i}}^{min}|CI_i \qquad \boldsymbol{\gamma}(f_{i\hat{i}}^{max}) = |\beta_i - f_{i\hat{i}}^{max}|CI_i \tag{5.6}$$

Algorithm 3 gives a pseudocode representation of constructing a continuous *PowerCost* message at a leaf node v_i. The minimum and maximum flow along $t_{i\hat{i}}$ is calculated using Equation (5.5) (lines 1 – 2). The power output that needs to be produced for the minimum flow $f_{i\hat{i}}^{min}$ is $\alpha_i = f_{i\hat{i}}^{min} - \beta_i$ (line 3). The *flowCO* element is constructed using $f_{i\hat{i}}^{min}$ and $f_{i\hat{i}}^{max}$ (line 4), and linked to the *OPCState* containing α_i indicating that v_i varies its output (line 5). The continuous *PowerCost* message is sent to $\hat{v}_i$ (line 6). For clarification, consider the following example where v_9 sends a continuous $PowerCost_{9\to5}$ message to v_5 using the real (and not discretised) electricity distribution network from 3.1(a) (compare with the previous example in Section 4.1.1.2 which sends a discrete $PowerCost_{9\to5}$ message). The following is the resulting continuous $PowerCost_{9\to5}$ message:

$$PowerCost_{9\to5} = \langle [-10.8, 19.8], 0.1, [0, 3.06] \rangle \to \langle [\alpha_5 = 0], v_9 \rangle \tag{5.7}$$

Thus, for the power flowing along t_{59} in the range of $[-10.8, 19.8]$, the CO_2 emissions will increase from 0kgCO_2/h to 3.06kgCO_2/h at a rate of 0.1 (i.e., CI_5). If 5kW of power were to flow from v_9 to v_5, $\alpha_5 = \alpha_5^{min} + |f_{59}^{min} - f_{59}| = 0 + |-10.8 - 5| = 15.8$kW

[4] For n generators at node v_i, the resulting continuous *PowerCost* message will contain up to n line segments (defined by n *flowCO* elements).

with a corresponding CO_2 emissions of $\boldsymbol{\gamma}(f_{59}^{min}) + (|f_{59}^{min} - f_{59}| \times \delta) = 0 + (15.8 \times 0.1) = 1.58\text{kg}CO_2/\text{h}$.

The total number of *flowCO* elements in the continuous $PowerCost_{9\to5}$ is 1. By contrast, compare the example discrete $PowerCost_{9\to5}$ message of D–DYDOP in Section 4.1.1.2 which has 31 *flowCO* elements, and the example discrete $R_{9\to13}(x_{13})$ message of max-sum in Section 3.3, which has 81 variable values. This further highlights the wasted computation that the discrete algorithms perform. The following section describes how continuous *PowerCost* messages are merged.

5.1.1.3 Merging Continuous *PowerCost* Messages

For each v_i that has at least one child, the *PowerCost* messages that it receives (be it continuous or discrete) must be processed in order to produce its own continuous *PowerCost* message that it sends to $\hat{v_i}$. Once v_i has received a *PowerCost* message from each of its children it can then calculate its own *PowerCost* message. The type of *PowerCost* messages v_i receives affects the way in which a new continuous *PowerCost* message is constructed as follows:

Constructing a continuous *PowerCost* message when v_i and all nodes in the subtree below it contain generators that are continuous This is the simplest case because each continuous *PowerCost* message constructed will be exactly the same type (i.e., Figure 5.1(a)).

Constructing a continuous *PowerCost* message when discrete and continuous generators are present This is more complex (containing three specific cases which will be explained in more detail later in this section) because each output of a discrete generator must be iterated through, and the resulting piecewise linear functions must be minimised and merged. Moreover, each continuous *PowerCost* message constructed could be of any type (i.e., Figure 5.1(a) – 5.1(d)).

The former case will be explained first followed by the latter.

First, v_i must calculate $f_{i\hat{i}}^{min}$ and $f_{i\hat{i}}^{max}$ based on $t_{i\hat{i}}$, β_i, and its children's continuous *PowerCost* messages:

$$f_{i\hat{i}}^{min} = \max\left(\alpha_i^{min} + \beta_i + \sum_{c \in chi(v_i)} f_{ci}^{min}, -t_{i\hat{i}}^{c}\right) \tag{5.8}$$

$$f_{i\hat{i}}^{max} = \min\left(\alpha_i^{max} + \beta_i + \sum_{c \in chi(v_i)} f_{ci}^{max}, t_{i\hat{i}}^{c}\right) \tag{5.9}$$

Algorithm 4 Merging continuous *PowerCost* messages.

```
  mergeContinuousMessages() {
1.  createFlowCOElements(); //Create flowCO elements, see Algorithm 5
2.  sendPowerCostMessageToParent();
  }
```

where $\sum_{c \in chi(v_i)} f_{ci}^{min}$ is the sum of the minimum power flow from each continuous *PowerCost* message of v_i's children and $\sum_{c \in chi(v_i)} f_{ci}^{max}$ is the sum of the maximum power flow from each continuous *PowerCost* message. As a result, v_i solves the following optimisation problem in order to calculate the optimal α_i and the corresponding flows to its children, f_{ci}, based on $f_{i\hat{i}}^{min}$:

$$\min \sum_{c \in chi(v_i)} PWL(PowerCost_{c \to i}) + CI_i \alpha_i \tag{5.10}$$

subject to the power flow balancing at v_i:

$$\alpha_i + \beta_i + \sum_{c \in chi(v_i)} f_{ci} - f_{i\hat{i}}^{min} = 0 \tag{5.11}$$

where $PWL(PowerCost_{c \to i})$ is the piecewise linear function of continuous $PowerCost_{c \to i}$. Using the calculated values for α_i and f_{ci}, v_i will then iteratively construct a piecewise linear function, in the form of a continuous *PowerCost* message, based on the gradients of its children's *flowCO* elements and v_i's own carbon intensity CI_i (see Algorithm 4 for a pseudocode representation of merging continuous *PowerCost* messages at node v_i).

The *flowCO* elements of the continuous *PowerCost* message are constructed (line 1), and then sent to $\hat{v_i}$ (line 2). See Algorithm 5 for a pseudocode representation of `createFlowCOMessageElements()`. The minimum and maximum flow along $t_{i\hat{i}}$ is calculated using Equations (5.8) – (5.9) (lines 1 – 2). The optimisation problem in Equations (5.10) – (5.11) is solved giving an optimal value for α_i and each f_{ci} for $f_{i\hat{i}}^{min}$ (line 3). The current flow $f_{i\hat{i}}$ is initialised to the minimum flow $f_{i\hat{i}}^{min}$ (line 4). While $f_{i\hat{i}}$ is smaller than $f_{i\hat{i}}^{max}$ or there is no more power `NMP()` available (line 5 – 11), we iteratively choose the best node to increase its power output (line 6). See Algorithm 6 for a pseudocode representation of `chooseBestNode()`. The available power at that particular cost (i.e., carbon intensity) is calculated (line 7), and a *flowCO* element is constructed (line 8). The *flowCO* element is linked to the *OPCState* containing α_i and each f_{ci} with an indicator of the node that varied its output (line 9). The current flow $f_{i\hat{i}}$ is updated (line 10).

Algorithm 6 gives a pseudocode representation of the steps for choosing the next best node to increase its output. We iterate through each immediate child of v_i (lines 1 – 8).

Algorithm 5 Creating *flowCO* message elements.

```
   createFlowCOElements() {
 1.  f_{ii^}^{min} ← minFlow(); //Calculate the minimum flow
 2.  f_{ii^}^{max} ← maxFlow(); //Calculate the maximum flow
 3.  solveForMinFlow(f_{ii^}^{min}); //Solve optimisation problem for the minimum flow,
                                     //giving optimal value for the power output of the
                                     //generator and each child distribution cable's flow
 4.  f_{ii^} ← f_{ii^}^{min} //Initialise current flow to the minimum
 5.  WHILE (f_{ii^} < f_{ii^}^{max}) OR NMP(){ //While the current flow does not exceed the
                                               //maximum flow, or there is no more power
                                               //available from the generator or its children
 6.    bestNode, bestElement ← chooseBestNode(); //Choose the next node to
                                                 //increase its output, see
                                                 //Algorithm 6
 7.    powerIncrease ← calculatePIncrease(bestNode); //Calculate the power
                                                     //increase
 8.    flowCO←createFlowCO(f_{ii^}, bestElement, powerIncrease); //Create flowCO
                                                                 //element and store
                                                                 //in PowerCost message
 9.    OPCState←linkToOPCState(flowCO); //Link flowCO element to OPCState
10.    f_{ii^} ← f_{ii^} + powerIncrease; //Update current flow
11.  }
   }
```

For each $v_c \in chi(v_i)$, the *flowCO* element at its current flow f_{ci} and the corresponding carbon intensity are retrieved (line 2). If `childCI` is smaller than the best carbon intensity observed so far, then the best carbon intensity `bestCI`, best *flowCO* element `bestElement`, and best node `bestNode` are recorded (lines 3 – 7). This means that given the current power flow f_{ci} of each distribution cable t_{ci} that connects v_c to v_i, the *flowCO* with the smallest carbon intensity (that contains f_{ci}) is chosen to be used because increasing the power from the corresponding v_c will have the least cost. However, if the carbon intensity of v_i is smaller than the chosen child node's `childCI`, and v_i is currently not outputting its maximum, v_i is the best node to use first (lines 9 – 13). The best node to increase its output and the corresponding *flowCO* element are returned (line 14).

As an example of merging continuous *PowerCost* messages, let's consider how the continuous $PowerCost_{5\to3}$ message (5.4) is constructed. First, v_5 must receive continuous $PowerCost_{9\to5}$ message (5.7) and continuous $PowerCost_{8\to5}$ message (5.12) from v_9 and v_8 respectively.

$$PowerCost_{8\to5} = \langle [-20.3, -0.6], 0.25, [5.1, 10.025] \rangle \to \langle [\alpha_4 = 20.4], v_8 \rangle \tag{5.12}$$

The minimum flow between v_5 and v_3 is $f_{35}^{min} = -40.6$kW. Using this value, v_5 solves the optimisation problem in Equations (5.10) – (5.11) giving optimal values of $\alpha_2 = 0$kW, $f_{58} = -20.3$kW, and $f_{59} = -10.8$kW for $f_{35} = -40.6$kW with 5.1kgCO_2 emissions. Thus, v_5 iteratively constructs *flowCO* elements based on the cleanest power available. The carbon intensity at each node's current flow is 0.7, 0.25, and 0.1 for v_5, v_8, and v_9 respectively. Therefore, the cleanest power is available from v_9 with *flowCO* element

Algorithm 6 Choosing the next node to increase its generation.

```
   bestNode, bestElement ← chooseBestNode() {
 1.  FOREACH (v_c ∈ chi(v_i)) { //For each child of v_i
 2.    childCI, childElement ← getCIElementAtCurrentFlow(f_ci); //Retrieve the carbon intensity and the flowCO
                                                                //element which corresponds to the child
                                                                //distribution cable's flow
 3.    IF (childCI < bestCI) { //If the carbon intensity of the child is smaller than the current best carbon
                               //intensity
 4.      bestCI ← childCI; //Update the best carbon intensity with the carbon intensity of the child
 5.      bestElement ← childElement; //Update the best flowCO element with the flowCO element of the child
 6.      bestNode ← v_c; //Update the best node with the child node
 7.    }
 8.  }
 9.  IF (CI_i < bestCI) AND (α_i < α_i^max) { //If the carbon intensity of v_i is smaller than the best carbon
                                              //intensity and the generator at v_i is not at its maximum output
10.    bestCI ← CI_i; //Update the best carbon intensity with the carbon intensity of v_i
11.    bestElement ← flowCO(α_i^min, α_i^max); //Update the best flowCO element with v_i's newly constructed flowCO
                                               //element
12.    bestNode ← v_i; //Update the best node with v_i
13.  }
14.  RETURN bestNode, bestElement;
   }
```

$\langle [-10.8, 19.8], 0.1, [0, 3.06] \rangle$ at a cost of 0.1kgCO_2/kWh and a maximum increase of 30.6kW. Thus, the first *flowCO* element of continuous $PowerCost_{5\to3}$ can be constructed and the flow along t_{59} can be increased to $f_{59} = 19.8$kW.

The *flowCO* element $\langle [-20.3, -0.6], 0.25, [5.1, 10.025] \rangle$ with a cost of 0.25kgCO_2/kWh and maximum increase of 19.7kW is chosen next. Thus, the second *flowCO* element of continuous $PowerCost_{5\to3}$ can be constructed and the flow along t_{58} can be increased to $f_{58} = -0.6$kW. Finally, v_5 chooses its own generator g_2 at a cost of 0.7kgCO_2/kWh and maximum increase of 20.1kW. Thus, the third and final *flowCO* element of continuous $PowerCost_{5\to3}$ can be constructed. There is no more power available so the continuous $PowerCost_{5\to3}$ can be sent to v_3.

Now, consider the more complex cases when there is a discrete generator at v_i or in the subtree below it. There are three cases we must consider:

1. If v_i contains the discrete generator (i.e., g_1 in Figure 3.1(a) when v_3 sends a continuous *PowerCost* message to v_1), to construct the final continuous *PowerCost* message, v_i must iterate through the power output values of the generator (i.e., 0kW and 11.2kW) in order to calculate all possible configurations of the generators. For each power output it can then iteratively construct a piecewise linear function, in the form of a continuous *PowerCost* message, based on the gradients of its children's *flowCO* elements.

2. If one of v_i's immediate children contains the discrete generator and is a leaf node (i.e., g_3 in Figure 3.1(a) when v_6 sends a continuous *PowerCost* message to v_4), v_i will receive a discrete *PowerCost* message with discrete *flowCO* elements as described in Equation (4.2). Thus, v_i must iterate through each discrete *flowCO* element. For each discrete *flowCO* element it can then iteratively construct a piecewise linear function, based on the gradients of its remaining children's continuous *flowCO* elements and the carbon intensity of g_i.

Algorithm 7 Merging continuous *PowerCost* messages.

```
  mergeContinuousDiscreteMessages() {
1.  FOREACH (αi ∈ Si) { //Iterate through each output of the generator
2.    createFlowCOMessageElements(); //Create corresponding flowCO elements
                                     //for generator power output, see Algorithm 5
3.  }
4.  createFinalMessageElements(); //Create final flowCO elements and store
                                  //in PowerCost message, see Algorithm 8
5.  sendPowerCostMessageToParent();
  }
```

3. The most complex case is when there is a discrete generator at a node, v_d, in the subtree of v_i that is not an immediate child leaf node (i.e., g_1 in Figure 3.1(a) when v_1 sends a continuous *PowerCost* message to v_0). The continuous *PowerCost* message v_i receives from the subtree which contains v_d will be one of the types specified in Figures 5.1(b) – 5.1(d). It will contain points where the discrete generator will change to another power output; a *step point*. For example, in Figure 5.1(c), for a power range of $range_0$ = [–49.8kW, 4.23kW] flowing along t_{13}, g_1 is producing 0kW. For a power range of $range_1$ = [4.23kW, 31.8kW], g_1 changes its output to 11.2kW. Thus, in order to iterate through each power output of v_3, v_1 can restrict the amount of power that could travel along t_{13} to $range_0$ and iteratively construct a piecewise linear function, based on the gradients of its children's continuous *flowCO* elements. It can then restrict the amount of power that could travel along t_{13} to $range_1$ and repeat the process. By doing this, v_1 is indirectly iterating through each power output of g_1 in order to calculate all possible configurations of the generators.

Essentially, each possible discrete generator power output must be iterated through in order to calculate all the possible generator configurations. This is done directly, if the discrete generator is situated at v_i and v_i is currently sending a continuous *PowerCost* message, or indirectly, by iterating through each step point of the child continuous *PowerCost* message that contains the discrete generator in its subtree. Once this process has finished, there will be multiple piecewise linear functions that give different costs for the same ranges of power flowing along $t_{i\hat{i}}$. Therefore, the minimum piecewise linear function must be constructed by merging the multiple piecewise linear functions. For clarification, we shall first detail a pseudocode representation (see Algorithm 7) and then present a detailed example.

Algorithm 7 is a pseudocode representation of merging continuous *PowerCost* messages in a network of continuous generators when there exists at least one discrete generator. For this pseudocode, we assume that the discrete generator is situated at v_i sending the continuous *PowerCost* message. Therefore, we iterate through each power output

Algorithm 8 Constructing the final continuous *PowerCost* message from a collection of *flowCO* elements.

```
  createFinalMessageElements() {
1.  flowPointArray ← getStartEndAndIntersectFlowPoints(); //Initialise flowPointArray with the start and
                                                          //end points of each flowCO element, and all
                                                          //intersection points. Sorted by smallest flow
                                                          //point
2.  currentFlowPoint ← flowPointArray[0]; //Initialise currentFlowPoint to first point in flowPointArray
3.  currentElement ← firstFlowCOElement(); //Initialise current flowCO element to the minimum flowCO (i.e.,
                                           //the one with the smallest minimum flow)
4.  FOREACH (nextFlowPoint ∈ flowPointArray) { //Iterate through each flow point in the flowPointArray
5.    nextElement ← getMinFlowCOElement(nextFlowPoint); //Initialise next flowCO element to the minimum
                                                        //flowCO for nextFlowPoint
6.    IF (nextElement != currentElement) {
7.      flowCO ←createFlowCO(currentFlowPoint, nextFlowPoint, currentElement); //Create final flowCO element,
                                                                               //store in PowerCost message
8.      OPCState←linkToOPCState(flowCO); //Link flowCO element to OPCState
9.      currentFlowPoint ← nextFlowPoint; //Initialise currentFlowPoint to nextFlowPoint
10.     currentElement ← nextElement; //Initialise current flowCO element to next flowCO element
11.   }
12. }
  }
```

of the discrete generator situated at v_i (lines 1 – 3).[5] For the current power output of the discrete generator, v_i creates a piecewise linear function based on the gradients of its children's continuous *flowCO* elements (line 2). Multiple piecewise linear functions which give different costs for the same ranges of power that could flow along $t_{i\hat{i}}$ must then be merged together (line 4). See Algorithm 8 for a pseudocode representation of how the final continuous *PowerCost* message is created by merging multiple piecewise linear functions. Finally, the continuous *PowerCost* message is sent to $\hat{v}_i$ (line 5).

Now, Algorithm 8 details the steps to merge the multiple piecewise linear functions, that are created during Algorithm 7 (lines 1 – 3), to produce the final continuous *PowerCost* message. The start and end flow points, and the intersection flow points, of all the *flowCO* elements, that were constructed in Algorithm 7, are initialised to an array (line 1). The first flow point is retrieved from the array (line 2), and the minimum *flowCO* element is retrieved (i.e., the one with the largest negative minimum power flow followed by the smallest minimum CO_2 emissions) (line 3). We iterate through each remaining flow point in the array (lines 4 – 12). For each flow point, the *flowCO* element with the minimum CO_2 emissions at `nextFlowPoint` is retrieved (line 5). It should be noted that if two *flowCO* elements intersect at their extremes (i.e., the maximum of one *flowCO* element lies on the same point as the minimum of another *flowCO* element), the *flowCO* element that intersects with its minimum is chosen. If `nextElement` is not equal to `currentElement` (line 6), a new *flowCO* element is created from `currentElement` between `currentFlowPoint` and `nextFlowPoint` (line 7), and then linked to the *OPCState* containing α_i and each f_{ci} with an indicator of the node that varied its output (line 8). Both `currentFlowPoint` and `currentElement` are updated (lines 9 and 10 respectively). If `nextElement` is equal to `currentElement` (line 6), this indicates that there is still more power available from `currentElement`, so skip to the next iteration.

[5] If there were multiple discrete generators, situated at v_i and further down the tree, v_i would iterate through every possible combination of power outputs from its own discrete generator and each step point from each child continuous *PowerCost* message.

Once this algorithm has finished, the resulting continuous *PowerCost* message consists of the minimum merged *flowCO* elements.

To exemplify the above process, consider how the continuous $PowerCost_{3\to1}$ message (depicted as a piecewise linear function in Figure 5.1(b)) is constructed. First, v_3 must receive continuous $PowerCost_{5\to3}$ message (5.4) from v_5. Using Algorithm 7, v_3 must iterate through each of g_1's power outputs (i.e., 0kW and 11.2kW) and for each power output, construct a piecewise linear function from the *flowCO* elements of continuous $PowerCost_{5\to3}$ message. When g_1 outputs 0kW, the minimum flow between v_1 and v_3 is $f_{13}^{min} = -49.8$kW. Using this value, v_3 solves the optimisation problem in Equations (5.10) – (5.11) giving optimal values of $\alpha_1 = 0$kW and $f_{35} = -40.6$kW for $f_{13} = -49.8$kW with 5.1kgCO_2 emissions. Thus, v_3 iteratively constructs *flowCO* elements based on the cleanest power available (since v_5 is the only child node, its *flowCO* elements are the only *flowCO* elements used):

$$\begin{aligned}
&\langle[-49.8,-19.2],0.1,[5.1,8.16]\rangle &&\to \langle[\alpha_1 = 0, f_{35} = -40.6], v_5\rangle \\
&\langle[-19.2,0.5],0.25,[8.16,13.085]\rangle &&\to \langle[\alpha_1 = 0, f_{35} = -10], v_5\rangle \\
&\langle[0.5,20.6],0.7,[13.085,27.155]\rangle &&\to \langle[\alpha_1 = 0, f_{35} = 9.7], v_5\rangle
\end{aligned} \tag{5.13}$$

Next, v_3 calculates that when g_1 outputs 11.2kW, the minimum flow between v_1 and v_3 is $f_{13}^{min} = -38.6$kW. Using this value, v_3 solves the optimisation problem in Equations (5.10) – (5.11) which gives optimal values of $\alpha_1 = 11.2$kW and $f_{35} = -40.6$kW for $f_{13} = -38.6$kW with 5.1kgCO_2 emissions. Thus, v_3 iteratively constructs *flowCO* elements based on the cleanest power available:

$$\begin{aligned}
&\langle[-38.6,-8],0.1,[5.1,8.16]\rangle &&\to \langle[\alpha_1 = 11.2, f_{35} = -40.6], v_5\rangle \\
&\langle[-8,11.7],0.25,[8.16,13.085]\rangle &&\to \langle[\alpha_1 = 11.2, f_{35} = -10], v_5\rangle \\
&\langle[11.7,31.8],0.7,[13.085,27.155]\rangle &&\to \langle[\alpha_1 = 11.2, f_{35} = 9.7], v_5\rangle
\end{aligned} \tag{5.14}$$

Figure 5.2 shows (5.13) and (5.14) depicted as piecewise linear functions on the same graph.

Finally, v_3 must merge the *flowCO* elements from (5.13) and (5.14) using Algorithm 8, to produce the piecewise linear function in Figure 5.1(b). The `currentElement` is initialised to $\langle[-49.8,-19.2],0.1,[5.1,8.16]\rangle$ and the `currentFlowPoint` is initialised to –49.8. From here, each flow point is used to retrieve the minimum *flowCO* element. Starting with –38.6, the *flowCO* element with the minimum CO_2 emissions at –38.6 is $\langle[-38.6,-8],0.1,[5.1,8.16]\rangle$. Therefore, a new *flowCO* element is created from `currentElement` with –49.8kW and –38.6kW, $\langle[-49.8,-38.6],0.1,[5.1,6.19]\rangle$, and linked to the *OPCState*. The `currentFlowPoint` is assigned –38.6 and `currentElement` is assigned $\langle[-38.6,-8],0.1,[5.1,8.16]\rangle$, and the process repeats. The result is the final

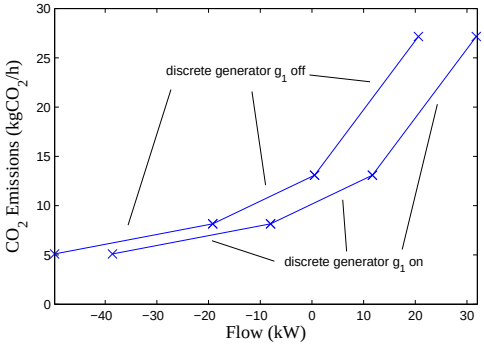


FIGURE 5.2: The two piecewise linear functions that are constructed, by changing the output of g_1 from 0kW to 11.2kW, and then merged to produce the final continuous $PowerCost_{3\to1}$ message sent from v_3 to v_1, Figure 5.1(b).

continuous $PowerCost_{3\to1}$ message which is sent to v_1:

$$PowerCost_{3\to1} = \begin{array}{ll} \langle [-49.8, -38.6], 0.1, [5.1, 6.19] \rangle & \to \langle [\alpha_1 = 0, f_{35} = -40.6], v_5 \rangle \\ \langle [-38.6, -8], 0.1, [5.1, 8.16] \rangle & \to \langle [\alpha_1 = 11.2, f_{35} = -40.6], v_5 \rangle \\ \langle [-8, 11.7], 0.25, [8.16, 13.085] \rangle & \to \langle [\alpha_1 = 11.2, f_{35} = -10], v_5 \rangle \\ \langle [11.7, 31.8], 0.7, [13.085, 27.155] \rangle & \to \langle [\alpha_1 = 11.2, f_{35} = 9.7], v_5 \rangle \end{array} \quad (5.15)$$

By comparing the two piecewise linear functions in Figure 5.2 with the final piecewise linear function sent to v_1 in Figure 5.1(b), it can be seen that the final continuous *PowerCost* message is created by using the minimum *flowCO* elements (i.e., the elements with the smallest CO_2 emissions) when two or more *flowCO* elements overlap the same range of power flow. The following section describes the second phase of C–DYDOP whereby power output values are propagated from the root node to the leaf nodes.

5.1.2 Phase 2: Value Propagation

Once the root node v_i has received *PowerCost* messages from all of its children, it calculates how much power to output in order to satisfy all the loads within the network and minimise CO_2 emissions. It does this by solving the optimisation problem in Equation (5.10) subject to the flows at v_i equaling zero:

$$\alpha_i + \beta_i + \sum_{c \in chi(v_i)} f_{ci} = 0 \quad (5.16)$$

This is a slightly simpler optimisation problem than when nodes merge *PowerCost* messages because the minimum flow to the parent node does not need to be calculated since the root does not have a parent. The solution to Equation (5.16) will produce an optimal power output α_i and an optimal flow value f_{ci} for each of its children. As with D–DYDOP, flow values are then sent to each of the root node's children telling them which of their *flowCO* elements resulted in the minimum CO_2 emissions. The children retrieve the correct *flowCO* element by matching the power flow value sent to them with

the range of flow from the *flowCO* message. The *OPCState* which is referenced by each child recipient's corresponding *flowCO* element tells the child exactly how much power to output. The child recipient can then send the power flow specified in the *OPCState* to each of its corresponding children. Power flow values are propagated in this manner to the leaf nodes, at which point each node in the network knows their optimum power output that results in the minimum CO_2 emissions for the entire network. It should be noted that if there is no solution to Equation (5.16), no solution exists for the real electricity distribution network (i.e., CPLEX would also report a solution does not exist) which C–DYDOP reports.

Having introduced C–DYDOP, we now present the completeness and correctness of the algorithm.

5.2 Completeness and Correctness

In what follows, we prove that C–DYDOP applied to acyclic networks is complete[6] and correct.[7]

Proposition 5.1. *C–DYDOP is complete.*

Proof. This proof follows on from Proposition 4.1. Leaf nodes, v_i, construct their continuous *PowerCost* messages by summarising the cost of producing power, whilst satisfying each $l \in \mathbf{L}(v_i)$, within the feasible range of power that will be flowing between v_i and $\hat{v_i}$. As continuous *PowerCost* messages are propagated up the tree, each v_i also calculates the cost of producing power within the feasible range of power flowing between v_i and $\hat{v_i}$. The root node receives continuous *PowerCost* messages which summarise the entire cost of the network to produce power for all feasible flows of each distribution cable within the network. Thus, at each node, all feasible ranges of flow are evaluated and the root node chooses the optimal state which minimises CO_2. Hence, the algorithm is complete. □

Proposition 5.2. *C–DYDOP is correct.*

Proof. This proof follows on from Proposition 5.1 and is exactly the same as Proposition 4.2. Any solution calculated by the algorithm will be valid as it has explicitly conformed to the local and global constraints of the entire network (since constraint checks are explicitly embedded in the algorithm). Hence, the algorithm is correct. □

Having presented the correctness and completeness of C–DYDOP, we now calculate the computational complexity.

[6]Complete in terms of finding the optimal solution calculated by CPLEX using Equations (3.1) – (3.4).

[7]Correct such that any solution returned by C–DYDOP is feasible given Equations (3.1) – (3.4).

5.3 Computational Complexity

Here, the worst-case complexity of C–DYDOP is calculated, with regard to the size of the network, in order to show its suitability for large optimal dispatch problems (Requirement III).

Proposition 5.3. *The size of continuous PowerCost messages sent by C–DYDOP grows polynomially with the size of the network.*

Proof. Consider a network of k nodes with each node containing one generator. Thus, $k = n$, where n is the number of generators in the network. The worst case is when the network is a line of connected nodes with a single leaf node v_i; since the computation must be completed sequentially. The size of v_i's continuous *PowerCost* message will be 1 since it contains one *flowCO* element describing the cost of g_i producing power α_i with a certain carbon intensity CI_i. When v_i's parent $\hat{v_i}$ calculates its own continuous *PowerCost* message, in the worst case, its continuous *PowerCost* message will contain two *flowCO* elements. Thus, as each consecutive node v_i calculates its own continuous *PowerCost* message, the size of their continuous *PowerCost* message will be $|PowerCost_{c \to i}| + 1$ where c is the child of v_i. The total size of the message sent by C–DYDOP will be:

$$\frac{n\,(n+1)}{2} \tag{5.17}$$

Therefore, the size of the messages C–DYDOP sends, in the worst case, grows polynomially in $O(n^2)$. □

Having presented C–DYDOP and analysed its theoretical properties, the following section provides an empirical evaluation against D–DYDOP and a highly optimised centralised approach based on MIP.

5.4 Empirical Evaluation

To highlight the improvements of C–DYDOP against D–DYDOP (presented in Chapter 4), we conducted two experiments on the two large real electricity distribution network topologies from Section 4.4 (see Figure 4.2), and one experiment on large random acyclic electricity distribution network topologies.[8] We benchmark C–DYDOP and D–DYDOP against a highly optimised centralised approach, which uses IBM's ILOG CPLEX 12.2.[9] CPLEX simply solves a large MIP without having to use message passing or decentralised

[8] We use random topologies in order to vary the branching factor of each node.

[9] Note, we do not benchmark C–DYDOP against max-sum as we have already benchmarked D–DYDOP against max-sum (in Section 4.4) and showed that D–DYDOP is more efficient at calculating a solution to the optimal dispatch problem than max-sum (in terms of computational time and number of message elements sent).

control. Thus, CPLEX is able to calculate a solution in under a second. The three experiments were conducted in order to test the following:

Experiment 1 Tests the effect of ω for CPLEX, C–DYDOP, and D–DYDOP on the two large real electricity distribution network topologies in Figure 4.2.

Experiment 2 Tests the effect of the size of the network for CPLEX, C–DYDOP, and D–DYDOP on the two large real electricity distribution network topologies in Figure 4.2.

Experiment 3 Tests the effect of the branching factor for CPLEX, C–DYDOP, and D–DYDOP on large random acyclic electricity distribution network topologies.

The remainder of this section is organised as follows: Section 5.4.1 describes the setup of the electricity distribution networks. Section 5.4.2 details experiment 1, Section 5.4.3 details experiment 2, and Section 5.4.4 details experiment 3. Finally, Section 5.4.5 draws conclusions from all three experiments.

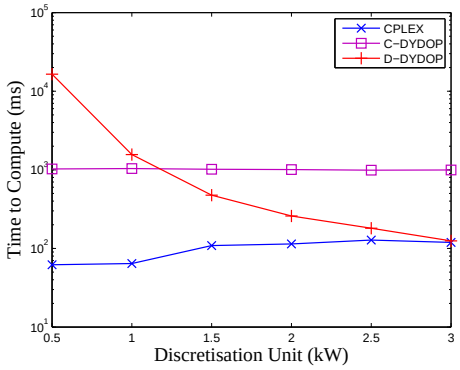
5.4.1 Experiment Setup

Each experiment was run in Java on a 2.67GHz Intel Xeon quadcore with 12GB of RAM. During each iteration, nodes are assigned a uniformly distributed load value in the range of [1kW, 5kW], and either a continuous or discrete generator with a uniformly distributed carbon intensity. There is a 90% chance that the generator will be continuous.[10] If the generator is continuous, it is assigned a uniformly distributed minimum power output in the range of [0kW, 2kW], and a uniformly distributed maximum power output in the range of [3kW, 20kW]. If the generator is discrete, it is assigned a uniformly distributed power output level η in the range of [3kW, 20kW] (i.e., each discrete generator can either be off, or produce ηkW). Each distribution cable in the network is assigned a uniformly distributed thermal capacity in the range of [10kW, 15kW]. The electricity distribution network constraints are then discretised with the equations in (3.7) (where ω is varied between 0.5kW and 3.0kW in experiment 1, and $\omega = 1$kW in experiments 2 and 3), in order to apply D–DYDOP. Having described the setup for each experiment, the following section details the first experiement.

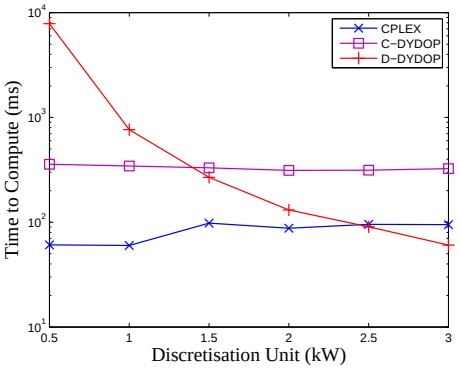
5.4.2 Experiment 1 : Impact of Varying Discretisation Unit

Experiment 1 was set up in order to test the effect of ω for CPLEX, C–DYDOP, and D–DYDOP. Using both the Indian and UK electricity distribution networks, the number

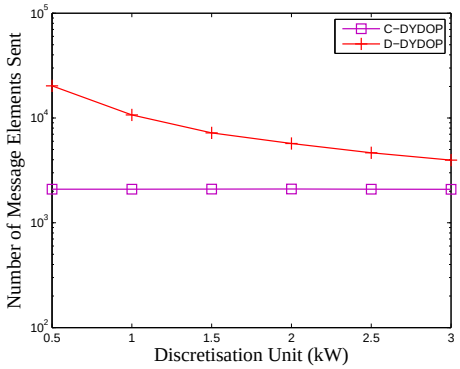
[10]This is an arbitrary number chosen so that the majority of the network contains continuous generators.



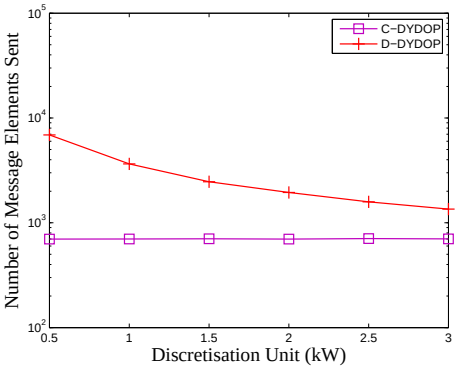
(a) Indian electricity distribution network.



(b) UK electricity distribution network.



(c) Indian electricity distribution network.



(d) UK electricity distribution network.

FIGURE 5.3: Experiment 1 tests the effect of ω for CPLEX, C–DYDOP, and D–DYDOP. Using both electricity distribution network topologies in Figure 4.2, the number of additional nodes at each substation was fixed at 8 and arranged as a random acyclic network with a maximum branching factor of 2. The discretisation unit was varied from 0.5kW to 3.0kW in 0.5kW steps, each with 50 iterations. 5.3(a) and 5.3(b) show how the discretisation unit ω affects computation time for CPLEX, C–DYDOP, and D–DYDOP on the Indian and UK electricity distribution networks respectively. 5.3(c) and 5.3(d) show how the discretisation unit ω affects the total number of message elements sent for C–DYDOP and D–DYDOP on the Indian and UK electricity distribution networks respectively. We use a logarithmic scale for the y-axis in all four plots.

of additional nodes at each substation was fixed at 8 and arranged as a random acyclic network with a maximum branching factor of 2.[11] Thus, the total number of nodes in the whole network was 596 and 203 for the Indian and UK electricity distribution networks respectively. The discretisation unit ω, used to discretise the electricity distribution network constraints, was varied from 0.5kW to 3.0kW in 0.5kW steps, each with 50 iterations.[12] During each iteration, the nodes and distribution cables were initialised as in Section 5.4.1.

[11] We choose a branching factor of 2 so that D–DYDOP can calculate a solution within a reasonable time frame.

[12] We found 50 iterations to be an adequate amount since further iterations did not improve the statistical significance of the results.

Figure 5.3 shows four plots of the results from the first experiment (error bars showing the standard error across the 50 iterations omitted due to being negligible). We use a logarithmic scale for the y-axis in all four plots. Figures 5.3(a) and 5.3(b) show how the computation time is affected by the discretisation unit ω for CPLEX, C–DYDOP, and D–DYDOP on the Indian and UK electricity distribution networks respectively. For CPLEX and C–DYDOP, regardless of the discretisation unit, the computation times remain constant. This is because both algorithms do not require the electricity distribution network constraints to be turned into a discrete problem before they can calculate a solution. However, for D–DYDOP the time complexity is exponential in ω.

Thus, it can be seen that even for small ω, the computation time for D–DYDOP suffers greatly (as discussed in Section 4.4.2). As the discretisation unit is increased, D–DYDOP has to calculate fewer state utilities, resulting in decreased computation time. For example, D–DYDOP computes a solution faster than C–DYDOP for a discretisation unit greater than 1.2 when applied to the Indian electricity disitribution network (Figure 5.3(a)). C–DYDOP must solve a small optimisation problem for each node and merge piecewise linear functions. This overhead means that in certain settings, a discrete algorithm may be faster to use (see Section 3.4 for a discussion of the settings when discrete algorithms can be used). In comparison, for the UK electricity distribution network in Figure 5.3(b), D–DYDOP is able to compute a solution faster than C–DYDOP for a discretisation unit greater than 1.3.[13] Moreover, C–DYDOP calculates a solution three times faster than it does on the Indian electricity distribution network. This difference between Figures 5.3(a) and 5.3(b) is because the section of UK electricity distribution network is smaller, and contains a number of nodes with higher branching factors. Both C–DYDOP and D–DYDOP are affected less by the branching factor of the network (compared with max-sum, which is affected much more by the branching factor of the network, see Section 4.4.2) as opposed to the number of nodes in the network.

Figures 5.3(c) and 5.3(d) show how the total number of message elements sent (i.e., the sum of the total message sizes) is affected by the discretisation unit ω for C–DYDOP and D–DYDOP. Note that CPLEX cannot be compared to the other algorithms using the discretisation unit because it does not use message passing to calculate a solution. As

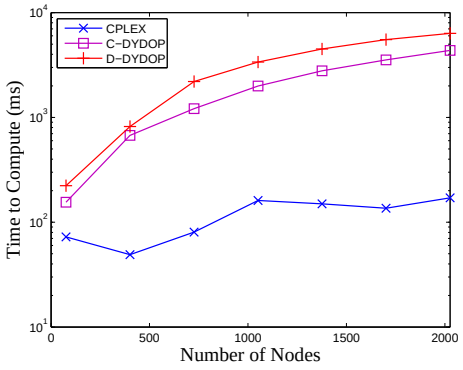
[13] In terms of accuracy of the resulting solution calculated by a discrete algorithm, when a small discretisation unit is used, D–DYDOP was able to calculate an optimal solution to the discretised electricity distribution network that was very close to the optimal solution calculated by CPLEX and C–DYDOP. For instance, when $\omega = 0.5$kW for the Indian and UK electricity distribution network topologies, D–DYDOP on average calculated a solution within 0.1% of the optimal solution (in terms of CO_2 emissions), with an average RMSE error of 0.07kW for each generator power output. However, as the discretisation unit is increased, the accuracy of the resulting solution calculated by D–DYDOP decreases. For instance, when $\omega = 3$kW for the Indian electricity distribution network, D–DYDOP on average calculated a solution which had CO_2 emissions that were 47% higher than the optimal solution, with an average RMSE error of 2.4kW for each generator power output. Similarly, when $\omega = 3$kW for the UK electricity distribution network, D–DYDOP on average calculated a solution which had CO_2 emissions that were 27% higher than the optimal solution, with an average RMSE error of 2.4kW for each generator power output. Thus, even though D–DYDOP can calculate a solution faster than C–DYDOP for certain settings, the accuracy of the resulting solution must be taken into consideration.

with Figures 5.3(a) and 5.3(b), the total message size of C–DYDOP is not affected by the discretisation unit. However, the total number of message elements D–DYDOP sends grows exponentially with regard to ω. For small ω, D–DYDOP sends a large number of message elements. As explained previously in this section, this is because D–DYDOP must iterate through a large number of states in order to calculate the utility for each resultant flow along a distribution cable, and consequently sends more message elements. It can be seen that even for a very large discretisation unit, D–DYDOP sends twice as many message elements compared to C–DYDOP. Thus, even though C–DYDOP sends fewer message elements, there is clearly some overhead with regard to computation from merging piecewise linear functions and solving an optimisation problem at each node. However, this overhead is small and remains constant. Having presented the first experiment, the following section details the second experiment.

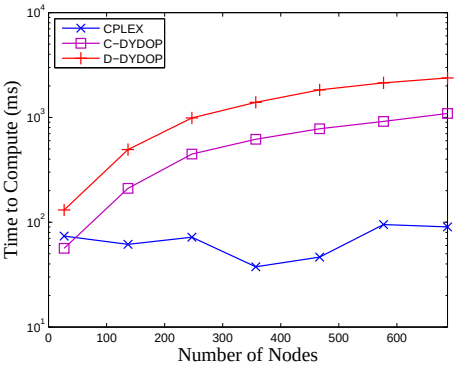
5.4.3 Experiment 2 : Impact of Varying Network Size

Experiment 2 was set up to demonstrate how the size of the network affects CPLEX, C–DYDOP, and D–DYDOP. Using both the Indian and UK electricity distribution networks, the number of additional nodes at each substation was varied from 0 to 30 in steps of 5, each with 50 iterations. At 30 additional nodes per substation, the total number of nodes in the network was 2026 and 687 for the Indian and UK distribution networks respectively. During each iteration, the nodes and distribution cables are initialised as in Section 5.4.1. Figure 5.4 shows four plots of the results from the second experiment (error bars showing the standard error across the 50 iterations omitted due to being negligible). We use a logarithmic scale for the y-axis in all four plots.

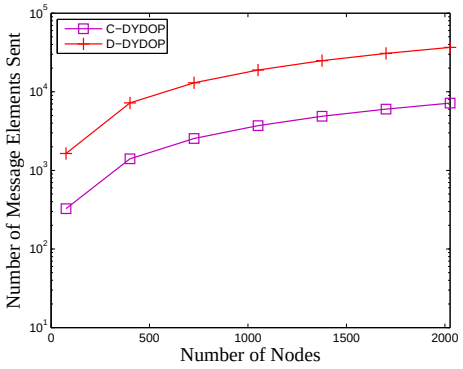
Figure 5.4(a) and 5.4(b) show how the computation time is affected by the number of nodes at each substation (and consequently the total number of nodes in the network) for CPLEX, C–DYDOP, and D–DYDOP, on the Indian and UK electricity distribution networks respectively. The time complexity of each algorithm is linear in the total number of nodes in the network. CPLEX has an almost constant computation time of 100ms on average (for very large networks, the effects of the network size on CPLEX would obviously be more apparent, but still linear). When comparing our two novel algorithms, C–DYDOP is faster at computing a solution for a given network size as apposed to D–DYDOP. For 2026 nodes in the Indian electricity distribution network, Figure 5.4(a), there is a reduction of computation time by a factor of 1.5 for C–DYDOP compared with D–DYDOP. For 687 nodes in the UK electricity distribution network, Figure 5.4(b), there is a reduction of computation time by a factor of 2 for C–DYDOP compared with D–DYDOP. This highlights the computational efficiency that C–DYDOP has over D–DYDOP, which is due to being able to summarise a range of power flows by a single function, instead of using discrete values.



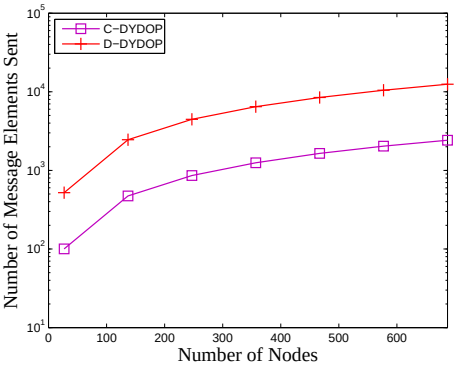
(a) Indian electricity distribution network.



(b) UK electricity distribution network.



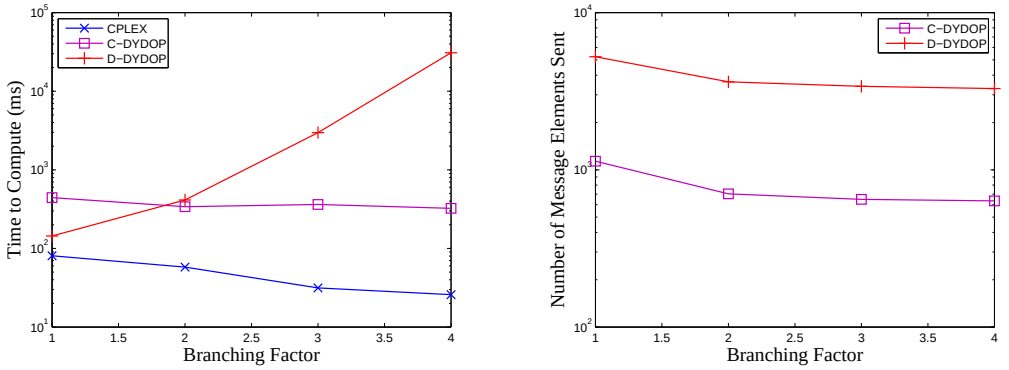
(c) Indian electricity distribution network.



(d) UK electricity distribution network.

FIGURE 5.4: Experiment 2 tests how the size of the network affects CPLEX, C–DYDOP, and D–DYDOP. Using both electricity distribution network topologies in Figure 4.2, the number of additional nodes at each substation was varied from 0 to 30 in steps of 5, each with 50 iterations. 5.4(a) and 5.4(b) show how the number of nodes in the network affects the computation time for CPLEX, C–DYDOP, and D–DYDOP on the Indian and UK electricity distribution networks respectively. 5.4(c) and 5.4(d) show how the number of nodes in the network affects the total number of message elements sent for C–DYDOP and D–DYDOP on the Indian and UK electricity distribution networks respectively. We use a logarithmic scale for the y-axis in all four plots.

Figures 5.4(c) and 5.4(d) show how the total number of message elements sent is affected by the number of nodes at each substation for C–DYDOP and D–DYDOP on the Indian and UK electricity distribution networks respectively. The total number of message elements sent for C–DYDOP and D–DYDOP grows linearly with regard to the total number of nodes in the network. C–DYDOP sends the smallest total number of message elements followed by D–DYDOP. Having presented the second experiment, the following section details the final experiment.



(a) (b)

FIGURE 5.5: Experiment 3 tests how the branching factor of the electricity distribution network affects CPLEX, C–DYDOP, and D–DYDOP. Using random acyclic electricity distribution network topologies with 200 nodes, the branching factor of the network was varied from 1 to 4 with 50 iterations for each. 5.5(a) shows how the branching factor of the network affects the computation time for CPLEX, C–DYDOP, and D–DYDOP. 5.5(b) shows how the branching factor of the network affects the total number of message elements sent for C–DYDOP, and D–DYDOP. We use a logarithmic scale for the y-axis in both plots.

5.4.4 Experiment 3 : Impact of Varying Branching Factor

Finally, Experiment 3 was set up to demonstrate how the branching factor of the network affects CPLEX, C–DYDOP, and D–DYDOP. Random acyclic electricity distribution network topologies were used to test this aspect of the algorithms since a variation of the branching factor was required. The number of nodes in the network was fixed at 200, and the branching factor of the network was varied from 1 to 4 with 50 iterations for each. During each iteration, a random acyclic electricity distribution network was generated with the nodes and distribution cables initialised as in Section 5.4.1. Figure 5.5 shows two plots of the results from the third experiment (error bars showing the standard error across the 50 iterations omitted due to being negligible). We use a logarithmic scale for the y-axis in both plots. Figure 5.5(a) shows how the computation time is affected by the branching factor of the network for CPLEX, C–DYDOP, and D–DYDOP. CPLEX and C–DYDOP have constant computation time with regard to the branching factor of the network. As discussed in Section 4.4.4, CPLEX is not affected by the branching factor because it does not use the structure of the network in order to solve the optimal dispatch problem. Similarly, C–DYDOP is not affected because each node chooses the best child messages to use given the constraints of its parent's distribution cable. Adding more children means that there are more message elements to choose from, but does not necessarily mean that every message element is used. Therefore, the additional children nodes have a negligible impact on the computation required. On the contrary, D–DYDOP has an exponential time complexity with regard to the branching factor of the network. This is because D–DYDOP must iterate through

every possible combination of generator power outputs and message elements in order to calculate a utility for each feasible power flow along a distribution cable. As the branching factor increases, the number of possible combinations grows exponentially, shown in Proposition 4.4.

Figure 5.5(b) shows how the total number of message elements sent is affected by the branching factor of the network for C–DYDOP and D–DYDOP. The total number of message elements sent for C–DYDOP and D–DYDOP decreases exponentially in the branching factor of the network. This is because as the number of children for each node is increased, the height of the network (i.e., from leaf node to root node) decreases meaning that fewer messages (and consequently fewer message elements) must be sent in order to calculate a solution (see Sections 4.1.1 and 5.1.1 for how D–DYDOP and C–DYDOP construct message elements respectively). The reasons why C–DYDOP sends fewer message elements compared with D–DYDOP, have already been explained in Section 5.4.2. The following section discusses the results from all three experiments.

5.4.5 Discussion

Our results show that both D–DYDOP and C–DYDOP significantly outperform a naïve implementation of max-sum, for the optimal dispatch problem, in terms of total number of message elements sent and computation time, by pruning the search space efficiently. Moreover, it is clear that C–DYDOP is a significant improvement compared to D–DYDOP and max-sum.[14] Firstly, the computation time of C–DYDOP is not affected by the discretisation of the network, whereas the computational complexity of both D–DYDOP and max-sum is exponential in the discretisation unit ω. Secondly, the computation time of C–DYDOP is less affected by the total number of nodes in the network compared with D–DYDOP or max-sum. Thirdly, the number of message elements that C–DYDOP must send is much lower than D–DYDOP and max-sum regardless of discretisation unit, total size of the network, or branching factor. Fourthly, the computation time of C–DYDOP is not affected by the branching factor of the network, whereas the time complexity for both D–DYDOP and max-sum is exponential with regard to the branching factor. Finally, we have shown that C–DYDOP can be applied to any real electricity distribution network problem as it does not require the electricity distribution network constraints to be discretised first (this is discussed in Section 3.4 as well as the justifications for when discrete algorithms can be used). Thus, C–DYDOP can flexibly coordinate generators in acyclic networks and can readily be applied to a wider range of network problems, compared to D–DYDOP and discrete max-sum. The following section concludes this chapter.

[14]Since we showed that D–DYDOP outperformed max-sum in Section 4.4, and we showed in Section 5.4 that C–DYDOP outperforms D–DYDOP, we can infer that C–DYDOP also outperforms max-sum.

5.5 Conclusions

In this chapter, we have shown how D–DYDOP can be adapted to consider continuous generator power outputs and perform faster than the discrete version. Thus, we believe that it can be readily applied to a wider variety of network problems compared with D–DYDOP. This is due to the decreased overhead when constructing continuous *PowerCost* messages, since there are fewer *flowCO* elements to calculate, and because C–DYDOP does not suffer from discretising the search space or from the branching factor of the network. C–DYDOP is able to handle arbitrary generator and power flow values without suffering from computational overheads. Thus, it is more readily applicable to real electricity networks.

As discussed in Section 3.4, in order for a solution to be generated by D–DYDOP or max-sum, the electricity distribution network constraints must be discretised. While the process can be completed in a distributed and decentralised way, this is obviously a disadvantage when compared with C–DYDOP, which does not require the electricity distribution network constraints to be discretised. A further disadvantage is that unless the appropriate discretisation unit is used (which is often very small resulting in an increased amount of computation) the solution produced is not always guaranteed to be applicable to the real distribution network.

However, there are a number of advantages for using D–DYDOP over C–DYDOP. Firstly, for low branching networks, and when the appropriate discretisation unit is large, D–DYDOP can outperform C–DYDOP in terms of computation time, as we have shown in Section 5.4.2. This is because C–DYDOP must merge piecewise linear functions and solve many small optimisation problems for each node. This creates an overhead that in certain settings is more computationally expensive than it takes D–DYDOP to exhaustively search every possible combination of power outputs for each generator. Secondly, if the electricity distribution network contains only discrete generators, then C–DYDOP is not appropriate for finding a solution and D–DYDOP must be used instead.

The following chapter provides a detailed summary of this thesis, as well as points of departure for future work in the area of optimal dispatch.

Chapter 6

Conclusions and Future Work

This chapter gives a detailed summary of the work presented in this thesis and introduces the future work in order to address all of the requirements in Section 1.1.

6.1 Summary and Conclusions

This thesis focused on decentralised and distributed message passing algorithms, to coordinate electricity generator power outputs, for a future smart grid. In particular, we investigated how an increased amount of generation can be incorporated into distribution networks without the need to install additional infrastructure by using ANM. This was motivated against a background where numerous global governments have agreed to reduce CO_2 emissions, as discussed in Chapters 1 and 2, and the increase in electricity generators embedded in electricity distribution networks as a result. In Chapter 3 we provided a new formalism of the optimal dispatch problem as a DCOP. We showed how this DCOP can be decomposed as a factor graph and solved using algorithms based on the GDL family, such as max-sum. We went on to show that max-sum applied naïvely in this setting performs a large number of redundant computations. Therefore, to address this issue, in Chapter 4 we presented a novel message passing algorithm, called D–DYDOP, which outperforms max-sum by using techniques based on local consistency to prune much of the search space.

The computational complexity of D–DYDOP, with regard to message size and number of states it must process, was analytically calculated to be $O(|\mathbf{V}|)$ and $O(M^{|chi(v_i)|})$ respectively. In order to demonstrate the computational efficiency of D–DYDOP, we empirically evaluated it against both an optimal centralised approach, based on the MIP solver CPLEX, and max-sum, on two real electricity distribution network topologies using three experiments (see Section 4.4). Experiment 1 varied the discretisation unit, Experiment 2 varied the size of the network, and Experiment 3 varied the branching factor of the network.

When varying the discretisation unit (Experiment 1), the time complexity for both D–DYDOP and max-sum was exponential in ω. Due to the redundant computation that a naïve implementation of max-sum performs, D–DYDOP outperformed max-sum in terms of computation time and number of message elements sent. When varying network size (Experiment 2), the time complexity for both D–DYDOP and max-sum, was linear in the total number of nodes in the network, with max-sum exhibiting the worst computational efficiency. In the worst case, for 2026 nodes in the Indian electricity distribution network, there was a reduction of computation time by a factor of 10 for D–DYDOP compared with max-sum. For 687 nodes in the UK electricity distribution network there was a reduction of computation time by a factor of 65 for D–DYDOP compared with max-sum. Finally, when varying the branching factor (Experiment 3) the time complexity for both D–DYDOP and max-sum was exponential in the branching factor of the network. Again, max-sum performed significantly worse in terms of computation time and number of message elements sent. For all three experiments, the computation time of CPLEX was highly linear and almost constant.

As can be seen from the results in Section 4.4, the computation time of both D–DYDOP and max-sum suffers greatly from the size of the discretisation unit ω, which is used to discretise the electricity distribution network constraints, and the branching factor of the network. Thus, to address these issues we presented an extension to D–DYDOP in Chapter 5, called C–DYDOP, which uses continuous variables for the generator power outputs and the distribution cable flows. C–DYDOP uses continuous piecewise linear functions to represent continuous *PowerCost* messages enabling it to greatly reduce the computation time necessary to calculate a solution to the optimal dispatch problem. In order to demonstrate the computational efficiencies of C–DYDOP, we empirically evaluated D–DYDOP, C–DYDOP, and CPLEX using the same two electricity distribution network topologies from Chapter 4 and the same three experiments (see Section 5.4):

When varying the discretisation unit (Experiment 1), for the Indian electricity distribution network, D–DYDOP computed a solution faster than C–DYDOP for a discretisation unit greater than 1.2. In comparison, for the UK electricity distribution network D–DYDOP was able to compute a solution faster than C–DYDOP for a discretisation unit greater than 1.3. However, in order for a discrete algorithm to produce accurate solutions, the discretisation unit must be small (see Section 3.4 for a discussion). Moreover, for small discretisation units, the computation time of D–DYDOP was much higher than C–DYDOP. With regard to message elements sent, in the worst case, D–DYDOP sent twice as many message elements compared to C–DYDOP. When varying the network size (Experiment 2), C–DYDOP was faster at computing a solution for a given network size as apposed to D–DYDOP. For 2026 nodes in the Indian electricity distribution network there was a reduction of computation time by a factor of 1.5 for C–DYDOP compared with D–DYDOP. For 687 nodes in the UK electricity distribution network there was a reduction of computation time by a factor of 2 for C–DYDOP compared with

D–DYDOP. Finally, when varying the branching factor (Experiment 3), C–DYDOP had constant computation time with regard to the branching factor of the network, whereas D–DYDOP had an exponential time complexity with regard to the branching factor of the network. Again, for all three experiments, the computation time of CPLEX was highly linear and almost constant.

The algorithms presented in Chapters 3, 4, and 5 have achieved a number of the requirements from Section 1.1. Requirement II has been achieved because the algorithms coordinate power within the electricity distribution network such that CO_2 emissions are minimised. Our algorithms scale up to large networks containing thousands of nodes achieving Requirement III. Requirement IV has been partially met since our algorithms are able to handle generations with different output types. However, the intermittency of generators (i.e., such as wind turbines) should be addressed in future work.

Requirement I has been met since the algorithms coordinate autonomously with minimal human interaction. Furthermore, by using agent-based message passing algorithms with a DCOP framework, Requirement V has been satisfied. Finally, Requirements VI and VII have been partially met. However, in terms of graceful degradation, we leave for future work the non-trivial extension for D–DYDOP and C–DYDOP to handle communication networks with faulty channels (such that messages could fail to be sent between nodes).

In order to apply the algorithms presented in Chapters 3, 4, and 5 to real world systems, a communication network that is constructed on top of existing electricity distribution networks would be required. This communication network would not require a high bandwidth, but the speed and accuracy (in terms of error free message sending) would have to be high.[1] Each node (consisting of a combination of generators, loads, and a substation) would have to contain a computation device and be able to monitor and control each connected generator and load with limited or no human interaction. Each node would communicate with its connected neighbouring nodes periodically to ensure that the network is configured optimally with regard to certain costs.[2] As discussed in Section 2.3.2, decentralising and distributing control in electricity networks has a number of advantages. For instance a distributed system that does not rely on a central authority for control is much more robust because there is no single point of failure.[3] In

[1] Coupled with this high accuracy of message sending would be the need to handle potentially error prone messages in our algorithms, see the following section on future work.

[2] Practically, depending on where this technology is deployed, multiple objective functions may be necessary. For instance, in an electricity distribution network of generators owned by different entities, where the power generated creates revenue for the generator owner, trying to reduce CO_2 emissions would require some owners to reduce their output, and hence reduce their revenue. Therefore, in this setting there must be a balance between reducing CO_2 emissions and revenue earned, as well as providing incentives for owners to allow the power output of their generators to be decreased if the CO_2 emissions of the electricity distribution network needs to be reduced.

[3] For a decentralised and distributed system, if the computation device was to fail at a node, the required computation could be outsourced to neighbouring nodes. However, if the computation device of a centralised system was to fail, the generators would be unable to calculate the optimum output with regard to their neighbours, which could lead to overloaded distribution cables.

terms of contributions to the DCOP research area, this thesis proves that DCOPs can be applied to the coordination of generators in electricity distribution networks. Our approaches could be generalised to other settings which exhibit similar global and local constraints.[4]

Therefore, with regard to decentralising and distributing electricity network control, our algorithms set the benchmark for the deployment of agent-based coordination algorithms to solve the optimal dispatch problem in electricity distribution networks. Moreover, our work highlights the challenge of constructing a DCOP from an electricity distribution network (i.e., ensuring that an acyclic electricity distribution network remains acyclic when transformed into a DCOP). This is challenging because of the coupled nature of electricity networks. Therefore, future DCOP decompositions must ensure that they address this challenge directly.

In terms of implementing each algorithm, we ran into a number of issues with regard to rounding error when calculating a solution using max-sum, D–DYDOP, and C–DYDOP. Rounding error was particularly an issue when merging messages because a small rounding error for a particular variable state can mean the difference between choosing a feasible configuration for the generators and choosing an infeasible configuration. The problem was that we had to round merged message values (to at least 10 decimal places) in order to evaluate them against each other; since computers introduce small rounding errors when storing floating point numbers. As these rounding errors were propagated around the network, eventually it would cause a wrong variable state to be chosen. In order to solve this issue, we used whole number values for the electricity distribution network model. It should be noted that this problem is not due to the algorithms, but is a limitation of how floating point numbers are stored and used in computers. Moreover, this phenomenon happened rarely and only for large electricity distribution networks (typically over 1000 nodes). The following section provides points of departure for future work.

6.2 Future Work

In terms of future work, we would like to consider a number of possible extensions. Firstly, each algorithm presented handles a particular instance of an electricity distribution network (i.e., a one-shot optimisation problem). Therefore, it is a non-trivial extension to factor time into our model.[5] This would allow our algorithms to continuously manage a real world electricity distribution network. Moreover, time would allow

[4] For example, our algorithms could be applied to a factory supply chain setting where it is imperative that the total amount of supply must equal the total amount of demand (in order to minimise wastage for instance), and the goal is to decide the quantity of each item whilst minimising the monetary expenditure of the entire system.

[5] We would extend our model to incorporate consumption that varies over time periods, instead of the static consumption in our current model.

us to model the latency[6] and intermittency[7] of the generators. As a result, the optimal dispatch problem becomes a much more complex time dependent problem. Coordinating electricity generators in an electricity distribution network over time involves taking into account the latency of each generator, the resources available (i.e., wind and solar), and the predicted loads across the day. Possible solutions may involve running the coordination algorithm every thirty minutes, giving each generator a range of power that it can output between and still be optimal (so that small fluctuations of power consumption and production can be satisfied). If the consumption changes significantly (as determined by a variation in the frequency), the coordination algorithm can be run again to calculate new optimal power output ranges for each generator.

Secondly, in this thesis we have considered acyclic electricity distribution networks (including electricity distribution networks that have been configured into acyclic topologies). However, electricity distribution networks could contain cycles during operation. Cyclic electricity networks present a number of additional challenges because of the resulting tightly coupled optimisation problem. This makes it non-trivial to: (i) split into independent subproblems, (ii) know exactly how much power will be travelling along each distribution cable, (iii) calculate the utility of a certain amount of power travelling along a certain distribution cable.

Initially, we decomposed a cyclic electricity distribution network into a DCOP and applied max-sum; since there exists extensive empirical evidence of its effectiveness on cyclic graphs (Aji et al., 1998; Weiss, 2000; Farinelli et al., 2008; Vinyals et al., 2010; Winsper and Chli, 2012). However, we found that due to the cycles in the electricity distribution network, each message sent from function to distribution cable flow variable, or generator power output variable, contained a number of message elements with the same optimal utility value. In order to resolve the deadlock between variable states required an additional value propagation phase which did not always result in the correct answer. Therefore, using the techniques that we apply to acyclic electricity distribution networks (either with max-sum, D–DYDOP, or C–DYDOP) cannot be applied to cyclic networks in the same way. In order to use distributed techniques with cyclic electricity networks, the duality of optimisation problems using Lagrangian techniques and the APP (Cohen, 1980) must be exploited. The advantage of both these techniques is that they allow the decomposition of a coupled problem into subproblems which can be solved independently, suggesting a distributed agent-based approach would be applicable (Kim and Baldick, 1997; Bakirtzis and Biskas, 2003; Granada et al., 2008; Kraning et al., 2013). Moreover, additional parameters and constraints must be considered when calculating power flow within a cyclic network (such as voltage phase angle at nodes and susceptance of distribution cables) which we excluded from our original electricity network model in Section 3.1.

[6]The time it takes a generator to change its power output.

[7]For example, wind turbines or photovoltaic panels are intermittent because they depend on intermittent resources.

Finally, another non-trivial extension would be to implement and thoroughly test robust versions of D–DYDOP and C–DYDOP to incorporate message failures and situations when whole nodes in the electricity distribution network could fail. To achieve this would involve a number of non-trivial additions. Firstly, a message passing framework must be constructed which delivers the messages to each agent.[8] Secondly, coupled with this must be a model of each message channel along with failure rates. Finally, robustness to message failure and corruption would need to be built into both D–DYDOP and C–DYDOP in order for each algorithm to handle the uncertain nature of actual communication networks (as discussed in the previous section with regard to an accurate communication network). The intricate details of communication networks, whilst interesting, is beyond the scope of this thesis as it would require considerable changes to our model and algorithms.

[8] Currently messages are received instantly, there is no latency associated with a real message delivery.

References

S M Aji, G B Horn, and R J McEliece. Iterative decoding on graphs with a single cycle. In *Proceedings of the 1998 IEEE International Symposium on Information Theory*, page 276, Cambridge, Massachusetts, USA, 1998.

S M Aji and R J McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

A Alarcon-Rodriguez, G Ault, and J McDonald. Planning the development of highly distributed power systems. In *Proceedings of the Nineteenth International conference on Electricity Distribution–(CIRED)*, pages 1–4, 2006.

A G Bakirtzis and P N Biskas. A decentralized solution to the DC-OPF of interconnected power systems. *IEEE Transactions on Power Systems*, 18(3):1007–1013, 2003.

U Bertele and F Brioschi. *Nonserial dynamic programming.* Academic Press, Inc., 1972.

E Bowring, M Tambe, and M Yokoo. Multiply-constrained distributed constraint optimization. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1413–1420, Hakodate, Japan, 2006.

A S Chuang and C W Gellings. Demand-side Integration in a Restructured Electric Power Industry. In *CIGRE session*, pages 96–105, 2008.

G Cohen. Auxiliary problem principle and decomposition of optimization problems. *Optimization Theory and Applications*, 32(3):277–305, 1980.

E M Davidson, M J Dolan, S D J McArthur, and G W Ault. The use of constraint programming for the autonomous management of power flows. In *Proceedings of the Fifteenth International Conference on Intelligent System Applications to Power Systems*, pages 1–7, Curitiba, Brazil, 2009.

M Davis and H Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

R Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, pages 211–219, Portland, Oregon, USA, 1996.

R Dechter. *Constraint processing.* Morgan Kaufmann Publishers Inc, 2003.

R Dechter and J Pearl. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34(1):1–38, 1987.

Department for Business Enterprise and Regulatory Reform. Active network management (ANM) technology. Technical report, 2008.

European Commission. European smartGrids technology platform. *Available at: http://www.smartgrids.eu/*, 2006.

A Farinelli, A Rogers, and N Jennings. Bounded approximate decentralised coordination using the max-sum algorithm. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence*, pages 46–59, Pasadena, California, USA, 2009.

A Farinelli, A Rogers, A Petcu, and N R Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 639–646, Estoril, Portugal, 2008.

A Farinelli, M Vinyals, A Rogers, and N R Jennings. Distributed Constraint Handling and Optimization. In G Weiss, editor, *Multiagent Systems*, chapter 12. MIT Press, MA, USA, 2nd edition, 2013.

L H Fink, H G Kwatny, and J P McDonald. Economic dispatch of generation via valve-point loading. *IEEE Transactions on Power Apparatus and Systems*, PAS-88 (6):805–811, 1969.

T Gönen. *Electric power distribution system engineering.* McGraw-Hill New York, 2nd edition, 2007.

M E Granada, M J Rider, J R S Mantovani, and M Shahidehpour. Multi-areas optimal reactive power flow. In *Proceedings of the Transmission and Distribution Conference and Exposition*, pages 1–6, Bogota, Colombia, 2008.

L L Grigsby. *Electric power generation, transmission and distribution.* CRC Press LLC, 2012.

Intergovernmental Panel on Climate Change. Climate Change 2013: The Physical Science Basis. In *Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, pages 1–2216, 2013.

International Energy Agency. IEA Member Countries. *Available at: http://www.iea.org/countries/*, 2011.

R Kaye and F Wu. Analysis of linearized decoupled power flow approximations for steady-state security assessment. *IEEE Transactions on Circuits and Systems*, 31(7): 623–636, 1984.

B H Kim and R Baldick. Coarse-grained distributed optimal power flow. *IEEE Transactions on Power Systems*, 12(2):932–939, 1997.

M Kraning, E Chu, J Lavaei, and S Boyd. Dynamic network energy management via proximal message passing. *Optimization*, 1(2):1–54, 2013.

F R Kschischang, B J Frey, and H A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

A Kumar, B Faltings, and A Petcu. Distributed constraint optimization with structured resource constraints. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 923–930, Budapest, Hungary, 2009.

D J C MacKay. *Sustainable Energy - Without the Hot Air*. UIT Cambridge, 2008.

R Mailler and V Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 438–445, 2004.

R Mailler and V Lesser. Asynchronous partial overlay: A new algorithm for solving distributed constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 25(1):529–576, 2006.

S Miller, S D Ramchurn, and A Rogers. Optimal Decentralised Dispatch of Embedded Generation in the Smart Grid. In *Proceedings of the Eleventh International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 281–288, Valencia, Spain, 2012.

S Miller, S D Ramchurn, and A Rogers. Agent-based Message Passing Techniques for Controlling Electricity Generators in Distribution Networks. *Artificial Intelligence Journal (In Review)*, 2013.

P J Modi, W M Shen, M Tambe, and M Yokoo. ADOPT: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1):149–180, 2005.

National Grid. National Electricity Transmission System Seven Year Statement. Technical report, 2011.

T J Overbye, X Cheng, and Y Sun. A comparison of the AC and DC power flow models for LMP calculations. In *Proceedings of the Thirty-seventh Annual Hawaii International Conference on System Sciences*, pages 1–9, Hawaii, USA, 2004.

A Petcu and B Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, volume 5, pages 266–271, Edinburgh, Scotland, UK, 2005.

G Platt. The Decentralised Control of Electricity Networks- Intelligent and Self-Healing Systems. In *Proceedings of Grid Interop Forum*, 2007.

S Rahman and M Pipattanasomporn. Modeling the simulation of a DG-integrated intelligent microgrid. *SERDP Project SI-1650*, 2010.

S Rahman, M Pipattanasomporn, and Y Teklu. Intelligent distributed autonomous power systems (IDAPS). In *Proceedings of the IEEE Power & Energy Society General Meeting*, pages 1–8, Tampa, Florida, USA, 2007.

S D Ramchurn, P Vytelingum, A Rogers, and N R Jennings. Putting the "smarts" into the smart grid: a grand challenge for artificial intelligence. *Communications of the ACM*, 55(4):86–97, 2012.

D Roberts. 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*, 2004.

A Rogers, A Farinelli, R Stranders, and N R Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2): 730–759, 2011.

D W Ross and S Kim. Dynamic economic dispatch of generation. *IEEE Transactions on Power Apparatus and Systems*, PAS-99(6):2060–2068, 1980.

F Rossi, P Van Beek, and T Walsh. *Handbook of constraint programming.* 2006.

J M Solanki, S Khushalani, and N N Schulz. A multi-agent solution to distribution systems restoration. *IEEE Transactions on Power Systems*, 22(3):1026–1034, 2007.

R Stranders, A Farinelli, A Rogers, and N R Jennings. Decentralised coordination of continuously valued control parameters using the max-sum algorithm. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 601–608, Budapest, Hungary, 2009.

J Sun and L S Tesfatsion. DC optimal power flow formulation and solution using QuadProgJ. Staff general research papers, Iowa State University, Department of Economics, Tampa, Florida, USA, March 2007. URL: http://ideas.repec.org/p/isu/genres/12558.html.

P Taylor, T Xu, S McArthur, G Ault, E Davidson, M Dolan, C Yuen, M Larsson, D Botting, D Roberts, and Others. Integrating voltage control and power flow management in AuRA-NMS. In *Proceedings of the Conference on SmartGrids for Distribution*, pages 1–4, Frankfurt, Germany, 2008.

P C Taylor, T Xu, N S Wade, M Prodanovic, R Silversides, T Green, E M Davidson, and S McArthur. Distributed voltage control in AuRA-NMS. In *Proceedings of the*

IEEE Power & Energy Society General Meeting, pages 1–7, Minneapolis, Minnesota, USA, 2010.

UK Department of Energy and Climate Change. *Smarter Grids: The Opportunity*. The Stationary Office, 2009a.

UK Department of Energy and Climate Change. *The UK Low Carbon Transition Plan: National strategy for climate and energy*. The Stationary Office, 2009b.

US Department of Energy. "Grid 2030" - A national vision for electricity's second 100 years. Technical report, 2003.

M Vinyals, J Cercuides, A Farinelli, and J A Rodriguez-Aguilar. Worst-case bounds on the quality of max-product fixed-points. *Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems*, pages 2325–2333, 2010.

B M Weedy and B J Cory. *Electric Power Systems*. John Wiley & Sons, 4th edition, 2004.

Y Weiss. Correctness of local probability propagation in graphical models with loops. *Neural computation*, 12(1):1–41, 2000.

M Winsper and M Chli. Decentralised supply chain formation using max-sum loopy belief propagation. *Computational Intelligence*, 29(2):281–309, 2012.

A J Wood and B F Wollenberg. Optimal Power Flow. In *Power Generation, Operation, and Control*, chapter 13. John Wiley & Sons, 2nd edition, 1995.

M Wooldridge and N R Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Yu-Chi Wu, A S Debs, and R E Marsten. A direct nonlinear predictor-corrector primal-dual interior point algorithm for optimal power flows. *IEEE Transactions on Power Systems*, 9(2):876–883, 1994.

M Yokoo and E H Durfee. Distributed constraint optimization as a formal model of partially adversarial cooperation. *CSE-TR-101-91*, pages 1–12, 1991.