Algorithm Selection for Power Flow Management

James Edward King

School of Electrical and Electronic Engineering

Newcastle University

This dissertation is submitted for the degree of

Doctor of Philosophy

June 2016

For my family.

Declaration

I hereby declare that this thesis is a record of work undertaken by myself, that it has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged.

Parts of this work, as indicated in the text, have been the subject of previous publications:

1. J. E. King, S. C. E. Jupe, and P. C. Taylor, "Autonomic control algorithm selection in decentralised power systems: a voltage control case study," in *22nd International Conference and Exhibition on Electricity Distribution (CIRED)*, Stockholm, 2013.

2. J. E. King, S. C. E. Jupe, and P. C. Taylor, "Performance evaluation of control algorithms for active distribution networks - the potential for algorithm selection," in *CIGRÉ Session*, Paris, 2014.

3. J. King, S. Jupe, and P. Taylor, "The potential of network state-based algorithm selection to improve power flow management," in *2014 IEEE PES General Meeting*, Washington DC, 2014, pp. 1–5.

4. J. E. King, S. C. E. Jupe, and P. C. Taylor, "Network State-Based Algorithm Selection for Power Flow Management Using Machine Learning," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2657–2664, 2015.

James Edward King
June 2016

Acknowledgements

Although a PhD is a qualification granted to an individual, this work would not have been possible without the generous help of a number of people.

Firstly, my employer, WSP | Parsons Brinckerhoff, has been extraordinarily generous in fully funding my PhD and granting me time to pursue my research. Within the company, I am particularly grateful to Katherine Jackson, who provided me with the opportunity to pursue a PhD while remaining at the company, and supported my pursuit of research.

My sincere thanks go to my supervisors: Prof. Phil Taylor, my academic supervisor, Dr. Samuel Jupe, my industrial supervisor, and Dr. Haris Patsios, my academic co-supervisor. They have provided me with excellent support, wise guidance, and necessary challenge throughout my research, even through changes in organisation. It really has been a privilege to have received such excellent supervision.

My fellow researchers at Durham, Newcastle, and from elsewhere on the Autonomic Power Systems project have made the day-to-day process of research more interesting and less of a solo enterprise. In particular, my thanks go to Barbara and Ivan for their excellent company and for the time spent bouncing around research (and non-research) ideas.

During my research I have used a number of software tools, which have made the research possible and also more efficient, such as the Python programming language, MATPOWER, WEKA, LaTeX, and matplotlib. These are all free and open source, so I am grateful to all the people who have volunteered their time and expertise in developing them.

I owe an enormous amount to my parents and family. They brought me up and allowed me to develop my interest in science and engineering from an early age. They made sure I had the right education and attitude so that I could even pursue a PhD. They are my most important people and knowing their faith in me has got me through the hardest times – not *all* of them being during my PhD!

My greatest thanks go to my wife, Kirsteen, for her unwavering support throughout the most intense and demanding period of work in my life. She provided encouragement, took on all the housework, was my proof reader, and shouldered the burden of a lot of other things while I wrote up (buying a house, moving home, wedding planning, etc.). Without her love and support I would not have been able to complete this work.

Abstract

Algorithms are essential for solving many important problems, including in power systems control, where they can allow the connection of new demand and generation whilst deferring or avoiding the need for network reinforcement. However, in many problem domains no algorithm always delivers the best performance for all problems, so better performance can be achieved by using algorithm selection to select the best algorithms for *each* problem.

This work applies algorithm selection to power systems control, with power flow management using generator curtailment examined as a representative power systems control task. The first half of this work focuses on whether *potential* performance benefits are available if algorithms are selected optimally for each network state. Five power flow management algorithms are implemented, which use diverse approaches such as optimal power flow, constraint satisfaction, power flow sensitivity factors, and linear programming. Four case study power systems – an 11 kV radial distribution system, a 33 kV meshed distribution system, the IEEE 14-bus system, and the IEEE 57-bus system – are used to test the algorithms over a extensive range of network states. None of the algorithms give the most effective performance for every state, in terms of minimising either the number or energy of overloads, whilst minimising curtailment. By optimally selecting algorithms for each state there are potential performance benefits for three of the four case study systems

In the second half of this work, algorithm selection systems (selectors) are created in order to exploit and *deliver* the observed potential performance benefits of per-state algorithm selection. Existing techniques for creating algorithm selectors are adapted and extended for the power flow management application, which includes the development of a training method that allows selectors to consider two objectives simultaneously. The selectors created take measurements of network state as input and use machine learning models to make algorithm selection decisions. The models either directly predict which algorithm is likely to be the most effective, or predict the performance of each algorithm, with the algorithm with the most effective predicted performance then being selected. Both of these approaches are shown to be effective in creating algorithm selectors for power flow management that deliver statistically significant performance benefits. In some cases, the selectors are able to match the optimum performance that could be achieved by selecting between the algorithms.

Table of contents

List of figures **xvii**

List of tables **xxi**

Nomenclature **xxvii**

1 Introduction **1**

1.1 Background 1
1.2 Motivating example 2
1.3 Research objectives 4
1.4 Structure of this work 5

2 Algorithms for power systems control **9**

2.1 Control in power systems 9
2.2 Power flow management: literature survey 12
2.2.1 OPF-based methods 12
2.2.2 Sensitivity factor-based methods 13
2.2.3 Other methods 14
2.2.4 Comparison of methods 16
2.2.5 Selection of methods to be implemented 16
2.3 Power flow management: implemented algorithms 19
2.3.1 PFM-CSP 20
2.3.2 PFM-OPF 22
2.3.3 PFSF-Egal 26
2.3.4 PFSF-TMA 28
2.3.5 PFSF-LP 29
2.4 Conclusions 32

3 Control algorithm testing environment 33
3.1 Requirements for testing environment . . . 33
3.2 Overview of testing environment . . . 35
3.2.1 Main components . . . 36
3.2.2 Components within a `TestCell` . . . 38
3.3 Power system modelling . . . 40
3.3.1 Component parameters . . . 41
3.3.2 Power system component modelling capabilities . . . 41
3.3.3 Load flow . . . 46
3.4 Communications system modelling . . . 47
3.4.1 Functions . . . 47
3.4.2 Communications system objects . . . 47
3.5 Control algorithm modelling . . . 48
3.6 Running single tests . . . 49
3.7 Running multiple tests . . . 51
3.7.1 Process . . . 51
3.7.2 Failure tolerance . . . 53
3.8 Evaluation against requirements . . . 53
3.9 Conclusions . . . 58

4 Performance of power flow management algorithms 59
4.1 Methodology . . . 61
4.1.1 Case study power systems . . . 61
4.1.2 Performance evaluation . . . 64
4.1.3 Statistical analysis of performance . . . 65
4.2 Case study: 11 kV radial distribution system . . . 67
4.2.1 System description . . . 67
4.2.2 Test states . . . 67
4.2.3 Baseline performance . . . 68
4.2.4 Algorithm performance . . . 68
4.3 Case study: 33 kV meshed distribution system . . . 70
4.3.1 System description . . . 70
4.3.2 Test states . . . 72
4.3.3 Baseline performance . . . 72
4.3.4 Algorithm performance . . . 73
4.4 Case study: IEEE 14-bus system . . . 75
4.4.1 System description . . . 75

4.4.2 Test states 76
4.4.3 Baseline performance 76
4.4.4 Algorithm performance 78
4.5 Case study: IEEE 57-bus system 81
4.5.1 System description 81
4.5.2 Test states 81
4.5.3 Baseline performance 83
4.5.4 Algorithm performance 83
4.6 Cross-case study analysis 86
4.6.1 Comparison of algorithm performance 86
4.6.2 Analysis of algorithm performance: PFM-OPF 89
4.6.3 Analysis of algorithm performance: PFM-CSP 91
4.6.4 Analysis of algorithm performance: PFSF-based algorithms 93
4.6.5 Execution times 97
4.7 Conclusions 99

5 Potential performance benefits from per-state selection of algorithms 101
5.1 Method for assessing the potential performance benefit 101
5.2 Potential performance benefits for each system 103
5.2.1 11 kV radial distribution system 103
5.2.2 33 kV meshed distribution system 104
5.2.3 IEEE 14-bus system 105
5.2.4 IEEE 57-bus system 106
5.2.5 Summary for all case study systems 109
5.3 Algorithm selection frequency 112
5.4 Performance with different sets of algorithms 114
5.5 Conclusions 118

6 Previous work on algorithm selection 121
6.1 The Algorithm Selection Problem 121
6.1.1 Rice's model 121
6.1.2 Framing the problem 122
6.1.3 Application to power flow management 123
6.2 Creating a selection mapping 124
6.3 Machine learning 126
6.4 Using machine learning for algorithm selectors 127
6.4.1 Timing 127

6.4.2 Model output 128
6.4.3 Selector output 128
6.4.4 Learning 129
6.4.5 Model type 130
6.4.6 Features 130
6.5 Previous applications 131
6.6 Conclusions 131

7 Design and development of algorithm selectors for power flow management 133
7.1 Preparation of training data for direct selectors 134
7.1.1 Selectors using unweighted training examples 134
7.1.2 Selectors for a single objective using weighted training examples . . 138
7.1.3 Selectors for two objectives using tuned weights 140
7.2 Preparation of training data for EPM-based selectors 146
7.3 Machine learning models 149
7.3.1 Models used in this work 149
7.3.2 Descriptions of machine learning model categories 150
7.3.3 Model variations 154
7.4 Feature construction and selection 154
7.4.1 Feature construction 155
7.4.2 Feature selection 155
7.4.3 Features used in this work 157
7.5 Training data volume 157
7.6 Conclusions 159

8 Performance of algorithm selectors for power flow management 161
8.1 Preliminary remarks 162
8.1.1 Direct selectors 162
8.1.2 EPM-based selectors 162
8.1.3 Machine learning model types 163
8.1.4 Objectives 163
8.2 Performance overview 163
8.3 Most effective selectors 166
8.4 Relationship between objectives and performance 175
8.4.1 33 kV meshed distribution system 175
8.4.2 IEEE 14-bus system 176
8.4.3 IEEE 57-bus system 177

8.4.4 Summary 179
8.5 Effect of considering different algorithm sets 179
8.5.1 Method 179
8.5.2 Summary of results 181
8.6 Effect of different model types 182
8.6.1 Effect of model type on direct selectors 182
8.6.2 Effect of model type on EPM-based selectors 185
8.6.3 Summary 195
8.7 Effect of tuning on direct weighted selectors 195
8.8 Conclusions 197

9 Discussion 201
9.1 Implementation outline 201
9.2 Selector design choices 201
9.3 Data requirements 202
9.4 Time requirements 205
9.5 Financial assessment 207

10 Conclusions 209
10.1 Evaluation against research objectives 209
10.2 Contributions 212
10.3 Outlook and future work 213

References 217

Appendix A Data model for power system model (`SysModel`) 229

Appendix B Case study system data 243
B.1 11 kV radial distribution system 243
B.2 33 kV meshed distribution system 244
B.3 IEEE 14-bus system 247
B.4 IEEE 57-bus system 249

Appendix C Additional power flow management algorithm performance data 257

List of figures

1.1 System schematic and results for pilot study into voltage control algorithm selection . . . 3

2.1 General overview of control . . . 9
2.2 Control architectures . . . 10
2.3 Flowchart for the processes of the PFM-CSP algorithm . . . 21
2.4 Flowchart for the processes of the PFM-OPF algorithm . . . 23
2.5 Bus real power flexibility as used in the PFM-OPF algorithm . . . 24
2.6 Cost curves for generators within PFM-OPF . . . 25
2.7 Flowchart for the processes of the PFSF-Egal and PFSF-TMA algorithms . 27
2.8 Derivation of $P_l^{\max}$ within the PFSF-LP algorithm . . . 28
2.9 Flowchart for the processes of the PFSF-LP algorithm . . . 30
2.10 Relationship of real power variables used in the PFSF-LP algorithm . . . 31

3.1 Overview of the architecture of the testing environment . . . 37
3.2 Overview of the modelling components within a `TestCell` . . . 39
3.3 Examples of generator real power operation modes . . . 45
3.4 Flow chart of process for running a single test within a `TestCell` . . . 50
3.5 Sequence diagram for running tests in parallel . . . 52

4.1 Overview of methodology for testing the performance of power flow management algorithms . . . 60
4.2 Single line diagram of the 11 kV radial distribution system used in this work 67
4.3 Overview of power flow management algorithm performance when applied to the 11 kV radial distribution system . . . 69
4.4 Single line diagram of the 33 kV meshed distribution system used in this work 71
4.5 Overview of power flow management algorithm performance when applied to the 33 kV meshed distribution system . . . 74
4.6 Single line diagram of the IEEE 14-bus system as used in this work . . . 75

4.7 Frequency of simultaneous overloads for the IEEE 14-bus system 77
4.8 Overview of power flow management algorithm performance when applied to the IEEE 14-bus system 79
4.9 Single line diagram of the IEEE 57-bus system as used in this work 82
4.10 Frequency of simultaneous overloads for the IEEE 57-bus system 83
4.11 Overview of power flow management algorithm performance when applied to the IEEE 57-bus system 84
4.12 Comparison of the relative performance of every algorithm across the case study systems, for each of the three performance measures 87
4.13 Distribution of the amount of curtailment applied by the PFM-OPF algorithm for the initially overloaded states in the IEEE 57-bus system, for different numbers of overloads remaining 90
4.14 Simple power system for illustrating overload performance weaknesses of PFM-CSP algorithm 92
4.15 Percentage of initially overloaded states in the IEEE 57-bus system where the PFSF-Egal and PFSF-TMA algorithms report errors 96
4.16 Algorithm execution times for the overloaded states within each case study system 98

5.1 Potential performance gains for each system that could be achieved by optimally selecting algorithms on a per-state basis 111
5.2 Frequency of algorithms appearing as the sole algorithm in the selection sets represented by the oracles for each of the case study systems 113
5.3 Comparison of the potential performance for the 33 kV meshed distribution system with different algorithm combinations considered for selection (oracle 1) 115
5.4 Comparison of the potential performance for the IEEE 14-bus system with different algorithm combinations considered for selection (oracle 1) 116
5.5 Comparison of the potential performance for the IEEE 57-bus system with different algorithm combinations considered for selection (oracle 2) 117

6.1 Rice’s conceptual model of the Algorithm Selection Problem 121

7.1 Structure of a direct algorithm selector 134
7.2 Visualisation of the state space for the IEEE 57-bus system, showing the states where a single algorithm is most effective at minimising the number of overloads whilst also minimising the curtailment applied 135
7.3 Example of different options for labelling “don’t care” states where no algorithm (from a pair of algorithms) is uniquely the most effective 136

7.4 Effect of varying weight multiplier c_{CT} with $c_{\mathrm{OL}} = 1.0$ for a selector using a `RandomForest` model to select between the PFM-OPF and PFSF-TMA algorithms, trained and tested on states from the 33 kV meshed distribution system 142
7.5 Example of the operation of the procedure for tuning selector weight multiplier c_2 144
7.6 Example structure of an EPM-based algorithm selector 146
7.7 Amount of power flow management algorithm performance data per system for use training and testing algorithm selectors 158

8.1 Performance of the direct and EPM-based selectors for the 33 kV meshed distribution system, with respect to the number of overloads and the amount of curtailment 167
8.2 Performance of the direct and EPM-based selectors for the IEEE 14-bus system, with respect to the number of overloads and the amount of curtailment 168
8.3 Performance of the direct and EPM-based selectors for the IEEE 14-bus system, with respect to overload energy and the amount of curtailment . . . 169
8.4 Performance of the direct and EPM-based selectors for the IEEE 57-bus system, with respect to the number of overloads and the amount of curtailment 170
8.5 Performance of the direct and EPM-based selectors for the IEEE 57-bus system, with respect to the overload energy and the amount of curtailment . 171
8.6 Comparison of the proportion of effective selectors (for either overload objective) produced per objective set for the 33 kV meshed distribution system 175
8.7 Comparison of the proportion of effective selectors (for each of the overload objectives) produced per objective set for the IEEE 14-bus system 176
8.8 Comparison of the proportion of effective selectors (for each of the overload objectives) produced per objective set for the IEEE 57-bus system 178
8.9 Matrix comparing the performance of the most effective selectors created by each machine learning model type for each system and overload objective . 183
8.10 Matrix comparing the performance of the most effective selectors created by each combination of primary and secondary EPM model types to PFM-OPF for the 33 kV meshed distribution system 187
8.11 Matrix comparing the performance of the most effective selectors (for minimising the number of overloads whilst minimising curtailment) created by each combination of primary and secondary EPM model types to PFM-OPF for the IEEE 14-bus system 189

8.12 Matrix comparing the performance of the most effective selectors (for minimising overload energy whilst minimising curtailment) created by each combination of primary and secondary EPM model types to PFSF-TMA for the IEEE 14-bus system 190
8.13 Matrix comparing the performance of the most effective selectors (for minimising the number of overloads whilst minimising curtailment) created by each combination of primary and secondary EPM model types to PFM-OPF for the IEEE 57-bus system 193
8.14 Matrix comparing the performance of the most effective selectors (for minimising overload energy whilst minimising curtailment) created by each combination of primary and secondary EPM model types to PFM-OPF for the IEEE 57-bus system 194

9.1 Effect of training set size on overload performance of unweighted selectors created for the IEEE 14-bus system, with the selectors only considering minimising overloads 203
9.2 Box plot of time taken to train the 10 machine learning model types that had the longest training time 205

C.1 Comparison of the performance of oracle 1 for the 11 kV radial distribution system with different algorithm combinations considered for selection . . . 258
C.2 Comparison of the performance of oracle 2 for the 11 kV radial distribution system with different algorithm combinations considered for selection . . . 259
C.3 Comparison of the performance of oracle 1 for the 33 kV meshed distribution system with different algorithm combinations considered for selection . . . 260
C.4 Comparison of the performance of oracle 2 for the 33 kV meshed distribution system with different algorithm combinations considered for selection . . . 261
C.5 Comparison of the performance of oracle 1 for the IEEE 14-bus system with different algorithm combinations considered for selection 262
C.6 Comparison of the performance of oracle 2 for the IEEE 14-bus system with different algorithm combinations considered for selection 263
C.7 Comparison of the performance of oracle 1 for the IEEE 57-bus system with different algorithm combinations considered for selection 264
C.8 Comparison of the performance of oracle 2 for the IEEE 57-bus system with different algorithm combinations considered for selection 265

List of tables

2.1 Comparison of surveyed power flow management algorithms 18

3.1 Evaluation of testing environment against requirements 54

4.1 Baseline loadings for the overloaded branches within the 11 kV radial distribution system 68
4.2 Baseline loadings for the overloaded branches within the 33 kV meshed distribution system 73
4.3 Baseline loadings for the overloaded branches within the IEEE 14-bus system 77
4.4 Statistical significance of differences in the distribution of the number of overloads for the power flow management algorithms applied to the IEEE 14-bus system 78
4.5 Statistical significance of differences in the distribution of overload energy for the power flow management algorithms applied to the IEEE 14-bus system 80
4.6 Statistical significance of differences in the distribution of the amount of curtailment for the power flow management algorithms applied to the IEEE 14-bus system 80
4.7 Baseline loadings for the overloaded branches within the IEEE 57-bus system 85
4.8 Ranking of the power flow management algorithms against the different performance measures across the four case study systems 88
4.9 Additional metrics related to the performance of the PFSF-based algorithms 94

5.1 Overview of algorithm performance and the potential performance from optimally selecting algorithms on a per-state basis for the 11 kV radial distribution system 103
5.2 Overview of algorithm performance and the potential performance from optimally selecting algorithms on a per-state basis for the 33 kV meshed distribution system 105

5.3 Overview of algorithm performance and the potential performance from optimally selecting algorithms on a per-state basis for the IEEE 14-bus system 106
5.4 Overview of algorithm performance and the potential performance from optimally selecting algorithms on a per-state basis for the IEEE 57-bus system 107
5.5 Performance comparison of the most effective algorithms for each system against the potential performance that could be obtained by per-state algorithm selection 109

7.1 Machine learning models used in this work 151

8.1 Summary of direct and EPM-based selectors giving performance benefits for each system and overload objective 165
8.2 Performance of the most effective power flow management algorithms, selectors (direct and EPM-based) and oracles for each of the case study systems 173
8.3 Configurations of the most effective direct selectors 174
8.4 Configurations of the most effective EPM-based selectors 174
8.5 Analysis of decline in selector performance when a larger set of power flow management algorithms is considered 180
8.6 Model types that created effective selectors for every case study system . . 196
8.7 Performance differences from applying tuning to (untuned) weighted selectors 197

A.1 System data model 230

B.1 Bus data for 11 kV radial distribution system 243
B.2 Circuit data for 11 kV radial distribution system 243
B.3 Transformer data for 11 kV radial distribution system 244
B.4 Load data for 11 kV radial distribution system 244
B.5 Generator data for 11 kV radial distribution system 244
B.6 Bus data for 33 kV meshed distribution system 244
B.7 Circuit data for 33 kV meshed distribution system 245
B.8 Transformer data for 33 kV meshed distribution system 246
B.9 Load data for 33 kV meshed distribution system 247
B.10 Generator data for 33 kV meshed distribution system 247
B.11 Bus data for IEEE 14-bus system 247
B.12 Circuit data for IEEE 14-bus system 248
B.13 Transformer data for IEEE 14-bus system 248
B.14 Load data for IEEE 14-bus system 249
B.15 Generator data for IEEE 14-bus system 249

B.16 Bus data for IEEE 57-bus system 249
B.17 Circuit data for IEEE 57-bus system 251
B.18 Transformer data for IEEE 57-bus system 253
B.19 Load data for IEEE 57-bus system 254
B.20 Generator data for IEEE 57-bus system 255

List of Algorithms

7.1 Procedure used to tune selector weight multiplier c_2 143

Nomenclature

Roman Symbols

$\mathscr{A}$	algorithm space in an Algorithm Selection Problem
a	algorithms in an Algorithm Selection Problem
B	set of buses
c	number of features in an Algorithm Selection Problem
c_y	weight component multiplier for performance measure y
d	number of performance measures in an Algorithm Selection Problem
$\mathscr{F}$	feature space in an Algorithm Selection Problem
f	features in an Algorithm Selection Problem
G	set of generators
i	state (of a power system)
k	size of domains in PFM-CSP algorithm
L	set of branches
l	iteration counter
$l_{\max}$	max iterations
N	sample size
n	number of generator in PFM-CSP algorithm
$\mathscr{P}$	problem space in an Algorithm Selection Problem

p	performance measures in an Algorithm Selection Problem
P_b	sum of real power injections of generators aggregated at bus b [MW]
P_b^-	"down" flexibility available from generators aggregated at bus b [MW]
P_b^+	"up" flexibility available from generators aggregated at bus b [MW]
P_b^{bias}	flexibility "bias" of generators aggregated at bus b [MW]
$P_b^\star$	real power injection at bus b from OPF solution [MW]
P_b^{min}	minimum operating point for generators aggregated at bus b [MW]
P_b^{max}	maximum operating point for generators aggregated at bus b [MW]
P_b^{target}	target operating point for generators aggregated at bus b [MW]
P_g	real power injection from generator g [MW]
P_g^{lim}	real power output limit for generator g [MW]
P_l	real power flow along branch l [MW]
P_l^{max}	maximum allowable real power flow along branch l [MW]
p_r	hypothesis r in a family of hypotheses
Q_l	reactive power flow along branch l [MVAr]
r	index of threshold p-value
S	selection mapping in an Algorithm Selection Problem
s	number of hypotheses in a family
S_l^{rating}	thermal rating of branch l [MVA]
$w_{i,a}$	weight for training example representing state i and algorithm a
x	problem instance in an Algorithm Selection Problem
$\mathscr{Y}$	performance space space in an Algorithm Selection Problem

Greek Symbols

α	significance level

ε_{μ_N}	relative error of sample
Φ	"egalitarian" proportional signal from PFSF-Egal algorithm
Φ^{-1}	inverse Gaussian cumulative probability distribution
μ_N	sample mean
σ_N^2	sample variance

Superscripts

Y	set of performance measures
y	performance measure index

Subscripts

b	bus index
g	generator index
l	branch index

Other Symbols

δ	target relative error
$\triangle P_b^+$	dummy variable, decrease in real power injection from generator g [MW]
$\triangle P_b^+$	dummy variable, increase in real power injection from generator g [MW]
$\triangle P_l^\star$	calculated change in real power along branch l [MW]
$\triangle P_l$	required change of real power along branch l [MW]

Acronyms / Abbreviations

ANM	active network management
AuRA-NMS	Autonomous Regional Active Network Management System
AVC	automatic voltage control
AVR	automatic voltage regulator
CBR	case-based reasoning

CSP	constraint satisfaction problem
CT	curtailment
DSR	demand side response
EPM	empirical performance model
FACTS	Flexible AC Transmission Systems
GA	genetic algorithm
HDF5	Hierarchical Data Format (version 5)
LP	linear program
OE	overload energy
OL	number of overloads
OLTC	on-load tap changer
OPF	optimal power flow
PFSF	power flow sensitivity factor
PSO	particle swarm optimisation
pu	per unit
PV	photovoltaic
RTDS	real-time digital simulator
SCADA	supervisory control and data acquisition
SVM	support vector machines
TMA	technically most appropriate
UK	United Kingdom
unw.	unweighted
US	United States
wei.	weighted
XML	Extensible Markup Language

Chapter 1

Introduction

This first chapter provides background on the topic of this work including a motivating example (as published in [1]). The scope and objectives of the research are then outlined, followed by an overview of the structure of this work.

The essence of this work is to explore if selecting power system control algorithms on a per-state basis – where the *state* of a system is the conditions that exist for a particular instant or period in time – can improve the performance available from automated control, over and above the performance from using the same algorithm for all states.

1.1 Background

Electric power systems are one of the most essential infrastructures in the modern world. The energy they deliver lights and heats our homes, drives many of the wheels of our transportation networks, provides the power behind industry and manufacturing, and even saves lives. Much of human progress in the last century would have been impeded – or even have been impossible – without electric power systems.

Control must be exerted within a power system under both normal and abnormal circumstances in order to maintain satisfactory operating conditions, such as maintaining frequency, voltages and equipment loading within limits. The integration of new demands – such as through the electrification of heat and transport – and of new generation – such as intermittent solar and wind – increases the stress on a power system but also adds new sources of flexibility. Due to both the increased scale and the speed of the interactions between the elements of a power system, the complexity of control is only going to increase, necessitating increased adoption of automated control approaches throughout a power system [2]. Furthermore, adoption of control systems such as active network management (ANM) may allow investment in additional network infrastructure to be reduced, deferred, or avoided [3].

Control systems that rely on digital computers use algorithms to make automated control decisions. Algorithms are sequences of instructions describing processes to solve problems, which can be concretely expressed computer programs [4]. Given a problem to solve, then a natural question to ask is which algorithm should be used in order to obtain the best performance? This could be in terms of the time taken to solve the problem, or some measure of the quality of the solution, such as in optimisation.

For search and optimisation algorithms at least, there is no single answer to the question of which algorithm is the best to be used on all problems. The "No Free Lunch" theorems of Wolpert and Macready [5], which were stated for search and optimisation, essentially state that over a broad enough selection of problems the average performance of different algorithms will be the same. Thus, no algorithm can outperform all others in all problems, so the only way to achieve the best possible performance is to select the best algorithms for each problem, rather than always using the same algorithm.

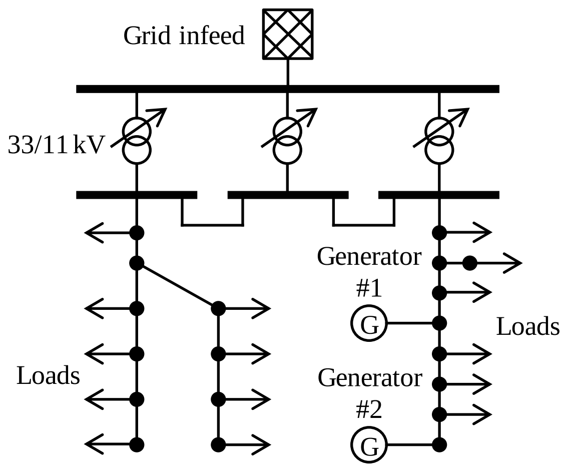
The idea that different algorithms give the best performance has been exploited in numerous problem domains – from planning [6] and search [7] within computer science, to bioinformatics [8], computational chemistry [9], and financial trading [10] – in order to obtain improved performance by selecting algorithms for specific problems. This work explores whether such fine-grained selection of algorithms can be applied to power systems control. In particular, this work examines whether any performance benefit can be obtained by selecting control algorithms on a *per-state* basis, where different algorithms may be used as the system conditions change.

1.2 Motivating example

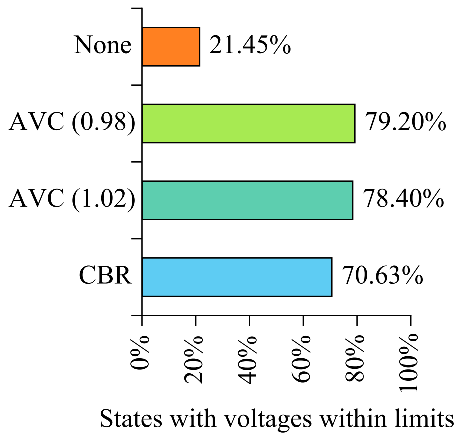
A small-scale pilot study was conducted into whether per-state algorithm selection could be of benefit in power system control applications [1]. In particular, the study examined voltage control within a model of a radial distribution system (shown in Figure 1.1a) that featured an infeed at 33 kV, transformation down to 11 kV using transformers with on-load tap changers (OLTCs), and two 11 kV feeders, one containing two generators. 2625 different system states were modelled, with the OLTC tap position, load level and the outputs of the two generators varied in discrete steps, and it was observed that under these conditions, for 21.45% of the states the voltages within the network fell outside the limits assumed for the study.

In addition to modelling the base condition of the network, the conditions following the application of three different algorithms were modelled:

- AVC (0.98): the first algorithm was an implementation of automatic voltage control (AVC) [11], which acted on the transformer tap positions only in order to achieve



(a) System schematic



(b) Results for each control algorithm

Fig. 1.1 System schematic and results for pilot study into voltage control algorithm selection

a voltage set point target of 0.98 pu on the low voltage terminals of the distribution transformers.

- AVC (1.02): the second algorithm also implemented AVC, but with a voltage set point target of 1.02 pu.
- CBR: the third algorithm used an artificial intelligence approach, case-based reasoning (CBR) that acted on the OLTC tap positions and the outputs of the generators [12]. This algorithm contained a "case base" of previous voltage excursion events and the pre-determined actions taken to mitigate the excursions. When a new voltage excursion event is encountered, it is matched to a previous event based on the similarity of network conditions, and the control actions taken in the previous event are then applied to mitigate the current excursion.

As Figure 1.1b shows, each of the algorithms was able to reduce the number of states in which the voltages within the network were outside limits, although no algorithm was able to keep voltages within limits for every state. However, for each state there was at least one algorithm that could keep voltages within limits, so it was possible to keep voltages within limits if the algorithm used for control was varied between states, thus there was a potential performance benefit by per-state algorithm selection.

To examine if this potential performance benefit could be realised, a machine learning algorithm (the C4.5 tree learning algorithm [13]) was used to create an algorithm selector that predicted which algorithm would be able to keep voltages within limits, based on measurements of the system state. 66% of the modelled system states were selected at random to be used in the training data for creating the algorithm selector. The remaining 34% of states were used for testing the algorithm selector and to compare its performance to that of each of the voltage control algorithms. The best performing algorithm for the states used to test the selector was AVC (0.98), which could keep voltages within limits for 78.92% of the test states, a similar proportion to that algorithm's performance on all the modelled states. The machine learning-based algorithm selector, however, could outperform this and was able to keep voltages within limits for all the test states, by selecting appropriate algorithms for each state. Thus, the algorithm selector was able to exploit the potential performance benefit from per-state algorithm selection.

1.3 Research objectives

The pilot study not only demonstrated that there was a potential performance benefit from per-state algorithm selection, but also that an algorithm selector could be created -— using machine learning — to exploit the performance benefit. Although the results were promising, the scope of the study was limited, restricting the possibility to generalise the findings. The main weaknesses were, in particular:

1. Only one system was studied. In order to generalise the findings, studies on networks of different scales and topologies would be needed.

2. The discrete state space may ignore subspaces where the performance of the algorithms is more varied; furthermore, only a limited, and finite, number of states were available for creating and testing algorithm selectors. To generalise the findings would require continuous state spaces, with states distributed to represent a range of system conditions and to more fully characterise the performance of the algorithms used.

3. Although three control algorithms were tested, two of these were just different parameterisations of the same algorithm (AVC), so only two diverse approaches were tested. There could be other algorithms that are more effective for some or all states.

4. Only one machine learning algorithm was used to create an algorithm selector. The results of the pilot study are insufficient to say whether that algorithm would produce effective selectors for other power systems, states, and sets of control algorithms.

Another weakness is that variations in voltage are a localised phenomenon, so voltage control tends to be restricted to limited regions within a power system. Other phenomena and their related control tasks may operate on wider scales – such as the control of frequency

– or operate on a range of scales – such as controlling power flows. The absence of these characteristics from voltage control limits the potential to generalise any findings about per-state algorithm selection for voltage control to many other power systems control tasks.

In order to examine what benefits per-state algorithm selection could have more generally for power systems control, and considering the weaknesses of the pilot study, the research presented in this work has the following objectives:

1. Examine if potential performance benefits for power systems control can be derived by selecting the algorithms on a per-state basis. This objective has the following sub-objectives:
   (a) Identify a power systems control task that has characteristics shared with many other power systems control tasks, so the results for the one control task are likely to be generalisable to other tasks.
   (b) Implement and test several power systems control algorithms for the chosen control task, which represent diverse approaches to tackling the control task.
   (c) Test the algorithms on power system models that represent different network designs.
   (d) Simulate a wide range of conditions within the power system models, in order to exercise the performance of the power system control algorithms.
2. If the answer to research objective 1 is affirmative, the research shall investigate if algorithm selection systems (*algorithm selectors*) can be created to exploit the potential performance benefits for power systems control from per-state algorithm selection. This includes:
   (a) Identifying existing algorithm selection techniques that could be applicable to developing algorithm selectors for power systems control.
   (b) Exploiting the existing algorithm selection techniques to create algorithm selectors for power systems control.
   (c) Extending the existing techniques and, when necessary, developing new techniques to allow algorithm selection to be applied to power systems control.

1.4 Structure of this work

This work splits into two sets of chapters, which address each of the research objectives in turn. The first set (Chapters 2 to 5) addresses research objective 1, exploring the performance of several power system control algorithms before examining if any performance benefit exists from using per-state algorithm selection. The second set of chapters (Chapters 6 to 8) deal with research objective 2, and thus investigate if any potential performance benefit from algorithm selection can be exploited by creating algorithm selectors.

Chapter 2 follows the present chapter and focusses on algorithms for power systems control. An overview of control within power systems is provided, before focussing on power flow management, which is the example power system control task that is used throughout this work. Existing power flow management algorithms are surveyed, in order to identify the main approaches and to determine a set of diverse algorithms to be implemented for this work. The implementation of each of those algorithms is then described in detail.

Chapter 3 describes the software system that was developed in order to allow large-scale testing of the control algorithms described in Chapter 2. This includes detail of power system modelling capabilities and the particular design features that allowed for repeatable testing of multiple control algorithms in parallel.

Having established the control algorithms to be tested in Chapter 2, and the environment used for testing them in Chapter 3, Chapter 4 presents results from testing the algorithms on four case study power systems. First, the testing and evaluation methodology is described, including the performance measures and statistical tests that are used for the remainder of this work. Then, for each case study system, a description of the system and the states simulated within it are provided, followed by the baseline performance of the system and of each control algorithm. A cross-case study analysis is then provided, which includes analysis of the performance of each algorithm and their execution times.

While Chapter 4 presents results for each control algorithm that were aggregated across all states, in Chapter 5 the same results are examined on a state-by-state basis, to determine whether there are any potential performance benefits from per-state algorithm selection. Any potential performance benefits are quantified and their significance assessed, in order to satisfy research objective 1.

Chapter 6 is the first of three chapters that concentrate on developing algorithm selectors for power systems control. Specifically, this chapter surveys previous work on algorithm selection, starting with describing the *algorithm selection problem* in general, before reviewing previous approaches and applications. An overview of machine learning is also provided, as that is highly pertinent to the creation of algorithm selection systems, and is also used in the subsequent chapters of this work.

Chapter 7 is based on the findings of Chapter 6 and develops two main algorithm selector designs to be applied to per-state algorithm selection for power flow management. These selector designs rely on measurements of a power system's state in order to make selection decisions, and do not know *a priori* which algorithm will provide the most effective performance for any state. Although based on existing algorithm selection approaches, the selector designs and their variants presented in the chapter include new developments to tailor them to the power flow management application. The design choices specific to each design and those that are common to both are described, as well as stating the design options that have been investigated in this work.

Chapter 8 presents the results of using the selector designs described in Chapter 7 to select power flow management algorithms within the case study systems. The effectiveness of the the selectors is assessed, along with the effect of each of the design choices.

Chapter 9 is a discussion of aspects relevant to the implementation of algorithm selectors, based on the results presented in earlier chapters.

Chapter 10 concludes the work. The outcomes of research are assessed against the objectives set in Section 1.3, and the major contributions of the work identified. Finally, prospects for applying algorithm selection more generally within power systems control are outlined, along with future work that could develop this research further.

Chapter 2

Algorithms for power systems control

This chapter provides background on control within power systems, before focusing on the control task that this work is concerned with: power flow management. Existing algorithms for power flow management are surveyed and several diverse approaches are identified for implementation. The implementations of these algorithms for this work are then described.

2.1 Control in power systems

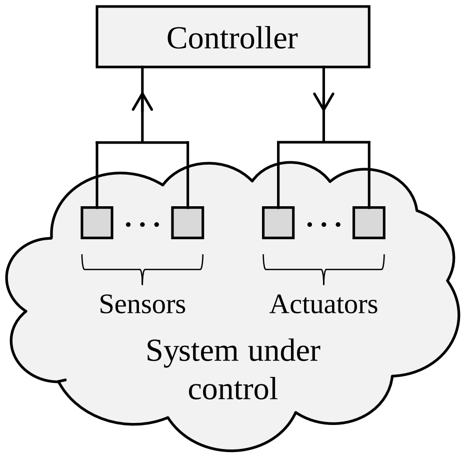


Fig. 2.1 General overview of control

Control is the act of exerting influence on a system to achieve a desired state. Illustrated in Figure 2.1, a controller observes the system state through sensors and exerts influence on the the system under control via actuators.

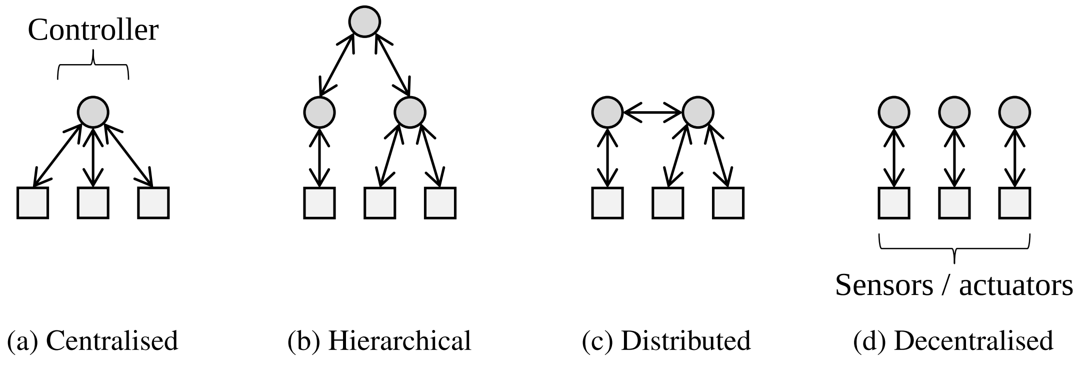


Fig. 2.2 Control architectures

There may be one or more controllers acting on a system, and, as shown in Figure 2.2, four control architectures can be distinguished, as described in [14]:

- Centralised (Figure 2.2a): a single controller interacts with all sensors and actuators.
- Hierarchical (Figure 2.2b): an extension of a centralised architecture where there is one controller that coordinates the actions of all actuators through intermediate controllers.
- Distributed (Figure 2.2c): multiple controllers coexist and coordinate their actions through communications between the controllers.
- Decentralised (Figure 2.2d): multiple controllers coexist and act independently of each other; there is no direct communication apart from interactions within the controlled system (known as stigmergy [15]).

Within the context of power systems, control is necessary to maintain acceptable operating conditions while the system is subject to continual perturbations. These perturbations come from different sources and vary in magnitude and duration. Generation is subject to plant outages, either due to faults or due to maintenance or upgrades, and is also subject output changes caused by variations in the energy source – for renewable generation such as wind or solar – or driven by market conditions. Demand changes throughout a day due to customer behaviour and weather, and the daily patterns vary across the days in a week, between different seasons, and between years; with changes in demand over longer timescales coming from the uptake of new devices such as through the electrification of heat and transport. The network itself changes, with equipment outages, due to faults or for maintenance or upgrades, and changes in equipment performance due to age and external factors, such as the weather.

There are a wide variety of actuators within a power system that can be used to effect control. These include adjusting generator real and reactive power output, transformer tap changers, transformer phase shifting, switching shunt devices such as capacitor banks, Flexible AC Transmission Systems (FACTS) devices, demand response such as load shedding, and network switching. Furthermore, there are wide variety of sensors within a power system that could be used as inputs for control, particularly at the transmission voltage levels, such as current and voltage transformers, weather stations, and smart meters.

Controlling a power system is a complex task. However, due to the varied nature of the perturbations affecting power systems and the different characteristics of actuators, it is possible to split the overall control task into a number of decoupled control tasks [16]. The control tasks can be decomposed in time; for example, on shorter timescales are tasks such as protection (which acts on the order or milliseconds), and on longer timescales are tasks such as tap changer control (which acts on the order of seconds to minutes). Control tasks can be decomposed in space; from the scale of device-level controllers, such as generator automatic voltage regulators (AVRs), to system-wide control, such as generator scheduling and dispatch. Control tasks can also be decomposed according to the physical quantity controlled; for instance, frequency and voltage can be controlled independently.

In this work, the potential for applying algorithm selection to power systems control is explored through investigating a particular control task: power flow management. Power flow management is a type of congestion management, which is a general term for actions associated with maintaining system operation within thermal, voltage and stability limits. Congestion management includes control but has also been used in reference to other aspects of power systems, such as market structures and determining the installation locations of new FACTS devices [17]. In this work, power flow management is defined as a type of congestion management that is concerned with adjusting the real power injection of generators in order to alleviate the thermal overloading of branches within a power system. Generators whose output is adjusted to below their target output level are said to be “curtailed”, and power flow management can be extended to also consider the adjustment of the real power draw of loads.

Power flow management shares a number of features with other power systems control tasks. It considers the same actions as other control tasks, for example, adjusting generator outputs is fundamental to economic dispatch and can also be considered for voltage control [18]. Power flow management can be used to control devices across a wide area, similar to network reconfiguration and frequency control. Furthermore, several algorithms have been proposed for power flow management (as described in the literature survey that follows), and there are numerous other control tasks for which several algorithms have been proposed. These shared features indicate that power flow management is generally representative of a number of power systems control tasks, and therefore results of the application of algorithm selection to power flow management could be applied more broadly in power systems control.

2.2 Power flow management: literature survey

This section surveys existing power flow management algorithms in order to identify a number of algorithms that will be implemented for testing in later chapters.

2.2.1 OPF-based methods

Optimal power flow (OPF), introduced by Carpentier [19], is a family of related optimisation problems that consider a set of power flow equations within their constraints [20].

The numerous formulations of OPF are in general non-linear and non-convex, and consider different objectives, decision variables and constraints, for example:

- Objectives: minimisation of generation costs, minimisation of system losses.
- Decision variables: these consider different controllable items within the power system, and can be either continuous – such as generator output levels – or discrete – such as transformer tap settings.
- Constraints: these can include equality constraints – such as the power flow equations – and inequality constraints – such as circuit thermal ratings and bus voltage limits.

There are numerous OPF formulations within the large body of OPF literature (the IEEE lists over 2000 publications that mention “optimal power flow” [21]), along with various solution methods. These include deterministic methods such as gradient methods [22], sequential linear programming [23], and interior point methods [24], as well as non-deterministic approaches such as genetic algorithms [25, 26] and particle swarm optimisation [27]. Particular solution methods are more suited for particular OPF formulations, and convergence speed and convergence guarantees vary between the methods. Convergence to the global optimum of a non-convex OPF problem is often not guaranteed; however, recent work by Lavaei and Low [28] has shown that under certain conditions, non-convex OPF problems have an equivalent convex problem (a semidefinite program). If the OPF is feasible, a globally optimal solution of the equivalent problem can be found and then translated in to a globally optimal solution of the original non-convex OPF.

Power flow management can be formulated as a form of OPF problem where the set of constraints include circuit ratings and the decision variables include controllable items that affect power flows. This was the approach taken for the Autonomous Regional Active Network Management System (AuRA-NMS) project [29–31], where the formulation considered minimising the amount of generator curtailment while keeping branch loadings within limits. The objective function also included scaling terms for the curtailment variable of each generator, which allows a curtailment priority order to be enforced. The OPF was implemented within the commercially-available PowerWorld [32] power system analysis package and applied to distribution network models with both radial (11 kV) and meshed (33 kV) topologies. In addition to the PowerWorld-based algorithm, there was also some work on a bespoke interior point solver [33]. Recent work by members of the same research group [34] has extended the OPF approach to multiple time instants with temporal constraints considered: the so called “dynamic optimal power flow”.

Alnaser and Ochoa [35] describe another OPF-based approach to power flow management. They formulate an OPF problem within the AIMMS [36] optimisation software that maximises the output of controlled generators (thus minimising curtailment) while satisfying limits for branch flow, voltage and generator power factor. Along with generator curtailment, the formulation also considers changing transformer tap positions and the reactive power setpoints of the generators. In a case study where voltage was the binding constraint on maximising the power export from generators, considering these other control actions was found to significantly reduce the amount that generators were curtailed.

2.2.2 Sensitivity factor-based methods

A number of authors have developed power flow management algorithms that use sensitivity factors that relate changes in real power injections at buses to changes in power flows along branches (variously referred to as power flow sensitivity factors (PFSFs) [37], power transfer distribution factors [38], or generation shift factors [39]). These sensitivity factors are derived from load flow solutions and represent a linearisation of power system behaviour around a particular operating point.

Jupe et al. [37, 40, 41] present different methods for power flow management based on sensitivity factors:

- One method curtails generators according to a priority order, in this case the order of connection (last-in first-out – LIFO), with the amount that each generator is curtailed calculated from the magnitude of the overload and the generator’s sensitivity factor in relation to the overloaded branch. Different priority orders (termed *principles of access*) can have an effect on the overall amount of generator curtailment [42].

- There is another method that is similar to the above, but uses a priority order based on the magnitude of each generator’s sensitivity factor relating to the overloaded branch. This order means that the generators that have a greater influence on a particular overload – that is, those that are “technically most appropriate” (TMA) to remove an overload – are curtailed first. Chang and Hsu [43] describe a similar approach but use an order based on the total power change that each generator could contribute to alleviating an overload, which is calculated from the sensitivity factors and the maximum amount that each generator can be curtailed.

- Another method uses the sensitivity factors and the amount that each generator can be curtailed to calculate an “egalitarian” curtailment signal that curtails all generators by the same proportion of their present output.

Each of the methods was implemented in a system that included a full AC load flow engine, which was used to validate that the calculated curtailments did alleviate overloads. Case studies on different power system models, including meshed network topologies, indicate that each method is able to alleviate overloads, but with significant differences in the amount of curtailment applied to the generators.

Sensitivity factors provide linear relationships between the key variables involved in power flow management, and, if linear power flow constraints are assumed, allow for power flow management to be formulated as a linear program (LP). This is the approach taken by Skokljev et al. [44], whose formulation considers increases and decreases in power injections at each bus, so can be applied to both generator curtailment and load shedding. The objective of the LP is to minimise the magnitude of changes applied to the generators and loads within the network, subject to power flow constraints represented as linear inequalities. The approach was demonstrated on the IEEE Reliability Test System model [45], which has a meshed network topology.

Sensitivity factors can be pre-calculated and can be manipulated using simple, linear operations, which allow algorithms that use them to be lightweight and execute quickly. However, a major weakness of sensitivity factors is that they are a linearisation of the (non-linear) system around one operating point, and that they can become inaccurate as the system moves away from that operating point.

2.2.3 Other methods

Another approach developed during the AuRA-NMS project was to take a method from computer science, constraint programming, and apply it to power flow management [29, 31, 46, 47]. In particular, power flow management was formulated as a constraint satisfaction problem (CSP), consisting of the following three elements:

- Variables: these represent the control variables, which are the amount each generator is curtailed and, in recent work [47], the amount that responsive demand that is shed.

- Domains: these describe the possible values that each variable is allowed to take; in particular, discrete domains containing four or five values were used for AuRA-NMS.
- Constraints: these are additional restrictions on what values one or more variables are allowed to take. In the AuRA-NMS work, three constraints were considered:
  1. Overloads: an assignment of the values to variables – the curtailment and load shedding levels – must not result in any branch being overloaded. This constraint is evaluated by applying candidate variable assignments to a power system model, executing a full AC load flow, and checking branch loadings.
  2. Order: the order in which the generators are curtailed (or loads shed) must satisfy a particular priority order, defined within principles of access. The ability to directly represent these kind of constraints – which may be defined within connection contracts – is a particular advantage of using CSP.
  3. Preference: this soft constraint is used to favour valid assignments of values to variables that have the least amount of generator curtailment (or load shedding).

A solution to a CSP is any assignment of values to variables where the values are drawn from the domains and all constraints are satisfied. Solutions are found using search algorithms, and in the AuRA-NMS work an off-the-shelf best-first backtracking search algorithm is used. The *best-first* aspect of the search algorithm refers to it using some heuristic to determine which is the most promising search direction, while *backtracking* refers to search directions being abandoned during the search if partial solutions are found to be invalid. A CSP may have a number of valid solutions, which is why the preference constraint (3) is used to sort through these to return solutions with minimal curtailment (or load shedding).

The CSP approach has been successfully demonstrated on case study distribution system models, of both radial (11 kV) and meshed (33 kV) network topologies. Although it appears adaptable to different network topologies, the CSP approach to power flow management is limited by poor scalability characteristics. The worst-case time complexity of the algorithm is of the order $\mathscr{O} \sim k^n$ [29], where k is the size of the domains and n the number of generators. This limits the CSP approach to small numbers of generators (and loads), and small domain sizes that give only coarse curtailment values.

Currie et al. [48] present an approach to power flow management that uses pre-set branch operating margins to determine when to “trim” (partially curtail) or “trip” (fully curtail) generators’ output. These margins are calculated for each branch that could potentially become overloaded, with values based on the branch thermal ratings and the maximum real power ramp rates of the generators that contribute to the overloading of a branch. The method does not calculate by how much a generator needs to be curtailed to alleviate an overload; instead, the “trim” signal simply instructs generators to start reducing their output until the branch rating goes below the “trim” margin. Inherent in the method is the identification of which generators contribute to the overloading of particular circuits, which is required for determining the operating margins and which generators each set of margins applied to. This is straightforward for radial networks, but it is unclear whether the method could be generalised to meshed networks also.

2.2.4 Comparison of methods

Most of the surveyed algorithms have only been tested in isolation, so it is difficult to compare the performance of the methods. Comparative studies have been performed by Dolan et al. [31] – for the CSP- and OPF-based approaches developed for AuRA-NMS – and Jupe et al. [37] – for the LIFO, TMA and “egalitarian’ sensitivity factor-based methods. However, these studies were limited to small sets of algorithms and a small number of test cases – either in terms of the number of networks used for testing, or the number of network states that the algorithms were tested on. Therefore, there is a need to perform comparative studies of a number of the surveyed algorithms, on a number of networks and also across a number of network states (which aligns with research objectives 1(b) and 1(d)).

2.2.5 Selection of methods to be implemented

Table 2.1 presents a summary of the surveyed power flow management algorithms, considering the following aspects:

- Reference: a descriptive identifier for each algorithm and publication references.
- Network type: if the algorithm can be applied to any network topology (radial or meshed), otherwise it can be only be applied to radial networks.
- Generator controls: whether the algorithm can be used to control generators.
- Load controls: whether the algorithm can be used to control loads.
- Voltage limits: whether the algorithm also considers bus voltage limits along with branch thermal limits.
- Availability: whether an implementation of the algorithm is available as a whole or in part. For some algorithms, source code is available (either open-source, or closed-source available to the author), whereas for others, only the description in literature is

available. Also indicated is where solvers for part of the algorithm's process are easily available.

- Comments: any further observations about each method.

In selecting which algorithms to implement, the characteristics of the algorithms were considered in isolation and also as part of a set of algorithms. Firstly, it was vital that each algorithm was able to be implemented. Secondly, the set of algorithms needed to be representative of the diverse range of methods that have been surveyed. Thirdly, the algorithms needed to act on the same set of control devices – specifically, generators, for power flow management as defined in this work – otherwise, differences in algorithm performance could be attributed to the difference in what control devices were used rather than the algorithm's processes. Finally, each algorithm within the set must be capable of running on the same networks and range of states within those networks as all the other algorithms in the set, otherwise performance comparisons could not be made. This included the need for algorithms that could be applied to a sequence of states (simulating network operation for a time period) and also to individual "snapshot" states.

The first two OPF methods in Table 2.1 consider controlling generators to alleviate branch overloads. However, the Alnaser and Ochoa OPF approach considers additional control devices (tap changer settings and generator reactive power) as well as considering bus voltage limits. These additions are not shared by any of the other algorithms in the table, and, if ignored, would make the algorithm essentially the same as the AuRA-NMS OPF approach. For this reason, the Alnaser and Ochoa OPF approach is disregarded.

Non-deterministic OPF approaches such as GA and PSO, represented in the third row in Table 2.1, could be adapted for use as power flow management algorithms. Although an appropriate formulation of power flow management for input in to a non-deterministic optimisation algorithm would need to be created, the optimisation algorithms themselves could be implemented from readily available libraries. However, by their very nature, non-deterministic algorithms may give different results when repeatedly applied to exactly the same problem. Therefore, to characterise their performance on a problem requires multiple tests – which would result in additional tests compared with the other algorithms – and necessitate looking at their *average* performance – which complicates analysis and comparison to the other (deterministic) algorithms. For these reasons, non-deterministic approaches were disregarded. Although this means a distinct approach for power flow management has not been examined in this work, the set of algorithms considered for implementation still represents a range of diverse approaches.

Table 2.1 Comparison of surveyed power flow management algorithms

Reference	Network type	Generator controls	Load controls	Voltage limits	Availability	Comments
AuRA-NMS OPF [29–31]	Any	Yes	–	–	Description in literature only, OPF solvers available	Can accept priority order for generators
Alnaser and Ochoa OPF [35]	Any	Yes	–	Yes	Description in literature only, uses commercial optimisation software	Also considers tap settings and generator reactive power
Non-deterministic OPF (e.g. GA [25, 26], PSO [27])	Any	Yes	Possible	Possible	General descriptions in literature, GA and PSO libraries available	Would require re-formulation of OPF for power flow management
Jupe et al. LIFO [37, 40, 41]	Any	Yes	–	–	Source code available	Requires generator priority order
Jupe et al. TMA [37, 40, 41]	Any	Yes	–	–	Source code available	–
Jupe et al. “egalitarian” [37, 40, 41]	Any	Yes	–	–	Source code available	–
Chang and Hsu [43]	Any	Yes	Yes	–	Description in literature only	Very similar to Jupe et al. TMA
Skokljev et al. LP [44]	Any	Yes	Yes	–	Description in literature only, LP solvers available	–
AuRA-NMS CSP [29, 31, 46, 47]	Any	Yes	Yes	–	Description in literature only, CSP solvers available	–
Currie et al. “trim” and “trip” [48]	Radial	Yes	–	–	Description in literature only, patented	–

Both the AuRA-NMS OPF and Jupe et al. LIFO algorithms recognise generator priorities, which allows generators to be curtailed in a specific order (principles of access). However, this feature is not shared with the other algorithms in Table 2.1. For the AuRA-NMS OPF algorithm, priority lists are optional and can be ignored, but for the Jupe et al. LIFO algorithm, priority lists are essential for the method to work. For this reason, the Jupe et al. LIFO algorithm is disregarded.

The Jupe et al. TMA algorithm and the algorithm of Chang and Hsu have little difference in their approaches. It was more straightforward to implement the Jupe et al. TMA algorithm as its source code was made available to the author, and its description in literature provides more implementation detail. For this reason, the Chang and Hsu algorithm is disregarded.

All of the surveyed algorithms can be applied to both radial and meshed networks, except for the Currie et al. “trim and trip” algorithm. That algorithm only appears to be applicable to radial networks, which would severely limit the range of networks that the set of algorithms could be tested on. The algorithm also relies on past information about the system under control (specifically, the past outputs of generators, in order to determine ramp rates), so it could not be applied to determine control actions for “snapshot” problems where only a single state is simulated, whereas the other methods can be applied to such problems. Furthermore, as it is a patented method, it would be less straightforward to implement than the other algorithms, if not impossible. For these reasons, the Currie et al. “trim and trip” algorithm is disregarded.

2.3 Power flow management: implemented algorithms

Following the analysis in Section 2.2.5, five algorithms remain, which were implemented for this work:

1. AuRA-NMS CSP
2. AuRA-NMS OPF
3. Jupe et al. “egalitarian”
4. Jupe et al. TMA
5. Skokljev et al. LP

The implementations of these algorithms that were used for this work are described within this section, including any changes or assumptions made from the algorithms’ original description in literature.

2.3.1 PFM-CSP

The PFM-CSP algorithm used for this work is an implementation of the CSP-based power flow management algorithm developed for the AuRA-NMS project [29, 31, 46, 47]. Figure 2.3 is a flowchart of the processof the algorithm when applied to an individual state of a power system. This process essentially consists of: 1) checking if there are overloaded branches or curtailed generators in the network, 2) formulating and solving the CSP, if there are overloads or curtailed generators, and 3) taking one of the CSP solutions and applying it to the generators in the network. Rather than specifying output setpoints, the PFM-CSP algorithm determines output limits for the generators within the network, which they will then operate within.

For PFM-CSP, the three constituents of the CSP problem are as follows:

- Variables: each variable represents the output limit, in relation to each generator's rated output, to be applied to one of the controllable generators in the network.
- Domains: these are discrete lists of the possible values each variable is allowed to take. Similar to previous work, the domain of each variable is $\{0\%, 50\%, 100\%\}$, where 100% represents no output limit applied (generator can run uncurtailed), and 0% represents an output limit that would fully curtail a generator.
- Constraints: two of the three constraints from AuRA-NMS are implemented:
  - The "overloads" constraint is implemented by applying an assignment of variables (the output limits), defined by a candidate solution, as output setpoints to generators within a network model internal to the algorithm. A load flow is then executed and branch loadings checked, in order to determine whether the "overloads" constraint has been violated. The internal network model is updated using measurements from the network under control before the CSP solution process starts, and is reset to that condition before each constraint check.
  - The "preference" constraint is implemented by sorting through the CSP solutions, finding the solution with the least generator curtailment, and then applying that solution to the network.
  - The "order" constraint was not implemented as it was assumed that there was no priority ranking of generators.

To find feasible solutions to the CSP a best-first backtracking search algorithm is used, as per the AuRA-NMS approach. In particular, the PFM-CSP implementation uses the open-source "constraint" Python package [49] to both formulate and solve the CSP.

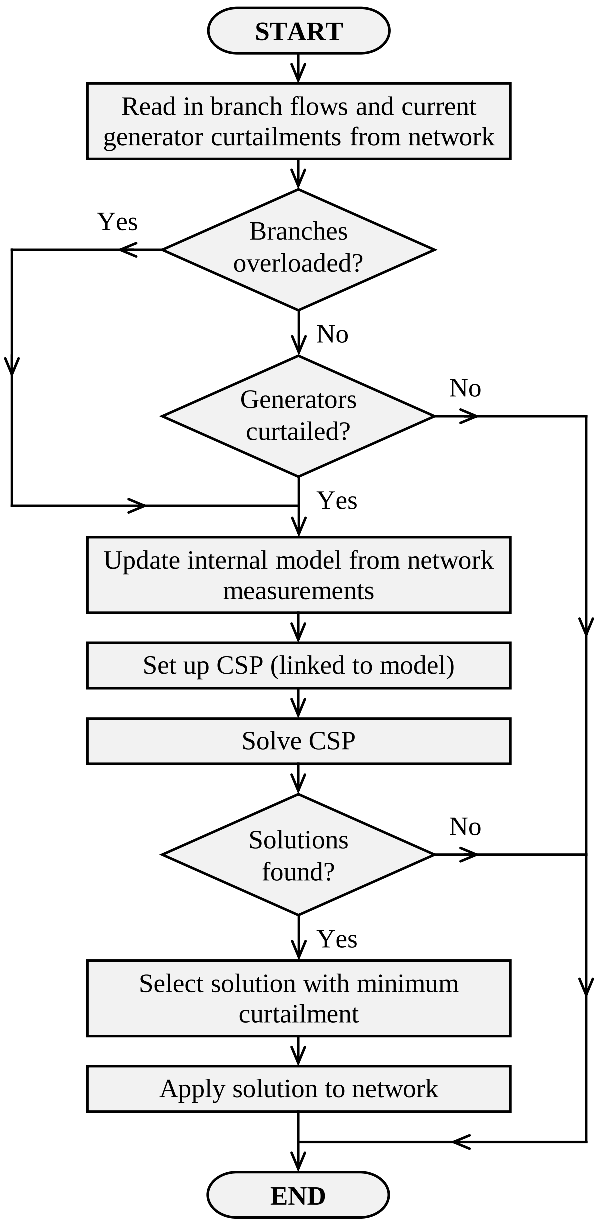

START
Read in branch flows and current generator curtailments from network
Branches overloaded?
Yes
No
Generators curtailed?
No
Yes
Update internal model from network measurements
Set up CSP (linked to model)
Solve CSP
Solutions found?
No
Yes
Select solution with minimum curtailment
Apply solution to network
END

Fig. 2.3 Flowchart for the processes of the PFM-CSP algorithm

2.3.2 PFM-OPF

The PFM-OPF algorithm is based on the OPF-based power flow management algorithm developed for the AuRA-NMS project [29–31]. The algorithm's process, shown in flowchart form in Figure 2.4, is similar to that of PFM-CSP, and can be summarised as: 1) checking if there are overloaded branches or curtailed generators in the network, 2) formulating and solving the OPF problem, if there are overloads or curtailed generators, and 3) taking the OPF solution and applying it to the generators in the network. Unlike PFM-CSP, PFM-OPF is used to determine generator output setpoints, rather than output limits.

The "PyPower" package [50], which is a Python translation of the MATPOWER [51] power systems analysis toolbox, is used for the formulation and solution of the OPF. The input provided to PyPower is in the form of a "case", which is a set of matrices describing the buses, branches, generators and generator cost functions of the network. The case is a model internal to the PFM-OPF algorithm and can be used for load flow as well as OPF. The case for a particular network is generated before the algorithm's process based on data about the network being controlled, and is updated during PFM-OPF's process from network measurements. Once the case is updated, a load flow is executed to check that the model is in close agreement with the network measurements. Once the case is verified, PyPower's built-in primal-dual interior point solver is used to solve the OPF.

The OPF within PyPower has the objective function of minimising generator costs, subject to a standard set of constraints, namely: the power flow equations, branch thermal limits, bus voltage magnitude limits, and generator real and reactive power limits. The extensible OPF framework within PyPower (MATPOWER) allows for the OPF formulation to be augmented to suit a particular application. This framework was utilised to tailor the standard formulation for PFM-OPF with respect to: 1) the control variables, 2) the cost functions, and 3) the constraints.

Rather than using the output of each individual generator as a control variable, PFM-OPF aggregates all the generators at a particular bus into a single generator whose output is determined by the solution process. This reduces the number of variables in the OPF, speeding up its solution. For each aggregated generator, a number of variables are defined that describe the real power "flexibility" at a bus, b, as shown in Figure 2.5. In the figure, P_b represents the sum of the present power injections of the aggregated generators, P_b^{target} represents the sum of the target operating points for the aggregated generators (as the generators may be operating away from their target operating points), while $P_b^{\min}$ and $P_b^{\max}$ are the sum of the minimum and maximum operating points for the aggregated generators, respectively. Based on these variables, the "down" and "up" flexibility at a bus can be defined, respectively, as: $P_b^- = P_b^{\text{target}} - P_b^{\min}$ and $P_b^+ = P_b^{\max} - P_b^{\text{target}}$; along with a flexibility "bias", P_b^{bias}, to account

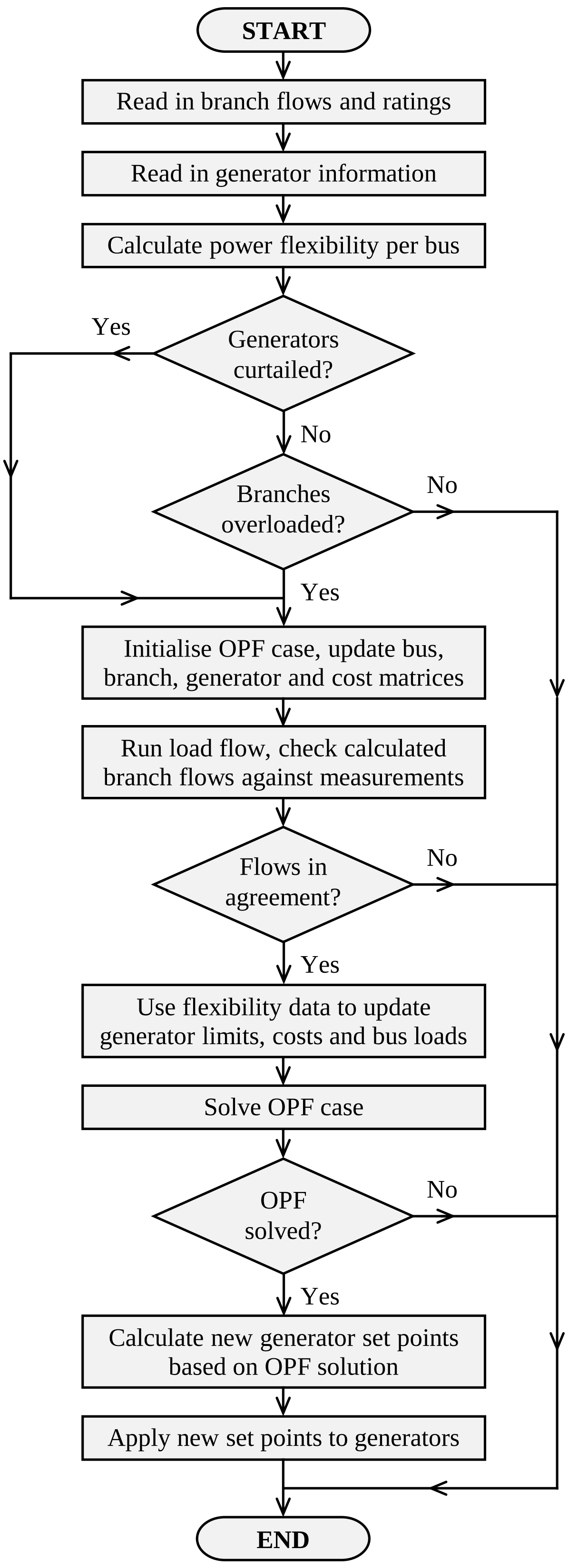

START
Read in branch flows and ratings
Read in generator information
Calculate power flexibility per bus
Generators curtailed?
Yes
No
Branches overloaded?
No
Yes
Initialise OPF case, update bus, branch, generator and cost matrices
Run load flow, check calculated branch flows against measurements
Flows in agreement?
No
Yes
Use flexibility data to update generator limits, costs and bus loads
Solve OPF case
OPF solved?
No
Yes
Calculate new generator set points based on OPF solution
Apply new set points to generators
END

Fig. 2.4 Flowchart for the processes of the PFM-OPF algorithm

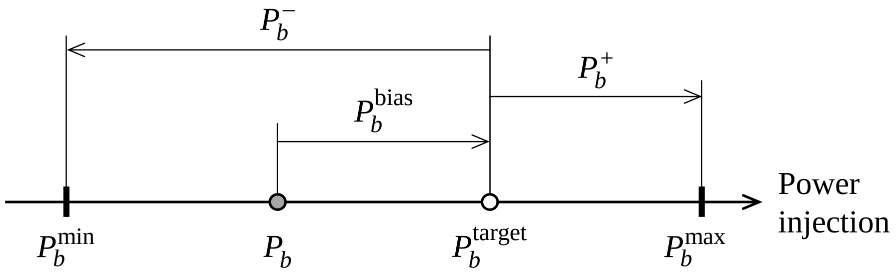


Fig. 2.5 Bus real power flexibility as used in the PFM-OPF algorithm

for any differences between the present and target real power injections at a bus. Although PFM-OPF only considers controllable generators, it would be trivial to include adjustable loads when aggregating real power flexibility.

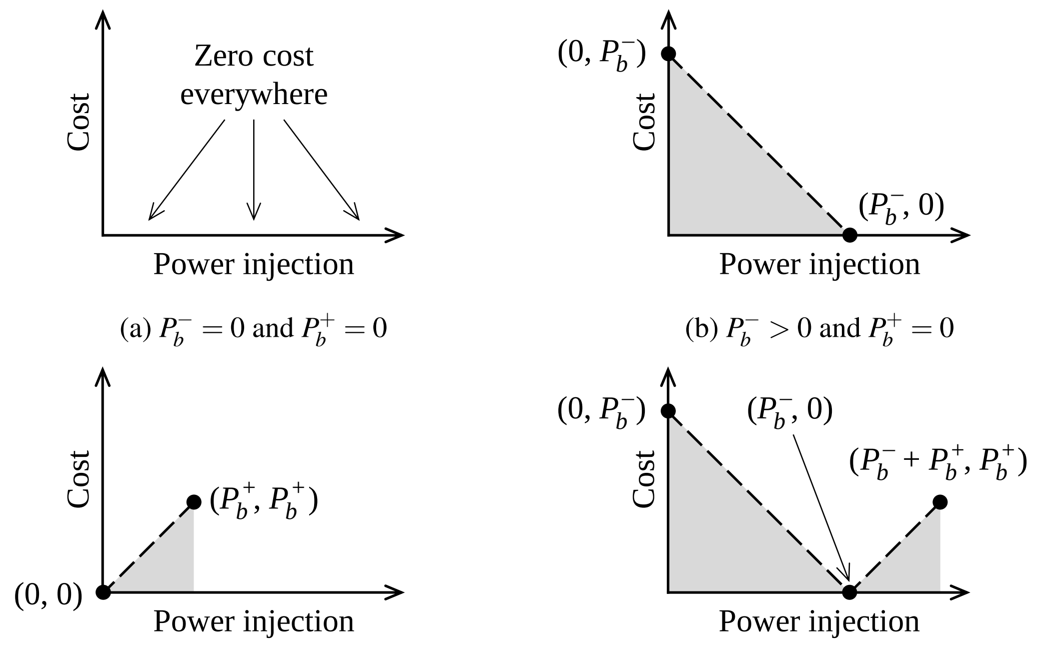
The standard OPF objective function of minimising generator costs is used; however, the cost functions of the generators are modified so that the OPF minimises the deviation from the target operating points (P_b^{target}) at each aggregated generator bus. For each aggregated generator, one of the piecewise-linear cost functions shown in Figure 2.6 is used, depending on the flexibility available:

- If no flexibility is available, then a flat cost of zero is assumed for all real power injection values (Figure 2.6a).

- If only "down" flexibility is available, then a cost function with a single line section is assumed (Figure 2.6b). This has a minimum cost (zero) where the power injection results in the target operating point, P_b^{target}, being achieved (when the power injection is equal to the "down" flexibility); and a maximum cost when the power injection results in the maximum curtailment (when the power injection is equal to zero). The gradient of the line section is $-^1/_2$.

- If only "up" flexibility is available, then a cost function with a single line section is assumed (Figure 2.6c). Similar to the case with only "down" flexibility, the minimum cost (zero) is where the power injection results in the target operating point being achieved, in this case when the power injection is zero. The maximum cost is when the power injection is equal to the "up" flexibility, P_b^+. The gradient of the line section is $+^1/_2$.

- If both "down" and "up" flexibility are available, then a cost function with two line sections is assumed (Figure 2.6d). The cost function contains the "down" flexibility curve (as per Figure 2.6b) followed by the "up" flexibility curve (as per Figure 2.6c), so



(a) $P_b^- = 0$ and $P_b^+ = 0$ (b) $P_b^- > 0$ and $P_b^+ = 0$

(c) $P_b^- = 0$ and $P_b^+ > 0$ (d) $P_b^- > 0$ and $P_b^+ > 0$

Fig. 2.6 Cost curves for generators within PFM-OPF

that the minimum cost (zero) is where the power injection equals the target operating point. This cost function ensures that deviations away from the target operating point are penalised by the same amount in either the “down” or “up” direction.

Within the OPF problem, the real power constraints for each of the aggregated generators are set to $[0, P_b^- + P_b^+]$, while reactive power is constrained to zero (except for the generator on the slack bus). To account for generators that are not operating at their minimum outputs, the difference between the present output P_b and the minimum output P_b^{min} is added to the fixed demand at the relevant bus.

The OPF formulation includes voltage constraints for each bus. However, these are not relevant for power flow management so are set to a wide range ($\pm 0.50\,\mathrm{pu}$) so that voltage is unlikely to become a binding constraint. For the slack bus a much narrower voltage range of $\pm 0.01\,\mathrm{pu}$ is used.

The OPF solution will indicate the real power flexibility usage per bus, so it is necessary to disaggregate these results to the individual generators at each bus. This is achieved by calculating the proportion of flexibility used ($P_b^\star / (P_b^- + P_b^+)$, where $P_b^\star$ is the power injection at the bus from the OPF solution) and applying this to each generator at the bus.

2.3.3 PFSF-Egal

This algorithm is based on the PFSF-based “egalitarian” algorithm of Jupe et al. [37, 40, 41]. It calculates an “egalitarian” proportional signal of present real power output, Φ, by which all generators are curtailed.

The PFSFs relate changes in generator real power injections P_g to changes in branch real power flows P_l: $\partial P_l/\partial P_g$. The PFSFs used within PFSF-Egal are calculated offline by executing a load flow on a network model, retrieving the Jacobian matrix, and deriving the sensitivity factors from its inverse. PFSFs are linearisations of the non-linear power flow equations around a particular operating point, so are only an approximation of the actual change in power flows. It is assumed that any changes of injections are compensated at the slack bus, maintaining overall power balance.

Although source code from Jupe et al. was available to the author, this algorithm (and the next, PFSF-TMA) were re-implemented from scratch so that they could be integrated with the test environment described in Chapter 3.

Figure 2.7 shows the flowchart of the algorithm’s process. This begins by PFSF-Egal reading in measurements from the network under control and using these to update an internal network model. The main loop of the algorithm then starts, which iterates until an exit condition is satisfied or an iteration limit is reached. In this implementation the limit is 10 iterations, in order to restrict the algorithms to executing in a reasonable time.

Each iteration of the main loop begins by retrieving branch loadings (from the internal network model) and selecting the branch with the largest loading on rating. If this branch is not overloaded, then no other branches are overloaded, so the algorithm can exit – applying any curtailments defined in previous iterations first. If the branch is overloaded, and the iteration limit has not been reached, the algorithm continues.

The next step is to calculate the required change in real power along the branch $\triangle P_l$ to bring the branch loading within rating:

$$\triangle P_l = \begin{cases} P_l^{\max} - P_l, & \text{if } P_l \geq 0 \\ -P_l^{\max} - P_l, & \text{otherwise} \end{cases} \tag{2.1}$$

In (2.1), the maximum real power $P_l^{\max} = \sqrt{(S_l^{\text{rating}})^2 - Q_l^2}$, Q_l is the reactive power flow along the branch, and S_l^{rating} is the branch thermal rating (adjusted to 99% to account for the PFSFs being approximations). The relationship between these variables is illustrated in Figure 2.8.

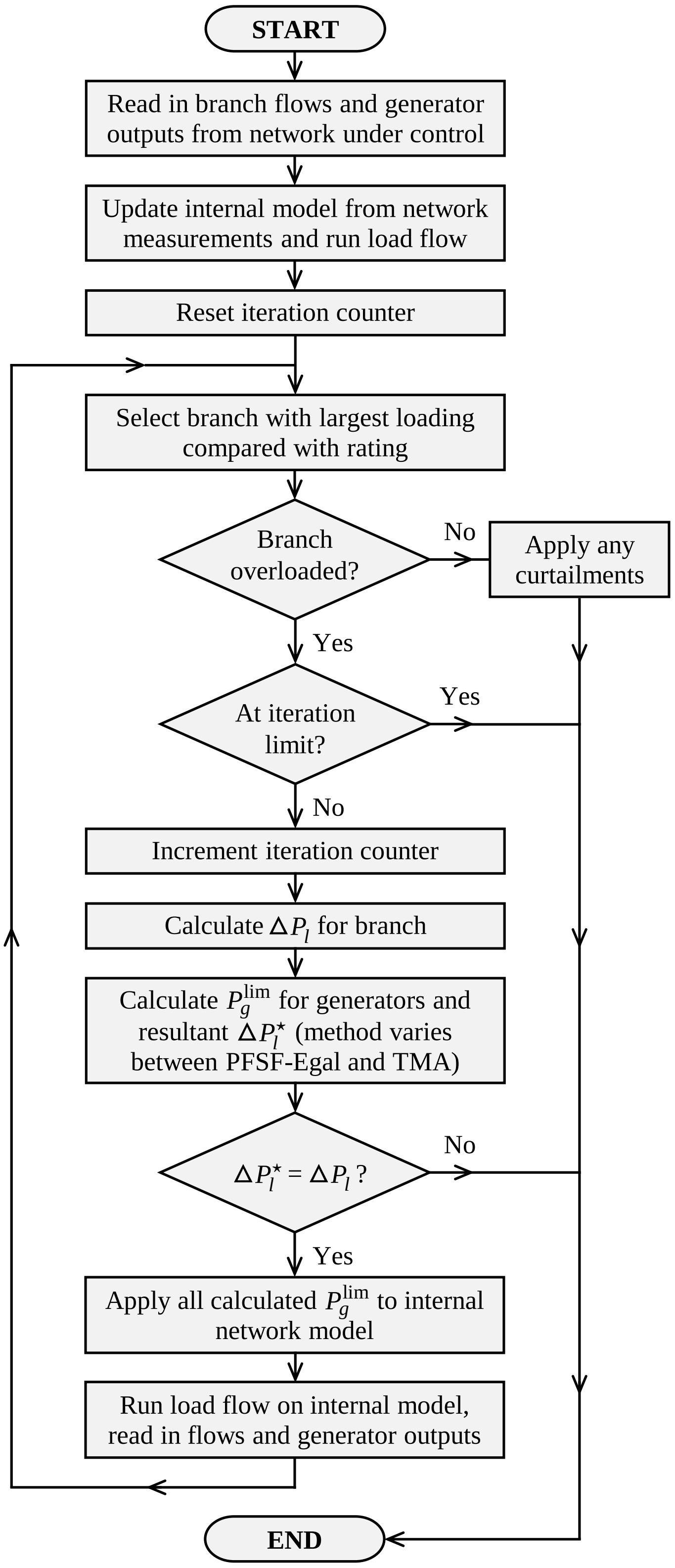

START
Read in branch flows and generator outputs from network under control
Update internal model from network measurements and run load flow
Reset iteration counter
Select branch with largest loading compared with rating
Branch overloaded?
No
Apply any curtailments
Yes
At iteration limit?
Yes
No
Increment iteration counter
Calculate ΔP_l for branch
Calculate P_g^{lim} for generators and resultant $\Delta P_l^\star$ (method varies between PFSF-Egal and TMA)
$\Delta P_l^\star = \Delta P_l$?
No
Yes
Apply all calculated P_g^{lim} to internal network model
Run load flow on internal model, read in flows and generator outputs
END

Fig. 2.7 Flowchart for the processes of the PFSF-Egal and PFSF-TMA algorithms

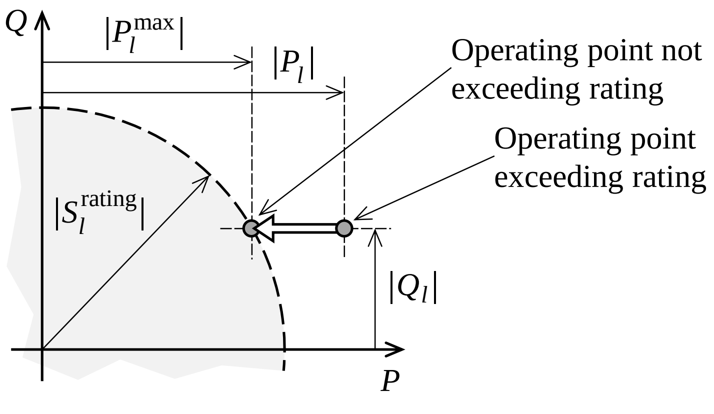

Q
$|P_l^{\max}|$
$|P_l|$
Operating point not exceeding rating
Operating point exceeding rating
$|S_l^{\text{rating}}|$
$|Q_l|$
P

Fig. 2.8 Derivation of $P_l^{\max}$ within the PFSF-LP algorithm

The output limits $P_g^{\lim}$ for each generator in the set of generators G are derived from $\triangle P_l$ and Φ:

$$\Phi = \frac{\triangle P_l}{\sum_{g\in G} -(\partial P_l/\partial P_g)P_g} \tag{2.2}$$

$$P_g^{\lim} = \begin{cases} (1-\Phi)P_g, & \text{if } 0 \leq \Phi \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

The effect of the generator output limits are calculated, for comparison with $\triangle P_l$:

$$\triangle P_l^\star = \sum_{g\in G} (\partial P_l/\partial P_g)(P_g^{\lim} - P_g) \tag{2.4}$$

If $\triangle P_l^\star \neq \triangle P_l$, then the generators cannot alleviate the overload and the algorithm exits early. Otherwise, the output limits are applied to the generators within the internal network model. A load flow is then executed to provide a starting point for the next iteration.

2.3.4 PFSF-TMA

This algorithm is based on the PFSF-based "technically most appropriate" (TMA) algorithm of Jupe et al. [37, 40, 41]. The implementation is almost exactly the same as the PFSF-Egal algorithm (the flowchart of Figure 2.7 describes both algorithms), and only differs in the calculation of the generator output limits P_g^{lim} and the change in branch real power flow they achieve $\triangle P_l^\star$.

Rather than curtailing all generators by the same proportion, PFSF-TMA ranks generators in an order determined by their ability to change the power flow along the overloaded branch, assessed by the magnitude and direction of their PFSF relating to the overloaded branch. The algorithm then calculates P_g^{lim} for each generator in order, until $\triangle P_l^\star = \triangle P_l$ or all generators having PFSFs that can positively influence the overload are fully curtailed.

P_g^{lim} is calculated from:

$$P_g^{\text{lim}} = \max(0, P_g + \frac{\triangle P_l - \triangle P_l^\star}{(\partial P_l / \partial P_g)}) \tag{2.5}$$

Whereas $\triangle P_l^\star$ is set initially to zero and then updated for each P_g^{lim} calculated:

$$\triangle P_l^\star := \triangle P_l^\star + (\partial P_l / \partial P_g)(P_g^{\text{lim}} - P_g) \tag{2.6}$$

2.3.5 PFSF-LP

PFSF-LP is based on the sensitivity factor-based method of Skokljev et al. [44] that formulates power flow management as a linear program. Figure 2.9 is a flowchart of the process followed by the algorithm, which is similar to the process of PFM-OPF, and can be summarised as: 1) checking if there are overloaded branches or curtailed generators in the network, 2) formulating and solving the LP problem, if there are overloads or curtailed generators, and 3) taking the LP solution and applying it to the generators in the network.

Another similarity to the PFM-OPF algorithm is that PFSF-LP aggregates generators together at each bus. This reduces the number of variables in the problem, speeding up solution time, as well as allowing PFSF-LP to use common code with PFM-OPF, reducing the algorithm development time.

PFSF-LP considers increases ($\triangle P_b^+$) and decreases ($\triangle P_b^-$) in real power injection from the generators, around a target operating point. How these variables relate to the real power flexibility variables previously described for PFM-OPF (see Figure 2.5) is shown in Figure 2.10.

The objective of the LP is to minimise deviation away from the target operating points for each bus b from a set of buses B:

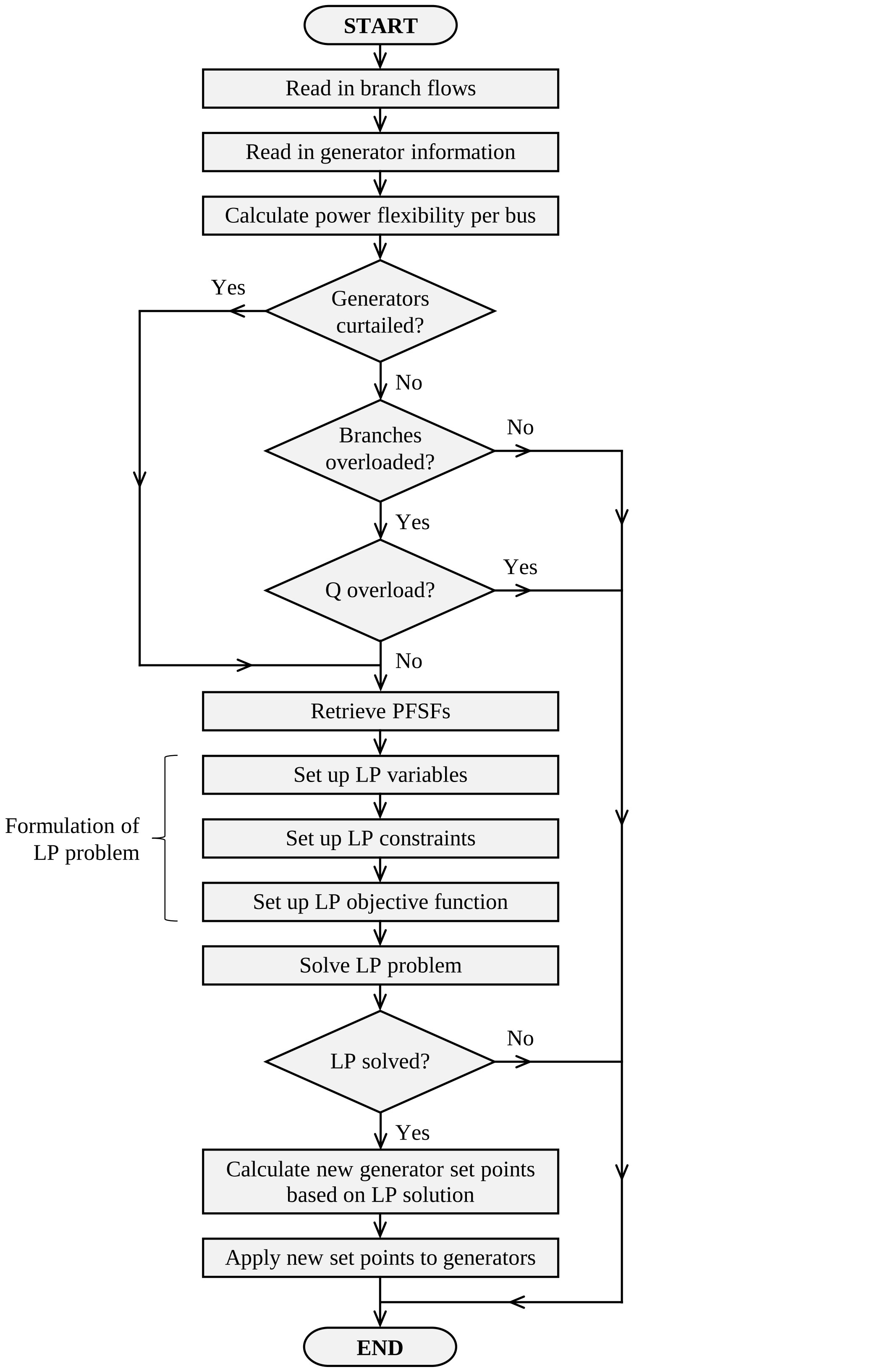

START
Read in branch flows
Read in generator information
Calculate power flexibility per bus
Generators curtailed?
Yes
No
Branches overloaded?
No
Yes
Q overload?
Yes
No
Retrieve PFSFs
Set up LP variables
Set up LP constraints
Set up LP objective function
Formulation of LP problem
Solve LP problem
LP solved?
No
Yes
Calculate new generator set points based on LP solution
Apply new set points to generators
END

Fig. 2.9 Flowchart for the processes of the PFSF-LP algorithm

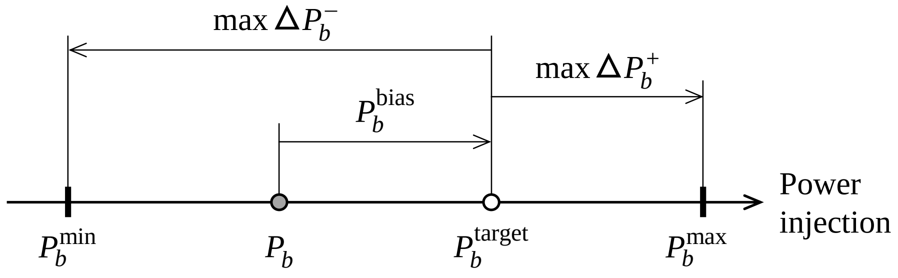


Fig. 2.10 Relationship of real power variables used in the PFSF-LP algorithm

$$\min \sum_{b \in B} \triangle P_b^+ + \triangle P_b^- \tag{2.7}$$

This objective is subject to constraints relating to: 1) branch maximum real power flows, 2) flexibility limits, and 3) non-negativity of decision variables. For the first constraint – branch maximum real power flows – two inequality constraints are created for each branch, from the set of branches L, that relate to flows in either direction.

For flows in the positive direction, the following constraint is defined:

$$\sum_{b \in B} [(\partial P_l / \partial P_b)(P_b^{\text{bias}} + \triangle P_b^+ - \triangle P_b^-)] + P_l \leq +P_l^{\max} \; \forall \; l \in L \tag{2.8}$$

This relates the change in real power injections ($\triangle P_b^+$ and $\triangle P_b^-$, with the initial power injections accounted for in the P_b^{bias} term) and initial branch real power loading P_l to the maximum real power flow $P_l^{\max}$. A similar constraint is defined in the reverse flow direction:

$$\sum_{b \in B} [(\partial P_l / \partial P_b)(P_b^{\text{bias}} + \triangle P_b^+ - \triangle P_b^-)] + P_l \geq -P_l^{\max} \; \forall \; l \in L \tag{2.9}$$

Constraints on the changes in real power injections ensure that the flexibility limits are not exceeded:

$$\triangle P_b^+ \leq P_b^{\max} - P_b^{\text{target}} \; \forall \; b \in B \tag{2.10}$$

$$\triangle P_b^- \leq P_b^{\text{target}} - P_b^{\text{min}} \; \forall \; b \in B \tag{2.11}$$

Furthermore, there are additional non-negativity constraints imposed on the changes in real power injections:

$$\triangle P_b^+ \geq 0 \; \forall \; b \in B \tag{2.12}$$

$$\triangle P_b^- \geq 0 \; \forall \; b \in B \tag{2.13}$$

The method for disaggregating the per-bus injection changes to individual generators is the same as that described for PFM-OPF.

2.4 Conclusions

This chapter has outlined the subject of control within power systems and introduced the particular power systems control task that this work concentrates on. This task is power flow management, which shares a number of features with other power systems control tasks and is therefore not unrepresentative of power systems control in general.

Existing algorithms for power flow management have been surveyed, and a diverse set of these algorithms have been identified and implemented for testing in the subsequent chapters of this work. These algorithms consider the same control actions – changes in generator real power output – and can be applied to both radial and meshed network topologies. However, the poor scalability properties of one algorithm (PFM-CSP) does limit the algorithms' application to networks with only a small number of controllable generators.

The literature survey also identified that existing comparative studies of power flow management algorithms are limited, in terms of the numbers of studies available, the number of algorithms tested, and the range of networks and networks states that the algorithms are tested on. Therefore, there is a need to perform wider-ranging comparative studies of power flow management algorithms, which is something that the diverse set of implemented power flow management algorithm enables when coupled with the automated test environment described in following chapter, Chapter 3.

Chapter 3

Control algorithm testing environment

This chapter introduces the software system that was developed in support of research objective 1, which was used for testing the performance of power system control algorithms applied to different scenarios within a variety of power system models.

In Section 3.1 the requirements defined for the system are outlined, followed in Section 3.2 by an overview of the testing environment that was developed to fulfil the requirements. The subsequent sections then describe the main components of the testing environment, including the power system modelling capabilities (Section 3.3), the communication system modelling (Section 3.4), and the modelling of control algorithms (Section 3.5). How the various components work together to execute tests is described in Section 3.6 for single tests, and Section 3.7 for multiple tests. Finally, in Section 3.8 the test environment is evaluated against the requirements, and the chapter concludes with Section 3.9.

3.1 Requirements for testing environment

Before developing the testing environment, requirements for its functionality were defined based on the research objectives and with consideration of experimental design [52, 53] – as the testing of power systems control algorithms can be treated as an *experiment* – and power system modelling [54, 55]. These requirements are listed below:

1. Requirements for tests:

   (a) Repeatability: each test shall be deterministic and it must be possible to prescribe and record all factors that can affect the outcome of a test, so that any test can be repeated and the same outcome obtained.

   (b) Observability: it must be possible to observe and record any parameter relevant to a test, in order to gather results for analysis and also to allow tests to be repeated if necessary with the same conditions.
   (c) Flexibility: as required by research objective 1(c), it must be possible to run tests on a variety of networks; furthermore, as required by research objective 1(d) it must be possible to simulate different operating scenarios within each network including conditions that change over time.
   (d) Automation: the environment must be capable of running multiple tests and recording their results, without user intervention, in order to enable the large scale testing envisaged by research objective 1(d).
   (e) Reliability: the environment must be tolerant of failed tests. If a test fails for whatever reason, then the failure should be recorded. Furthermore, the failure of one test should not influence the outcome of any other test.
   (f) Speed: the environment should allow testing to be completed as quickly as possible within the resources available. However, the environment is not required to operate in real-time and match the "wall clock" time like a real-time digital simulator (RTDS) [56].

2. Requirements for power system modelling:
   (a) The environment must be capable of running full AC load flow.
   (b) The environment must allow for the simulation of real-time network operation over a time period, comprising a series of steady states separated by a constant time interval.
   (c) The environment must allow for representations of standard power system components and for components that could be expected in future power systems, in order to allow existing and emerging control algorithms to be tested.
   (d) The environment must allow time-based characteristics of components to be simulated, in order to reflect real-world operation.

3. Requirements for modelling control algorithms and communications system:
   (a) The environment must support the simulation of multiple control algorithms, in order to fulfil research objective 1(b).

(b) It must be possible to specify a set of components that each control algorithm can control. Any control action request that falls outside an algorithm's specified set of controllable devices shall be ignored, so that the control scope of individual algorithms can be restricted to particular zones of operation [2, 57].

(c) Algorithms must only be able to modify the operational parameters of components. Modification of fixed parameters – such as circuit impedances – or internal parameters – such as the state of charge of a battery storage unit – shall not be allowed, in order to reflect real-world operation.

(d) Any interaction that an algorithm has with the power system model must be observable and should be able to be captured for inspection and analysis, such as for debugging when prototyping a control algorithm.

(e) It should be possible for the environment to be extended to include modelling of communications system characteristics, such as delays and data drop-outs, which could be used as factors to influence the behaviour of control algorithms.

(f) Any information that is preserved by an algorithm between executions must be available for inspection. Furthermore, it should be possible to capture and restore this preserved information, so that tests can be restarted at any point without invalidating the requirement for repeatability.

3.2 Overview of testing environment

A testing environment was developed using the Python programming language in order to meet the requirements listed in Section 3.1. Python has a number of features that made it suited to the creation of the testing environment. Firstly, it is an object-oriented language, which was useful for representing different elements of a modelled system, and other aspects of the testing environment needed to execute tests. Secondly, there is an extensive library of free add-on packages which allowed development time to be reduced as certain features (such as database interfaces) did not need to be developed from scratch. Thirdly, it is an interpreted language, which was beneficial when prototyping and debugging, although this can be at the expense of slower execution compared with a compiled language such as C.

Figure 3.1 provides an overview of the architecture of the testing environment. The components within this architecture can be split in to two categories: firstly, at the top of the figure are *in-memory* software objects that exist at run time; secondly, at the bottom are *stored* objects that reside within the file system of the computer that hosts the environment.

The in-memory components can be further split down according to their function. On the left of Figure 3.1, the `DriverProcess` is used to administer series of tests, while on the right, the `TestProcess` objects are used to execute individual test runs. In between are two queues that are used to pass information between the objects. This structure allows the objects to operate in parallel on separate cores of the computer's processor, which presents a significant speed advantage when running multiple tests.

3.2.1 Main components

The main components of the testing environment are described below:

- `DriverProcess`: this object is responsible for administering a series of tests. It communicates with the runs database to determine what test runs need to be completed, and then adds these to the task queue. Any completed test runs are retrieved from the results queue, and the `DriverProcess` then records the results of these within the runs database and data archive file.

- `TestProcess`: these objects run individual tests. A `TestProcess` will begin by interrogating the task queue to obtain a test run that needs to be completed. Test runs are set up within the `TestCell` contained within the `TestProcess`, and then the test is executed and results gathered. The results of a test run are then placed in to the results queue, ready for the `DriverProcess` to collect them. The `TestProcess` essentially acts as a "wrapper" around a `TestCell` to enable parallel processing.

- Task queue: this contains details of test runs that need to be completed. Each test run in the queue is specified in terms of:
  - The control algorithm that should be tested.
  - The power system that is to be simulated.
  - The scenario that is to be applied to the power system, which defines a period of time to be studied including the initial conditions and any changes that occur to the power system over time – such as changes of load or generation.

- Results queue: this contains results of test runs that have been completed by the `TestProcess` objects. Each set of results includes any control actions taken by the control algorithm and measurements from the studied power system for each time step within the scenario. Furthermore, the test run to which the result belongs is identified (in particular, the power system, scenario and algorithm tested).

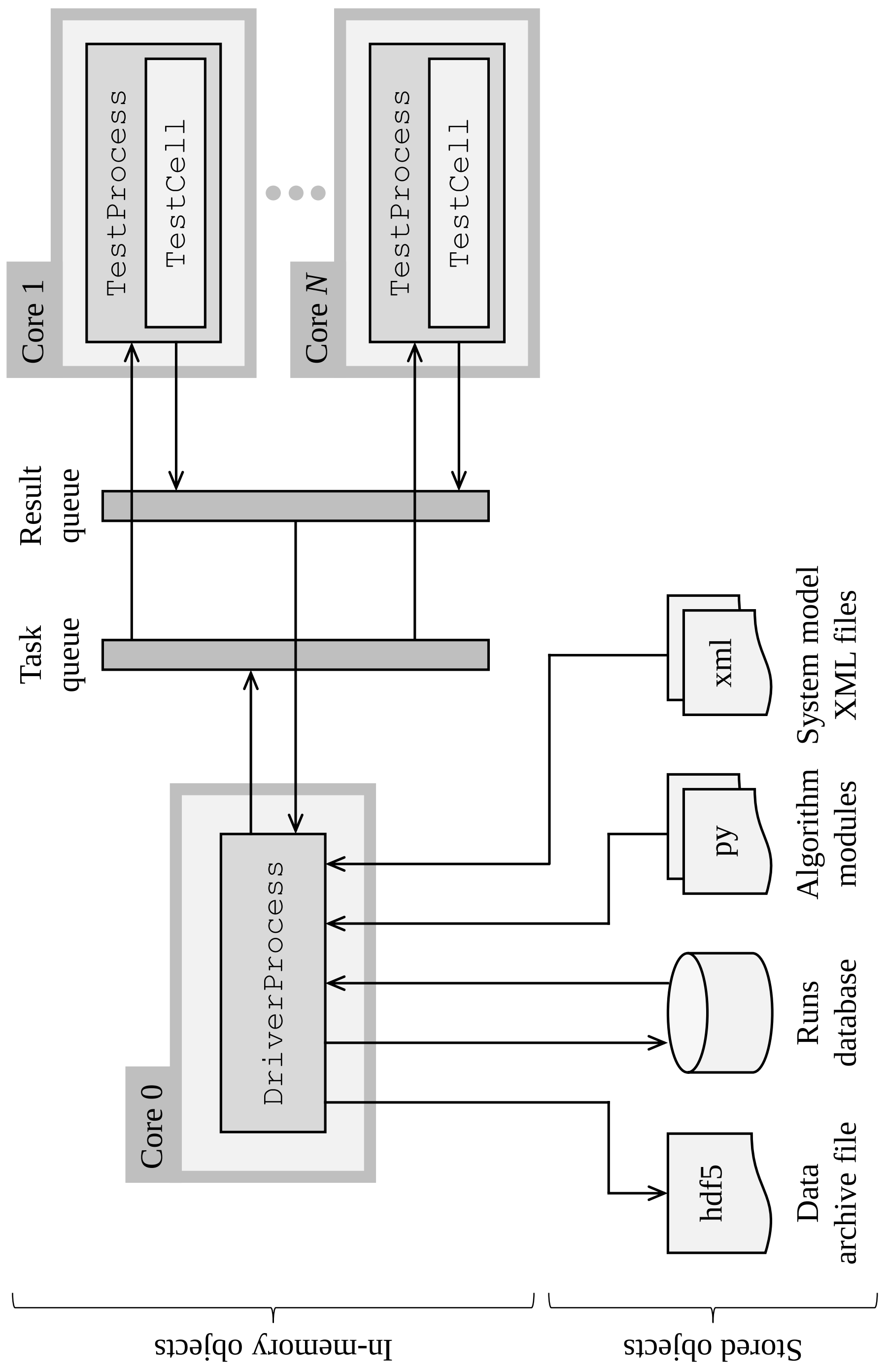


Fig. 3.1 Overview of the architecture of the testing environment

- Data archive file: this is used for bulk storage of test run results. The HDF5 (Hierarchical Data Format, version 5) format allows for structured storage of large datasets, which was ideal for the purposes of the testing environment as running several thousands tests – a quantity required by the testing described in Chapter 4 – can result in Gigabytes of data.

- Runs database: this contains records of each power system, scenario (initial conditions and changes to these) and algorithm, along with their combinations, which define the test runs. The runs database stores each test run that needs to be completed and each test run that has been completed; however, only a minimal amount of data about completed test runs is stored, with the majority of results data being stored in a data archive file.

- Algorithm modules: these are Python files that define each control algorithm.

- System model XML files: these are used to store representations of the power systems to be studied, encoded using an XML format.

There is an additional in-memory object, the `StatsProcess`, that has been omitted from Figure 3.1 to aid clarity. This object polls the `DriverProcess` and each `TestProcess`, in order to provide status information to the user of the testing environment, such as the size of the queues and the number of test runs completed.

3.2.2 Components within a `TestCell`

The `TestCell` objects shown in Figure 3.1 contain objects within them that are used to model the control algorithms being tested, the power system that the algorithms are being tested on, and a communications system that sits between the algorithms and the power system. The objects within a `TestCell` and their relationships to each other are shown in Figure 3.2. In the figure, the components are arranged vertically in three layers according to what they model:

- Power system: this is modelled by the three objects in the bottom layer in the figure, namely `SysModel`, `LFEngine`, and `PyPower`. The `SysModel` is a data structure representing the components and parameters of the power system being modelled; however, this is not an electrical model, so `PyPower` is used to perform load flow calculations using an electrical model generated on-the-fly from the `SysModel`. The `LFEngine` acts as an interface to `PyPower`, updating the `PyPower` electrical model from the `SysModel` data and storing the results of load flow calculations. `LFEngine` provides a layer of

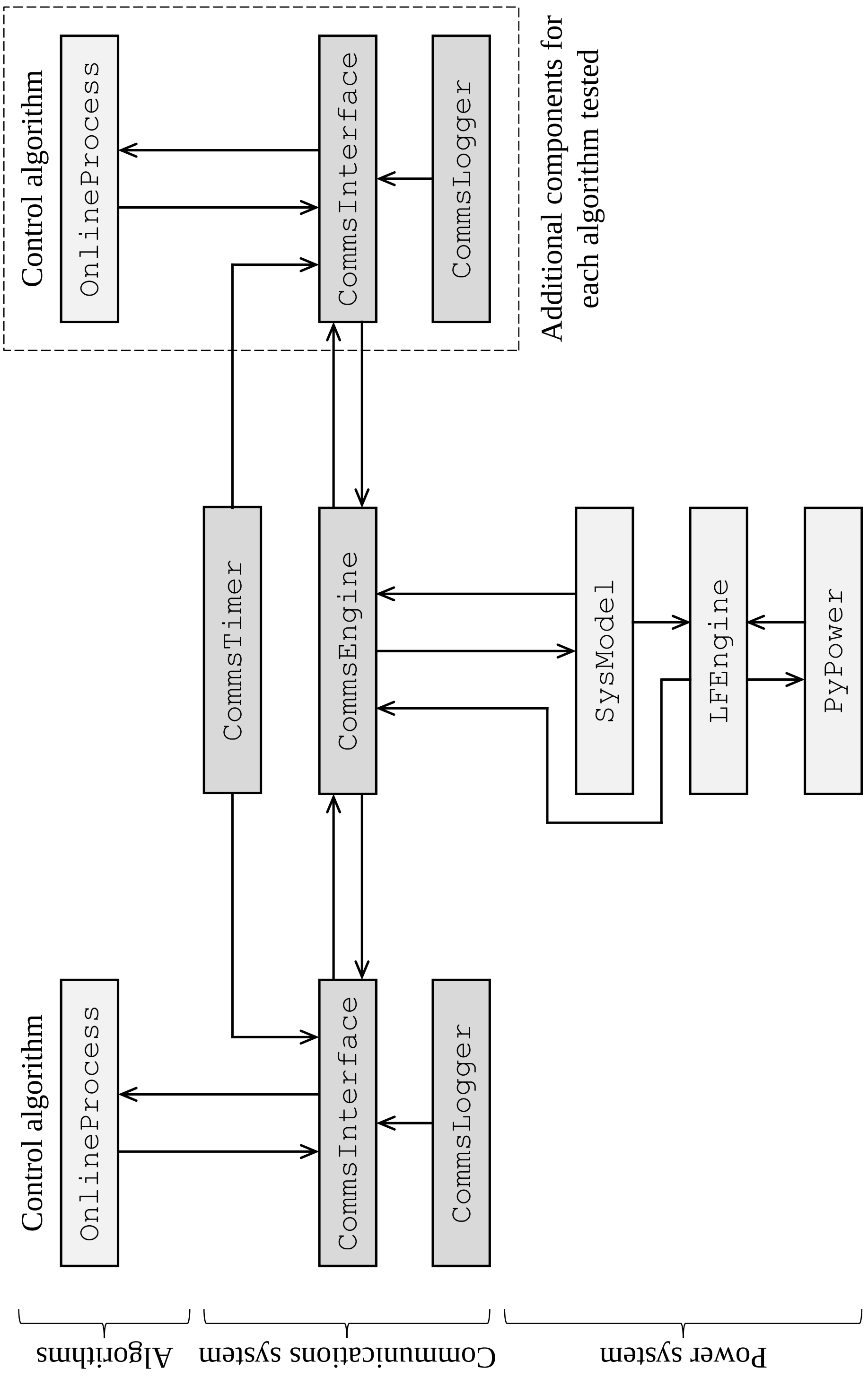


Fig. 3.2 Overview of the modelling components within a `TestCell`

abstraction above `PyPower` so a different load flow calculation engine could be used without needing to modify any of the objects in the testing environment aside from `LFEngine`. Modelling of power systems is described in further detail in Section 3.3.

- Communications system: the `CommsEngine` is the main object in this layer and handles all communications between the algorithms and the power system modelling objects. Algorithms can either request data or provide control actions, which are routed to the appropriate objects by the `CommsEngine`. Data requests are either sent to the `SysModel`, in the case of component parameter data, or sent to the `LFEngine`, in the case of load flow results. Control actions are passed on to the `SysModel`, which modifies the parameters of the power system components that the control actions related to. Each control algorithm has a dedicated `CommsInterface`, which simply serves to forward communications on to the `CommsEngine` for processing. The `CommsLogger` objects record the communications traffic from each algorithm, while the `CommsTimer` is used to set the current time step being studied. Modelling of the communications system is described in further detail in Section 3.4.

- Algorithms: each `OnlineProcess` object represents a single control algorithm, which determine control actions to be applied to the modelled power system based on data obtained from the power system system model via the communications system. Although Figure 3.2 shows multiple algorithms, only a single algorithm at a time was tested during all the tests described in this work. Modelling of the control algorithms is described in further detail in Section 3.5.

An overview of the process used by these objects to execute single and multiple algorithm tests is described in Sections 3.6 and 3.7.

3.3 Power system modelling

Within the testing environment, power systems are represented as data structures (`SysModel` objects) that contain information about every component within a particular system. Which types of power system component are allowed and what parameters are valid for each component type is defined within a data model.

3.3.1 Component parameters

Each power system component has a number of parameters that determine how it is modelled, as defined in the data model. The testing environment has several types of parameter, for different purposes, as described below:

- *ID*: an identifier that is unique for each component of a particular type. Each component can only have one ID-type parameter.
- *Static*: properties that do not change over time, such as busbar nominal voltages and generator output limits.
- *Dynamic*: properties that change over time but are not controllable by the control algorithms, such as demands or circuit ratings.
- *Control*: properties that change over time and can be controlled by the control algorithms, such as the output of a generator.
- *Mode*: a property that indicates a particular mode of operation, taken from within a given set of modes (defined by a *mode list*-type parameter). Control algorithms are allowed to change component modes.
- *Mode list*: a list of available modes.
- *Internal*: a variable required for internal calculation, such as the last power setting of a storage unit, which is used to calculate state of charge.
- *Measurement*: these are read-only parameters determined by load flow calculations, such as the real and reactive power flows along a circuit. The values of measurement parameters are stored within `LFEngine` objects rather than within a `SysModel`.

3.3.2 Power system component modelling capabilities

The different types of power system component that the testing environment can model are described below. A summary of each component type is presented, including their name (such as `bus`), an overview of their main parameters and some essential features of their operation. Appendix A provides a full listing of the parameters for each component type.

Busbars (`bus`)

Each busbar has an assigned nominal voltage and has voltage magnitude and angle available as measurements. The bus type for load flow ($V\theta$, PQ or PV) is determined automatically by the types of components that are connected to the bus. The number of components that can be connected to each bus is unlimited.

Circuits (`cct`)

Circuits can represent either overhead lines or cables, depending on the parameters entered (resistance, reactance and charging). Each circuit has a static power rating that can be overridden by a dynamic rating value. The real and reactive power flows at each end of a circuit are available as measurements.

Transformers (`tx`)

These are modelled in a similar way to circuits, but with the addition of having a variable voltage ratio and phase shift. The voltage ratio and phase shift can be adjusted in discrete tapping steps. The transformer representation can limit the number of voltage or phase shifting tap steps taken within a time step, simulating lock-out delays. The real and reactive power flows at each end of a transformer are available as measurements.

DC interconnectors (`dcx`)

These represent point-to-point DC links, with four-quadrant AC/DC converters on both ends. The real power transfer across the link is defined as a controllable parameter and is subject to losses, which are proportional to the power transfer. The reactive power at each end is independently controllable, with each end being in one of three control modes:

1. `Q`: reactive power import or export is set directly
2. `PF`: a fixed power factor is assumed, so reactive power import or export is proportional to the real power import or export at the end in this control mode
3. `V`: voltage-control mode, with reactive power import or export modulated automatically to control the voltage at the connecting bus.

Changes in real power transfer and reactive power at each end are subject to ramp rate limits. At for circuit (`cct`) components, `dcx` components have a static power rating that can be overridden by a dynamic rating value. The real and reactive power flows at each end of the link are available as measurements.

In the PyPower electrical model, each end of a dcx is represented in the same way as a gen component.

Circuit breakers (brk)

Each circuit breaker can be associated any number of circuits, which are taken out of service if the circuit breaker is set to “open”.

Loads (ld)

Loads represent aggregated demand, such as at a metering point, and include a block of demand side response (DSR) that can be activated by the control algorithms. The real and reactive power of both the demand and the DSR can be varied over throughout a simulation.

Generators (gen)

These components have the flexibility to represent a variety of different generation units, including fully dispatchable plant, plant with limited dispatchability – such as a wind farm that can whose output can be curtailed – and non-dispatchable plant – such as domestic solar photovoltaic (PV). The “prime mover” power of each generator can be time-varying, as is the amount that this can be adjusted “up” or “down”. This allows plant with different dispatching abilities to be represented. For example, for a wind generator, the “prime mover” power would be the power available from the wind, the “up” adjustment of real power would be zero – as the power output cannot exceed the amount of power collected from the wind – while the “down” adjustment of real power can be equal to the current “prime mover” power – representing a wind generator that can be curtailed to zero real power output.

Control algorithms that control generators can apply one of two real power operating modes, which are illustrated in Figure 3.3 and explained below:

1. CAP: as shown in Figure 3.3a, the real power output matches the prime mover power within “cap” limits set by the control algorithm. If the prime mover power is greater than the upper cap limit, then the real power output is set to the greater of the upper cap limit value, or the prime mover power minus the down power adjustment value. Similarly, if the prime mover power is less than the lower cap limit, then the real power output is set to the lesser of the lower cap limit value, or the prime mover power plus the up power adjustment value.

2. SET: in this mode, shown in Figure 3.3b, the control algorithm sets a desired real power output value. This set point value will be used as the real power output of the generator

so long as it is within the bounds of the up and down adjustments around the current prime mover power. If the real power set point value is outside the adjustment bounds, the generator output is set to the value of the closet bound.

The reactive power output of each generator can be set to one of three operating modes previously defined for DC interconnectors (`dcx`, namely `Q`, `PF` or `V`. The reactive power mode is set independently from the real power mode, and separate real and reactive power ramp rates can be set.

Storage devices (`sto`)

These represent storage devices that have a power electronics interface. They operate in almost exactly the same way as `gen` components, with the same operating modes and ramp rate limitations, except that the "prime mover" power is further limited by the state of charge of the storage. The storage capacity is fixed, and a bi-directional conversion efficiency can be specified. Furthermore, the charge and discharge rates can be set to different values.

Capacitor/reactor banks (`qbk`)

These consist of fixed steps of reactance, either inductive or capacitative. A control algorithm can activate any number of steps, though the number of steps that can be changed at one time can be restricted, in the same way that tap operations for a transformer can be time limited.

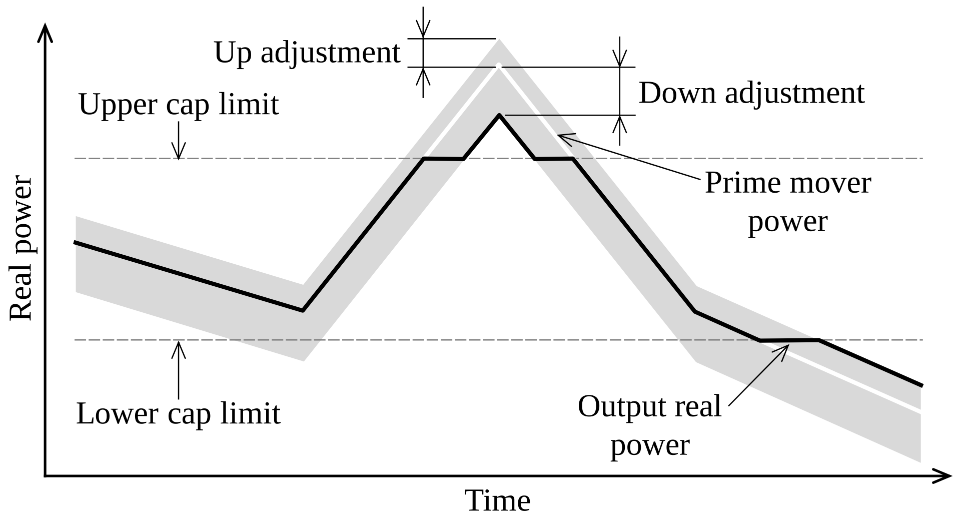
Slacks (`slk`)

These component transform the busbar they are connected to into a slack bus. Their voltage magnitude and angle are adjustable parameters, whilst the amount real and reactive power they infeed are available as measurements.

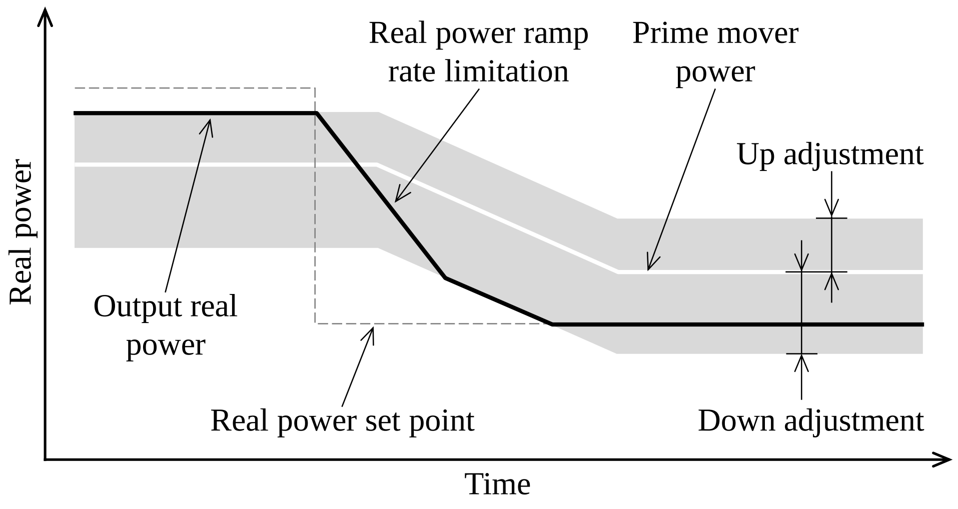
Zones (`zn`)

Zones are virtual components that define which components each control algorithm may act upon. Each zone contains a list of busbars that are considered to be within the zone. Each busbar can only be assigned to a single zone so that no zones overlap, although it is not necessary for zones to be contiguous. Control actions determined by a control algorithm are only applied to the components connected at the busbars within that algorithm's zone.

Within each power system model a "slack" zone is defined containing the slack bus. This zone is in addition to the zones required for the control algorithms, and ensures the algorithms are not allowed to control the slack bus.



(a) `CAP` mode



(b) `SET` mode

Fig. 3.3 Examples of generator real power operation modes

3.3.3 Load flow

The data model described above is used as a template for the data structure within the `SysModel` objects, which contain the information necessary to represent the components of a power system and their parameters. This information encompasses the *independent* state of the power system, but does not include the *dependent* state of the system, such as the voltages at each busbar and the currents flowing through each circuit.

In order to derive the dependent state of the power system it is necessary to execute a load flow calculation. However, the `SysModel` does not support this calculation directly. Instead, the `LFEngine` object is called, which creates a lower-level electrical model based on the `SysModel` and uses a separate solver (`PyPower`) to execute a load flow using the electrical model. The resultant dependent state is stored within the `LFEngine` where it can be accessed as measurements via the `CommsEngine`.

The `PyPower` [50] solver is used for the load flow calculation as: 1) it is a direct port of MATPOWER [51], so benefits from MATPOWER's active development and established support within the research community; 2) it allows for full AC load flow to be calculated using a Newton-Raphson algorithm; 3) `PyPower` is Python-based so allows for easy integration with the other objects within the testing environment.

The electrical model that is provided as input to `PyPower` is in the form of data matrices:

- Bus matrix: each busbar in the `SysModel` is represented as a row within this matrix. An aggregate real and reactive power demand for each bus is calculated by summing the contributions from all `ld`, `gen` and `dcx` components connected to the busbar. Any `gen` or `dcx` components in `V` control mode are ignored, as their reactive power contribution can only be determined during the load flow calculation. A similar aggregation process is used to sum the contributions of all the `qbk` components connected to a bus.

- Branch matrix: this contains impedance, charging, rating, voltage ratio, phase shift and status data for each circuit (`cct`) and transformer (`tx`).

- Generator matrix: each row in this matrix contains data for either a generator (`gen`) component or for one of the ends of each DC interconnector (`dcx`). Additional rows are included for each slack within the system. The data in each row is updated from the relevant parameters within the `SysModel`; for example, reactive power limits within the generator matrix are set based on the last reactive power setting and ramp rate limits given in the `SysModel` for each relevant component. If a `gen` or `dcx` end is not in `V` control mode, the corresponding row in the generator matrix is disabled, and the component's real and reactive power contribution is added to the total real and reactive power for its connecting bus, as recorded in the bus matrix.

- Generator cost matrix: this contains cost curves for each row of the generator matrix. Although cost data is not needed in the load flow calculation, the solver requires the matrix to be provided so it is initialised to the correct dimensions but left empty.

3.4 Communications system modelling

The testing environment includes a separation between the power system model and the control algorithms that can act on the components within the power system, in which sits a layer of objects that represent a communications system.

3.4.1 Functions

The primary functions of the communications system in the testing environment are threefold:

1. To provide an interface for algorithms to access data about the power system model, including component parameters and measurements. As would be expected in a real system, the access provided does not include parameters internal to the components.
2. To restrict which components the algorithms are able to control, if needed for a test.
3. To log the communications traffic associated with each algorithm, allowing for inspection and debugging.

The version of the testing environment used for this work does not include modelling of communications system characteristics such as delays and failures, and does not restrict the component parameters that are visible to the algorithms (aside from parameters internal to the components). However, the architecture of the testing environment would allow these characteristics to be added without making it necessary to modify the control algorithms or the power system modelling objects.

3.4.2 Communications system objects

The communications system layer is implemented with the following objects, which are shown schematically in the middle layer in Figure 3.2:

- `CommsEngine`: this is the main component of the communications system, and there is only ever one `CommsEngine` object per `SysModel`. The `CommsEngine` has methods that can be called to retrieve and to set values within the testing environment, so long as the requests are valid. Values are retrieved from either the `SysModel` for component parameters or from the `LFEngine` for measurements. Only values relating to component parameters in the `SysModel` can be set.

- `CommsInterface`: each control algorithm has an associated `CommsInterface` object, which sits between the algorithm and the `CommsEngine`. This arrangement limits the interfaces that are exposed to each algorithm and allows the algorithm's action to be restricted to within its zone only, as each `CommsInterface` is associated with a particular zone. Additionally, any communications across a `CommsInterface` triggers the associated `CommsLogger` object.

- `CommsLogger`: these objects are used to store the communications traffic between a control algorithm and the power system it is controlling. There is one `CommsLogger` per `CommsInterface`, and, therefore, per control algorithm.

- `CommsTimer`: when a number of sequential states are modelled, this object is used to track which state has been reached.

3.5 Control algorithm modelling

Each control algorithm that is tested within the testing environment exists as a Python module that must contain the following two classes: an `OnlineProcess`, which is a runtime representation of an algorithm, and an `OfflineProcess`, which is used to automatically configure the algorithm to be applied to a new power system model.

The `OnlineProcess` contains a number of "housekeeping" methods that are needed at runtime, such as to start, stop and reset an algorithm. There is also the vital `run` method that encapsulates the calculations and logic that define how each algorithm determines control actions. When the `run` method is called, the algorithm will use its associated `CommsInterface` to retrieve any power system parameters and measurements that it requires, use these to determine any control actions, and then communicate these actions back to the power system model via the `CommsInterface`.

The processes of some control algorithms rely on random number generators. However, this can lead to the algorithm producing different control actions when applied to exactly the same power system and state, contradicting the requirement for repeatability within the testing environment. However, if started with the same seed, a random number generator will output the same sequence of random numbers, and an algorithm that used the output of that generator would produce the same control actions. Therefore, each `OnlineProcess` must accept a pre-defined integer that is used as the seed for any random number generator within the algorithm. This removes any variance between tests due randomness within an algorithm's process.

In addition to a pre-set random seed, another aid to repeatability is that any information that is preserved between executions of an `OnlineProcess` is stored as an accessible Python data structure. This allows the preserved information to be interrogated, stored and reinstated, so that test runs can be restarted from the beginning or part-way through.

An algorithm's `OfflineProcess` is only used once per power system model, to calculate parameters for the `OnlineProcess` that will tailor the algorithm to a particular power system model. For example, the PFSF-Egal algorithm (see Section 2.3.3) has an `OfflineProcess` that calculates PFSFs for any power system model that it needs to be applied to.

3.6 Running single tests

A `TestCell` object is used to perform a single test of a control algorithm. As shown previously in Figure 3.2, the `TestCell` contains objects that model the power system, communications system and algorithms necessary for a test; however, the `TestCell` is responsible for executing a test and sequencing the modelling objects during test execution.

Before a test can begin, the `TestCell` object must be initialised and the parameters of the test set up, namely:

- The power system that the test will be performed on, supplied as a `SysModel` object
- The scenario to be tested, consisting of the initial system state and any changes to the parameters within the `SysModel` that are to be made over a number of time steps
- The algorithm to be tested, supplied as an `OnlineProcess` object
- The random seed to be used by the algorithm
- A list of parameters and measurements (load flow results) that should be captured from the system state at every time step.

The process that the `TestCell` follows to execute a test is illustrated in the flow chart of Figure 3.4. The first steps of this process set up the test, and begin by creating a copy of the `SysModel` for use during the test, so that the original `SysModel` is not altered by the testing process. The communications system objects (`CommsEngine`, `CommsInterface`, `CommsLogger` and `CommsTimer`) are then initialised and linked in to the other modelling objects (`SysModel`, `LFEngine` and the algorithm's `OnlineProcess`). The initial control

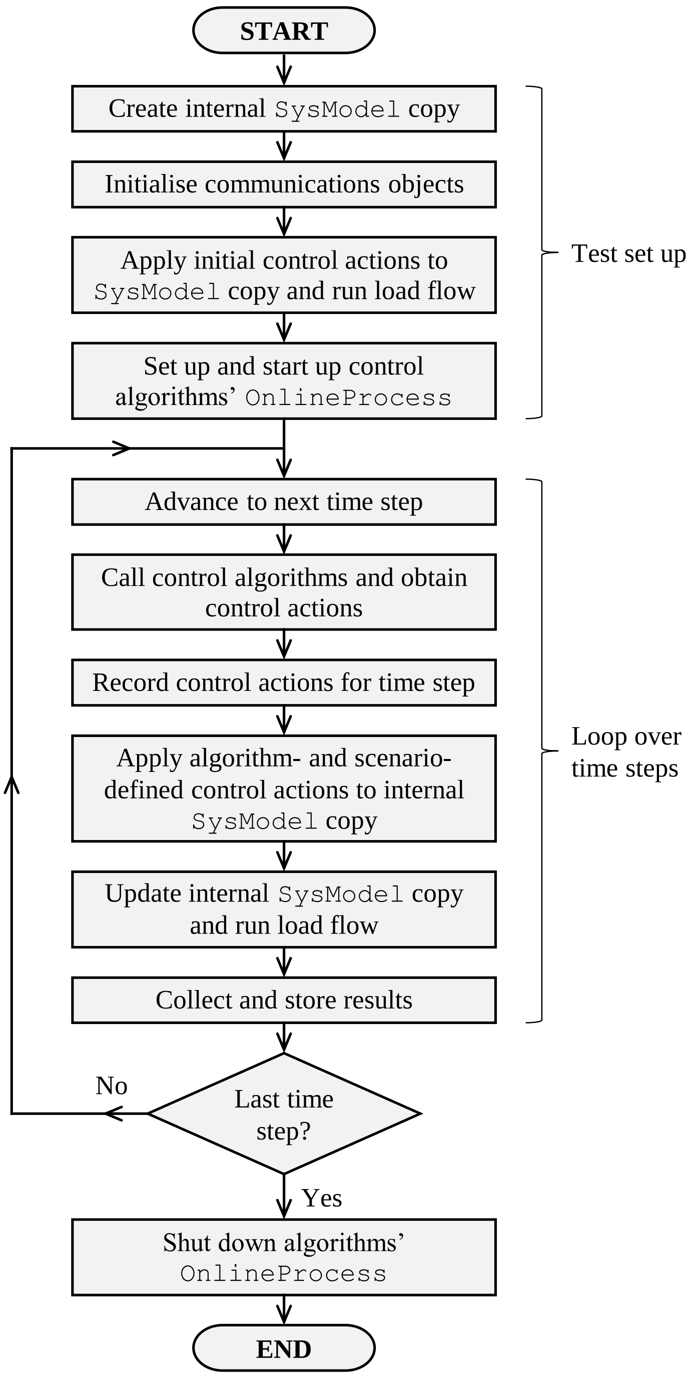

START
Create internal SysModel copy
Initialise communications objects
Apply initial control actions to SysModel copy and run load flow
Set up and start up control algorithms' OnlineProcess
Test set up
Advance to next time step
Call control algorithms and obtain control actions
Record control actions for time step
Apply algorithm- and scenario-defined control actions to internal SysModel copy
Update internal SysModel copy and run load flow
Collect and store results
Loop over time steps
Last time step?
No
Yes
Shut down algorithms' OnlineProcess
END

Fig. 3.4 Flow chart of process for running a single test within a `TestCell`

actions, as defined by the scenario, are then applied to the `SysModel` copy and a load flow is performed based on the updated power system model. This ensures that the `SysModel` and the load flow results in the `LFEngine` represent the initial time step of the scenario being studied. The last step before running the test is to set up and start the control algorithm's `OnlineProcess`, which includes linking it to the `CommsInterface`.

The `TestCell` then loops over the time steps within the scenario, where the control algorithms are called to provide control actions, these actions are then applied to the `SysModel` along with any control actions defined in the scenario, the `LFEngine` is then updated from the `SysModel` and a load flow executed, with the results stored temporarily in memory. In modelling the evolution of a power system's state over time, it is assumed that the algorithm's control actions are not applied to the power system model instantaneously. Instead, the control actions determined by an algorithm for one time step are applied at the next time step, in addition to any controls for that time step as defined by the scenario.

The execution time of the control algorithm is recorded for each time step. This is achieved by monitoring the elapsed processor time for the computer process within which the algorithm is being executed. This gives a more accurate indication of the computational resources required the algorithm than measuring the real "wall clock" time that has elapsed, as that can be significantly affected by other processes that are being executed on the processor at the same time.

3.7 Running multiple tests

All the objects shown in Figure 3.1 are used to run multiple tests of algorithms in an automated manner. A key feature of the testing environment is that the objects within it are able to operate in parallel, allowing the process that administers a series of tests (the `DriverProcess`) to run at the same time as multiple processes that are executing tests (the `TestProcess` objects). Queues are used in between these processes to achieve inter-process communication.

Section 3.2.1 described the functions of the objects related to running multiple tests. This section describes how these objects work together in order to achieve automated testing.

3.7.1 Process

The sequence starts with the `DriverProcess` interrogating the runs database to retrieve test runs that need to be completed, which are then put onto the task queue. In the diagram, this is illustrated by point (a), where 2 test runs are put onto the test queue.

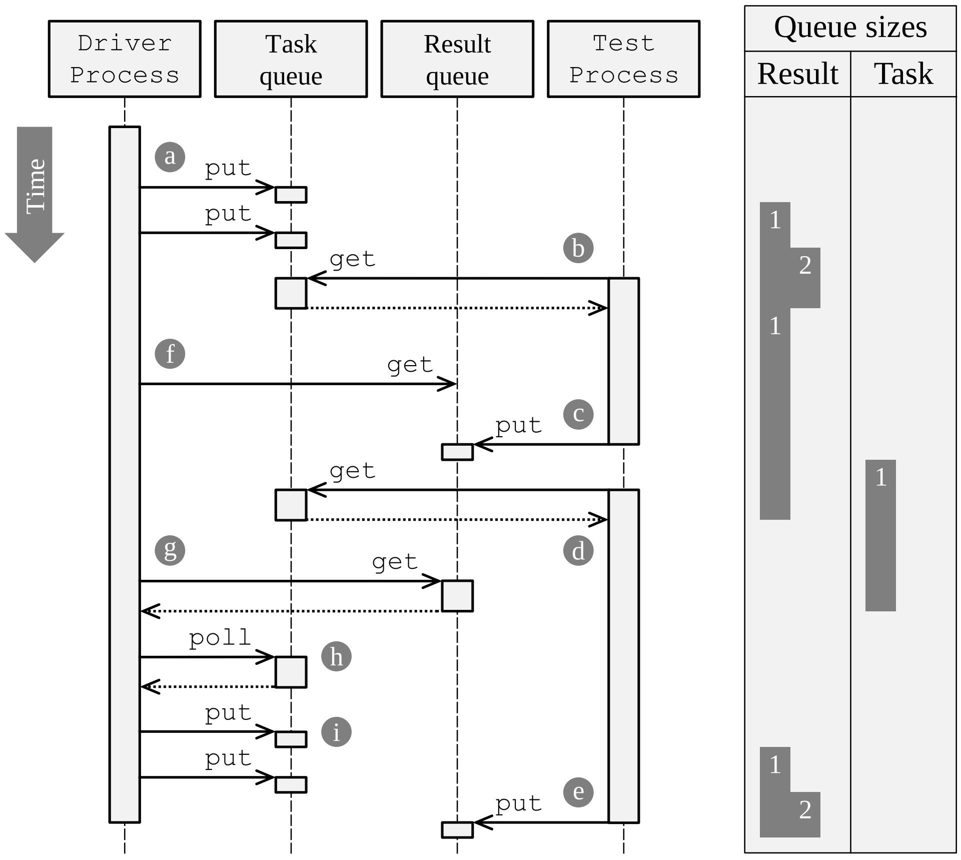


Fig. 3.5 Sequence diagram for running tests in parallel

The `TestProcess` monitors the task queue and retrieves the first available test that needs to be completed (point (b)). Although only one `TestProcess` is shown in the figure, multiple `TestProcess` objects can have access to the queue and can retrieve tests from it without conflict. The `TestProcess` will then set up and execute the test that it has retrieved from the task queue. Once completed, the results of the test are taken from the `TestCell` within the `TestProcess` and put onto the result queue (c). The `TestProcess` can then repeat the process: retrieving another test from the task queue (d), executing the test and then putting the results onto the results queue (e). The `TestProcess` will continue to repeat this process until there are no new test runs added to the task queue.

While the `TestProcess` is executing tests, the `DriverProcess` is acting to store the results of completed tests and to keep the task queue populated with new tests to be completed. The `DriverProcess` achieves this by repeatedly checking the results queue ((f) and (g)). If there are completed tests in the results queue, these are retrieved (g) and then stored: in the runs database, the test is marked as done and a small set of details are recorded, while the data archive is used for storing the complete set of results from the test. The `DriverProcess` will then check the status of the task queue (h). If that queue is found to be under-populated, the `DriverProcess` will interrogate the runs database for additional test runs that need to be completed, and these will be added to the task queue (i).

3.7.2 Failure tolerance

In order to allow automated testing, it was essential for the testing environment to be robust to failures without requiring user intervention. This was achieved by introducing error handling at multiple points throughout the environment. For example, the `TestCell` within each `TestProcess` was encapsulated so that any error arising during a test – such as algorithms being unable to determine control actions, or the load flow not converging – would be intercepted and not cause the `TestCell` to crash. Any test runs that failed in this way would be identified by the `DriverProcess`, reported to the user, and barred from being added to the task queue in the future (as a failed test run will appear within the runs database as a test run that needs completing). The testing environment also allowed the used to interrupt and halt testing at any time without test data becoming corrupted.

3.8 Evaluation against requirements

Table 3.1 evaluates whether each requirement listed in Section 3.1 has been met, partly met, or not met, by the testing environment that has been developed, as described in this chapter.

Most of the requirements have been met, with requirements 1(f) and 3(a) only partly met. However, despite these limitations, the test environment was suitable for the testing, described in Chapter 4, which was required in pursuit of research objective 1.

Table 3.1 Evaluation of testing environment against requirements

No.	Requirement	Met?	Comment
1 (a)	Repeatability: each test shall be deterministic and it must be possible to prescribe and record all factors that can affect the outcome of a test, so that any test can be repeated and the same outcome obtained.	Yes	Achieved through using the test runs database to store all parameters that affect each test, including the value used to seed any random number generator used by an algorithm.
1 (b)	Observability: it must be possible to observe and record any parameter relevant to a test, in order to gather results for analysis and also to allow tests to be repeated if necessary with the same conditions.	Yes	Full state information for the power system being modelled is available within the `SysModel` and `LFEngine` objects, and these are captured and stored in a data archive file for each test. Furthermore, the state of each algorithm can be captured, stored and reinstated at any point.
1 (c)	Flexibility: as required by research objective 1(c), it must be possible to run tests on a variety of networks; furthermore, as required by research objective 1(d) it must be possible to simulate different operating scenarios within each network including conditions that change over time.	Yes	The `SysModel` object can be used to represent any power system that comprises the component types listed in Section 3.3.2, and can be used to represent a particular operating condition by altering component parameters. The runs database allows scenarios with changing conditions to be stored and used for testing.

(continued on next page)

Table 3.1 (continued from previous page)

No.	Requirement	Met?	Comment
1 (d)	Automation: the environment must be capable of running multiple tests and recording their results, without user intervention, in order to enable the large scale testing envisaged by research objective 1(d).	Yes	Achieved, mainly by the `DriverProcess` that automates test runs that are pre-defined in the runs database.
1 (e)	Reliability: the environment must be tolerant of failed tests. If a test fails for whatever reason, then the failure should be recorded. Furthermore, the failure of one test should not influence the outcome of any other test.	Yes	Achieved through the failure-tolerant design of the `DriverProcess` and `TestProcess` objects (please refer to Section 3.7.2).
1 (f)	Speed: the environment should allow testing to be completed as quickly as possible within the resources available. However, the environment is not required to operate in real-time and match the “wall clock” time like a real-time digital simulator (RTDS) [56].	Partly	The architecture of the testing environment is based on parallel processing, which allows testing to exploit the speed advantage of multi-core processors. However, the use of Python for implementing the test environment may result in speed penalties compared with other programming languages.
2 (a)	The environment must be capable of running full AC load flow.	Yes	The `PyPower` load flow engine provides full AC load flow calculation, using the Newton-Raphson solution method.
2 (b)	The environment must allow for the simulation of real-time network operation over a time period, comprising a series of steady states separated by a constant time interval.	Yes	Each test in the runs database consists of an initial state followed by any number of condition changes, representing a series of steady-states at a constant time step.

(continued on next page)

Table 3.1 (continued from previous page)

No.	Requirement	Met?	Comment
2 (c)	The environment must allow for representations of standard power system components and for components that could be expected in future power systems, in order to allow existing and emerging control algorithms to be tested.	Yes	Section 3.3.2 details which components can be represented. The mix of component types includes some that are prevalent in today's power systems, such as transformers and generators, while other types are still emerging, such as energy storage units and embedded DC links.
2 (d)	The environment must allow time-based characteristics of components to be simulated, in order to reflect real-world operation.	Yes	Section 3.3.2 describes a number of components that can be represented within a `SysModel` that can include time-based characteristics that are updated between time steps, such as the state of charge of an energy storage (`sto`) unit.
3 (a)	The environment must support the simulation of multiple control algorithms, in order to fulfil research objective 1(b).	Partly	The architecture of the test environment is flexible to allow different algorithms to be implemented within it, and for tests be run on each; although the version implemented does not allow multiple algorithms to execute simultaneously during a single test run. However, this limitation did not impinge on the scope of testing performed for this work, in particular that presented in Chapter 4.

(continued on next page)

Table 3.1 (continued from previous page)

No.	Requirement	Met?	Comment
3 (b)	It must be possible to specify a set of components that each control algorithm can control. Any control action request that falls outside an algorithm's specified set of controllable devices shall be ignored, so that the control scope of individual algorithms can be restricted to particular zones of operation [2, 57].	Yes	Zone (`zn`) components can be used to control which components each algorithm can control down to the per-busbar level. Additionally, each component has an `enable` parameter that disallows algorithm control at a per-component level, regardless of zone.
3 (c)	Algorithms must only be able to modify the operational parameters of components. Modification of fixed parameters – such as circuit impedances – or internal parameters – such as the state of charge of a battery storage unit – shall not be allowed, in order to reflect real-world operation.	Yes	The components within the `SysModel` can have different types of parameters, as listed in Section 3.3.1, and algorithms are only allowed to modify some of these (namely, control- and mode-type parameters).
3 (d)	Any interaction that an algorithm has with the power system model must be observable and should be able to be captured for inspection and analysis, such as for debugging when prototyping a control algorithm.	Yes	This is achieved by having the communications system between the algorithms and the power system model, as described in Section 3.4.
3 (e)	It should be possible for the environment to be extended to include modelling of communications system characteristics, such as delays and data drop-outs, which could be used as factors to influence the behaviour of control algorithms.	Yes	The communications system objects described in Section 3.4 could be modified to represent those characteristics, without necessitating modifications to any of the other objects within the testing environment.

(continued on next page)

Table 3.1 (continued from previous page)

No.	Requirement	Met?	Comment
3 (f)	Any information that is preserved by an algorithm between executions must be available for inspection. Furthermore, it should be possible to capture and restore this preserved information, so that tests can be restarted at any point without invalidating the requirement for repeatability.	Yes	The structure of each algorithm's `OnlineProcess` requires that any preserved information is available in an externally-accessible data structure.

3.9 Conclusions

This chapter has introduced the testing environment that was used to test power system control algorithms. The requirements for the testing environment were elaborated, including those relating to the ability to run automated tests, model particular power system components and characteristics, implement a communications system model and also those related to the representation of control algorithms under test. Most of those requirements were met by the environment that was developed, and those that were not met fully have not limited the work described in subsequent chapters.

The development of testing environment was essential to enable testing of control algorithms to take place, as required by research objective 1. For the next chapter, Chapter 4, the testing environment was used to test the five power flow management algorithms described in Chapter 2 on four different case study power systems, with 10,000 or more states simulated for each system.

Chapter 4

Performance of power flow management algorithms

This chapter addresses research objectives 1(b), 1(c), and 1(d), by evaluating the performance of the five power flow management algorithms introduced in Chapter 2 when they are tested on four different case study power systems.

Firstly, in Section 4.1, the methodology for the test and evaluation of the power flow management algorithms is outlined, including the performance measures used and the method for statistical analysis of the algorithms' performance. Each of Sections 4.2 to 4.5 is dedicated to one of the case study power systems, and includes a description of the system, the states tested, and the performance of the algorithms. A cross-case study analysis follows in Section 4.6, which examines the performance of each algorithm, including aspects of their designs that lead to particular performance traits. Finally, in Section 4.7, conclusions are drawn on the performance of the power flow management algorithms.

Some of the work presented in this chapter has been the subject of previous publications by the author, in particular, the results of testing the power flow management algorithms on three of four case study power systems. Results from testing on the 33 kV meshed distribution system were presented in [58], results from the IEEE 14-bus system were in [59], while in [60] results for both of those systems and the IEEE 57-bus system were presented. However, results from the 11 kV radial distribution system (Section 4.2) have not been published previously, nor has the statistical analysis of the algorithms' performance or the cross-case study analysis presented in Section 4.6.

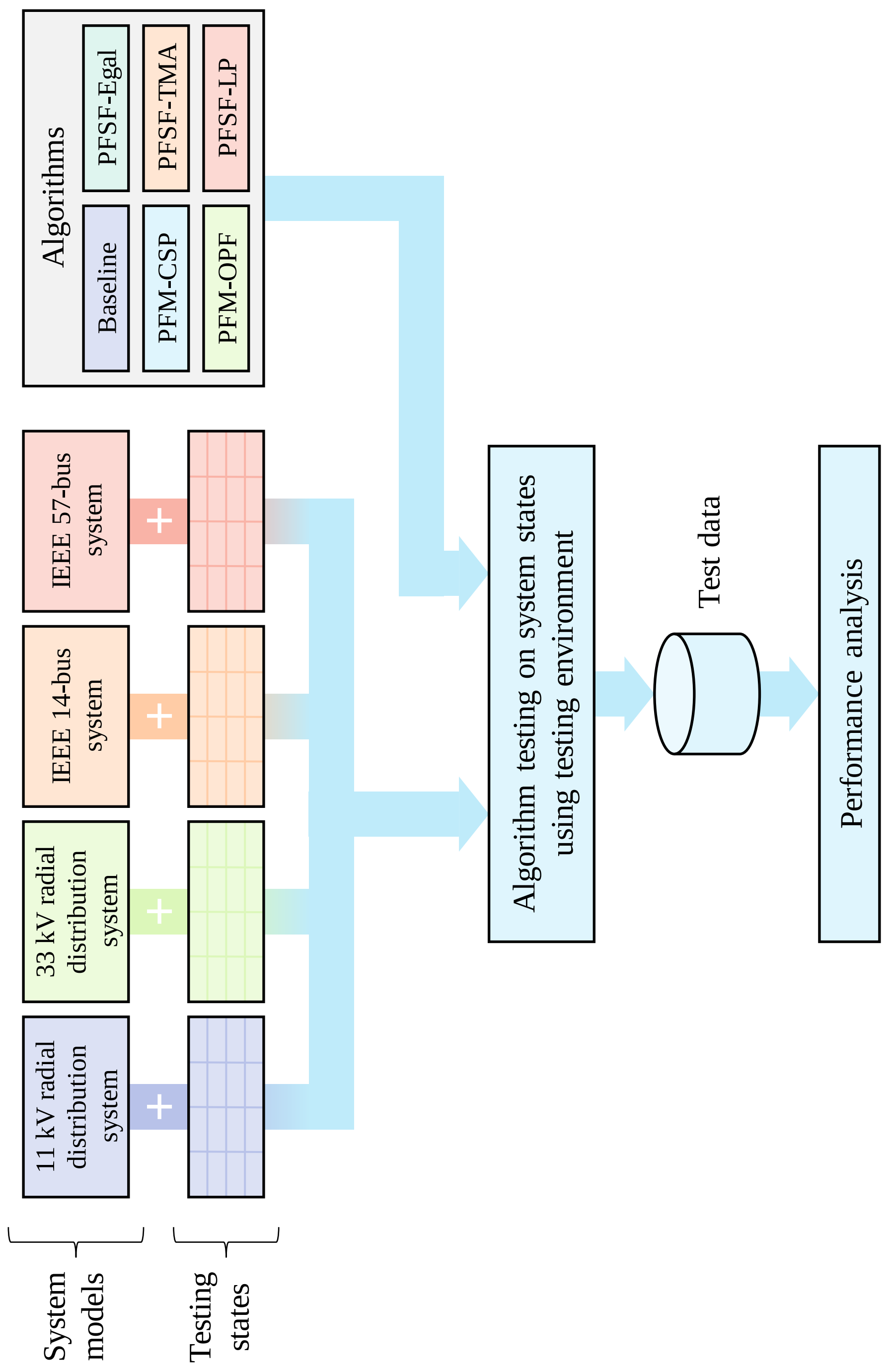


Fig. 4.1 Overview of methodology for testing the performance of power flow management algorithms

4.1 Methodology

The methodology for testing the performance of the power flow management algorithms is illustrated in Figure 4.1 and consists of taking each of the algorithms described in Chapter 2 (and listed in the figure) and applying them to four case study power systems. For each system, a number of states are simulated and each of the power flow management algorithms is then applied separately to each state. As the algorithms are deterministic, it is only necessary to test each algorithm once on each state in order to characterise its performance on that state. The testing environment described in Chapter 3 is used for simulating the operation of the power systems with the various control algorithms applied, and for automating the testing process. Data about the resultant network state after the application of each algorithm is captured, and from this data performance measures are derived and analysed.

4.1.1 Case study power systems

Four case study power systems are used in this work, which are either benchmark systems or have been used in previous work to evaluate the performance of power flow management algorithms: an 11 kV system derived from a real UK distribution network, a second system derived from a real UK distribution network at 33 kV, the IEEE 14-bus system, and the IEEE 57-bus system.

Previous work evaluating the performance of power flow management algorithms have typically used one [35, 37, 40, 43, 44, 47] or two [29–31] power systems as case studies. However, four systems are used in this work to represent a more diverse range of networks and thus the performance of the power flow management algorithms can be more fully characterised. Due to their distribution or transmission origins, the systems feature either radial or meshed network topologies, and with branches mainly consisting of either overhead line or cable, with some transformers. The systems vary in scale in terms of the number of buses and branches, along with featuring different numbers of branches that can become overloaded, either independently or simultaneously.

Due to the poor scalability of the PFM-CSP algorithm to controlling multiple generators (please refer to Section 2.2.3), the systems selected feature four or fewer generators so that the PFM-CSP algorithm can be executed in a reasonable amount of time. Although this precludes the algorithms' performance being characterised for networks with many more generators, there is still a variation in the number and the size of generators, which allows for some characterisation of the effects of scale on the performance of the algorithms. Furthermore, Chapter 2 found that the PFM-CSP is a distinctive approach to power flow management and therefore it is worthwhile to evaluate its performance alongside the other algorithms.

Modifications to the case study systems

For each of the case study systems it was necessary to make modifications so that the generators within the systems could cause branches to become overloaded, thus creating conditions where power flow management algorithms could be applied. As it was desirable to observe the performance of each algorithm on a significant number of states with overload conditions, the modifications were made in such a way that the systems would frequently produce overload conditions. Furthermore, the modifications also ensured that any encountered overload condition could be resolved by adjusting the outputs of the generators within the systems. Only branch ratings and the location and sizing of generators were modified to these ends, using the following iterative process:

1. Execute a series of load flows for: (a) a number of states with generators off and all loads scaled in 10% steps between 0% and 100%, and (b) 1000 states with all loads and the output of each generator scaled by independent random variables between 0% and 100%, sampled from uniform distributions. For each state within each of (a) and (b), capture real, reactive and apparent power of each branch.

2. Find the maximum real, reactive, and apparent power for each branch in (a) and (b).

3. Calculate a minimum rating for each branch based on the maximum apparent power in dataset (a) (so that loads alone cannot cause branch overloads) and the absolute maximum reactive power in both datasets (so that reactive power alone cannot cause branch overloads).

4. Calculate the maximum percentage loading for each branch, based on the minimum rating and the maximum apparent power within datasets (a) and (b).

5. If none or few of the branches are overloaded, or the maximum overload is negligible, then the generation in the network is adjusted (either by increasing the maximum output of individual generators or by changing the locations of the generators) and the process is restarted from step 1.

6. Select a number of the most overloaded branches, and increase their ratings to above the minimum rating but below the maximum loading, such that at least 25% of states in dataset (b) have overloaded branches.

7. For the remaining branches, set their ratings above the minimum rating so that those branches should not become overloaded for any network state.

Data for each case study system can be found in Appendix B.

Test states

For each of the case study systems, the states that the algorithms were tested on represent different levels of general load and output levels for each of the generators within the systems. In contrast to previous work, where some of the power flow management algorithms were tested on only a small number of test states [29–31, 35, 43, 44, 47], in this work at least 10,000 states are used per system for testing, so that the performance of each power flow management algorithm is more fully characterised. The large number of states tested also provides a large sample size for statistical analysis of the algorithms' performance.

For three out of the four case study systems, the states were generated randomly by a Monte Carlo approach, sampling independent uniform random distributions to determine values for the load and generator output levels. This random approach allows for an arbitrary number of test states to be generated and gives uniform coverage of the state space – where the state variables are a scaling factor for all loads and the outputs levels for each of the generators – and could expose performance strengths and weaknesses of the algorithms at the more extreme limits of network operation.

In Monte Carlo approaches it is important that the sample size (the number of states in this case) is sufficient to characterise the system behaviour. Monte Carlo simulations in [61] and [62] use a stopping rule to determine when a sample is sufficiently large enough that the relative error of the sample, ε_{μ_N}, is equal to or below a target maximum relative error, δ. ε_{μ_N} was calculated for the stopping rules in [61, 62] using:

$$\varepsilon_{\mu_N} = \frac{\Phi^{-1}\left(1-\frac{\delta}{2}\right)\sqrt{\frac{\sigma_N^2}{N}}}{\mu_N} \tag{4.1}$$

Where Φ^{-1} is the inverse Gaussian cumulative probability distribution (mean of 0, standard deviation of 1), N is the sample size, μ_N is the sample mean, and σ_N^2 is the sample variance. This stopping rule was adapted for this work as a post-hoc test applied to the randomly generated test states to assess whether the number of states was sufficient to give $\varepsilon_{\mu_N} \leq \delta$. The test was repeated for each of the performance measures (detailed in the next section, 4.1.2) for the baseline performance of a system across all tested states. A target relative error of $\delta = 0.05$ was used, in alignment with the significance level used in the statistical analysis of algorithm performance (introduced in Section 4.1.3). The results of applying the test can be found in the subsequent sections for each of the systems whose states were generated randomly (Sections 4.2, 4.4, and 4.5).

In real power systems the distribution of the state variables may not be uniform, and the state variables may be interrelated and therefore not independent. For these reasons, the states for the remaining system (the 33 kV distribution system) were generated from monitored data, rather than randomly, and thus the distribution and dependences of the state variables follows those of the underlying real data. This system was chosen in particular as the profile data available aligned with the generation types (wind and hydro) known to be in the network. More information about the generation of states is presented in the individual sections for each case study system (Sections 4.2 to 4.5).

4.1.2 Performance evaluation

The algorithm testing environment allows for a large number of parameters to be captured for each test of an algorithm on a particular state. Based on the captured data, the following performance measures are used to evaluate the performance of the power flow management algorithms and the baseline performance of a system with no algorithm applied:

1. **Number of overloads**: the number of branches that remain overloaded after the application of an algorithm.
2. **Overload energy**: calculated from the sum of loadings above rating for each state, multiplied by the time period each state represents.
3. **Total curtailment**: the sum of the generator output power curtailed in each state after the application of an algorithm, multiplied by the time period that each state represents.

In the subsequent analysis, it is assumed that the objective of power flow management is firstly to minimise the number (or the energy) of overloads, and secondly to minimise the curtailment applied to the generators. Thus, when comparing the performance of two algorithms, the algorithm that has the smallest number (or energy) of overloads is deemed to be better performing; and in the case of tied performance, the algorithm that then applies the least curtailment is deemed to be better performing. In practice, this absolute prioritisation of minimising the number (or energy) of overloads above minimising the amount of generator curtailment may not always hold; for example, if there is a cost associated with generator curtailment, it could be more economic to allow some overloads when the marginal increase in replacement costs (due to decreased asset lifespan) associated with an overload is outweighed by the curtailment reimbursement cost owed to a generator.

In addition to the three performance measures relating to the ability of the algorithms to solve power flow management problems, the computational performance of each algorithm is assessed by timing their execution, in order to understand their suitability for real-time control. The background processor load when an algorithm is executed can affect its execution time, so to ensure a fair comparison of execution times the background processor load is assessed before executing a series of tests (every 10 minutes during testing) by timing the execution of a dummy program that has a fixed number of instructions. As the execution time of the dummy program is directly proportional to the background processor load, it can be used to scale the execution times of the subsequent algorithm tests to remove the influence of background processor load. Furthermore, as each algorithm test is executed on a separate processor core, there should not be any influence on execution times from other algorithm tests being run concurrently.

4.1.3 Statistical analysis of performance

The results presented for each case study system aggregates the performance of the baseline system, and each algorithm, across all states tested for that system. However, statistical analysis is required to determine if differences in performance are statistically significant.

As the algorithms are tested on the same set of states, the performance of any two algorithms for any of the performance measures can be considered as paired samples. For testing whether there is a difference in the distributions of two paired samples, the paired t-test or the non-parametric Wilcoxon signed-rank test can be used [63]. The paired t-test assumes the data are normally distributed and compares the *mean* of differences between the pairs. The Wilcoxon signed-rank test is used as an alternative to the t-test when the distribution of the data is severely non-normal, and compares the *median* of the differences between the paired data.

Large sample sizes are used in this work (10,000 or more states), so, under the Central Limit Theorem, it is assumed that the distribution of the mean differences in performance between algorithms approximates a normal distribution. Under this assumption, the paired t-test is used to evaluate the following null hypothesis when considering the performance of algorithms A and B for performance measure O (either the number of overloads, overload energy, or amount of generator curtailment) on system S: *algorithms A and B have the same distribution of performance for measure O on system S.* If the performance of the algorithms A and B are such that this null hypothesis fails to be rejected, then there is no statistically significant difference in those algorithms' performance. On the other hand, if the data allows for this null hypothesis to be rejected, then there is statistically significant difference in the performance of the algorithms.

Each comparison of two sets of performance data using the paired t-test returns a p-value. If the p-value is less than or equal to a value known as the *significance level* (α) then the null hypothesis defined above can be rejected, which indicates that the differences in algorithm performance are statistically significant. In this work, a significance level of $\alpha = 0.05$ is used, which is a conventional choice of significance level for research [64].

When sets of data are compared using the t-test, such as for the multiple pair-wise comparisons performed in this chapter, each comparison constitutes a separate hypothesis. Multiple hypotheses are known as a *family* when they are a collection "for which it is meaningful to take into account some combined measure of error" [65]; in other words, when they are related to one another. In this work, the pair-wise comparisons of algorithm performance are considered as separate families of hypotheses for each case study power system due to the comparisons of algorithm performance being made separately for each system. However, for each system, the comparisons for different performance measures are considered as being within the same family of hypotheses.

When there is a family of multiple hypotheses, increasing the number of hypotheses tested increases the likelihood of a result appearing to be significant by chance alone. Therefore, it becomes necessary to correct for making multiple comparisons in order to control for type I errors (known as *false discoveries* or *false positives*). In this work the Benjamini-Hochberg procedure [66] is used to correct for making multiple comparisons, which consists of:

1. For each family of s hypotheses, rank the p-values of each hypothesis (a comparison using the paired t-test) in ascending order $p_1 \dots p_s$.
2. For the significance level α, find the largest r such that $p_r \leq (r/s) \times \alpha$.
3. For the hypotheses up to the threshold r ($p_1 \dots p_r$), the null hypothesis can be rejected (therefore the t-test indicates statistically significant differences); for any remaining hypotheses the null hypothesis fails to be rejected.

The Benjamini-Hochberg procedure is a common way to correct for multiple comparisons [67], although other methods exist. For example, with the Bonferroni correction [68] the significance of each hypothesis is tested against a modified significance value, which is simply α/s. Although straightforward to apply, the Bonferroni correction becomes conservative as the size of a family of hypotheses increases, and can increase the likelihood of making type II errors (*false negatives*). A less conservative method, the Holm-Bonferroni procedure [69], is similar to Benjamini-Hochberg as p-values are ranked and compared to a modified significance level. However, Benjamini-Hochberg has more detection power than Holm-Bonferroni [67], so is used in this work when correcting for multiple comparisons.

In the next chapter (Chapter 5), the algorithm performance data used in this chapter are supplemented by additional data for what are essentially two other algorithms, and extra pair-wise comparisons of the combined performance data are performed using the paired t-test. As the extra comparisons are related to the tests performed in this chapter, they constitute single families of hypotheses for each system. In order to account for the extra comparisons, the Benjamini-Hochberg procedure was applied only to the combined data for each system to determine the statistical significance of every comparison made. Therefore, the corrections for multiple comparisons in this chapter account for the extra comparisons, although the results of those extra comparisons are not described until Chapter 5.

4.2 Case study: 11 kV radial distribution system

4.2.1 System description

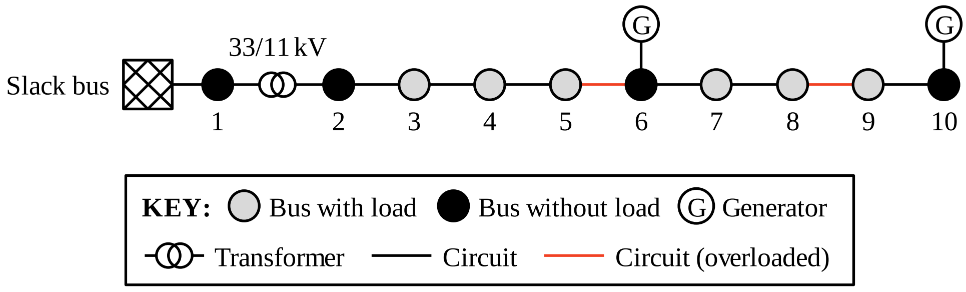


Fig. 4.2 Single line diagram of the 11 kV radial distribution system used in this work

This system is based on network data for part of an 11 kV distribution network from the south east of England [70], which was used in previous AuRA-NMS work testing the PFM-OPF and PFM-CSP algorithms [29–31]. As shown in Figure 4.2, the system has an infeed at 33 kV, transformation down to 11 kV, and a single feeder with a radial topology. The system features 10 buses (6 with loads), 8 cable sections, a single transformer, and 2 generators, which are situated at the towards the remote end of the feeder. As per the descriptions within previous work using this system [29–31], the two circuits highlighted in red in Figure 4.2 (5-6 and 8-9) have had their ratings reduced so that they can potentially become overloaded due to power exported from the generators. Appropriate ratings to achieve this were determined by using the process outlined in Section 4.1.1.

4.2.2 Test states

10,000 system states were generated randomly for the 11 kV radial distribution system in order to test the power flow management algorithms. The following parameters were varied for each state:

Table 4.1 Baseline loadings for the overloaded branches within the 11 kV radial distribution system

Branch	Rating [MVA]	Min. loading [%]	Mean loading [%]	Max. loading [%]	No. of overloads [count]	Mean overload [%]
Circuit 5-6	2.00	5.70	62.81	164.30	1472	117.42
Circuit 8-9	1.50	0.34	59.60	132.09	1872	112.79

- System load: all loads within the system were scaled by the same amount, between 0% and 100%. The scaling factor was drawn from a uniform random distribution.
- Generator output: the real power output of each of the generators within the system were scaled between 0% and 100%. Separate scaling factors were used for each generator, and these were drawn from independent uniform random distributions.

4.2.3 Baseline performance

Each of the 10,000 states was simulated for the system with no algorithm applied in order to understand the baseline performance of the system. 2462 (24.62%) of the states tested featured overloads, and 882 (35.82%) of these have both circuits overloaded. This gives a total of 3344 overloads, which have a total energy of 871.95 MVAh.

Table 4.1 provides a summary of the loadings along the two overloaded circuits within the 11 kV radial distribution system. Circuit 5-6 can become more heavily overloaded as it has both generators feeding in to it, although circuit 8-9 is more frequently overloaded.

Applying the relative error test (Section 4.1.1) for the baseline performance with respect to the number of overloads across the 10,000 tested states yields $\varepsilon_{\mu_N} = 0.037$, whereas for overload energy $\varepsilon_{\mu_N} = 0.048$. Both of these values are below the target maximum relative error ($\delta = 0.05$) so generating additional states and testing the system's behaviour on those states was not required. Curtailment was not considered for the relative error test as that performance measure had the same value (0.0 MWh) for all states.

4.2.4 Algorithm performance

Figure 4.3 summarises the performance of the power flow management algorithms when applied to the 10,000 test states in the 11 kV radial distribution system. Four of the algorithms are able to remove all the overloads, and thus have identical performance with respect to the number and energy of overloads. PFM-OPF is the only algorithm that fails to remove all overloads, leaving 2 overloads in the system with a total energy of 0.10 MVAh.

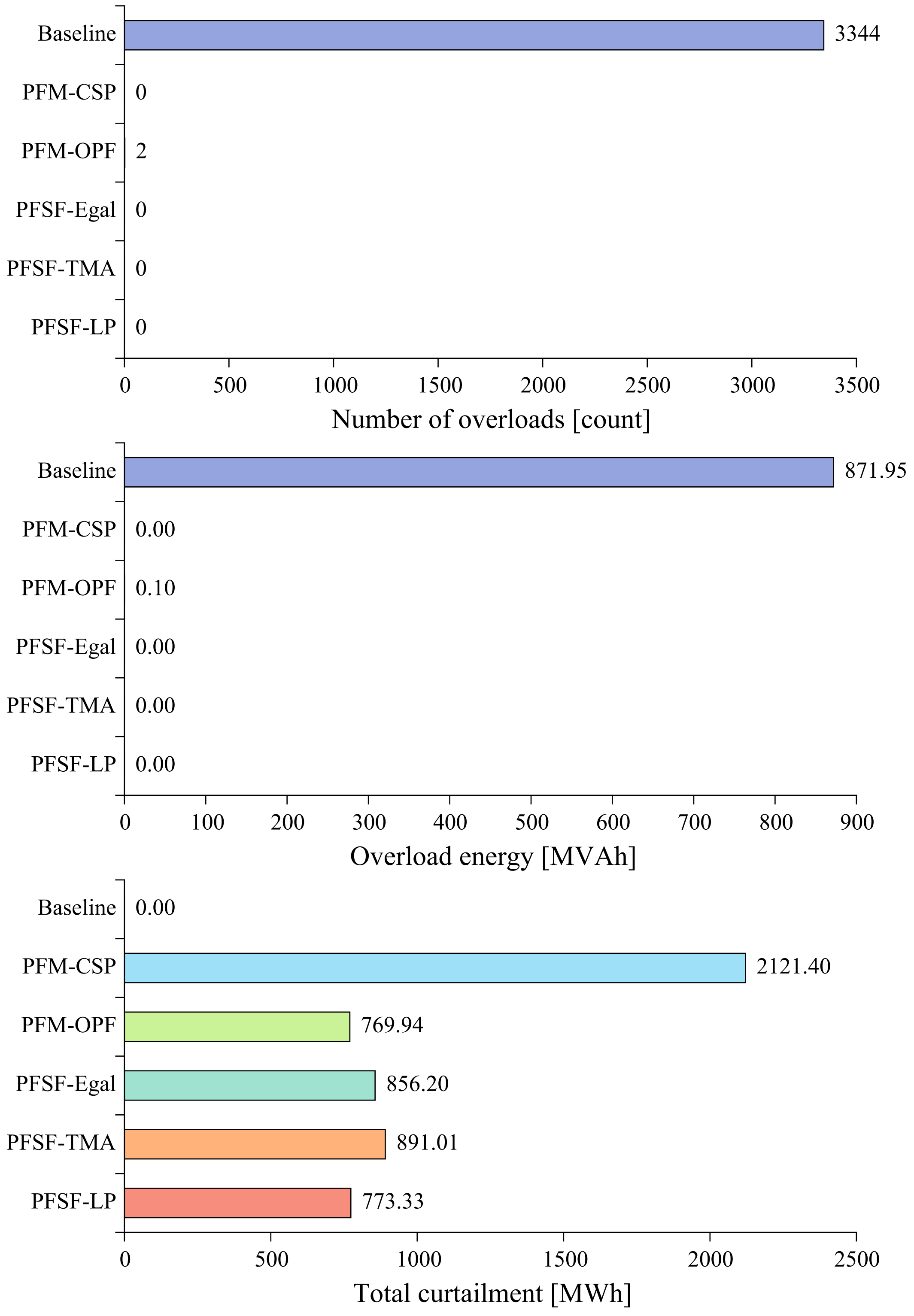


Fig. 4.3 Overview of power flow management algorithm performance when applied to the 11 kV radial distribution system

Comparing the distribution of performance of PFM-OPF against any of the four other algorithms using the paired t-test yields p-values of 0.1573 and 0.1627 for the number and energy of overloads, respectively. The Benjamini-Hochberg correction for multiple comparisons (as described in Section 4.1.3) was applied to these p-values – along with the p-values of all the other pair-wise t-test comparisons in the family of hypotheses for this case study system – to determine their statistical significance. This revealed that the results of the t-tests comparing the overload performance of PFM-OPF to each of the other power flow management algorithms fail to reject the null hypothesis (described in Section 4.1.3) that the distributions of the algorithms' performance are the same.

While there is no statistically significant difference in the algorithms' performance with respect to removing overloads, the amount of curtailment applied varies significantly between the algorithms. PFM-OPF applies the least curtailment, with PFSF-LP applying just 0.44% more. PFSF-Egal applies 11.20% more curtailment than PFM-OPF, and PFSF-TMA applies 15.72% more. PFM-CSP is the worst performing algorithm, applying over double the amount of curtailment of any other algorithm and 175.53% more than PFM-OPF.

Although the difference in the amount of curtailment applied by PFM-OPF and PFSF-LP appears small, using the paired t-test to compare the distribution of curtailment performance for these two algorithms yields a p-value < 0.0001. Applying the same analysis to all other pair-wise combinations of algorithms reveals similar p-values (< 0.0001). After correcting for multiple comparisons using the Benjamini-Hochberg procedure, all of these p-values reject the null hypothesis that the distributions of performance are the same, so there are statistically significant differences in the distribution of curtailment applied by the algorithms.

Based on these results, PFM-OPF is the power flow management algorithm that gives the best performance for the 11 kV radial distribution system, as it applies the least curtailment whilst minimising the number and energy of overloads to a level where there is no statistically significant difference to the other algorithms. However, if the absolute number of overloads is considered, then the amount of curtailment, PFSF-LP is the most effective algorithm.

4.3 Case study: 33 kV meshed distribution system

4.3.1 System description

Similar to the 11 kV radial distribution system, this system was used previously in the AuRA-NMS work for testing the PFM-OPF and PFM-CSP algorithms [29–31]. It is based on network data for part of a 33 kV distribution network within Wales [71], and has a meshed network topology as shown in Figure 4.4.

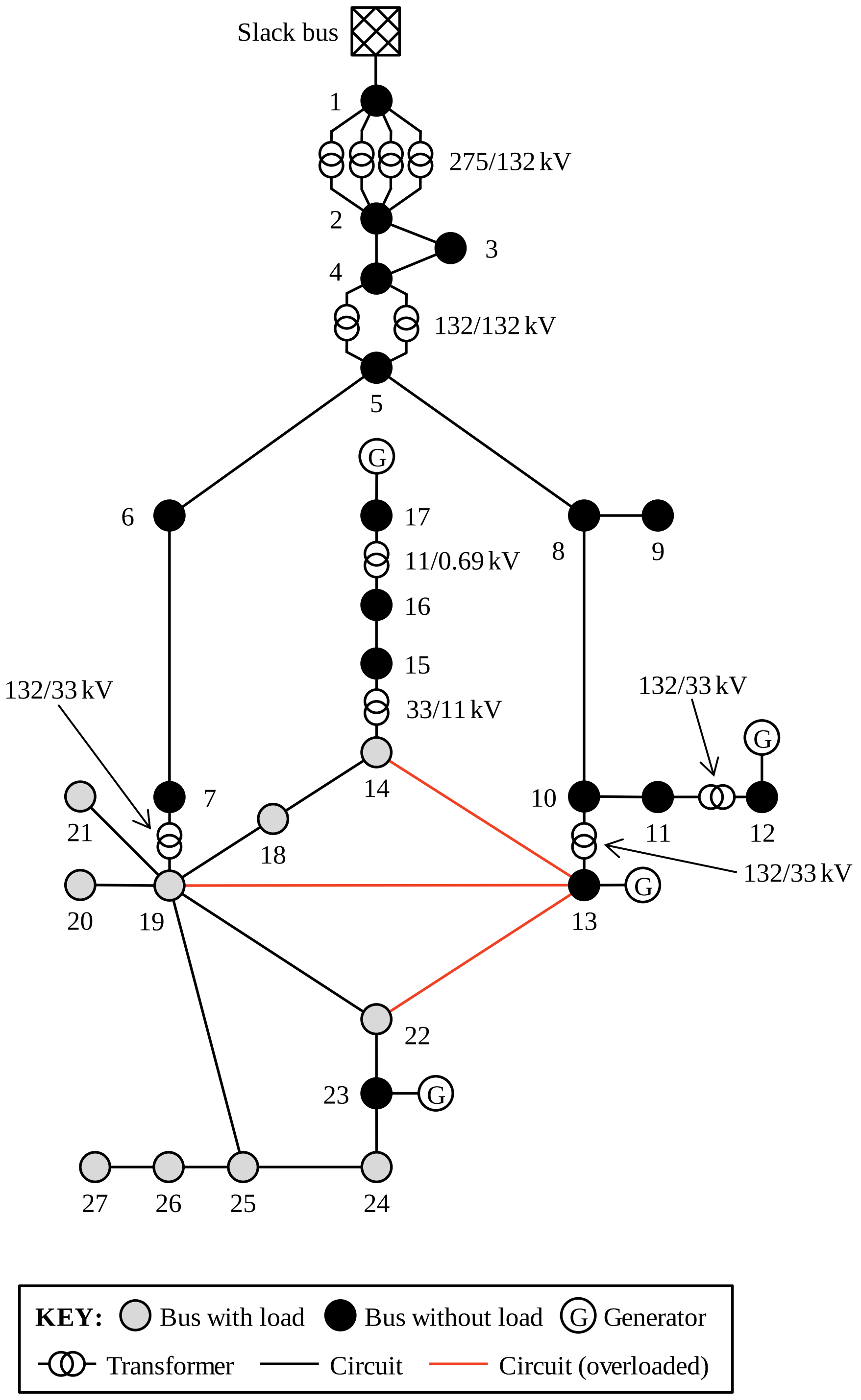


Fig. 4.4 Single line diagram of the 33 kV meshed distribution system used in this work

The system features an infeed from the transmission network at 275 kV, represented by a slack bus. There is then transformation down to 132 kV and then to 33 kV, which is the predominant voltage level within the system model. There are 27 buses and 10 of these have loads connected. There are 26 circuits, almost all constructed of overhead lines, and 9 transformers. The three circuits highlighted in the figure were allowed to become overloaded due to generator export, in line with previous work [29–31]. It was necessary to use the process outlined in Section 4.1.1 in order to determine appropriate ratings for those circuits.

The system features a number of wind or hydroelectric generation sites. In the source network data some of these appeared as multiple generators connected to a bus, representing, for example, the individual turbines at a wind farm. However, this resulted in a large number of generators, which made execution of the PFM-CSP algorithm intractable due to its poor scalability to systems with multiple controllable generators (please refer to Section 2.2.3). To mitigate against this, where a bus had multiple generators connected, the generators were lumped together, leaving a total of four generators in the system.

4.3.2 Test states

For the 33 kV meshed distribution system the test states were generated from real load and generation profiles. Profile data from the United Kingdom Generic Distribution Systems (UKGDS) project [72] was used to scale each of the loads (by a profile representing an average domestic customer) and the outputs of the generators (by either a "wind" or "hydro" profile, as appropriate). Each profile covers a whole year at a half-hour time step, resulting in 17,520 test states in total for this system.

4.3.3 Baseline performance

With no algorithm applied in the 33 kV meshed distribution system, 6719 (38.35%) of the 17,520 test states feature overloads. Out of these overloaded states, 1449 (21.57%) have a single circuit overloaded, 2480 (36.91%) have two circuits overloaded, and the remaining 2790 (41.52%) have all three circuits overloaded (highlighted in Figure 4.4). This gives a total of 14,779 overloads and a total overload energy of 9445.74 MVAh.

Table 4.2 provides a summary of the loading for the three overloaded circuits within the system. Circuit 13-22 is both the most frequently overloaded and the most heavily loaded circuit. Circuits 13-14 and 13-19 are loaded at a similar frequency to each other and have similar loadings.

Table 4.2 Baseline loadings for the overloaded branches within the 33 kV meshed distribution system

Branch	Rating [MVA]	Min. loading [%]	Mean loading [%]	Max. loading [%]	No. of overloads [count]	Mean overload [%]
Circuit 13-14	7.00	25.25	82.55	144.96	4193	112.80
Circuit 13-19	12.00	37.30	84.06	139.08	4189	110.20
Circuit 13-22	10.00	39.37	90.72	152.70	6397	115.64

4.3.4 Algorithm performance

Figure 4.5 summarises the performance of the power flow management algorithms when applied to the 17,520 test states within the 33 kV meshed distribution system.

PFM-OPF is the best performing algorithm as it is the only one that is able to remove all overloads within this system. PFM-CSP is the second best algorithm at removing and reducing overloads, while the PFSF-based algorithms perform worse by at least an order of magnitude. PFSF-Egal removes the least overloads (24.40% remain), and although it also reduces the total overload energy the least, it still manages to reduce the overload energy to 0.83% compared with the baseline performance of the system.

PFSF-TMA applies the least curtailment, with PFSF-LP applying a slightly higher amount (0.12% more), which is correlated with fewer overloads remaining (4.41% less). PFM-OPF, despite removing all overloads, applies just 1.39% more curtailment than PFSF-TMA. PFSF-Egal applies 50.91% more curtailment than PFSF-TMA, and PFM-CSP, the worst performing algorithm in terms of the amount of curtailment, applies 93.76% more than PFSF-TMA.

For each pair-wise combination of algorithms, comparing the distributions of each of the three performance measures shown in Figure 4.5 using the paired t-test gives p-values < 0.0001. After correcting for multiple comparisons using the Benjamini-Hochberg procedure (described in Section 4.1.3), all of these p-values successfully reject the null hypothesis that the distributions of performance are the same. Therefore, the differences in performance of the algorithms are statistically significant for the 33 kV meshed distribution system for all three of the performance measures.

These results show that PFM-OPF is the best performing algorithm in terms of minimising the number and energy of overloads for this case study system, whilst also reducing the amount of curtailment applied.

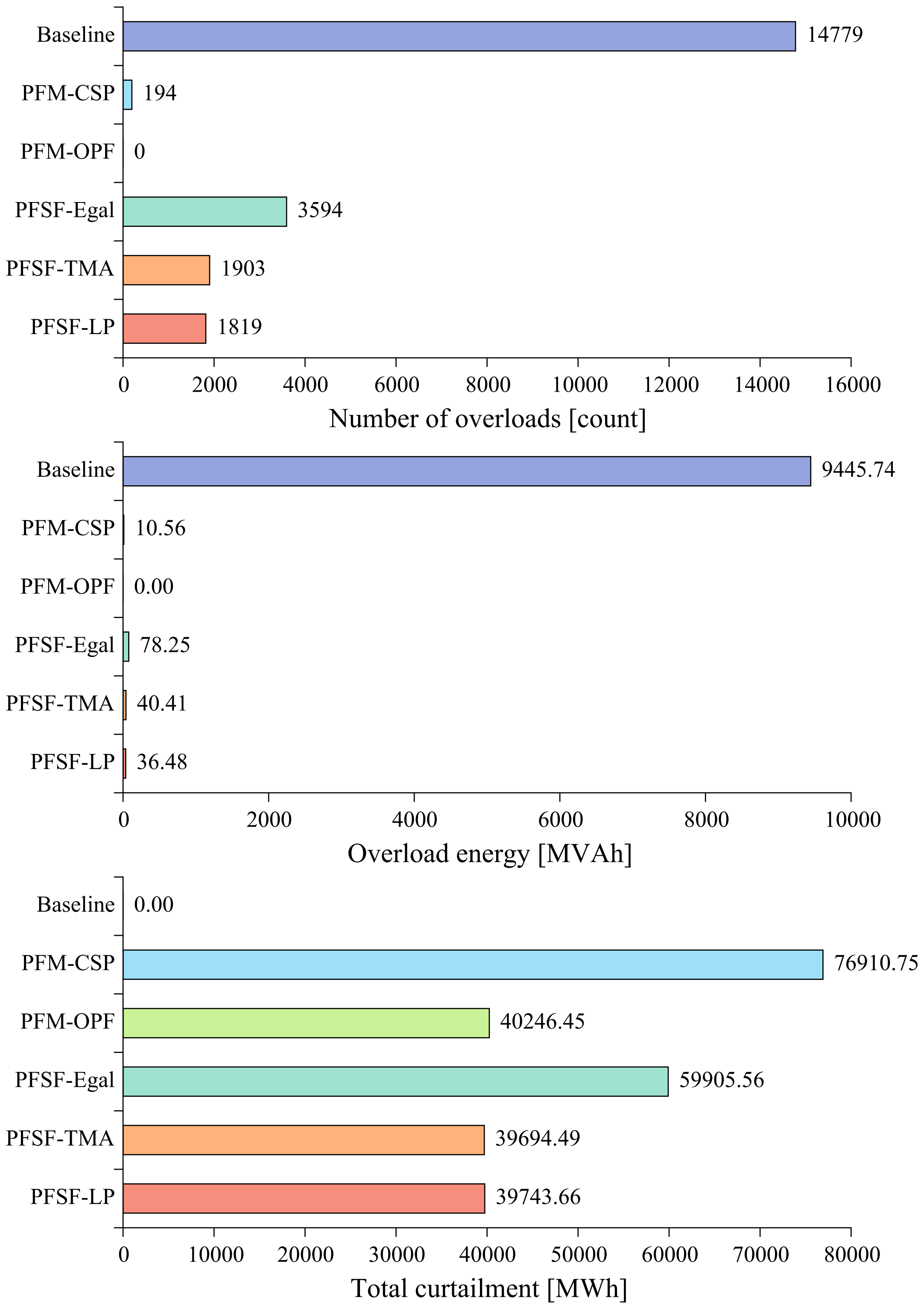


Fig. 4.5 Overview of power flow management algorithm performance when applied to the 33 kV meshed distribution system

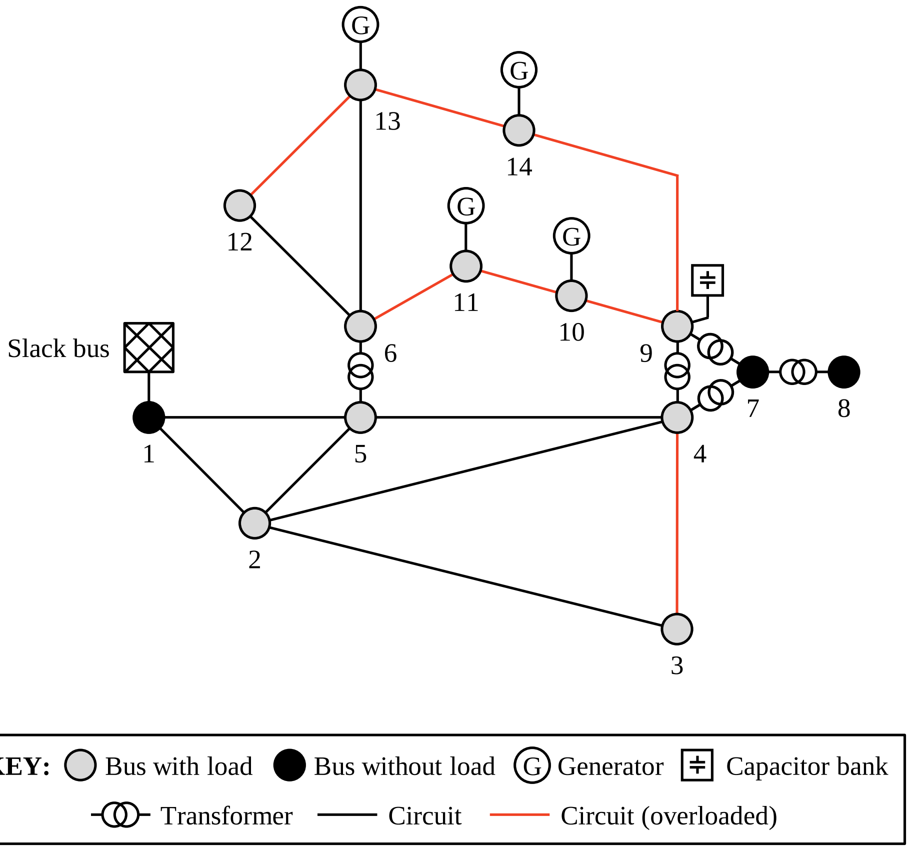


Fig. 4.6 Single line diagram of the IEEE 14-bus system as used in this work

4.4 Case study: IEEE 14-bus system

4.4.1 System description

Figure 4.6 shows the version of the IEEE 14-bus system used in this work. The network topology and electrical parameters of transformers and circuits are taken from a common repository for network data [73]. However, the generators and the branch ratings have been modified using the process defined in Section 4.1.1 in order to create a system in which the generators can cause overloads and thus power flow management algorithms can be applied.

The IEEE 14-bus system is originally derived from a US transmission system and comprises 14 buses, 15 circuits, and 5 transformers (if the 3-winding transformer between buses 4, 8, and 9 is treated as a set of 2-winding transformers). Due to its transmission origin, the circuits are mainly overhead lines whose X/R ratio is between 2 and 4; furthermore, the original data does not include ratings for the circuits and transformers. The slack bus is located at bus 1, and the original configuration from [73] features a single generator (at bus 2) and three synchronous condensers (at buses 3, 6, and 8). Loads are spread amongst 11 of the buses, with bus 9 also featuring a shunt capacitor.

The original configuration of a single generator and branches without ratings does not present a power flow management problem, creating the need to modify the system. As shown in Figure 4.6, the generator and synchronous condensers have been removed, and instead four generators have been added at buses 10, 11, 12, and 14. The generators are assumed to operate with a unity power factor, and their sizes were determined using the process described in Section 4.1.1. That process was also used to determine ratings for the branches within the system, and the 7 branches whose ratings allow them to become overloaded due to generator output are highlighted red in the figure.

4.4.2 Test states

The same method as used for the 11 kV radial distribution system was used to randomly generate 10,000 test states for the IEEE 14-bus system, with random variables drawn from independent uniform distributions used to scale:

- All loads by a common scaling factor.
- Each of the four generators, by separate scaling factors.

4.4.3 Baseline performance

Out of the 10,000 states tested using the IEEE 14-bus system, 4,008 (40.08%) result in overloads. As shown in Figure 4.7, a number of states feature more than one overload, giving 5345 overloads, with a total energy of 9116.25 MVAh.

Table 4.3 tabulates a summary of the loadings for each of the seven circuits within the IEEE 14-bus system that can become overloaded. No circuit is overloaded for more than 25% of all states. Circuits 9-10 and 10-11 are the most heavily loaded, in terms of: 1) the number of overloads, with these two circuits appearing in 58.71% and 37.62% of the overloaded states, respectively; and 2) the magnitudes of the overloads, with these two circuits having both the largest mean overload (120.79% and 113.24%, respectively) and maximum loading (190.58% and 157.00%, respectively).

Applying the relative error test (Section 4.1.1) yields $\varepsilon_{\mu_N} = 0.027$ for the number of overloads and $\varepsilon_{\mu_N} = 0.035$ for overload energy. Both of these values are below the target maximum relative error ($\delta = 0.05$) so the number of states generated and tested is adequate.

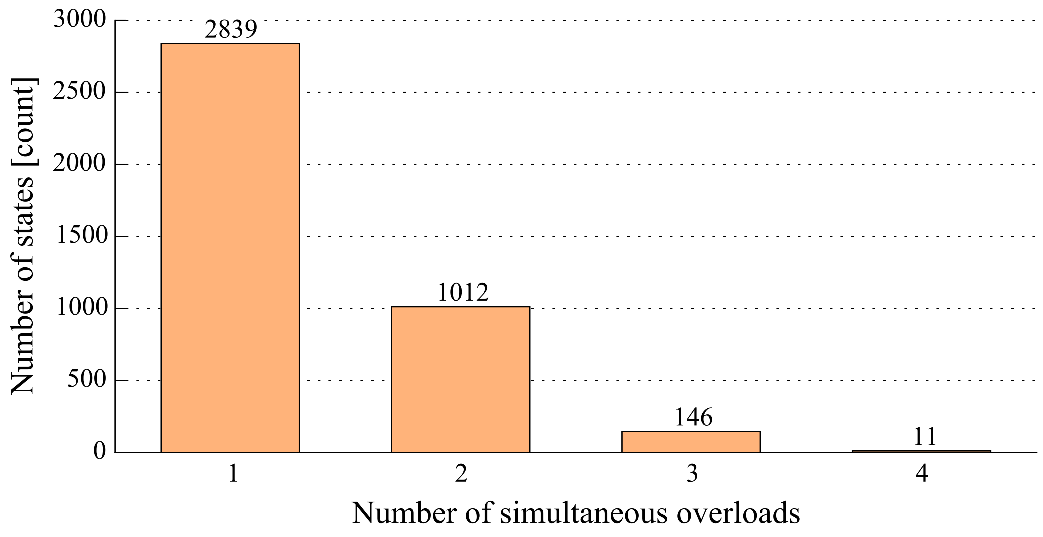


Fig. 4.7 Frequency of simultaneous overloads for the IEEE 14-bus system

Table 4.3 Baseline loadings for the overloaded branches within the IEEE 14-bus system

Branch	Rating [MVA]	Min. loading [%]	Mean loading [%]	Max. loading [%]	No. of overloads [count]	Mean overload [%]
Circuit 3-4	30.00	10.28	60.82	116.32	558	104.55
Circuit 6-11	10.00	6.91	45.14	133.72	147	107.48
Circuit 9-10	10.00	2.44	75.14	190.58	2353	120.79
Circuit 9-14	15.00	0.62	50.74	142.03	738	112.10
Circuit 10-11	10.00	10.32	66.14	157.00	1508	113.24
Circuit 12-13	10.00	1.61	40.08	101.98	2	101.53
Circuit 13-14	10.00	2.28	36.95	117.62	39	104.18

Table 4.4 Statistical significance of differences in the distribution of the number of overloads for the power flow management algorithms applied to the IEEE 14-bus system

Algorithm	Performance [count]	p-values (**bold** if not statistically significant)					
		Baseline	PFM-CSP	PFM-OPF	PFSF-Egal	PFSF-TMA	PFSF-LP
Baseline	5345	–	0.0000	0.0000	0.0000	0.0000	0.0000
PFM-CSP	45		–	0.0366	0.0462	**0.1432**	0.0000
PFM-OPF	24			–	0.0001	0.0007	0.0000
PFSF-Egal	66				–	**0.3546**	0.0000
PFSF-TMA	60					–	0.0000
PFSF-LP	1172						–

4.4.4 Algorithm performance

Figure 4.8 summarises the performance of the power flow management algorithms when applied to the 10,000 test states in the IEEE 14-bus system.

In terms of minimising the number of overloads, PFM-OPF is the most effective algorithm as it leaves only 24 overloads, compared with 45 or more remaining after the application of any of the other algorithms. The overload performance for PFM-CSP, PFSF-Egal, and PFSF-TMA are within the same order of magnitude, whereas the number of overloads remaining after the application of PFSF-LP is two orders of magnitude higher.

The statistical analysis in Tables 4.4, 4.5, and 4.6 show the p-values of the pair-wise t-test comparisons of the algorithms' performance for each of the performance measures. Values that fail to reject the null hypothesis of no difference in algorithm performance, following the application of the Benjamini-Hochberg procedure, are shown in **bold**. The remaining p-values indicate statistically significant differences in performance. With respect to the differences in performance for the number of overloads, Table 4.4 shows that the differences between PFSF-TMA and both of PFSF-Egal and PFM-CSP are not statistically significant; however, the differences in performance between all other pair-wise combinations of algorithms are statistically significant.

Although PFM-OPF removes the most overloads, it is not the most effective algorithm at reducing the total overload energy. Rather, PFSF-TMA is the most effective, and is able to reduce overload energy by an additional 96.68% compared with PFM-OPF. When compared with the baseline performance of the system, this represents a 99.98% reduction in overload energy. PFSF-Egal reduces overload energy by a similar amount as PFSF-TMA, and the analysis in Table 4.5 indicates that the difference in performance for these two algorithms is not statistically significant. The performance of PFM-CSP sits between PFSF-TMA (and PFSF-Egal) and PFM-OPF, while PFSF-LP gives the worst performance, with overload

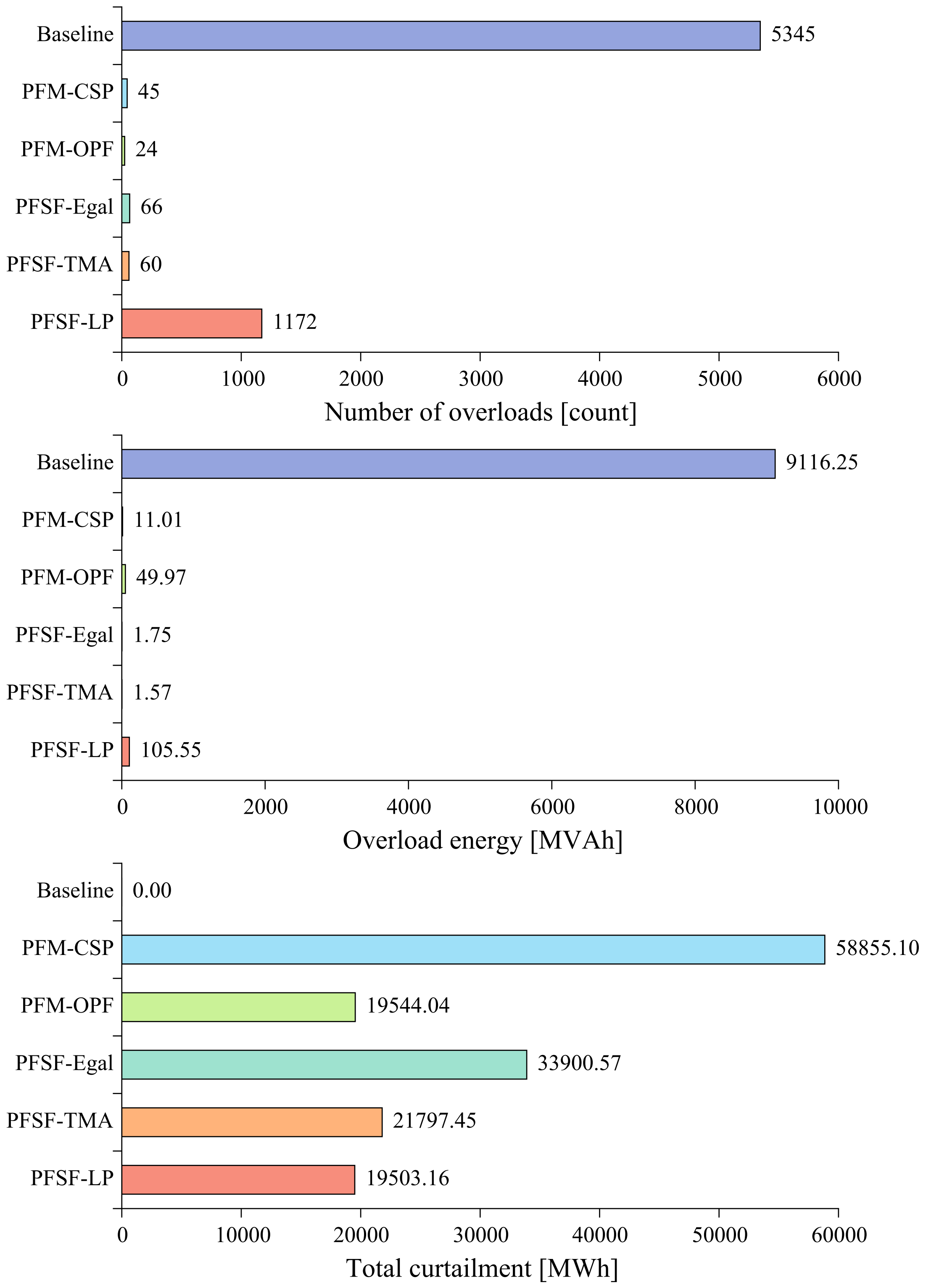


Fig. 4.8 Overview of power flow management algorithm performance when applied to the IEEE 14-bus system

Table 4.5 Statistical significance of differences in the distribution of overload energy for the power flow management algorithms applied to the IEEE 14-bus system

		p-values (**bold** if not statistically significant)					
Algorithm	Performance [MVAh]	Baseline	PFM-CSP	PFM-OPF	PFSF-Egal	PFSF-TMA	PFSF-LP
Baseline	9116.25	–	0.0000	0.0000	0.0000	0.0000	0.0000
PFM-CSP	11.01		–	0.0211	0.0000	0.0000	0.0000
PFM-OPF	49.97			–	0.0040	0.0039	0.0014
PFSF-Egal	1.75				–	**0.3197**	0.0000
PFSF-TMA	1.57					–	0.0000
PFSF-LP	105.55						–

Table 4.6 Statistical significance of differences in the distribution of the amount of curtailment for the power flow management algorithms applied to the IEEE 14-bus system

		p-values (**bold** if not statistically significant)					
Algorithm	Performance [MWh]	Baseline	PFM-CSP	PFM-OPF	PFSF-Egal	PFSF-TMA	PFSF-LP
Baseline	0.00	–	0.0000	0.0000	0.0000	0.0000	0.0000
PFM-CSP	58855.10		–	0.0000	0.0000	0.0000	0.0000
PFM-OPF	19544.04			–	0.0000	0.0000	**0.6957**
PFSF-Egal	33900.57				–	0.0000	0.0000
PFSF-TMA	21797.45					–	0.0000
PFSF-LP	19503.16						–

energy of over the double that of PFM-OPF, although PFSF-LP still removes 98.84% of overload energy when compared with the system baseline performance.

PFSF-LP applies the least curtailment out of the five algorithms; however, Table 4.6 shows there is no statistically significant difference in the curtailment it applies compared with PFM-OPF. This is despite the PFM-OPF being the most effective algorithm at reducing the number of overloads, while PFSF-LP is the least effective for the same performance measure. The differences in the amount of curtailment applied by the other three algorithms is stark, however, and is surprising given their similar performance at reducing the number and energy of overloads. PFM-CSP applies the most curtailment, 73.61% higher than the next worst algorithm, PFSF-Egal. PFSF-Egal, in turn, applies 55.53% more curtailment than the next worst algorithm, PFSF-TMA. The amount of curtailment applied by PFSF-TMA is of a similar order to that applied by PFM-OPF, although it is 11.53% higher and, as Table 4.6 indicates, this is a statistically significant difference in performance.

Out of the algorithms tested on the IEEE 14-bus system, which algorithm performs most effectively depends on whether the number or energy of overloads is of most interest. If the objective is minimising the number of overloads, while also minimising the curtailment applied, then PFM-OPF is the most effective algorithm. However, if the objective is minimising the energy of overloads, while minimising the curtailment applied, then PFSF-TMA is the most effective algorithm.

4.5 Case study: IEEE 57-bus system

4.5.1 System description

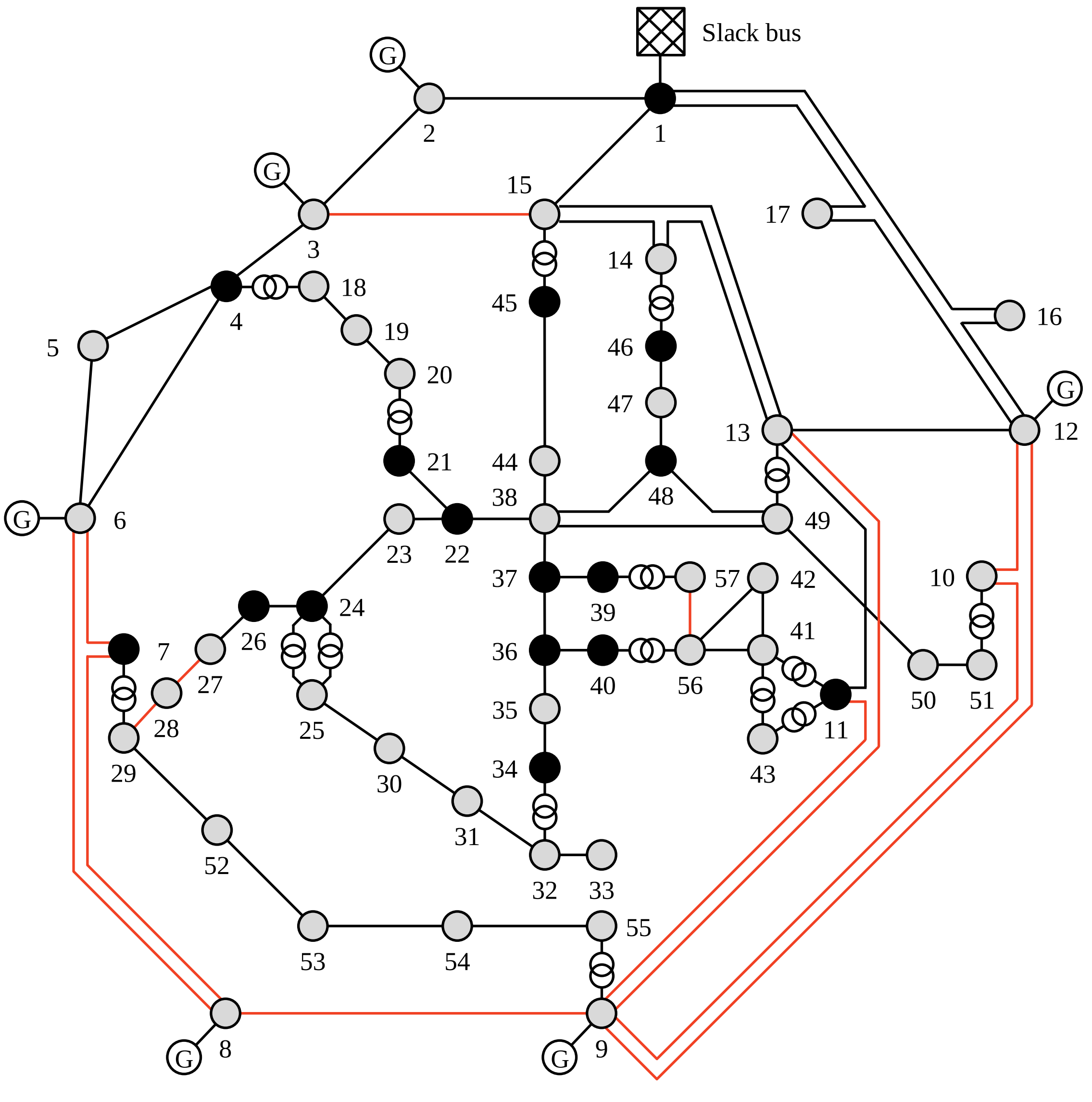
Figure 4.9 shows the version of the IEEE 57-bus system used in this work. The network topology and electrical parameters of transformers and circuits are taken from same common repository of network data as for the IEEE 14-bus system [73]. The generator locations are unchanged from the original data; however, the generator and branch ratings have been modified using the process defined in Section 4.1.1 in order to create a system in which the generators can cause overloads and thus power flow management algorithms can be applied.

The IEEE 57-bus system is originally derived from a US transmission system and comprises 57 buses (of which 41 have loads connected), 62 circuits, and 18 transformers. The circuits are mainly overhead lines whose X/R ratio is between 1.5 and 5, and no ratings data is available. The slack bus is located at bus 1, and there are six generators within the system. The source data has three of these generators (at buses 2, 6, and 9) acting as synchronous condensers, with the remainder (at buses 3, 8 and 12) export real power.

The system was modified in order to create conditions where generator real power export could create branch overloads, using the process described in Section 4.1.1. The modified branches are shown in red in Figure 4.9. The ratings of the three real power-exporting generators were increased. All the generators were left in voltage control mode, and their reactive power ranges were increased to avoid load flow non-convergence due to voltage collapse. Ratings for the circuits and transformers were assigned, with the 13 circuits highlighted in the figure given ratings that could result in them becoming overloaded due to generator real power export.

4.5.2 Test states

Similar to the 11 kV radial distribution and IEEE 14-bus systems, 10,000 states were generated randomly for testing the power flow management algorithms on. In each state, all loads and each of the generators' output were scaled by independent uniform random variables.



KEY: ● Bus with load ● Bus without load Ⓖ Generator
-ꝏ- Transformer —— Circuit —— Circuit (overloaded)

Fig. 4.9 Single line diagram of the IEEE 57-bus system as used in this work

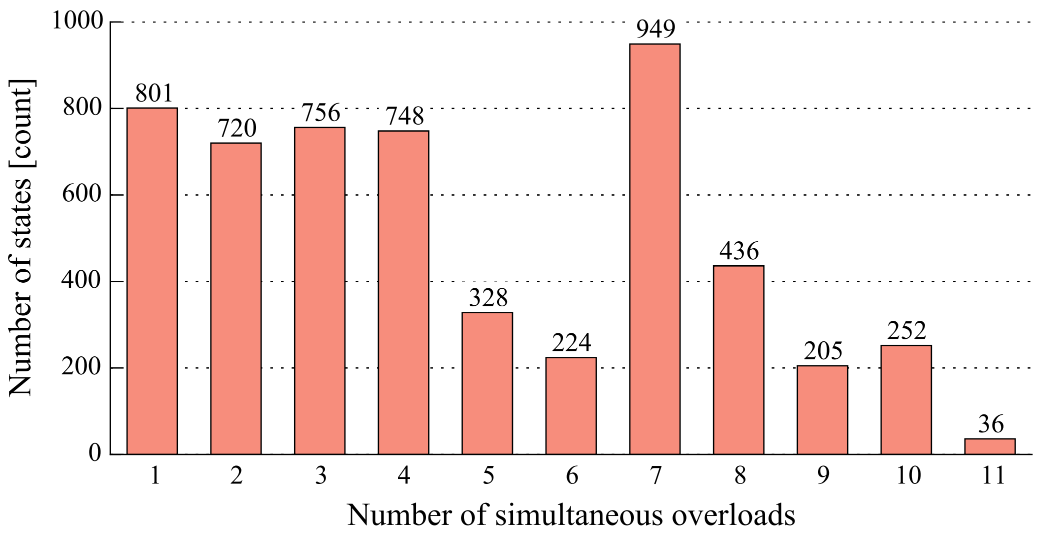


Fig. 4.10 Frequency of simultaneous overloads for the IEEE 57-bus system

4.5.3 Baseline performance

Out of the 10,000 states tested using the IEEE 57-bus system, 5,455 (54.55%) result in overloads. As shown in Figure 4.10, the majority of these states feature more than one overload, giving 25,377 overloads with a total energy of 678,836.34 MVAh.

Table 4.7 summarises the loadings for the 13 circuits whose ratings are such that they can become overloaded. Circuit 8-9 is the most heavily loaded and most frequently overloaded branch. It is overloaded for 51.03% of states and is amongst the seven branches that are overloaded more than 1,000 times. Circuits 9-13 and 10-12 are overloaded significantly less frequently than the other branches, being overloaded for just 0.48% and 0.18% of the 10,000 states, respectively.

Applying the relative error test (Section 4.1.1) yields $\varepsilon_{\mu_N} = 0.024$ for the number of overloads and $\varepsilon_{\mu_N} = 0.027$ for overload energy. Both of these values are below the target maximum relative error ($\delta = 0.05$) so the number of states generated and tested is adequate for this system.

4.5.4 Algorithm performance

Figure 4.11 summarises the performance of each power flow management algorithm when applied to the 10,000 test states in the IEEE 57-bus system.

PFM-OPF is the most effective algorithm at minimising the number of overloads, with 5.39% overloads remaining after its application. PFM-CSP is the next most effective algo-

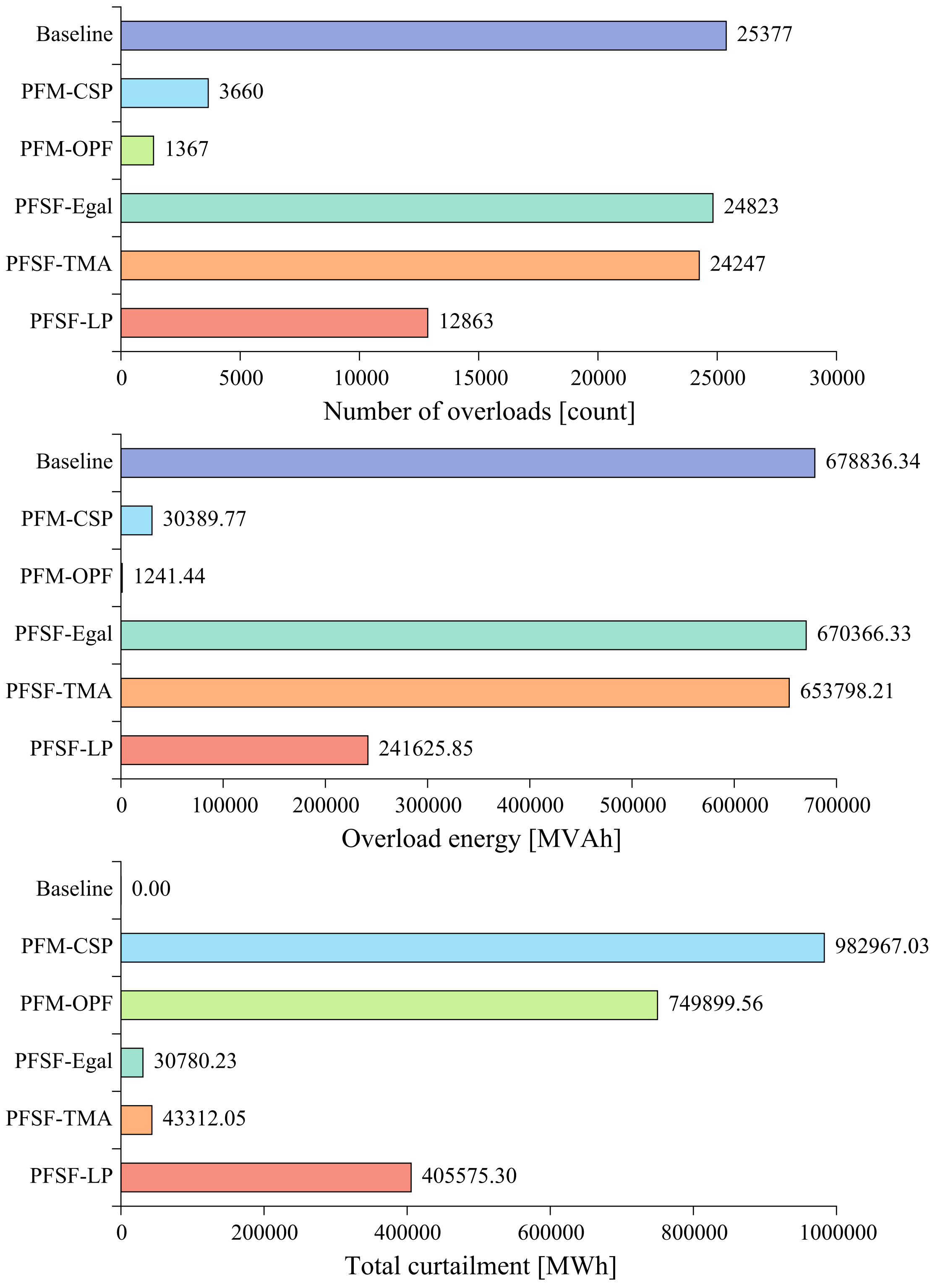


Fig. 4.11 Overview of power flow management algorithm performance when applied to the IEEE 57-bus system

Table 4.7 Baseline loadings for the overloaded branches within the IEEE 57-bus system

Branch	Rating [MVA]	Min. loading [%]	Mean loading [%]	Max. loading [%]	No. of overloads [count]	Mean overload [%]
Circuit 3-15	70.00	4.27	56.52	125.47	554	108.07
Circuit 6-7	80.00	18.58	56.46	121.28	619	106.62
Circuit 6-8	70.00	23.02	75.53	160.60	3039	122.38
Circuit 7-8	70.00	16.90	91.04	189.96	4412	137.52
Circuit 8-9	150.00	33.46	106.83	227.53	5103	155.45
Circuit 9-10	40.00	37.82	86.72	187.05	3659	130.50
Circuit 9-11	95.00	12.60	58.03	113.72	700	104.07
Circuit 9-12	40.00	37.44	75.59	174.46	2559	123.69
Circuit 9-13	100.00	10.69	52.25	104.39	48	101.33
Circuit 10-12	60.00	13.81	38.84	106.35	18	102.14
Circuit 27-28	25.00	0.38	64.64	126.81	2027	111.75
Circuit 28-29	25.00	0.93	65.95	127.64	2170	112.60
Circuit 57-56	3.00	47.86	72.93	111.10	469	103.78

rithm, and leaves 14.42% of overloads in the system, which is 167.74% more then PFM-OPF. PFSF-LP leaves 50.69% of the overloads. PFSF-TMA and PFSF-Egal perform similarly, although PFSF-Egal is the worst performing of the two and leaves 95.57% of overloads after its application.

The ranking of the algorithms regarding overload energy follows the same pattern as their performance at reducing the number of overloads. PFM-OPF is the most effective algorithm as it minimises total overload energy. It also produces the lowest overload energy as a proportion of the overloads remaining, achieving 0.91 MVAh per overload compared with 8.30 MVAh per overload for PFM-CSP and higher for the other algorithms.

For the IEEE 57-bus system, the ranking for the amount of curtailment applied by the algorithms is almost the inverse of the ranking for the other two performance measures, with PFSF-Egal applying the least, followed by PFSF-TMA and PFSF-LP. PFM-OPF applies the second highest amount of curtailment, while PFM-CSP applies the highest amount. Despite applying much more curtailment, PFM-CSP does not perform as well as PFM-OPF with respect to removing overloads.

For each pair-wise combination of algorithms, comparing the distributions of each of the three performance measures using the paired t-test gives p-values < 0.0001. All of these p-values successfully reject the null hypothesis that the distributions of performance are the same, after using the Benjamini-Hochberg procedure to correct for multiple comparisons.

Thus the differences in performance are statistically significant for the IEEE 57-bus system, for every performance measure and pair-wise combination of algorithms.

Considering the performance of all the algorithms, PFM-OPF is the most effective, as it both minimises the number and energy of overloads. However, it does not apply the least amount of curtailment, although this is to be expected as curtailment is required to remove overloads, and the algorithms that apply less curtailment perform significantly worse at removing overloads.

4.6 Cross-case study analysis

This section analyses the performance of the algorithms across all the case study systems. This includes consideration of the reasons behind the differences in the performance of the algorithms.

4.6.1 Comparison of algorithm performance

Figure 4.12 compares the performance of the power flow management algorithms across all the case study systems, for the three performance measures relating to the number of overloads, overload energy, and curtailment applied. Also shown is the baseline performance of each system. The height of each bar gives the relative performance of an algorithm (or the baseline) compared with the maximum performance value for each system. For the overload performance measures, the baseline systems give the maximum values so the algorithms' performances are scaled relative to those values. For curtailment, PFM-CSP consistently applies the most curtailment across all systems, and the performance of the other algorithms are scaled relative to the amount of curtailment applied by PFM-CSP for each system.

With respect to the number of overloads, there is a pattern in how the relative performances of the algorithms change between systems. For the 11 kV radial distribution system the algorithms are able to remove all (or almost all) overloads. For the IEEE 14-bus system, there is an increase in the number of overloads relative to the baseline performance of the system, though this increase is to 1.23% or less for all algorithms except PFSF-LP, whose relative performance is an order of magnitude higher at 21.93%. The 33 kV meshed distribution system sees an increase in the relative number of overloads for all algorithms except PFM-OPF. All algorithms perform relatively worse on the IEEE 57-bus compared with the other systems, particularly the PFSF-based algorithms, with PFSF-Egal failing to remove almost all overloads.

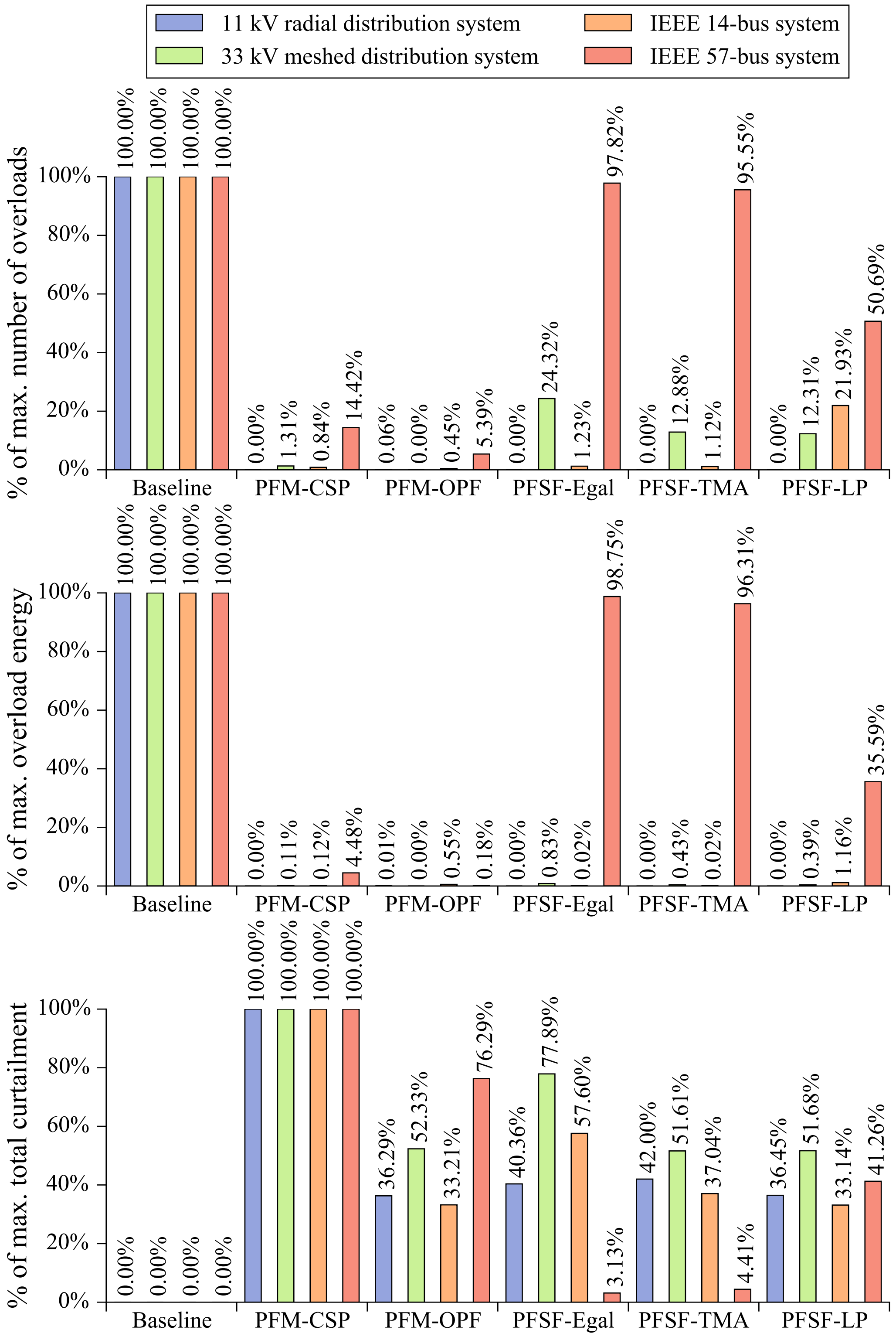


Fig. 4.12 Comparison of the relative performance of every algorithm across the case study systems, for each of the three performance measures

Table 4.8 Ranking of the power flow management algorithms against the different performance measures across the four case study systems

Performance measure	Algorithm	Ranking – for each performance measure and system				
		11 kV radial	33 kV meshed	IEEE 14-bus	IEEE 57-bus	Average
Total number of overloads	PFM-OPF	3	1	1	1	1½
	PFM-CSP	3	2	3	2	2½
	PFSF-LP	3	3	5	3	3½
	PFSF-TMA	3	4	3	4	3½
	PFSF-Egal	3	5	3	5	4
Total overload energy	PFM-OPF	3	1	4	1	2¼
	PFM-CSP	3	2	3	2	2½
	PFSF-TMA	3	4	1½	4	3⅛
	PFSF-LP	3	3	5	3	3½
	PFSF-Egal	3	5	1½	5	3⅝
Total curtailment	PFSF-LP	2	2	1½	3	2⅛
	PFM-OPF	1	3	1½	4	2⅜
	PFSF-TMA	4	1	3	2	2½
	PFSF-Egal	3	4	4	1	3
	PFM-CSP	5	5	5	5	5

The pattern of changes in relative performance for overload energy is similar to that for the number of overloads, although the degree of change is less for all systems apart from the IEEE 57-bus system.

With respect to the amount of curtailment, the pattern of the relative curtailment applied (compared with PFM-CSP) is fairly consistent. The relative curtailment applied by the algorithms on the 33 kV system is more than what they apply on the 11 kV system, while for the IEEE 14-bus system it is less (with the exception of PFSF-Egal, which applies an amount between what it applies for the 11 kV and 33 kV distribution systems). For the IEEE 57-bus system, there is different pattern in the amount of curtailment applied by the algorithms, with PFM-OPF applying its highest amount of curtailment while PFSF-Egal and PFSF-TMA apply their least.

Table 4.8 ranks the performance of the algorithms. For each performance measure, the ranks of the algorithms for each system is shown, along with average ranks across all systems. Where there are two or more algorithms with no statistically significant difference in their performance – such as for the number of overloads on the 11 kV radial distribution system – those algorithms are all given the same average rank.

As seen in Figure 4.12 and in the rankings in Table 4.8, PFM-OPF is, on average, the most effective algorithm at reducing the number and energy of overloads. For the same two performance measures, PFM-CSP is the second most effective algorithm, followed by PFSF-LP and PFSF-TMA, with PFSF-Egal the least effective algorithm. The rankings for PFSF-LP and PFSF-TMA swap between the two performance measures, with PFSF-LP ranking higher with respect to minimising the number of overloads and PFSF-TMA ranking higher with respect to minimising overload energy.

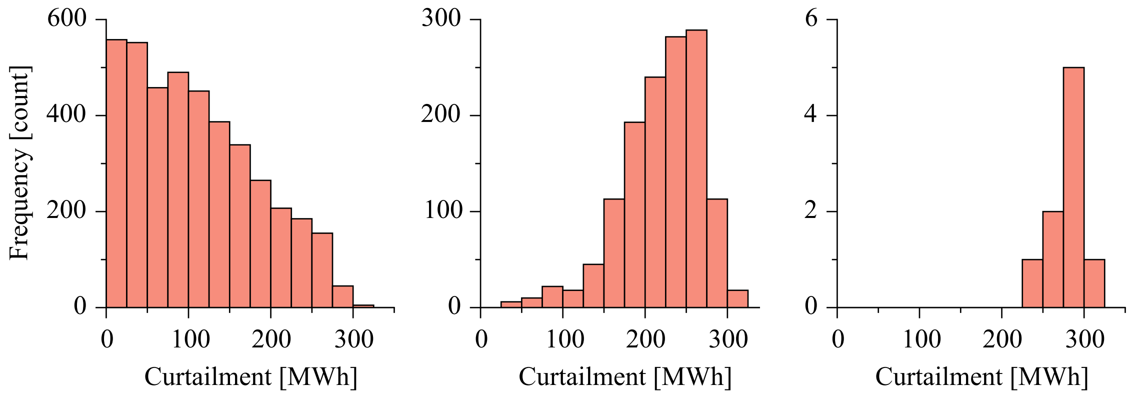
With respect to minimising the curtailment applied to the generators, it could be expected that the algorithm that removes the least overloads on average (PFSF-Egal) would also apply the least curtailment on average, as applying curtailment is necessary to remove overloads. However, this is not the case, as PFSF-LP applies the least curtailment, on average, across the case study systems; despite not ranking last with respect to minimising the number or energy of overloads. PFSF-OPF ranks second, followed by PFSF-TMA, PFSF-Egal, and PFM-CSP, which is the worst performing algorithm for this performance measure as it consistently applies the most curtailment on every system.

4.6.2 Analysis of algorithm performance: PFM-OPF

Although PFM-OPF is the most effective algorithm on average for reducing the number and energy of overloads, it fails to remove some overloads within each of the systems, except for the 33 kV meshed distribution system. For the 11 kV radial distribution and the IEEE 14-bus systems, PFM-OPF fails to remove overloads for 2 and 11 states, respectively, and all of these are due to non-convergence of the OPF algorithm. For the IEEE 57-bus system, however, there are no convergence issues for the 10,000 states tested, despite the larger scale of that system, and PFM-OPF's failure to remove overloads stems from a different cause.

For the 5455 states with overloads in the IEEE 57-bus system, PFM-OPF always detects the overloads and applies some curtailment, but fails to remove all overloads for 1358 states. In 1349 (99.34%) of the states with overloads remaining, PFM-OPF leaves a single overload in the system, while in the remaining 9 (0.66%) states, two overloads are left. Figure 4.13 shows the distribution of curtailment applied across all states that initially contain overloads, with the data split according to the number of overloads that remain after PFM-OPF is applied. The figure shows an association between increased numbers of overloads remaining and an increased amount of curtailment being applied.

The link between the amount of curtailment and the number of remaining overloads for PFM-OPF in the IEEE 57-bus system can be explained by considering two factors. The first factor is that any curtailment applied will cause the system under control to deviate away from its initial state, but larger amounts of curtailment will cause this deviation to be more significant. This becomes pertinent when considering the second factor, which is the model of the system under control that is used internally within the PFM-OPF algorithm.



(a) Overloads remaining = 0 (b) Overloads remaining = 1 (c) Overloads remaining = 2

Fig. 4.13 Distribution of the amount of curtailment applied by the PFM-OPF algorithm for the initially overloaded states in the IEEE 57-bus system, for different numbers of overloads remaining

The model internal to PFM-OPF assumes that only the slack bus is in voltage control mode (with tolerance limits of ± 0.01 pu), whereas the IEEE 57-bus system used in this work has generators at other buses in voltage control mode in addition to the slack bus. At the start of the solution process, the internal model is updated to match the initial state of the system under control, which ensures that the reactive power contributions from the generators in voltage control mode are appropriately represented. However, once the OPF solution process starts and potential curtailments are trialled, the state of the internal model will deviate from the initial state. Crucially, due to the assumptions about voltage control, the state of the internal model will also deviate from the equivalent state of the system under control if the same curtailments were applied. This means that PFM-OPF may find a solution that removes all overloads in its internal model, but some overloads may remain when the same curtailments are applied to the system under control. This is particularly apparent when larger amounts of curtailment are applied, which results in larger deviations between the initial and final states of the internal model and thus larger differences between the final state of internal model and the state of the system under control. This explains the association between increased numbers of overloads remaining and increased amounts of curtailment being applied shown in Figure 4.13.

The results from the four case study systems show that although PFM-OPF most frequently minimises the number of overloads, there are states where one or more of the other algorithms can do the same but with less curtailment, such as for the 33 kV meshed distribution system. The key factor behind this is an assumption within PFM-OPF's internal model that adjusts branch ratings to 99% of their actual value, which results in PFM-OPF sometimes applying more curtailment than necessary to remove an overload. During prototyping, this assumption was found to be essential in ensuring that PFM-OPF would not leave marginal overloads in the system under control when its internal model had no overloads, due to numerical differences. The marginal overloads caused PFM-OPF to have very poor performance with respect to the number of overloads, so it was desirable to improve this. Similar assumptions are used in the other algorithms in order to account for approximations and help to improve the algorithms' performance with respect to the number of overloads, although with little effect of their performance with respect to overload energy and curtailment.

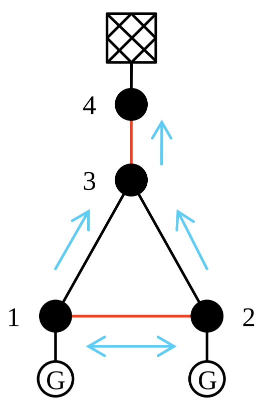
4.6.3 Analysis of algorithm performance: PFM-CSP

PFM-CSP is on average the second most effective algorithm at removing overloads. It failed to remove overloads on all three systems with a meshed network topology, due to a feature of its design rather than execution errors.

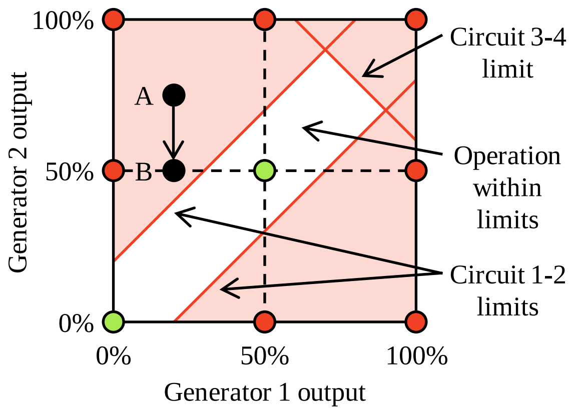
PFM-CSP uses an internal network model to check candidate generator output limits during the CSP solution process. This internal model is updated to match the initial state of the system under control, including the current output set points of the generators. When the internal model is used during the CSP solution process, candidate output limits are applied as new set points to the generators within the model to assess whether overloads would occur if the generators were operating at those limits. This ignores the available power from each generator, so, for example, a generator with only 50% of its maximum power available would have a output limit of 100% tested in the internal model by setting the appropriate generator to 100% output, rather than 50% output. This allows the maximum output limits to be determined, which the generators can increase their output to match, but means that PFM-CSP does not check that the overloads are removed when the output limits are applied to the generators in the system under control. A consequence of this is that PFM-CSP may fail to remove overloads within meshed systems, as explained below.

To explain how this phenomena occurs, consider the simple power system illustrated in Figure 4.14a. This example system has a meshed topology consisting of four buses, of which two feature generators and one is the slack bus. It is assumed that the generators are the same size and the parameters of all branches (except their ratings) are identical. With this configuration, the power flowing through circuit 3-4 combines the output of both generators, while power will only flow along circuit 1-2 if the output of the generators are different.

The state space for this system comprises two variables, which are the output levels of the two generators. Therefore, the state space can be represented in two dimensions, as shown in Figure 4.14b. Assuming that the ratings of circuits 1-2 and 3-4 are such as to cause a power flow management problem due to export of power from the generators, the operating



(a) System schematic



(b) System state space

Fig. 4.14 Simple power system for illustrating overload performance weaknesses of PFM-CSP algorithm

conditions that lead to overloads can be represented within the state space, as shown by the shaded areas in the figure. Also represented in the figure (by the nine coloured dots) are the output limits that the PFM-CSP algorithm would check within its internal model if an overload condition occurs. It is obvious that the only valid output limits that the algorithm would find are the points highlighted in green, which represent an output limit of 0% applied to both generators, and an output limit of 50% applied to both generators.

In Figure 4.14b point A is an example of an operating state in which circuit 1-2 is overloaded, due to the generators at bus 1 and 2 operating at different output levels (20% and 80%, respectively). The PFM-CSP algorithm would detect the overload and use its internal model to assess what output limits would lead to the overload being removed, by applying the nine operating points (represented by the coloured dots: (0%, 0%), (50%, 0%), etc.) to the generators in the internal model. Due to the algorithm's preference constraint of minimising curtailment (refer to Section 2.3.1), the operating limits from point (50%, 50%) would be determined to be both a valid and the preferred solution. Applying a 50% limit to both generators would result in the output of the generator at bus 1 staying the same (as 20% < 50 %), while the output of the generator at bus 2 would fall from 80% to 50%. Although the system would be in a new state (point B) within the output limits defined by the PFM-CSP algorithm, this new state is still inside the region of the state space in which circuit 1-2 is overloaded. Therefore, although PFM-CSP may find a solution that removes all overloads when applied as output limits to its internal model, when the same limits are applied to the system under control, overloads may remain.

Aside from the failure of PFM-CSP to remove certain overloads, it applies the most curtailment of all the algorithms on each of the case study systems. This is due to the discrete domains used in CSP process for the output limits, which can lead to more curtailment being applied than necessary. For example, if a generator operating at 100% (full output) only needed to operate at 99% output in order to remove an overload, PFM-CSP would not recognise this and would apply an output limit based on the closest value from the discrete domain that was less than or equal to that value, which in this case would be 50%.

The failure of PFM-CSP to remove certain overloads is down to a fundamental aspect of the algorithm's design and re-designing the algorithm to overcome that issue – for example, by determining an operating "envelope" for the generators to operate within rather than just output limits – is outside the scope of this work. The issue of PFM-CSP applying excessive curtailment can be resolved more simply by increasing the number of values in the domain to make the output limits less coarse. This has a significant performance drawback, however, due to the poor scalability characteristics of PFM-CSP to increased domain sizes and numbers of generators, as noted in Section 2.2.3. For example, for the IEEE 14-bus system with four generators, changing the domains from $\{0\%, 50\%, 100\%\}$ to $\{0\%, 25\%, 50\%, 75\%, 100\%\}$ increases the worst-case execution time by a factor of $5^4 \ / \ 3^4 \approx 7.72$.

4.6.4 Analysis of algorithm performance: PFSF-based algorithms

These three algorithms all use PFSFs and share some common code, with PFSF-Egal and PFSF-TMA only differing on a few key lines of their process (see Section 2.3.4). As these lines have a common effect on their performance, these algorithms are analysed together in this section. This analysis makes reference to Table 4.9, which tabulates a number of metrics regarding the performance of these three algorithms across the four case study systems, including the number of states where the algorithms report errors during their process. The tabulated metrics are numbered for ease of reference.

PFSFs linearise the power flow equations around a particular operating point and can be manipulated using basic arithmetic operations, so can speed up execution when compared with algorithms that use a full non-linear representation of the system. However, because PFSFs are a linearisation of a non-linear system around one state, they are approximations and can become inaccurate when used to calculate power flows for states that deviate from the initial state. The effect of this is that the PFSF-based algorithms, which use a common set of PFSFs that are calculated offline for a single state in each system, may overestimate the change in power flows for particular curtailment values, and thus leave overloads remaining in the system under control. Similarly, the algorithms may also underestimate the effect of some curtailments, applying more curtailment than necessary to remove an overload.

The algorithms employ various design features to mitigate the approximation inherent in using PFSFs. The first is to adjust ratings to 99% within their internal processes so that

Table 4.9 Additional metrics related to the performance of the PFSF-based algorithms

	1. Number of overloads			
Algorithm	11 kV radial	33 kV meshed	IEEE 14-bus	IEEE 57-bus
PFSF-Egal	0	3594	66	24823
PFSF-TMA	0	1903	60	24247
PFSF-LP	0	1819	1172	12863

	2. States with overloads			
Algorithm	11 kV radial	33 kV meshed	IEEE 14-bus	IEEE 57-bus
PFSF-Egal	0	3594	66	5165
PFSF-TMA	0	1903	60	4995
PFSF-LP	0	1819	1029	5262

	3. States with overloads and curtailment			
Algorithm	11 kV radial	33 kV meshed	IEEE 14-bus	IEEE 57-bus
PFSF-Egal	0	3406	56	503
PFSF-TMA	0	1715	50	951
PFSF-LP	0	1631	1019	5239

	4. States with overloads but no curtailment			
Algorithm	11 kV radial	33 kV meshed	IEEE 14-bus	IEEE 57-bus
PFSF-Egal	0	188	10	4662
PFSF-TMA	0	188	10	4044
PFSF-LP	0	188	10	23

	5. States with overloads, no curtailment, and algorithm error			
Algorithm	11 kV radial	33 kV meshed	IEEE 14-bus	IEEE 57-bus
PFSF-Egal	0	0	0	4639
PFSF-TMA	0	0	0	4021
PFSF-LP	0	0	0	0

	6. States with overloads, no curtailment, but no algorithm error			
Algorithm	11 kV radial	33 kV meshed	IEEE 14-bus	IEEE 57-bus
PFSF-Egal	0	188	10	23
PFSF-TMA	0	188	10	23
PFSF-LP	0	188	10	23

	7. States with no overloads but with curtailment			
Algorithm	11 kV radial	33 kV meshed	IEEE 14-bus	IEEE 57-bus
PFSF-Egal	2546	3186	4017	297
PFSF-TMA	2546	4877	4023	467
PFSF-LP	2546	4961	3054	200

	8. Mean remaining overload energy versus baseline (%) for overloaded states with curtailment			
Algorithm	11 kV radial	33 kV meshed	IEEE 14-bus	IEEE 57-bus
PFSF-Egal	–	10.25	17.78	6.95
PFSF-TMA	–	17.14	19.69	3.86
PFSF-LP	–	17.66	3.71	37.23

the algorithms will tend to overestimate the amount of curtailment needed, which can help remove overloads when the PFSFs overestimate the effect of curtailments. The second design feature is to run a full non-linear load flow to validate the curtailment values derived using PFSFs, which is used within PFSF-Egal and PFSF-TMA.

One design feature shared by these algorithms is that the logic used to detect overloads calculates the apparent power flowing through a branch from the average real and reactive power measured at both ends, rather than calculating the apparent power at both ends and comparing the maximum of these to the branch rating. This feature allows each branch to be associated with single values of real and reactive power, which simplifies the algorithms' logic and speeds up execution. For example, PFSF-LP could be adapted to consider power flow constraints at both ends of a branch, rather than looking at a constraint based on average flows; however, this would double the number of variables in the LP.

This design feature of the overload detection logic does not cause an issue for many of the tested states, due to the branch loading limit of 99% assumed to mitigate the approximation inherent in using PFSFs. This lowers the detection threshold, so the algorithms are more likely to correctly detect an overload, although, conversely, they may falsely detect an overload and apply curtailment when none is needed. This is quantified by metric 7 in Table 4.9, which shows the number of states within each system for which the algorithms falsely detect overloads and then apply curtailment. However, under certain conditions a branch may be overloaded at one or both ends, but the apparent power calculated from the average real and reactive powers at both ends may still be within the 99% limit used within the algorithms. Under these conditions, this design feature of the algorithms becomes a flaw as they will fail to detect overloads and thus no curtailment will be applied to remove the overloads. As shown in Table 4.9 (metric 6), for the 33 kV meshed distribution system, there are 188 states in which the PFSF-based algorithms fail to detect overloads; for the IEEE 14-bus system there are 10 such states; and for the IEEE 57-bus system there are 23. All overloads in the 11 kV radial distribution system are correctly detected.

For the 33 kV meshed distribution system and the IEEE 14-bus system, failure to detect overloads accounts for all the states where the PFSF-based algorithms do not apply any curtailment despite there being overloads. However, for the IEEE 57-bus system, there is a different aspect of designs of PFSF-Egal and PFSF-TMA that leads to these algorithms failing to apply curtailment when there are overloads, which is the iteration limit that ensures the algorithms execute in a reasonable time (described in Section 2.3.3). For a significant number of states (metric 5 in Table 4.9), the algorithms reach the iteration limit within their processes before the overloads within their internal models are removed, and then exit, reporting an error and applying no curtailment.

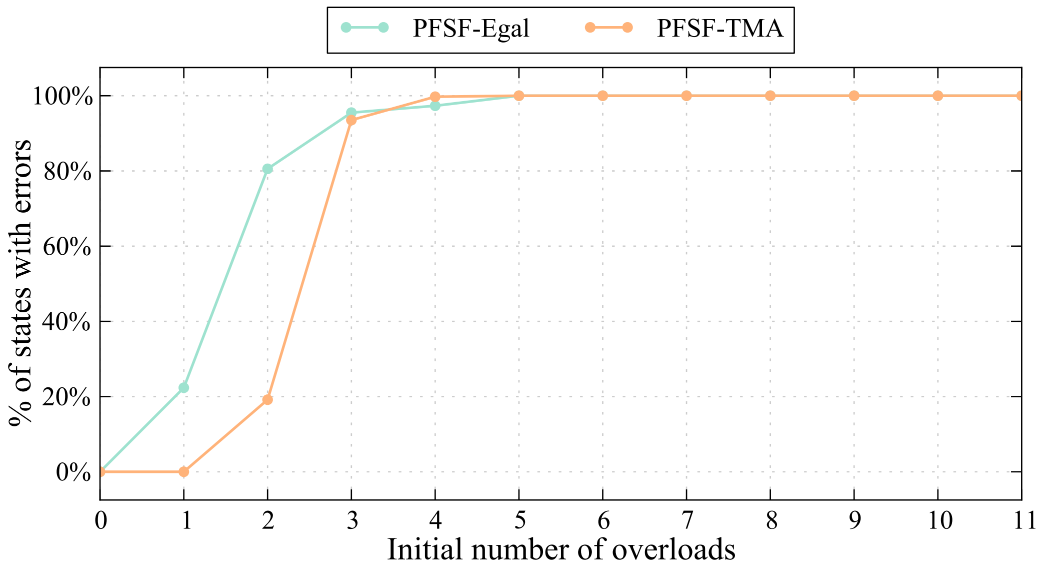


Fig. 4.15 Percentage of initially overloaded states in the IEEE 57-bus system where the PFSF-Egal and PFSF-TMA algorithms report errors

As can be seen in Figure 4.15, there is a link between the initial number of overloads in the system and the propensity for the algorithms to reach their iteration limit and report an error. PFSF-TMA is less likely than PFSF-Egal to reach its iteration limit and report an error for states with three or fewer overloads initially; however, for all states with 5 overloads or more, both algorithms always reach their iteration limits and report errors. The design of the algorithms is such that when the iteration limit is reached they will report an error and exit without applying curtailment. The consequence of this is that all overloads will remain in the system, resulting in the significant number of overloads for PFSF-Egal and PFSF-TMA on the IEEE 57-bus system, and making that the only system where these two algorithms are outperformed by PFSF-LP in terms of the reducing the number and energy of overloads.

Aside from the states in which overloads remain due to missed detection (metric 6 in Table 4.9) or algorithm error (metric 5), for the majority of states with overloads remaining the algorithms operate correctly according to their designs and apply some curtailment to the generators within the systems (metric 3). In these cases, the algorithms determine curtailments that are able to remove all overloads within their internal system models, but leave one or more overloads when applied to the actual system under control.

For the PFSF-LP algorithm, the difference between the internal solution and the external reality stems from the linearised form of the system response (the PFSFs) within the LP that are used to determine the curtailments. The curtailments calculated from the linearised system representation are directly applied to the non-linear system under control, so may incorrectly estimate the effect on power flows that the curtailments aim to achieve.

For the PFSF-Egal and PFSF-TMA algorithms, the curtailments calculated using a linearised representation of the system are then validated by performing an AC load flow using a full non-linear representation of the system under control. While this validation step should correctly determine the change in power flows due to the curtailments, the overload detection logic used in these algorithms may fail to detect marginal overloads that may remain in the internal models. As no overloads are detected in the internal model, the curtailments are then applied to the system under control, but overloads may still remain. However, overload energy will be reduced, as shown by metric 8 in Table 4.9.

4.6.5 Execution times

The execution times of the algorithms were measured in order to evaluate their suitability for real-time control operation, and also to understand the effect of the different case study systems. Figure 4.16 presents the execution times of the algorithms for the overloaded states in the four case study systems as box-and-whisker plots. In the figure, median execution times of the algorithms on each system are shown with a vertical black line with the value given underneath; mean execution times are shown as a black square; the 25th and 75th percentiles of the execution time data are shown by the extent of the boxes, while the 5th and 95th percentiles are shown by the whiskers. Outliers are plotted individually.

The execution times of the three PFSF-based algorithms lie within the same range and do not exceed 1.0 seconds, while the execution times of PFM-CSP and PFM-OPF are typically an order of magnitude higher for each case study system. The minimum median execution times for each algorithm occurs for the 11 kV radial distribution system, which is the simplest of the four case study systems as it contains the least buses, generators and potential overloads. With the exception of PFM-CSP, the maximum median execution times for each algorithm occurs for the IEEE 57-bus system, which has the most buses and potential overloads of all the case study systems. PFM-CSP has its maximum median execution time on the 33 kV meshed distribution system. Although this is not the largest system, it does feature four generators and PFM-CSP scales poorly to increased numbers of generators. The IEEE 14-bus system has the same number of generators, so the lower median execution time for that system must be because it is quicker to execute a load flow on the IEEE 14-bus system model internal to PFM-CSP compared with the internal model of 33 kV system.

The execution times on each system for PFSF-Egal and PFSF-TMA follow a similar pattern to each other, which is to be expected as those algorithms follow the same processes for the majority of their execution. PFSF-LP has the lowest median execution times for each

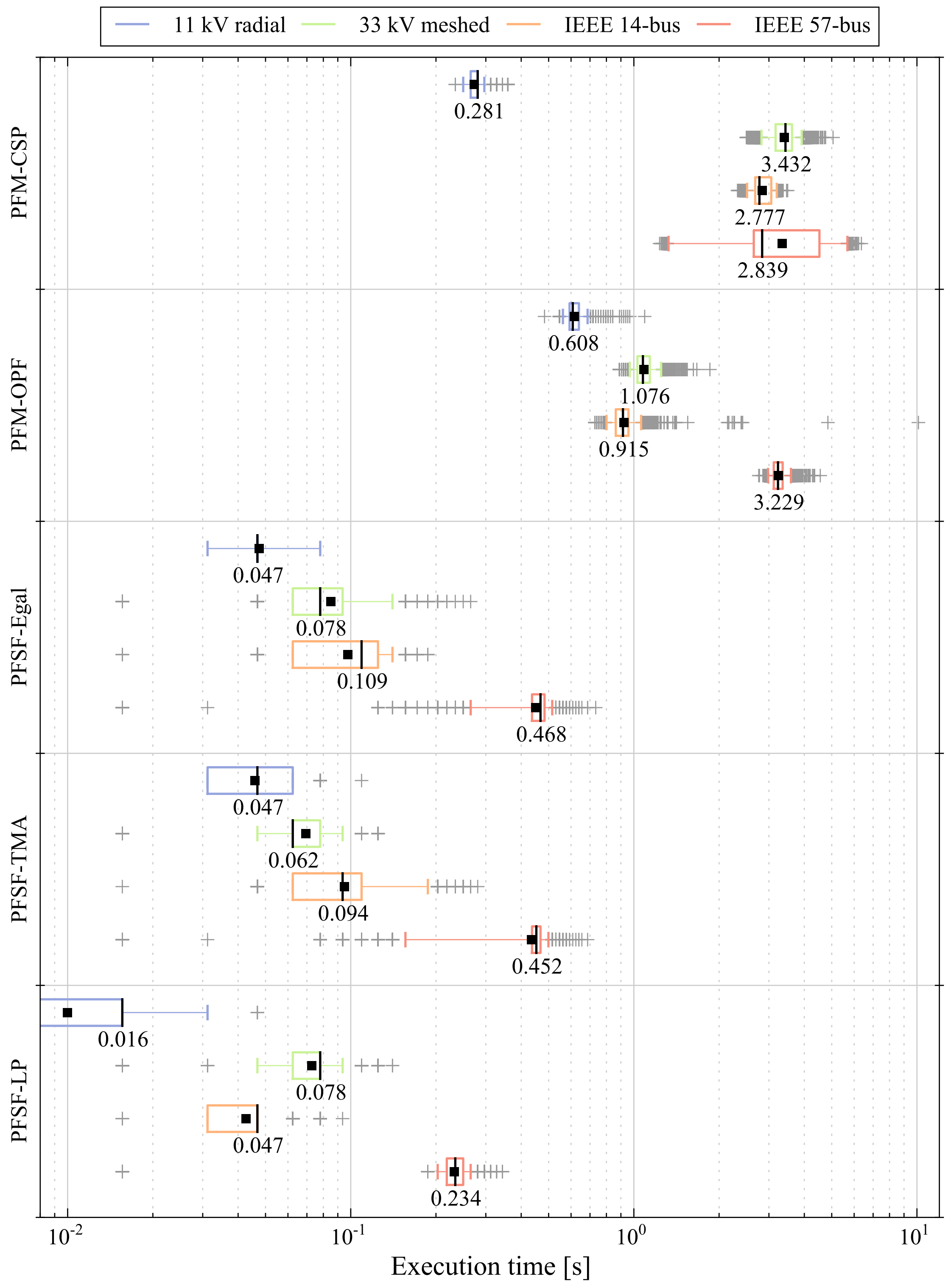


Fig. 4.16 Algorithm execution times for the overloaded states within each case study system

system except on the 33 kV system, where PFSF-TMA is the fastest algorithm based on median execution time.

This analysis shows that the execution times of the algorithms are relatively small when compared with the length of time each state represents. The PFSF-based algorithms are, on average, and order of magnitude faster than PFM-OPF and PFM-CSP. Although there is variation in the execution times observed across the systems, the algorithms would need to be tested on a larger number of systems in order to determine what characteristics of power systems influence changes in execution times.

4.7 Conclusions

In this chapter, five different power flow management algorithms (as described in Chapter 2) have been applied to four case study power systems, which represent different network topologies (radial, meshed), voltage levels (transmission, distribution) and geographies (UK, US). For each case study system, at least 10,000 different system states have been simulated, with many featuring overloads, to which the algorithms were applied and tested.

PFM-OPF was found to be the most effective algorithm overall for most of the systems, with respect to minimising the number and energy of overloads whilst also minimising the amount of curtailment. However, for the IEEE 14-bus system, PFSF-TMA was more effective at reducing overload energy, whilst minimising the amount of curtailment; and for the 11 kV radial distribution system, PFM-OPF could only provide a performance benefit in terms of reducing the curtailment applied, as there was no statistically significant difference in the number or energy of overloads removed by each of the power flow management algorithms tested. For all of the case study systems, the algorithms were found to execute in a reasonable time when compared with the time period that each simulated state represented; although additional systems would need to be tested in order to draw any empirical conclusions about the scalability of the algorithms.

The performance of each algorithm across all of the case study systems was examined. Design features of the algorithms which were most likely the causes of particular performance traits were identified, and the findings regarding these features could be helpful for research developing improved and novel power flow management algorithms.

This chapter has established the performance of the power flow management algorithms when aggregated across *all* the states simulated. In the next chapter, the performance of the algorithms is examined for *each* state, in order to understand if any potential performance benefits can be obtained by selecting different algorithms on a per-state basis.

Chapter 5

Potential performance benefits from per-state selection of algorithms

The previous chapter took the five power flow management algorithms introduced in Chapter 2 and assessed their performance on aggregate across *all* states tested for each of the four case study systems. This chapter aims to complete research objective 1, by examining the performance of the algorithms for *each* state individually, in order to determine if there is any *potential* performance benefit if algorithms were selected on a *per-state* basis, rather than selecting one algorithm to be used for all states.

This chapter is structured as follows: Section 5.1 presents the method used to determine the potential benefit that could be obtained from per-state algorithm selection, Section 5.2 applies that method to assess the potential benefits for each of the case study systems, Section 5.3 examines how frequently each algorithm is the most effective for each system, Section 5.4 examines how the potential performance benefits vary if different sets of algorithms are considered, and Section 5.5 concludes the chapter.

Some of the work in this chapter has been published previously. The potential performance benefits from per-state selection for the 33 kV meshed distribution system were published in [58], the benefits for the IEEE 14-bus system were in [59], while in [60] the benefits for both of those systems and the IEEE 57-bus system were presented.

5.1 Method for assessing the potential performance benefit

The potential performance benefit that could be obtained by selecting algorithms on a per-state basis is calculated using a two-step process, which is repeated for each of the overload performance measures of 1) the number of overloads, and 2) the energy of overloads:

1. First, the optimal algorithm selections are determined for each state. This is achieved by post-processing the performance data of all algorithms, and determining, for each state, the set of algorithms that minimises either the number or energy of overloads for that state. Considering that set of algorithms only, the subset that minimises curtailment is determined, referred to as the “selection set”. If the selection set is a singleton, then there is only a single most effective algorithm for the state with respect to minimising either the number or energy of overloads, while also minimising curtailment; and this algorithm represents the optimal selection. However, if a number of algorithms remain in the selection set, then which represents the optimal selection is arbitrary as the overload and curtailment performances of the remaining algorithms are identical.

2. Second, the optimal selections for each state are used to extract the performance data of the selected algorithms. This performance data is aggregated to give the theoretical limit for the performance that can be obtained by optimally selecting between the considered set of algorithms on a per-state basis. This performance can be compared with the performance of the individual algorithms to determine if there is any potential performance benefit from selecting different algorithms for each state.

As the sequence of selections is made post-hoc with perfect information about how all the algorithms perform on each state, the selection are referred to as being made by “oracles”, as the selections are always optimal for each state with respect to the performance measures considered. Performance results for two oracles are presented: 1) an oracle that minimises the number of overloads, then the amount of curtailment, and 2) an oracle that minimises the energy of overloads, then also minimises the amount of curtailment.

In the reporting of results for each case study system, the number of times each algorithm appears in the selection sets of each oracle is provided, along with the number of times each algorithm is the sole algorithm in the selection set (when the set is a singleton). This indicates how often each algorithm is uniquely the most effective algorithm.

In the subsequent analysis, the performance of the algorithms is compared to the potential performance that the oracles would allow. The statistical significance of any performance differences are assessed using the same method as used in the previous chapter for comparing the performance of the algorithms, as described in Section 4.1.3. In summary, for each system, the distributions of performance for every performance measure and pair-wise combination of algorithms (including the oracles) are compared using a paired t-test, yielding p-values. The Benjamini-Hochberg procedure is then used to determine which p-values (and therefore which differences in performance) are statistically significant, while correcting for making multiple comparisons.

Table 5.1 Overview of algorithm performance and the potential performance from optimally selecting algorithms on a per-state basis for the 11 kV radial distribution system

	performance					
Algorithm	Over-loads [count]	Over-loaded states	Overload energy [MVAh]	Total cur-tailment [MWh]	Algorithm in selection set (sole algorithm in set) Oracle 1	Oracle 2
Baseline	3344	2462	871.95	0.00	7538 (0)	7538 (0)
PFM-CSP	0	0	0.00	2121.40	7538 (0)	7538 (0)
PFM-OPF	2	2	0.10	769.94	9997 (2459)	9997 (2459)
PFSF-Egal	0	0	0.00	856.20	7454 (0)	7454 (0)
PFSF-TMA	0	0	0.00	891.01	7455 (1)	7455 (1)
PFSF-LP	0	0	0.00	773.33	7456 (2)	7456 (2)
Oracle 1	0	0	0.00	770.08	–	–
Oracle 2	0	0	0.00	770.08	–	–

5.2 Potential performance benefits for each system

5.2.1 11 kV radial distribution system

Table 5.1 summarises the performance of the power flow management algorithms and the potential performance that would be achieved if, for each state, only the most effective algorithms were selected and used. As explained in Section 5.1, the potential performance from optimally selecting algorithms on a per-state basis is represented by two "oracles", with oracle 1 optimally selecting algorithms that minimise the number of overloads for a state, and oracle 2 optimally selecting algorithms that minimise the overload energy for a state. When more than one algorithm gives the optimal performance, minimisation of curtailment is additionally taken into account for the oracles.

As well as showing performance, Table 5.1 also shows (in the right-hand columns) the frequency of each algorithm appearing in the selection sets of each oracle. In other words, this is the number of states that each algorithm is the amongst the most effective. For each algorithm, two figures are given per oracle: the first is the frequency of the algorithm appearing in the oracle's selection set; whereas the second figure (in parentheses) is the frequency of the algorithm appearing in the selection set when the set is a singleton, therefore, when the algorithm is *uniquely* the most effective. For example, PFM-OPF (the third row) appears in the selection sets of oracle 1 for almost all states (9997 out of 10,000), which means it is the most effective algorithm at minimising the number of overloads, while minimising curtailment, for those 9997 states. It can be inferred from the table that for many of those states, other algorithms give the same performance as PFM-OPF and are therefore also the most effective. However, for 2459 states, PFM-OPF is the uniquely the most effective, with all the other algorithms giving worse performance.

For this system, both oracles give identical performance. Compared with the single best algorithm, PFM-OPF, the oracles allow all overloads to be removed – giving identical performance to all the other algorithms except PFM-OPF – although this is at the expense of applying more curtailment. These differences in performance come from only 3 states where PFM-OPF is not in the selection set, and is thus not an optimal algorithm to select. Two of these states are when PFM-OPF fails to remove overloads and PFSF-LP is selected, whereas the other state is when PFSF-TMA applies less curtailment when removing an overload.

Comparing the distribution of performance of the oracles against PFM-OPF using the paired t-test gives p-values of 0.1573, 0.1627, and 0.1616, respectively, for the number of overloads, the overload energy, and the curtailment applied. These fail to reject the null hypothesis that the distributions of performance are the same, following application of the Benjamini-Hochberg procedure. Therefore, the performance differences between the oracles and PFM-OPF are not statistically significant with respect to overload and curtailment performance. Therefore, selecting algorithms on a per-state basis does not offer a potential performance benefit for this case study system.

5.2.2 33 kV meshed distribution system

Table 5.2 is similar to Table 5.1 and summarises the performance of the power flow management algorithms for the 17,520 states tested within the 33 kV meshed distribution system. Also shown is the potential performance that could be achieved if the algorithms are optimally selected for each state, represented by the two oracles (see Section 5.1), and the frequency of each algorithm appearing in the selection sets of the oracles.

The potential performance represented by the two oracles is identical for all the performance measures, and is the same as PFM-OPF with respect to overloads. As PFM-OPF is able to remove all overloads, there is no potential performance benefit from selecting algorithms on a per-state basis with respect to minimising the number or energy of overloads.

However, if algorithms are optimally selected on a per-state basis, potentially 0.91% less curtailment can be applied than PFM-OPF. This potential performance benefit comes from 2158 states (32.12% of the states with overloads) where PFM-OPF is not in the selection set and alternative algorithms are the most effective, removing the same number of overloads as PFM-OPF but doing so with less curtailment applied to the generators. For 1069 (49.54%) of these states, PFSF-TMA is the most effective algorithm, while PFSF-LP is most effective for 799 (37.03%), PFM-CSP for 82 (3.80%), and two or more algorithms are the most effective for each of the remaining 208 states (9.64%).

Table 5.2 Overview of algorithm performance and the potential performance from optimally selecting algorithms on a per-state basis for the 33 kV meshed distribution system

Algorithm	Over-loads [count]	Over-loaded states	Overload energy [MVAh]	Total cur-tailment [MWh]	Algorithm in selection set (sole algorithm in set)	
					Oracle 1	Oracle 2
Baseline	14779	6719	9445.74	0.00	10801 (0)	10801 (0)
PFM-CSP	194	194	10.56	76910.75	10883 (82)	10883 (82)
PFM-OPF	0	0	0.00	40246.45	15362 (4561)	15362 (4561)
PFSF-Egal	3594	3594	78.25	59905.56	10948 (0)	10948 (0)
PFSF-TMA	1903	1903	40.41	39694.49	12017 (1069)	12017 (1069)
PFSF-LP	1819	1819	36.48	39743.66	11539 (799)	11539 (799)
Oracle 1	0	0	0.00	39879.06	–	–
Oracle 2	0	0	0.00	39879.06	–	–

Comparing the distribution of curtailment that would be applied by the oracles against that of PFM-OPF using the paired t-test gives p-values < 0.0001. After applying the Benjamini-Hochberg procedure, these p-values successfully reject the null hypothesis and therefore the performance differences are statistically significant. Based on these results, algorithm selection on a per-state basis does potentially offer a statistically significant performance benefit for this case study system. This benefit is in terms of reducing the amount of curtailment applied to the generators, as no performance gain is possible in terms of reducing the number or energy of overloads.

5.2.3 IEEE 14-bus system

Table 5.3 summarises the performance of the power flow management algorithms across the 10,000 states tested within the IEEE 14-bus system, along with the performance achieved if the algorithms are optimally selected for each state, represented by the oracles. Also shown is the frequency of each algorithm being in the selection sets of each oracle.

Out of the 4008 states with overloads, there is only a single state where the selection set contains more than one algorithm, in this case PFSF-TMA and PFSF-LP. In the remaining 4007 states the selection set is a singleton. For 58.10% of these states, PFSF-LP is the most effective algorithm, in terms of minimising the number and energy of overloads, while also minimising curtailment, despite it having the worst performance overall against all the performance measures. All the other algorithms feature at least once in a singleton selection set, with PFSF-TMA being the algorithm that is second most frequently selected on its own (21.41%) compared with PFSF-LP, followed by PFM-OPF (18.64%), PFM-Egal (1.47%), and PFM-CSP (0.37%).

Table 5.3 Overview of algorithm performance and the potential performance from optimally selecting algorithms on a per-state basis for the IEEE 14-bus system

Algorithm	Over-loads [count]	Over-loaded states	Overload energy [MVAh]	Total cur-tailment [MWh]	Algorithm in selection set (sole algorithm in set)	
					Oracle 1	Oracle 2
Baseline	5345	4008	9116.25	0.00	5992 (0)	5992 (0)
PFM-CSP	45	45	11.01	58855.10	6007 (15)	6007 (15)
PFM-OPF	24	11	49.97	19544.04	6739 (747)	6739 (747)
PFSF-Egal	66	66	1.75	33900.57	5976 (59)	5976 (59)
PFSF-TMA	60	60	1.57	21797.45	6776 (858)	6776 (858)
PFSF-LP	1172	1029	105.55	19503.16	8246 (2328)	8246 (2328)
Oracle 1	0	0	0.00	19311.40	–	–
Oracle 2	0	0	0.00	19311.40	–	–

If the algorithms are selected optimally for each state, it is possible to remove all overloads, outperforming each of the algorithms if only one is used for all states. Furthermore, it is also potentially possible to reduce the amount of curtailment applied by at least 0.98%, which also outperforms each of the algorithms. Comparing the distributions of performance for each performance measure of each algorithm against the the potential performance represented by the oracles using the paired t-test yields p-values < 0.01. After applying the Benjamini-Hochberg procedure, each of these p-values successfully reject the null hypothesis that the distributions of performance are the same. Therefore, the potential performance benefits are statistically significant.

For the IEEE 14-bus system, selecting algorithms on a per-state basis clearly offers potential performance benefits, both in terms of minimising the number and energy of overloads, but also in terms of minimising the amount of curtailment applied.

5.2.4 IEEE 57-bus system

Table 5.4 summarises the performance of the power flow management algorithms when applied to the 10,000 test states within the IEEE 57-bus system, and the performance achieved if the algorithms are optimally selected for each state, represented by the oracles. Also shown is the frequency of each algorithm being in the selection sets of each oracle.

This is the only case study system for which the potential performance represented by the oracles differs, which is due to there being a number of states for which no algorithm can remove all overloads. If there are several algorithms that can minimise the number of overloads for a state, but none of the algorithms can remove all overloads, the energy of remaining overloads and the amount of curtailment applied by the algorithms may differ.

Table 5.4 Overview of algorithm performance and the potential performance from optimally selecting algorithms on a per-state basis for the IEEE 57-bus system

Algorithm	Over-loads [count]	Over-loaded states	Overload energy [MVAh]	Total curtailment [MWh]	Algorithm in selection set (sole algorithm in set) Oracle 1	Algorithm in selection set (sole algorithm in set) Oracle 2
Baseline	25377	5455	678836.34	0.00	4546 (0)	4545 (0)
PFM-CSP	3660	2882	30389.77	982967.03	5736 (1190)	5308 (763)
PFM-OPF	1367	1358	1241.44	749899.56	8364 (3819)	8784 (4239)
PFSF-Egal	24823	5165	670366.33	30780.23	4538 (0)	4546 (8)
PFSF-TMA	24247	4995	653798.21	43312.05	4792 (254)	4792 (254)
PFSF-LP	12863	5262	241625.85	405575.30	4729 (191)	4729 (191)
Oracle 1	768	760	4352.19	821087.30	–	–
Oracle 2	769	760	985.14	826709.24	–	–

Generally, increased curtailment results in a reduction in overload energy, so the selection sets of the oracles will differ as oracle 1 represents the case where the optimal algorithms to select are those that minimise the *number* of overloads and then minimise the amount of curtailment, whereas oracle 2 represents the case where the optimal algorithms to select are those that minimise the overload *energy* and then minimise the amount of curtailment.

As Table 5.4 shows, the number of overloads can be substantially reduced by optimally selecting algorithms on a per-state basis. The potential performance represented by oracle 1 allows for a reduction in the number of overloads by 43.82% compared with PFM-OPF, which is the most effective algorithm for that performance measure. Using the paired t-test to compare the distributions of this performance measure between each oracle and each algorithm results in p-values < 0.0001. Following the application of the Benjamini-Hochberg procedure, these p-values successfully reject the null hypothesis of no difference in performance, so the differences in performance are statistically significant. Although selecting algorithms for each state potentially allows each of the power flow management algorithms to be outperformed with respect to minimising the number of overloads, the differences between performances represented by the oracles for this performance measure are not statistically significant, as comparing the performance using the paired t-test yields a p-value of 0.3173, which fails to reject the null hypothesis.

There is a statistically significant difference in the performance represented the oracles with respect to overload energy. Indeed, applying the t-test to compare the performance for this measure of all pair-wise combinations of algorithms and oracles yields p-values of < 0.0001, all of which reject the null hypothesis of the performances being the same following the application of the Benjamini-Hochberg procedure. The optimal selections represented by oracle 2 would allow the overload energy to be reduced by 20.65% compared with PFM-

OPF, which is the most effective algorithm with respect to minimising this performance measure. However, the optimal selections represented by oracle 1 would result in an increase in overload energy by 250.58% compared with PFM-OPF, as the minimisation of the amount of curtailment is prioritised once the number of overloads has been minimised.

If the algorithms are selected optimally on a per-state basis to minimise overloads (number or energy) and then curtailment, at least 9.49% more curtailment would be applied compared with PFM-OPF. However, this extra curtailment translates into the removal of more overloads. Furthermore, the relative amount of curtailment applied per overload removed is not dissimilar that of PFM-OPF, as the optimal selections represented by oracle 1 would allow 24,609 overloads to be removed, compared with 24,010 removed by PFM-OPF, which translates into 33.37 and 31.23 MWh of curtailment per overload removed, respectively.

The optimal selections represented by the oracles differ in the curtailment that would be applied. The selections represented by oracle 1 would results in 0.68% less curtailment than the selection represented by oracle 2, with the difference due to the different prioritisation of objectives in the selections. Comparing the distributions of curtailment represented by the oracles to each other and to each of the algorithms yields p-values of < 0.0001. This p-value rejects null hypothesis that the distributions of performance are the same, following the application of the Benjamini-Hochberg procedure. Therefore the differences in curtailment are statistically significant.

The right-hand side of Table 5.4 shows the frequency of each algorithm appearing in the selection sets of each oracle, including the frequency that each algorithm is the sole algorithm in the selection set (when there is a singleton selection set). For oracle 1, only 1 of the 5455 states with overloads has a non-singleton selection set, in which no algorithm can reduce the number of overloads in the system. All overloaded states have a singleton selection set for oracle 2, meaning that for each of these states there is only a single most effective algorithm at minimising the overload energy, while also minimising the amount of curtailment.

PFM-OPF most frequently appears in the selection sets of both oracles, followed by PFM-CSP, PFSF-TMA and PFSF-Egal. The major difference in selection frequencies between the oracles is a shift from PFM-CSP for oracle 1 to PFM-OPF for oracle 2. PFM-CSP is selected for 427 fewer states by oracle 2, whereas PFM-OPF is selected for 420 more states. For the vast majority of these states, the two algorithms remove the same number of overloads; however, PFM-CSP does so with less curtailment (hence is selected by oracle 1), whereas PFM-OPF leaves less overload energy (hence is selected by oracle 2). For similar reasons, PFSF-Egal only appears in the selection sets of oracle 2.

For this case study system there is a statistically significant performance benefit from selecting between the algorithms on a per-state basis, as the number and energy of overloads

Table 5.5 Performance comparison of the most effective algorithms for each system against the potential performance that could be obtained by per-state algorithm selection

Values in bold indicate a statistically significant difference between the distribution of performance that could be achieved by per-state algorithm selection (represented by the oracles) and the algorithm each is compared to, after using the Benjamini-Hochberg procedure.

System	Oracle or algorithm for comparison	Number of overloads [count]	Overload energy [MVAh]	Total curtailment [MWh]
11 kV radial	Oracle 1 & 2	0	0.00	770.08
	PFSF-LP	0	0.00	773.33
33 kV meshed	Oracle 1 & 2	0	0.00	**39879.06**
	PFM-OPF	0	0.00	40246.45
IEEE 14-bus	Oracle 1	**0**	**0.00**	**19311.40**
	PFM-OPF	24	49.97	19544.04
	Oracle 2	**0**	**0.00**	**19311.40**
	PFSF-TMA	60	1.57	21797.45
IEEE 57-bus	Oracle 1	**768**	**4352.19**	**821087.30**
	PFM-OPF	1367	1241.44	749899.56
	Oracle 2	**769**	**985.14**	**826709.24**
	PFM-OPF	1367	1241.44	749899.56

remaining in the system are significantly reduced. This does result in increased curtailment compared with the most effective algorithm (PFM-OPF), although the relative amount of curtailment applied per overload removed is not dissimilar.

5.2.5 Summary for all case study systems

Table 5.5 summarises the potential performance for each case study system that could be achieved if the algorithms are selected optimally on a per-state basis. The oracles represent the potential performance that could be achieved if the optimal selections minimise the number (oracle 1) or energy (oracle 2) of overloads, whilst minimising curtailment. The potential performances are compared with the algorithms that are most effective overall with respect to the same objectives used when determining the optimal selections. Where the performance represented by the two oracles is the same, and one algorithm is most effective at both reducing the number and energy of overloads – such as for the 11 kV and 33 kV systems – then the results are grouped together to save duplication.

Bold values in Table 5.5 indicate where there is a statistically significant difference in the distribution of performance between the potential performance from per-state algorithm selection and the algorithm it is compared against. For example, for the IEEE 14-bus system the potential performance that could be obtained if algorithms are optimally selected for each state to minimise the number of overloads, whilst also minimising curtailment (represented by oracle 1), is compared with PFM-OPF, which is the most effective algorithm overall for those objectives. The bold values for performance represented by oracle 1 indicate that the potential performance if algorithms are optimally selecting on a per-state basis is statistically significantly different to the performance of PFM-OPF.

Figure 5.1 presents the data from Table 5.5 in terms of the potential performance gain from per-state algorithm selection, relative to the performance of the algorithms that are most effective overall. As in the table, the potential performance represented by each oracle is compared against the algorithms that are most effective overall with respect to the same objectives as used when determining the optimal selections. The result of this comparison is shown by the percentage performance gain (or loss). Positive values represent improved performance; for example, if algorithms are optimally selected on a per-state basis for the IEEE 14-bus system, there is potential to remove 100% of the overloads that the most effective algorithms would otherwise leave in the system.

For the 11 kV radial distribution system, all the algorithms tested except PFM-OPF could remove all overloads. Amongst these, PFSF-LP applied the least curtailment, so it is used to compare with the performance represented by the oracles. As it is possible to remove all overloads with a single algorithm, there is no performance gain in terms of reducing the number or energy of overloads. Per-state algorithm selection could potentially result in a 0.42% reduction in the amount of curtailment applied to achieve no overloads; however, as explained in Section 5.2.1, this is not a statistically significant performance improvement.

For the 33 kV meshed distribution system the potential performance from per-state algorithm selection is compared against PFM-OPF, as it was able to remove all overloads. Due to this, there is no potential performance gain in terms of reducing the number or energy of overloads. There is, however, the potential for a small but statistically significant improvement in the amount of curtailment applied, which can be reduced by 0.91% if algorithms are selected optimally for each state.

For the IEEE 14-bus system, every algorithm leaves some overloads remaining, but optimally selecting algorithms for each state can allow for all overloads to be removed. The potential performance represented by oracle 1 is compared with PFM-OPF as that algorithm minimised the number of overloads, whereas the potential performance represented by oracle 2 is compared with PFSF-TMA, which is the algorithm that minimised overload energy. Algorithm selection on a per-state basis potentially offers a statistically significant performance benefit in terms of minimising the number and energy of overloads on the

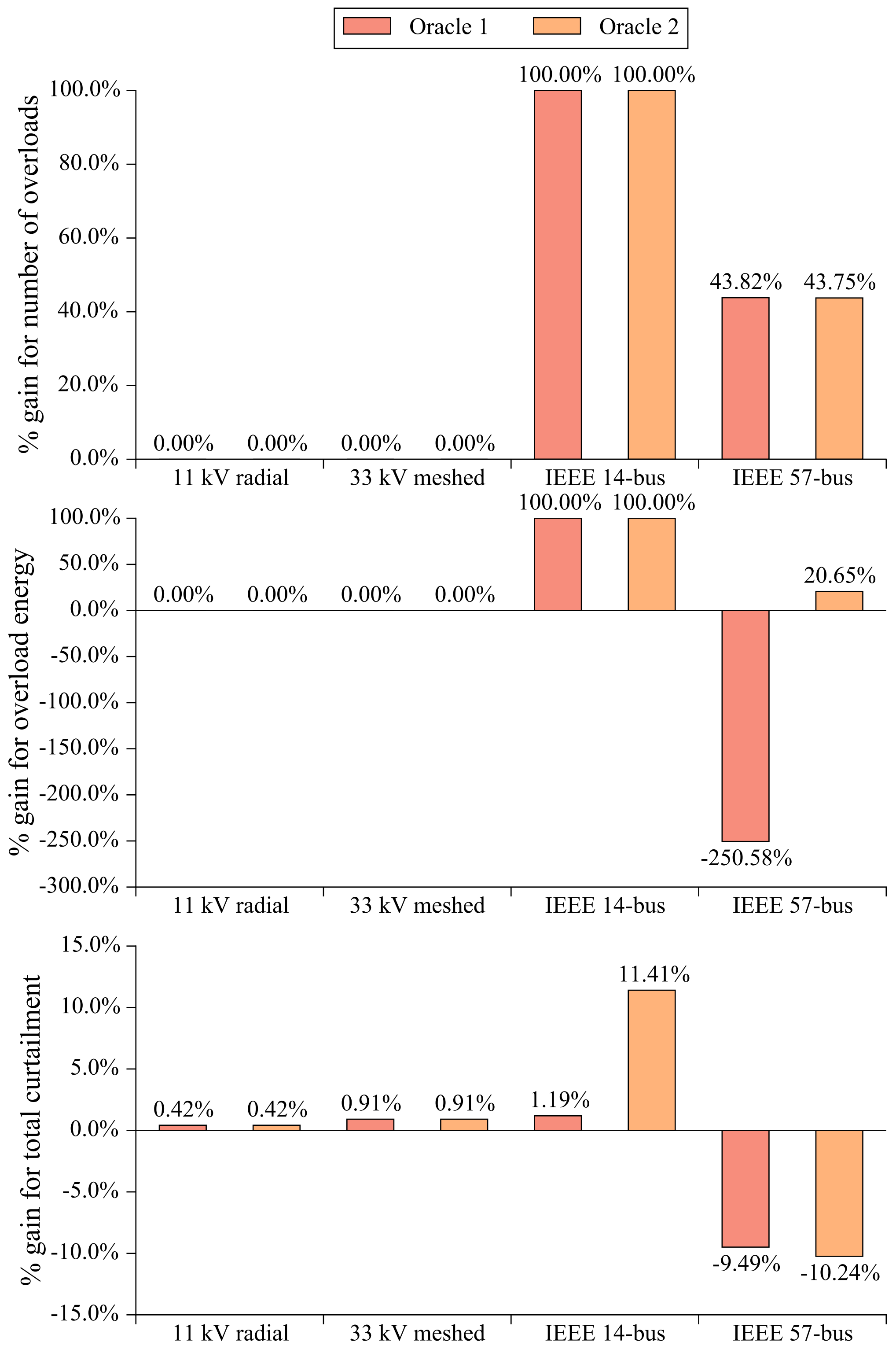


Fig. 5.1 Potential performance gains for each system that could be achieved by optimally selecting algorithms on a per-state basis

IEEE 14-bus system. Additionally, per-state algorithm selection can also allow statistically significantly less curtailment to be applied compared with the most effective algorithms for the two overload performance measures.

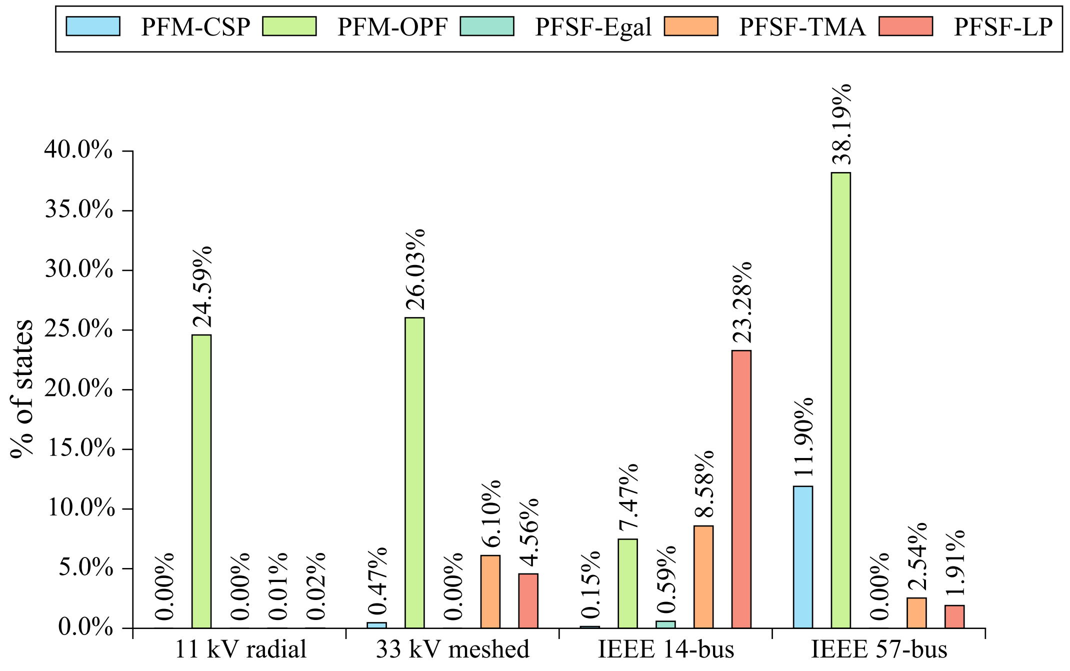
Similar to the IEEE 14-bus system, for the IEEE 57-bus system no algorithm can remove all overloads. The potential performances represented by both oracles are compared with PFM-OPF, as it is the most effective algorithm with respect to both minimising the number and energy of overloads. Per-state algorithm selection can potentially reduce the number of overloads by up to 43.82% and the overload energy by up to 20.65%, although it is not possible to remove all overloads completely. If minimising the number of overloads is prioritised (oracle 1), there is potentially a significant performance loss in terms of overload energy, with a 250.58% increase compared to PFM-OPF. This is because the optimal selections represented by oracle 1, though they minimise the number of overloads, also minimise the amount of curtailment applied, which can lead to larger overloads remaining for states where no algorithm can remove the overloads. However, if the optimal selections prioritise minimising the overload energy (oracle 2), there is a potential performance gain for both the number and energy of overloads, with the performance gain for the number of overloads being almost exactly the same as that represented by oracle 1. Per-state algorithm selection does not offer a potential performance benefit regarding the amount of curtailment applied, although this can be expected as reductions in overloads is associated with the application of additional curtailment.

5.3 Algorithm selection frequency

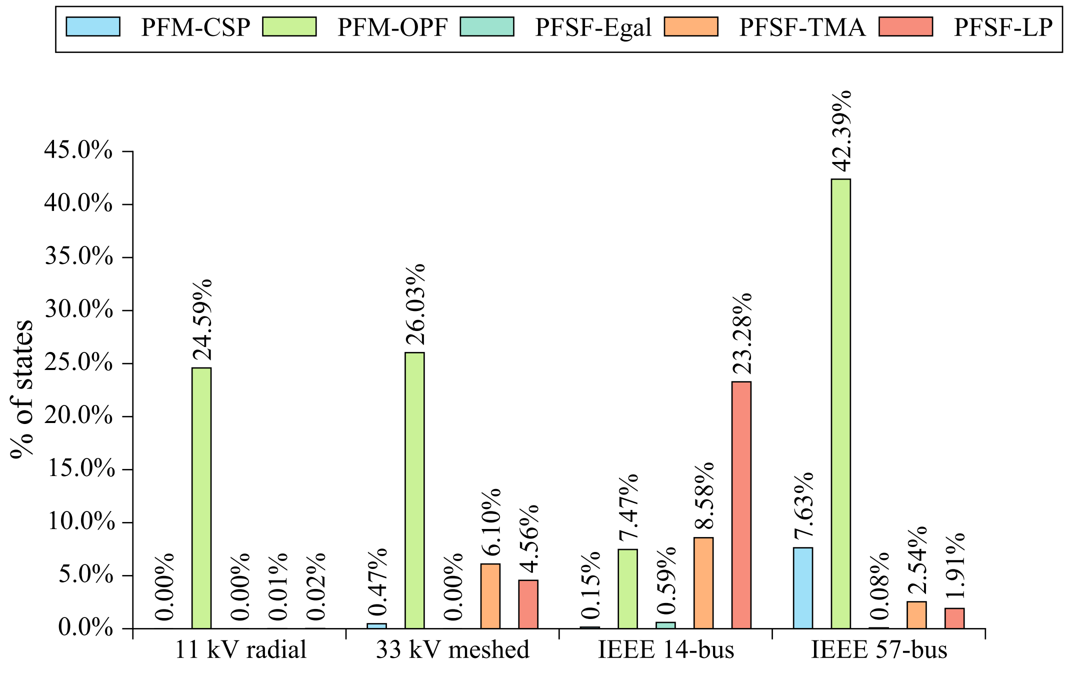
In Sections 5.2.1 to 5.2.4, the frequency of each algorithm being the most effective for individual states was reported for each case study power system. In this section, the frequency of each algorithm being the most effective – in other words, the frequency of each algorithm being the optimal algorithm to select – is analysed across all of the systems, in order to discern any trends in the selection frequency.

Figures 5.2a and 5.2b show, for each case study system, the percentage of states for which each algorithm appears individually in the selection sets represented by oracle 1 and oracle 2, respectively. These are for the states where there is a singleton selection set, when only one algorithm is uniquely most effective at minimising the number (or energy) of overloads with the least curtailment applied.

PFM-OPF, PFSF-LP and PFSF-TMA appear individually in the selection sets for each system, whereas PFM-CSP appears in the selection sets for three systems, and PFSF-Egal appears individually in the selection sets of only two systems.



(a) Oracle 1 (minimise number of overloads, then minimise curtailment)



(b) Oracle 2 (minimise energy of overloads, then minimise curtailment)

Fig. 5.2 Frequency of algorithms appearing as the sole algorithm in the selection sets represented by the oracles for each of the case study systems

PFM-OPF and is most frequently selected algorithm, across all systems except the IEEE 14-bus system, where PFSF-LP is the most frequently selected algorithm, despite being the least effective algorithm overall for that system. For the 11 kV radial distribution system, PFM-OPF is only absent from the selection sets of three states, where PFSF-LP or PFSF-TMA are selected instead.

The number of algorithms that appear individually in the selection sets varies across the systems: for the 11 kV radial distribution system, only 3 algorithms appear, for the 33 kV meshed distribution system, 4 algorithms appear, whereas all of the algorithms appear in the selection sets of the IEEE 14- and 57-bus systems. This suggests that some of the systems may have characteristics – perhaps relating to their scale or complexity – that particularly favour a more diverse range of algorithms.

The selection frequencies represented by the two oracles only differ for the IEEE 57-bus system, where minimising the number of overloads (represented by oracle 1) leads to PFM-CSP being selected more frequently than if the optimal selections are those that minimise overload energy (oracle 2).

This analysis shows that which algorithms are most effective on particular states (and appear in singleton selection sets) varies between the networks, and some algorithms are not solely most effective on any state for particular systems. For example, PFSF-Egal does not appear in the singleton selection sets for two of the case study systems, while some algorithms – such as PFM-OPF and PFSF-TMA – are selected on their own at least once for each system. Furthermore, the relative selection frequency of the algorithms varies across the systems, and the most frequently selected algorithm (for the singleton selection sets) may not necessarily be the most effective algorithm overall. For example, PFSF-LP is most frequently selected on its own for the IEEE 14-bus system, despite being the least effective algorithm overall for that system in terms of reducing the number and energy of overloads.

5.4 Performance with different sets of algorithms

The previous section looked at how frequently each algorithm appeared in the optimal selection sets, whereas this section considers the effect of considering different combinations of algorithms for selection. This allows the performance gain offered by the inclusion of each individual algorithm for selection to be assessed.

Figures 5.3, 5.4, and 5.5 show, for three of the case study systems, the differences in potential performance that could be obtained making the optimal algorithm selections for each state, but considering different sets of algorithms for selection. Each column in the figures represents different combinations of three algorithms from the set, with **bold** letters

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	14779	194	0	194	0	0	0
E T L	3594	3594	42	0	42	0	0	0
E **T** L	1903	1903	13	0	13	0	0	0
E T **L**	1819	1819	13	0	13	0	0	0
E **T** L	1903	1903	13	0	13	0	0	0
E T **L**	1819	1819	13	0	13	0	0	0
E **T** **L**	1819	1819	13	0	13	0	0	0
E **T** **L**	1819	1819	13	0	13	0	0	0

(a) Number of overloads [count]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	0.00	76910.75	40246.45	76894.78	40246.45	40230.24	40230.24
E T L	59905.56	57677.23	65971.97	40166.77	65956.11	40166.77	40150.83	40150.83
E **T** L	39694.49	39098.60	53549.92	39908.24	53538.15	39908.24	39893.69	39893.69
E T **L**	39743.66	39147.76	53214.32	39895.97	53202.56	39895.97	39881.42	39881.42
E **T** L	39694.49	39098.60	53549.92	39908.24	53538.15	39908.24	39893.69	39893.69
E T **L**	39737.32	39141.42	53214.32	39895.97	53202.55	39895.97	39881.42	39881.42
E **T** **L**	39727.48	39131.59	53211.91	39893.60	53200.14	39893.60	39879.06	39879.06
E **T** **L**	39727.48	39131.59	53211.91	39893.60	53200.14	39893.60	39879.06	39879.06

(b) Total curtailment applied [MWh]

Fig. 5.3 Comparison of the potential performance for the 33 kV meshed distribution system with different algorithm combinations considered for selection (oracle 1)

used to indicate which algorithms a column represents: baseline (the option to not apply any algorithm at all, denoted by "B"), PFM-CSP (denoted "C"), and PFM-OPF ("O"). Similarly, the rows represent different combinations of the remaining three algorithms: PFSF-Egal ("E"), PFSF-TMA ("T"), and PFSF-LP ("L"). Thus, each cell represents an intersection of these algorithm combinations and every possible combination of the five algorithms (plus the option of not applying any algorithm) is represented. For example, the second cell down in the third column of Figure 5.3a shows the number of overloads (42) that would occur in the 33 kV meshed distribution system if PFM-CSP ("C") and PFSF-Egal ("E") were optimally selected between on a per-state basis.

The shading of the cells in the figures shows the relative change in a performance measure with respect to the potential performance if all algorithms are optimally selected between. Cells coloured green (🟩) indicates improved performance, while red (🟥) indicates decreased performance. Note that, for the sake of brevity, similar figures for every system, performance measure, and oracle, are not shown here but can be found in Appendix C.

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	5345	45	24	45	24	0	0
E T L	66	66	0	0	0	0	0	0
E **T** L	60	60	0	1	0	1	0	0
E T **L**	1172	1169	16	1	16	1	0	0
E **T** L	42	42	0	0	0	0	0	0
E T **L**	46	46	0	0	0	0	0	0
E **T** **L**	45	45	0	0	0	0	0	0
E **T** **L**	42	42	0	0	0	0	0	0

(a) Number of overloads [count]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	0.00	58855.10	19544.04	58598.26	19544.04	19863.94	19863.94
E T L	33900.57	33855.96	33156.06	19652.35	33156.06	19652.35	19645.63	19645.63
E **T** L	21797.45	21735.54	22204.86	19544.02	22204.86	19544.02	19529.00	19529.00
E T **L**	19503.16	16188.67	32475.44	19471.93	32429.97	19471.93	19466.91	19466.91
E **T** L	21150.71	21124.05	21582.88	19445.80	21582.88	19445.80	19439.74	19439.74
E T **L**	24424.79	24396.11	24509.66	19416.65	24509.66	19416.65	19410.59	19410.59
E **T** **L**	20655.58	20629.02	21106.55	19372.05	21106.55	19372.05	19363.73	19363.73
E **T** **L**	20388.98	20362.42	20849.00	19317.46	20849.00	19317.46	19311.40	19311.40

(b) Total curtailment applied [MWh]

Fig. 5.4 Comparison of the potential performance for the IEEE 14-bus system with different algorithm combinations considered for selection (oracle 1)

Figure 5.3 shows the potential performance, with respect to the number of overloads and the curtailment applied, that different combinations of algorithms could achieve for the 33 kV meshed distribution system if optimal per-state selections are made that prioritise minimising the number of overloads (oracle 1). As noted in Section 4.3.4, PFM-OPF can remove all overloads for this system, though as Figure 5.3a shows, the combination of PFM-CSP and PFSF-TMA can remove all but 13 overloads. Although PFM-OPF can remove all overloads on the 33 kV system, selecting between it and other algorithms allows for less curtailment to be applied while still removing all overloads. Figure 5.3b shows what combination of algorithms is necessary to achieve this, which is all algorithms with the exception of PFSF-Egal (and the baseline option of not applying any algorithm). This explains why PFSF-Egal is not selected individually at all for this system, as shown in Figure 5.2a.

Figure 5.4 shows the potential performance, with respect to the number of overloads and the curtailment applied, that different combinations of algorithms could achieve for the IEEE 14-bus system if optimal per-state selections are made that prioritise minimising the

			B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E	T	L	-	678836.34	30389.77	1241.44	30385.38	1241.44	986.91	986.91
E	T	L	670366.33	670366.33	24974.82	1239.67	24974.82	1239.67	985.14	985.14
E	**T**	L	653798.21	653798.21	24421.55	1241.44	24421.55	1241.44	986.91	986.91
E	T	**L**	241625.85	241625.85	26967.29	1241.44	26967.29	1241.44	986.91	986.91
E	**T**	L	652049.04	652049.04	23296.29	1239.67	23296.29	1239.67	985.14	985.14
E	T	**L**	238886.58	238886.58	24768.12	1239.67	24768.12	1239.67	985.14	985.14
E	**T**	**L**	232140.18	232140.18	23782.87	1241.44	23782.87	1241.44	986.91	986.91
E	**T**	**L**	231512.94	231512.94	23155.66	1239.67	23155.66	1239.67	985.14	985.14

(a) Energy of overloads [MVAh]

			B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E	T	L	-	0.00	982967.03	749899.56	981644.69	749899.56	827822.94	827822.94
E	T	L	30780.23	30765.98	995253.47	750047.46	995253.47	750047.46	827970.85	827970.85
E	**T**	L	43312.05	43305.44	974117.95	748622.97	974117.95	748622.97	826579.29	826579.29
E	T	**L**	405575.30	405571.69	988936.13	748989.57	988936.13	748989.57	826917.82	826917.82
E	**T**	L	48519.55	48512.94	978436.37	748777.42	978436.37	748777.42	826733.74	826733.74
E	T	**L**	417997.22	417993.61	989173.76	749144.03	989173.76	749144.03	827072.27	827072.27
E	**T**	**L**	423089.93	423086.32	975305.54	748596.65	975305.54	748596.65	826554.79	826554.79
E	**T**	**L**	426519.49	426515.88	978687.63	748751.10	978687.63	748751.10	826709.24	826709.24

(b) Total curtailment applied [MWh]

Fig. 5.5 Comparison of the potential performance for the IEEE 57-bus system with different algorithm combinations considered for selection (oracle 2)

number of overloads (oracle 1). Although no algorithm can remove all overloads individually, there are several sets of two algorithms that can: PFM-CSP and PFSF-Egal, PFM-CSP and PFSF-TMA, PFM-CSP and PFM-OPF, or PFM-OPF and PFSF-Egal. The benefit of using more than two algorithms, however, is a reduction in curtailment, and the figure shows that all the power flow management algorithms are required in order to minimise the curtailment applied while also removing all overloads.

Figure 5.5 shows the potential performance, with respect to overload energy and the curtailment applied, that different combinations of algorithms could achieve for the IEEE 57-bus system if optimal per-state selections are made that prioritise minimising overload energy (oracle 2). Overload energy is minimised by a combination of three algorithms: PFM-OPF, PFM-CSP and PFSF-Egal. This is despite PFSF-Egal being the least effective algorithm overall at reducing overloads on the IEEE 57-bus system. Any combination missing one of these algorithms results in increased overload energy. To minimise the curtailment, while still minimising the overload energy, it is necessary to consider both of the remaining algorithms (PFSF-TMA and PFSF-LP).

Figure 5.5 also shows that selecting between applying an algorithm and not applying an algorithm can sometimes give better performance than always selecting an algorithm. On its own, PFM-CSP reduces the overload energy to 30389.77 MVAh. However, if the oracle selects between PFM-CSP and the baseline, this actually reduces to 30385.38 MVAh.

Similar to the analysis of algorithm selection frequencies in the previous section, this analysis has revealed that different sets of algorithms are required for each system to achieve particular objectives. For example, only PFM-OPF is required to minimise the number of overloads for the 33 kV distribution system, while three algorithms are required to minimise the energy of overloads for the IEEE 57-bus system. These sets may contain algorithms that have poor performance overall, but that can provide a performance benefit when selected amongst other algorithms, such as PFSF-Egal being needed to minimise the overload energy on the IEEE 57-bus system. Furthermore, sometimes a number of different sets of algorithms can achieve an objective on a particular system, as seen with the four pairs of algorithms that can minimise the number of overloads for the IEEE 14-bus system.

5.5 Conclusions

In this chapter, the performances of the power flow management algorithms tested in Chapter 4 have been examined for each state tested for the four case study power systems. This has revealed that none of the algorithms is always the most effective for every state, in terms of minimising either the number or energy of overloads whilst also minimising the amount of curtailment applied. If the most effective algorithms are selected on a per-state basis, potential performance benefits could be obtained for each system in comparison to the algorithms that were most effective overall:

- For the 11 kV radial distribution system, the potential performance benefit was a 0.42% reduction in the amount of curtailment applied in order to remove all overloads; however, this difference in curtailment was not statistically significant.
- For the 33 kV meshed distribution system, there was a statistically significant performance benefit in terms of a 0.91% reduction in the amount of curtailment needed in order to remove all overloads.
- For the IEEE 14-bus system, all overloads could be removed (a 100% improvement with respect to minimising the number or energy of overloads), and this could be

achieved with at least 1.19% less curtailment. Furthermore, both of these performance benefits were statistically significant.

- For the IEEE 57-bus system, there was a statistically significant performance benefit in terms of either a 43.82% reduction in the number of overloads, or a 20.65% reduction in overload energy. However, these reductions result in statistically significant increases in the curtailment applied, and, despite these increases, a number of overloads could not be removed completely.

To achieve the potential performance benefits from selecting algorithms for each system state, several, if not all, of the power flow management algorithms needed to be considered for selection. Conversely, for some systems there were particular algorithms that did not need to be considered for selection at all, as they were not uniquely the most effective algorithm for any state. The relative selection frequency of the algorithms varied across the systems, and the most frequently selected algorithm for a system was not necessarily the algorithm that was most effective overall. Furthermore, the analysis showed that even algorithms that perform poorly overall can provide a potential performance benefit if they are considered for selection alongside algorithms that are more effective overall.

Although optimally selecting algorithms on a per-state basis was found to offer statistically significant performance benefits for three of the case study systems – namely: the 33 kV meshed distribution system, the IEEE 14-bus system, and the IEEE 57-bus system – these are just *potential* benefits, obtained by deriving the optimal selections through post-hoc analysis of algorithm performance data, represented by abstract "oracle" selectors. The oracles in this work had objectives that considered performance measures relevant to power flow management: minimising the number or energy of overloads is important to network operators, whilst minimising the amount of curtailment is important to generator operators. However, use of different objectives – for example, to minimise the execution time of the algorithms, if control response time were sufficiently important – could give different results with respect to the potential performance benefits from per-state algorithm selection.

The selections represented by the oracles rely on perfect foresight, which is not available in reality. However, concrete *algorithm selectors*, which do not rely on perfect foresight, have been developed for a number of applications. The next three chapters examine the background, the design, and the performance of such systems when used to select power flow management algorithms for each system state, in an attempt to exploit and realise the potential performance benefits shown in this chapter.

In conclusion, this chapter has built upon the preceding chapters in order to provide an answer to research objective 1: yes, potential performance benefits for power systems control can be derived by selecting the algorithms on a per-state basis – at least for power flow management within three of the four case study power systems as used in this work. The remainder of this work addresses research objective 2, examining if algorithm selectors can be created to exploit the potential performance benefits.

Chapter 6

Previous work on algorithm selection

This chapter aims to satisfy research objective 2(a) by identifying existing algorithm selection techniques that could be applied to power system control; in particular for the selection of power flow management algorithms. First, a formal model for selecting algorithms is presented, which is then instantiated for the per-state selection of power flow management algorithms. This is followed by review of previous work that has created algorithm selectors for various applications, with a concentration on a particular suite of techniques – machine learning – as these have been frequently used to create algorithm selectors.

6.1 The Algorithm Selection Problem

6.1.1 Rice's model

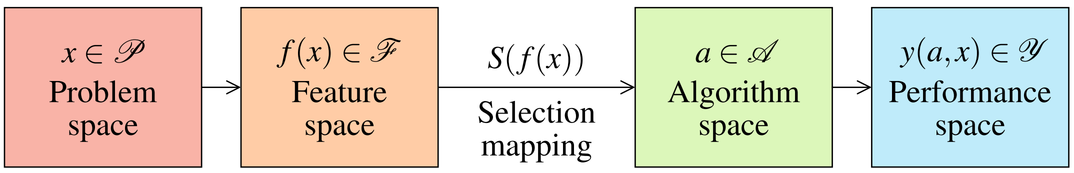


Fig. 6.1 Rice's conceptual model of the Algorithm Selection Problem

The problem of selecting "an effective or good or best" algorithm was formalised in 1976 by the computer scientist John Rice as *the Algorithm Selection Problem*. In the paper that formalised the problem [74], an abstract model of the problem was introduced, which is shown in Figure 6.1. This model contains the following elements:

- The **problem space**, $\mathscr{P}$, containing **problem instances**, x, drawn from a particular problem domain or application. One example given by Rice was for computer job scheduling, where each instance, x, represents a different set of jobs to be executed.
- The **feature space**, $\mathscr{F}$, containing **features**, $f(x)$, that are characteristics extracted from the problem instances. For c features, $\mathscr{F} = \mathbb{R}^c$, which may be of a lower dimension than $\mathscr{P}$. For the scheduling example given by Rice, the features could include the priorities of individual jobs, the expected processor time, and the expected memory usage.
- The **algorithm space**, $\mathscr{A}$, containing the **algorithms**, a, that can be applied to the problem instances. For the scheduling example given by Rice, each a represents an algorithm for scheduling the jobs contained within x.
- The **performance space**, $\mathscr{Y}$, containing **performance measures**, $y(a,x)$, describing the performance of an algorithm a applied to problem instance x. For d performance measures, $\mathscr{Y} = \mathbb{R}^d$. For the scheduling example, the performance measures could include the median and maximum turnaround time for jobs, and the total number of jobs processed per unit time.
- The **selection mapping**, $S(f(x))$, also referred to as an *algorithm selector*, which maps from problem instance features to an algorithm selection ($S : f(x) \to a$), in order to optimise an objective function of the performance measures.

6.1.2 Framing the problem

There is not a single Algorithm Selection Problem; rather, an instance of the Algorithm Selection Problem is formulated for each application to a problem domain that requires algorithms to be selected. The characteristics of the problem domain will influence what the elements within the model shown in Figure 6.1 contain.

What the problem instances represent and the scope of the problem space is an important aspect of formulating an Algorithm Selection Problem. Each problem instance should represent a distinct example of a single problem drawn from the problem domain. If problems can develop over time, then a decision has to be made as to what time period a problem instance can represent. Some problems allow for partial solutions, which allows the solution process to be paused and potentially completed by a different algorithm. If a problem can have partial solutions, then these may also be considered in the problem space as problem instances.

The scope of the problem space covers all potential problem instances considered from the problem domain. Depending on whether the problem instances are characterised by a fixed or changeable number of variables, and whether those variables are discrete or continuous, there could be a finite or infinite number of instances within the problem space.

The structure of some problems may allow them to be split in to separate sub-problems, which may allow for algorithms to be applied recursively to each sub-problem, such as in [75] for sorting, or in parallel, such as could be imagined for voltage control in separate areas of a power system. If the sub-problems are independent of each other, then they can be considered as problem instances in themselves.

The choice of features depends on what the problem instances represent, what aspects of them are relevant when selecting algorithms, and also whether the features can be computed. Depending on the choice, there may be a non-unique mapping between problem instances and features, so that some problem instances are not separated within the feature space.

The algorithms considered can simply be limited to those that are available to be applied to the problem instances. However, some work (such as [76]) does not limit the algorithm space to contain a fixed, small set of algorithms; instead the algorithm space is parameterised to contain all possible algorithm configurations.

The choice of performance measures is linked to the overall objectives that the algorithms are being applied for. The measures can relate to the quality of the solution process – such as the execution time required by the algorithms, which is a common performance measure in literature – or the quality of the solution – such as the prediction error of machine learning algorithms [77].

6.1.3 Application to power flow management

In this work, a separate instantiation of the Algorithm Selection Problem is considered for each case study power system, as each system is independent from the others. This allows for the following high-level formulation of the Algorithm Selection Problem to be created for each case study system:

- **Problem space**: each problem instance represents a state of the system, described by continuous state variables. Although the continuous state variables should mean the state spaces should be infinite, the states within the problem spaces are restricted to the finite sets of states considered in Chapter 4.

- **Feature space**: this contains features derived from the problem instances (states), such as the values of load and generation within a power system. Determining a set of features to use is a task during the creation of an algorithm selector, and which features are used in this work is discussion in Chapter 7.

- **Algorithm space**: this contains the five power flow management algorithms described in Chapter 2 (PFM-CSP, PFM-OPF, PFSF-Egal, PFSF-TMA, and PFSF-LP), along with the baseline option of not applying any control.
- **Performance space**: this contains the three performance measures considered in Chapter 4, namely: the number of overloads, the energy of overloads, and the amount of generator curtailment applied.
- **Selection mapping**: the objectives that the mappings should consider are the same as those used in Section 4.1.2 to evaluate the performance of the power flow management algorithms; namely: minimising either the number or energy of overloads, while minimising curtailment. Previous work on *creating* selection mappings is discussed in the next section.

6.2 Creating a selection mapping

Once a particular application has been formulated as an Algorithm Selection Problem, the next task is to derive a selection mapping (an algorithm selector) that is able to select algorithms to best achieve the desired objectives.

In the paper that introduced the Algorithm Selection Problem, Rice originally proposed that the tools of approximation theory could be used to create selection mappings. Approximation theory is concerned with deriving mathematical functions to closely represent other – typically more complex – functions. When applied to the Algorithm Selection Problem, approximation theory exposes such questions as whether a best selection mapping exists, whether it is unique, and how is "best" characterised? However, Rice considered these to be mostly irrelevant and emphasised that it is more important to find a *good* – rather than the *best* – selection mapping, and also to find a appropriate form for the approximation, which was left as an open research question.

One of the earliest examples of the concrete application of the Algorithm Selection Problem and the creation of an algorithm selector was the ATHENA system created by Rice, Houstis and others [78], which formed part of the ELLPACK system for solving partial differential equations. Each equation to be solved represented a problem instance, and ATHENA was used to select solver algorithms to apply to the equations in order to meet user-specified requirements for accuracy and execution time. The algorithm selector was based on an expert system – rather than approximation theory – that featured a knowledge base of performance profiles of each solver, which could be updated with new performance data each time a solver was used. For a new problem instance, past performance on similar problems was used to predict the performance of the solvers, which was then used to make a selection decision.

Another early example of a concrete algorithm selector is the work of Brodley [79], which had the objective of selecting classification algorithms in order to reduce classification error. Hand-crafted rules were used to select algorithms, which where related to a number of features of the classification data . Other authors have also developed algorithm selectors using hand-crafted rules, such as the work of Beck and Freuder [80] on selecting scheduling algorithms. Although hand-crafted selection rules were used successfully in these applications, formulating rules requires domain expertise that may be unavailable for other applications.

Lobjois and Lemaître [81] took a different approach to select between four branch and bound search algorithms that were used to solve constraint satisfaction problems, with the aim of minimising execution time. Rather than selecting algorithms based on rules or models of algorithm performance, the algorithms' execution time per problem instance was predicted by executing the algorithms for a short time on the instance. The prediction combined the measured execution time per search node with an estimate of the size of the search tree, and the algorithm with the lowest predicted time was then selected. This approach was shown to work well when compared with alternative strategies, such as executing the algorithms interleaved with one another or selecting algorithms at random. Although successful, this approach requires that the problem instances can be easily duplicated, which is not possible in some problem domains, such as if the problem instances represent a physical system that only has a single instantiation. Furthermore, the approach relies on the observed performance when initially solving a problem being a good predictor of the overall performance, which may not always be the case, if indeed it is possible to partially solve the problem in order to derive a prediction of execution time.

Although this early work trialled a variety of approaches and had some success, the majority of recent work on creating algorithm selectors has used *machine learning* applied to algorithm performance data. An introduction to machine learning is provided in the next section, followed by a review of how it has been applied to create algorithm selectors.

6.3 Machine learning

Machine learning is a branch of artificial intelligence concerned with creating systems that can learn (improve their performance) through experience (exposure to data) [82]. Although the roots of machine learning can be traced back at least as far as the work on perceptrons (a form of artificial neural network) in the late 1950s [83], there has been a recent resurgence in interest in the subject due to ever increasing computer power and access to large amounts of data, such as can be found on the world wide web. Machine learning has been used in a variety of diverse applications, from the automated translation of speech [84], to classifying astronomical objects [85], to self-driving vehicles [86].

Russell and Norvig [82] distinguish four paradigms of how a learning algorithm can be applied to data:

- **Supervised learning**: this is where the learning algorithm is given a set of data that consists of inputs labelled with outputs, and the task is to create a *model* or *hypothesis* from that data that maps from the inputs to the outputs. Some supervised learning models and algorithms allow for *online* learning, where new data can be taken in to account and the model updated. Others can only learn in *batch* mode, where the model is learnt once, offline, and cannot be updated.

  Supervised learning is subdivided according to the nature of the outputs: if the outputs come from a finite set of discrete values (or *classes*), then the task of predicting the output is known as *classification*; otherwise, if the values are real-valued, then the prediction task is known as *regression*.

  The use of supervised learning shares much of Rice's original vision of using approximation theory to create algorithm selectors. Indeed, one type of supervised learning model (artificial neural networks) have been shown to be universal approximators [87].

- **Unsupervised learning**: this is where only input data is given, and the task is to discover patterns in the data. An example of this is *clustering*, where algorithms such as k-means [88] are used to find groupings of data that are most self-similar.

- **Semi-supervised learning**: this sits in a continuum between supervised and unsupervised learning, where some training examples are labelled (perhaps with inaccuracies) and others are not. The learning task is similar to supervised learning: to create a mapping between inputs and outputs.

- **Reinforcement learning**: this is where the learning algorithm receives rewards for the actions it takes, so must learn which actions should be taken in order to maximise reward. Algorithms such as Q-learning [89] can be used to learn a mapping between system state, action taken, and reward gained. Reinforcement learning is particularly suited to online applications, where the learning algorithm interacts directly with the system that the actions are applied to.

6.4 Using machine learning for algorithm selectors

Numerous authors have used a variety of machine learning techniques to create algorithm selectors, and the most pertinent aspects of using machine learning are presented below. These aspects include the timing of the selection decisions, the form of output from the machine learning models, the form of output from the selector that the models are used within, how and when learning is incorporated in to the selector, the choice of machine learning model type, and the features used for selection.

6.4.1 Timing

When the selection decisions are made depends on the nature of the problem instances and influences how machine learning can be applied. Gagliolo and Schmidhuber [90] distinguish two types of selectors according to the timing of their selection decisions:

- **Static selectors** make a single selection decision for each problem instance, before any algorithm is executed. While they are straightforward to understand and have been implemented by many authors, static selectors assume that the performance of the algorithms is reasonably predictable *before* they are executed on a problem instance.
- **Dynamic selectors** make selection decisions while algorithms are executing to solve a problem instance. This allows information about how the algorithms are performing on a problem instance to be used to make selection decisions, so are useful if the performance of algorithms is only reasonably predictable *after* execution has started.

Dynamic selectors can be more complex than static selectors [90]. The timing of dynamic selection decisions can be critical, as short intervals between decisions may not allow algorithm performance to be observed adequately, while lengthy intervals may allow a previous poor choice of algorithm to waste resources. Dynamic selection decisions may be triggered in response to changes during the problem solution; for example, if the problem changes or the solution reaches a specific state, such as a branching point [75, 79]. Alternatively, if there are no obvious triggers, dynamic decisions may be made according to a predefined schedule, such as in [90].

When algorithms are used that return partial solutions when interrupted — so-called “anytime” algorithms — the selector can exploit the partial solutions by passing them to the next algorithms, possibly reducing the total time needed to solve a problem [91]. If partial solutions are to be passed on, there is a choice of which partial solutions are passed on to the next algorithms: those returned by the last algorithms executed [9], the best so far [91], or the last partial solution returned by the same algorithm (if it has already been executed on the problem) [92]. Which of these yields the best performance depends on characteristics implicit to the problems and algorithms, and can be resolved through experimentation.

6.4.2 Model output

A selector may contain a machine learning model that directly represents the selection mapping. In this setting, classification models can be used, with their inputs being the features extracted from the problem instances and the outputs of the models (classes) being the algorithm selections. Numerous authors have used this approach, with applications including the selection of sorting and matrix multiplication algorithms [93], quadratic assignment solvers [94], and simulation algorithms [95].

Reinforcement learning models can also be used for directly representing the selection mapping, such as in [75] for selecting sorting algorithms. For these models, the algorithm selections are represented as the actions taken by the models.

An alternative to directly predicting which algorithm is most likely to be the best performing for a problem instance is to use models to predict the performance of each algorithm, and then select the algorithm with the best predicted performance. Machine learning is used to create *empirical performance models* (EPMs) [96] of each algorithm, which approximate a mapping from the feature space to the performance space. Either regression or classification models can be used, depending on the nature of the performance measure being predicted. Often the EPMs are used to predict algorithm execution time; for example, in [97], ridge regression models are used to predict the runtime of solvers for boolean satisfiability problems, whereas in [98], a nearest-neighbour approach is used to predict the runtime of constraint satisfaction solvers. Other authors have predicted other performance measures; for example, Fink [99] predicted the probability of planning algorithms completing within a particular time bound.

6.4.3 Selector output

In the most straightforward setting, a selector will only select a single algorithm at a time. However, selectors can be created to select sets of algorithms – referred to as portfolios [100]

– to be applied to a problem instance. There are a number of different ways that multiple algorithms can be applied, which depend upon the characteristics of the problem instances and of the algorithms:

- Multiple algorithms operating sequentially on the same problem instance. This requires that partial solutions to the problem instances can be represented, and "anytime" algorithms that will provide partial solutions whenever their execution is interrupted. The selector will provide a sequence for the execution of the algorithms, perhaps including a time allocation for each, such as in [98] for constraint satisfaction solvers.
- Multiple algorithms operating in parallel on *independent copies* of the problem instance. Gagliolo and Schmidhuber [90] used this approach for creating dynamic selectors for constraint satisfaction and the auction winner determination problem. All algorithms were allowed to execute in parallel, and were periodically interrupted to observe their performance. The selector, which was based on reinforcement learning, then used this partial performance information in order to allocate time shares to the algorithms, with the most promising being prioritised.
- Multiple algorithms operating in parallel on a *shared copy* of the problem instance. This could be encountered if the problem instances represented a physical system that could not be duplicated. If the problems can be decomposed in to sub-problems, then the algorithms could be applied to each sub-problem independently. Otherwise, mechanisms would need to be put in place to allow the algorithms to co-operatively execute on a single instantiation of a problem instance.

6.4.4 Learning

The timing of *when* learning occurs to create or update a selector can be either:

- **Offline learning** which occurs once before the selector is applied to make any selection decisions. This is most suitable when supervised learning algorithms are used.
- **Online learning** involves updating the selector once selection decisions have been made. Reinforcement learning techniques are particularly suited to being used to create selectors with online learning. Online learning can be further sub-divided according to when the learning takes place: *inter-instance* learning occurs between problem instances, so that past performance can be used to improve the selector, such as in [90, 99]; whereas *intra-instance* online learning occurs while algorithms are

being executed on a problem instance, such as in [9, 91], so therefore can only occur in dynamic selectors.

As algorithm selectors must produce an output (the algorithm selections), then only supervised (including semi-supervised) and reinforcement learning techniques can be used. However, that does not preclude the use of unsupervised learning to assist the creation of selectors. For example, in [101] clustering in the feature space is used to select the neighbourhood size for a selector based on a nearest neighbour classifier.

6.4.5 Model type

The choice of machine learning model type is one of the most important when creating an algorithm selector. Although many different models have been used previously, few works have created selectors using a variety of model types, in order to see which tend to lead to selectors that perform well.

Kotthoff et al. [7] compared the performance of 30 different machine learning models used as direct and EPM-based algorithm selectors for five different types of constraint satisfaction and search problems. Most of the models could not outperform a “winner takes all” approach of simply selecting the algorithm that most often gave the best performance. However, support vector machines (SVMs), a type of classification model, were identified as being particularly promising to create algorithm selectors.

Hutter et al. [96] created EPMs for 11 combinatorial optimisation algorithms for 35 different problem instance distributions. Out of the 6 different regression models used to create EPMs, random forests were found to most often produce the most accurate performance predictions, followed by Gaussian processes.

Although these findings indicate particularly promising model types for creating algorithm selectors, there are no guarantees that their findings can apply to other problem domains. Furthermore, many machine learning models have parameters that can significantly alter their performance and could mean other model types are more promising.

6.4.6 Features

Developing and choosing relevant high-level features is one “one of the most important, yet nebulous, aspects of the Algorithm Selection Problem” [74], as well as being a major concern generally in machine learning. Features are high-level abstractions of the most important characteristics of the problem instances that relate to how algorithms perform. However, features do not need to be limited to simple metrics calculated from the problem instances.

For example, in dynamic selectors the features can include performance measures relating to the algorithms that are currently being executed on the problem instances. The performance predictions of EPMs may be used as features in a direct selector, such as in Kotthoff's work [102] on creating a "hybrid" selector for constraint satisfaction and search problems. The feature sets may also be augmented with the performance of simpler algorithms executed on a problem instance, that may be indicative of – or "landmark" – the performance of more complex algorithms [103].

The choice of features is problem domain specific and the creation of a candidate set of relevant features requires domain knowledge. This does not guarantee that the features are all relevant, but automatic techniques can be used to select the most predictive sub-sets, such as in [97] and [7]. Although past work on algorithm selection has relied on hand-crafted high-level features, recent advances in deep learning [104] give promise that relevant higher-level abstractions can be learned automatically from very low-level features.

6.5 Previous applications

Although the examples presented so far of work applying the Algorithm Selection Problem and creating algorithm selectors have been in computer science applications, there have been some wider applications. For example, in [10], an EPM-based selector is created for stock market trading algorithms, and selectors have been created to select prediction algorithms for both electrical load [105] and wind power [106]. However, apart from the author's own work there does not appear to be any other previous applications of the Algorithm Selection Problem to power systems problems.

6.6 Conclusions

This survey of the algorithm selection literature has revealed that machine learning is the predominant suite of techniques used for creating algorithm selectors. There are numerous variations in how machine learning can be used, although one of the major differentiations is in how the selection mapping is represented by the machine learning model: either the mapping is represented *directly*, or machine learning models are used as EPMs to predict the performance of each algorithm, with the algorithm predicted to best then selected. There is no consensus on which machine learning algorithms are most promising for creating algorithm selectors, particularly if they are being developed for new problem domains.

A high-level formulation of the Algorithm Selection Problem for power flow management was presented, which will be used as the basis for developing algorithm selectors in the next chapter. Particular aspects of the problem and its formulation restrict the scope of selector designs that can be considered. The power flow algorithms (described in Chapter 2) are not “anytime” algorithms and must be allowed to complete their execution before any solution is provided, therefore dynamic selectors are inapplicable. This means static selectors are to be produced; therefore offline, supervised learning techniques are most suitable. In summary, static selectors are to be created that select a single algorithm for each state, with purely offline learning, using either direct or EPM-based designs.

Chapter 7

Design and development of algorithm selectors for power flow management

This chapter contributes towards fulfilling research objective 2, in particular research objectives 2(b) and 2(c), by developing designs for algorithm selectors for per-state selection of power flow management algorithms. Based on the findings of the literature review in Chapter 6, designs have been developed for both direct and empirical performance model (EPM) based selectors. This chapter provides an overview of the design and development of the selectors, with their performance being the subject of the next chapter, Chapter 8.

This chapter splits in to two halves. In the first half, the methods that have been developed in this work for preparing training data to create per-state algorithm selectors for power flow management are described, with Section 7.1 describing the methods for direct selectors, and Section 7.2 describing the methods for EPM-based selectors. These descriptions include the design choices that are specific to each selector design, and the options that have been investigated in this work. The second half of the chapter discusses a number of design choices that are shared by all of the selector designs, which are: the choice of machine learning model (Section 7.3), the features used by the models (Section 7.4), and the volume of training data used (Section 7.5, which also discusses how additional data was generated in this work for training the selectors).

The work in this chapter relating to direct selectors (Section 7.1) has been published previously in a reduced form in [60]. However, the remaining content – such as the development of EPM-based algorithm selectors for power flow management, and the discussion of common design options – has not been published previously.

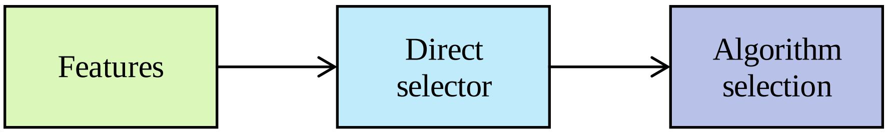


Fig. 7.1 Structure of a direct algorithm selector

7.1 Preparation of training data for direct selectors

This section describes three different methods for preparing data to train classifiers to be used as direct algorithm selectors. These selectors directly represent the selection mapping (as described in Section 6.4.2) and, as illustrated in Figure 7.1, use input features to predict which algorithm is likely to be the most effective.

As shown in Figure 7.2, where one algorithm is more effective than all other algorithms can be visualised within the state space of a system; in this case, the IEEE 57-bus system, where the algorithms shown are those that are solely most effective at minimising the number of overloads whilst also minimising curtailment for a state. Although the figure can only show two-dimensional combinations of the state variables, for some combinations it is clear to see regions where particular algorithms are more effective than others; for example, within the plot of the generator 8 and 12 outputs, there are distinct regions where each of PFM-OPF, PFM-CSP, and PFSF-TMA dominate. Intuitively, if the state variables are used as features, classifiers used as algorithm selectors can be thought of as identifying regions within the feature space where one algorithm is most effective and therefore should be selected.

The three methods outlined below are differentiated by the number of objectives that the selector can consider (either a single objective or multiple) and by whether examples are weighted or not during training.

7.1.1 Selectors using unweighted training examples

In this method, the training examples are unweighted, so misclassification cost is the same for each training example. Each state considered for training is represented by at most a single training example, which can only be labelled with a single power flow management algorithm. The labelling is based on determining the selection set for each state, using either a single or multiple objectives. The selection set for a state will contain either a single algorithm (the one that is most effective) or multiple algorithms (if they are equally most effective). Labelling a training example is straightforward for a state with a singleton selection set: the

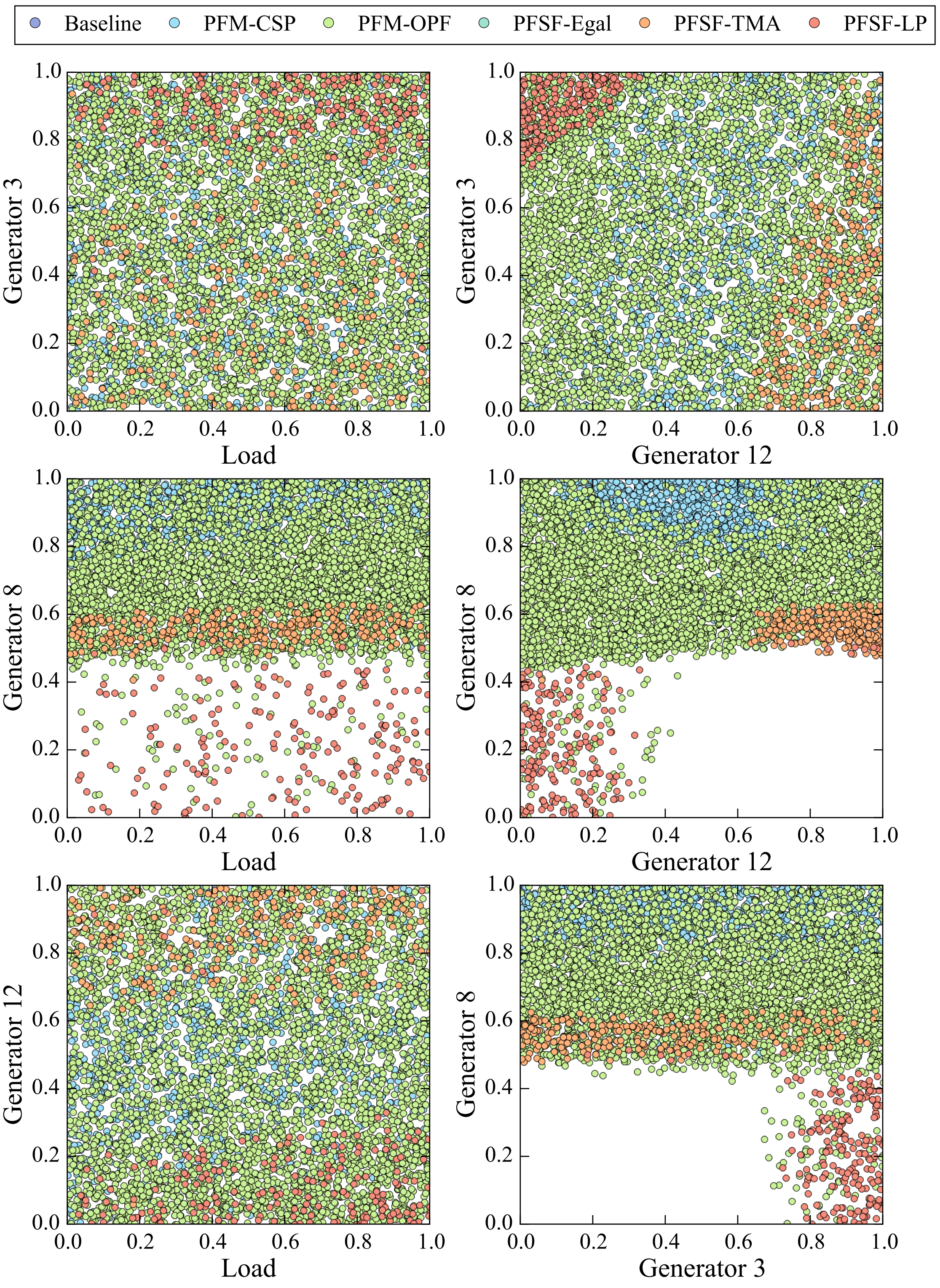
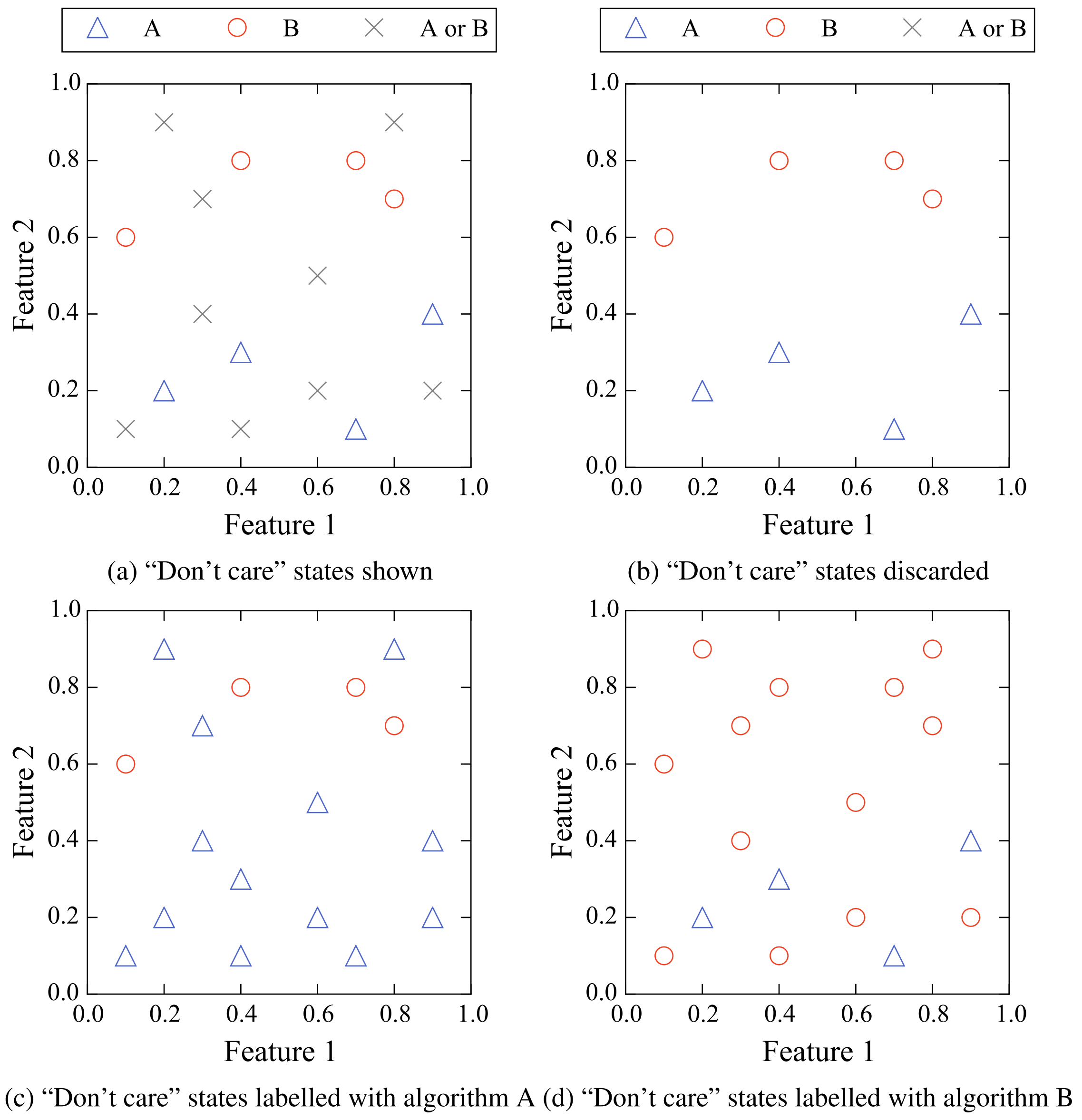


Fig. 7.2 Visualisation of the state space for the IEEE 57-bus system, showing the states where a single algorithm is most effective at minimising the number of overloads whilst also minimising the curtailment applied



(a) "Don't care" states shown (b) "Don't care" states discarded

(c) "Don't care" states labelled with algorithm A (d) "Don't care" states labelled with algorithm B

Fig. 7.3 Example of different options for labelling "don't care" states where no algorithm (from a pair of algorithms) is uniquely the most effective

example is labelled with the sole algorithm in the set. However, for non-singleton selection sets it is not obvious which algorithm the examples should be labelled with.

Labelling a training example with an arbitrary algorithm in the case of a non-singleton selection set can make the selection mapping harder to learn, and thus deleteriously affect the performance of a selector. To illustrate this, consider the two-dimensional feature space shown in Figure 7.3a, in which states are shown where one of two algorithms (A or B) is most effective and thus the selection set is a singleton, along with “don’t care” states where both algorithms deliver the same performance and thus there is a non-singleton selection set.

Figures 7.3c and 7.3d show the same feature space but with each state representing a training example and the “don’t care” states with non-singleton selection sets labelled with an arbitrary algorithm from the set. Figure 7.3c represents the case where algorithm A is used for the arbitrary label, while Figure 7.3d represents where the same states are labelled with algorithm B. Figure 7.3b is the same, except that all “don’t care” states have been discarded and are thus not used as training examples. When the “don’t care” states are discarded, it is clear to see where each algorithm should be selected, and a very simple classifier could be constructed to be used as an algorithm selector which only considers whether feature 2 is above or below a single value (0.5, for example). However, when the “don’t care” states are kept and arbitrarily labelled, multiple regions emerge where each algorithm should be selected, making the selection mapping inherently more complicated, which can reduce the accuracy of any classifier trained as an algorithm selector using those datasets.

With this method, it is possible to consider multiple objectives by iteratively determining, for each state, the selection sets of each objective. For example, if minimising the number of overloads and the amount of curtailment were both considered (with minimising the number of overloads prioritised), for each state the selection set that minimises the number of overloads would be determined first. If a state had a singleton selection set, then the sole algorithm in the set would be used to label the training example representing that state. However, if a state had a non-singleton set (where two or more of the considered algorithms remove the same number of overloads) then the minimisation of curtailment would also be considered to further refine the selection set. If the refined selection set was then a singleton, then the remaining algorithm would be used to label the training example; otherwise, the state would effectively be a “don’t care” state with no algorithm uniquely most effective, and the training example representing the state would be discarded.

There are three design choices specific to this design:

1. Objectives: the set of objectives used to determine the selection sets. In this work, the following design options for the sets of objectives are investigated:

- Single objectives (2 options): (1) minimising the number of overloads or (2) the overload energy. This option was not investigated for the 33 kV meshed distribution system as the PFM-OPF algorithm could remove all overloads.
- Two objectives (4 options): minimising the number of overloads whilst also minimising (1) the overload energy or (2) the amount of curtailment; or minimising the overload energy whilst also minimising (3) the number of overloads or (4) the amount of curtailment.
- Three objectives (2 options): (1) minimising the number of overloads whilst also minimising the overload energy and then the amount of curtailment; or (2) minimising the overload energy whilst also minimising the number of overloads and then the amount of curtailment.

2. Algorithms: which sets of algorithms are considered for selection. In this work, the following sets of algorithms were investigated as design options for the selectors developed for each system, based on the analysis of different algorithm sets presented in Section 8.5:

   - 33 kV meshed distribution system (5 options): PFM-OPF with either (1) PFM-CSP, (2) PFSF-Egal, (3) PFSF-TMA, or (4) PFSF-LP, plus (5) the set of all algorithms, including the baseline option of not applying any control.
   - IEEE 14-bus system (5 options): each of the pairs of algorithms that could remove all overloads were considered, namely: (1) PFM-CSP and PFSF-Egal, (2) PFM-CSP and PFSF-TMA, (3) PFM-CSP and PFM-OPF, and (4) PFM-OPF and PFSF-Egal; plus (5) the set of all algorithms).
   - IEEE 57-bus system (3 options): (1) PFM-CSP and PFM-OPF, which is the pair of algorithms that minimises the number and energy of overloads, when compared with all other pairs of algorithms for that system; (2) PFM-OPF, PFM-CSP, and PFSF-Egal, which is the set of algorithms that minimises the overload energy; and (3) the set of all algorithms.

3. Discarding "don't care" states: the states where the selection set is a non-singleton can either be retained, and labelled with an arbitrary algorithm; or discarded. In this work the single design option of discarding all "don't care" states is investigated.

7.1.2 Selectors for a single objective using weighted training examples

For the method described in the previous section, when only two algorithms are considered, the states with non-singleton selection sets are those where both algorithms are equally effective, and no information is lost when those states are disregarded from becoming training examples. However, when three or more algorithms are considered, there may be states with non-singleton selection sets where one or more of the algorithms are not as effective as the others. Therefore, when those states are disregarded as training examples, potentially important information about differences in algorithm performance – and therefore, which algorithms should be selected – can be lost, can deleteriously affect the performance of selectors trained using that data. Another issue is that for the states with singleton selection sets, the "cost" (in terms of difference in performance) may vary between states. For example, for one state the performance difference between using one algorithm over another could be the removal of a substantial number of overloads, while for another state the difference could only be a marginal decrease in the amount of curtailment. Without modification, the classifiers used as algorithm selectors will only assume the same cost of misclassification for every state, and thus will not recognise important cost differences.

The optimisation processes within the algorithms that are used to train classifiers seek to minimise misclassification error, which usually considers the cost of misclassifying a training example to be equal across all examples in the training dataset. However, when using a classifier as an algorithm selector the *actual* cost of misclassification – that is, choosing an algorithm that is not the most effective for a state – may vary between states, as explained above. Therefore, it is desirable to have selectors that can recognise the different costs of misclassification (selecting an algorithm that is not the most effective), and this can be achieved by *weighting* the training examples used when training a classifier to be used as an algorithm selector.

In contrast to using unweighted training examples (as in Section 7.1.1), when using weighted training examples it is not necessary to disregard states with non-singleton selection sets when creating the test dataset. In fact, every state is duplicated in the training dataset, with one example created per algorithm for each state. Each example is then weighted according to the algorithm's performance on the state, with the largest weight being assigned to the example representing the most effective algorithm (for the performance measure being selected for). For an algorithm a and state i, the weight for the corresponding training example is calculated as follows:

$$w_{i,a} = p_{i,y}^{\max} - p_{i,y,a} \tag{7.1}$$

Where $p_{i,y,a}$ is the performance of the algorithm on the state for performance measure y, and $p_{i,y}^{\max}$ is the maximum value of performance for all algorithms considered (in other words, the performance of the worst performing algorithm for the state). Thus, each training example is weighted according to the relative performance gain that can be achieved by selecting the algorithm that the example represents.

This selector design has the following specific design choices:

1. Objectives: which objective is used to determine the selection sets. In this work, two objectives are investigated as design options: (1) minimising the number of overloads, and (2) minimising the overload energy.

2. Algorithms: the set of algorithms considered for selection. In this work, the following sets of algorithms were investigated as design options for each system:

    - 33 kV meshed distribution system: as PFM-OPF can remove all overloads, and the objectives for this selector design only considered overloads, this particular selector design was not investigated for the 33 kV distribution system.
    - IEEE 14-bus system (5 options): the same sets of algorithms were considered as for the direct (unweighted) selectors (Section 7.1.1).
    - IEEE 57-bus system (3 options): the same sets of algorithms were considered as for the direct (unweighted) selectors (Section 7.1.1).

7.1.3 Selectors for two objectives using tuned weights

The method outlined in the previous section can be extended to consider multiple objectives, by weighting training examples according to all objectives considered. This can be achieved by calculating weight components for each performance measure y, and then summing the components with scalar multipliers (c_y) applied to prioritise particular objectives; using:

$$w_{i,a} = \sum_{y \in Y} c_y (p_{i,y}^{\max} - p_{i,y,a}) \tag{7.2}$$

The prioritisation of multiple objectives is an application-specific decision, which influences the values of the weight component multipliers, c_y, that are used when training classifiers to be used as selectors. Determining what multiplier values to use constitutes a multi-objective optimisation problem, where the decision variables are the multipliers and the objective function comprises the performance of the algorithms selected by a selector that has been trained with data using those multipliers. Since each evaluation of a candidate set of multipliers requires a classifier to be trained and then tested, the process of finding appropriate multiplier values is suited to so-called “black-box” or “derivative-free” optimisers, such as grid search or a genetic algorithm [107], which can optimise arbitrary functions.

Out of the three objectives considered in this work, two combinations have been used for creating selectors that consider multiple objectives, namely:

1. Minimising the number of overloads whilst minimising the amount of curtailment, or:
2. Minimising the overload energy whilst minimising the amount of curtailment.

For both of these combinations, the objectives are *ordered* such that the first objective is prioritised over and above the second objective. In other words, the performance of the algorithms selected by one selector compared with those selected by a different selector can only be considered superior if there is a reduction in the number or energy of overloads, or, in cases where number or energy of overloads removed is the same, there is a reduction in the amount of curtailment applied.

A procedure has been developed in this work for tuning the weight multiplier, c_2, associated with the second of two ordered objectives. This procedure is based on an observation made when developing a direct selector for the 33 kV meshed distribution system. During this development, `RandomForest` model (described in Section 7.3) was used and the selector considered the nested objectives of minimising the number of overloads whilst also minimising the amount of curtailment applied, with each objective having weight multipliers of c_{OL} and c_{CT}, respectively. A range of c_{CT} values were used to create training data with different weights, with a fixed value of $c_{\mathrm{OL}} = 1.0$. A basic set of features (described in Section 7.4.3) and the 10,000 system states from the "A" training split of power flow management performance data for the 33 kV distribution system (described in Section 7.5) were used to create the training data, which were used to train selectors. The selection decisions of each selector were then determined for the 17,520 test states for the 33 kV distribution system, with the performance of each selector calculated from the performance of the algorithms selected.

Figure 7.4 shows the effect of varying the weight multiplier c_{CT} on the performance of the selectors, with respect to the number of overloads and the amount of curtailment applied. The figure clearly shows the shift in the prioritisation of the objectives as the value of c_{CT} increases, with curtailment reducing at the same time as the number of overloads increases. Importantly, there is a range of c_{CT} values (< 0.01) where the amount of curtailment decreases but not at the expense of incurring additional overloads. The tuning procedure developed in this work exploits this observation that, for particular selectors, the weight multiplier of a second objective (c_2) can be increased to a non-zero value, thus improving performance against the second objective, without a performance penalty with respect to the first objective.

The procedure for tuning c_2 is outlined in Algorithm 7.1, with an example of its operation illustrated in Figure 7.5. In essence, the procedure searches along the c_2 axis to find the maximum c_2 value that, when used to train and test a selector, does not cause a decrease

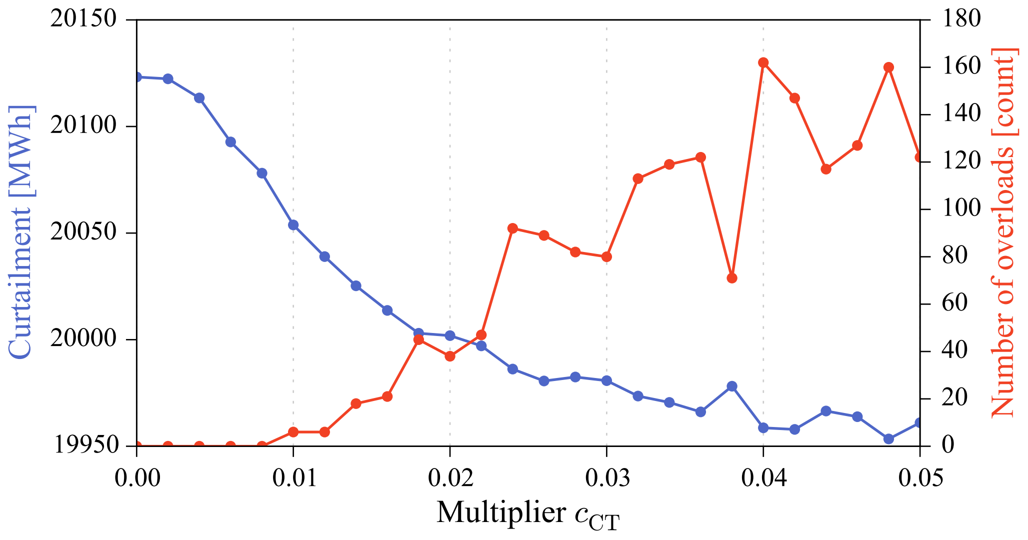


Fig. 7.4 Effect of varying weight multiplier c_{CT} with $c_{\mathrm{OL}} = 1.0$ for a selector using a `RandomForest` model to select between the PFM-OPF and PFSF-TMA algorithms, trained and tested on states from the 33 kV meshed distribution system

in the performance measure for the first objective, p_1. Each iteration, l, of the procedure comprises training and evaluating a selector for the c_2 value for that iteration (lines 3-6), and determining the next c_2 value to try (lines 7-27). The first value of c_2 trialled is 0.0, and the performance of the selector trained using that c_2 is captured with respect of the first objective as p_1^{target} (line 8). The search position is then incremented by a variable length step, $\triangle c_2$ (line 24), which for the first iteration is set to a minimum length of $\triangle c_2^{\mathrm{min}}$. For each of the subsequent c_2 values trialled, the progress of the search depends on the the performance, p_1, of the selector trained using that c_2 value.

If p_1 meets the value of p_1^{target} (lines 9-15), such as for iterations 2 and 3 in Figure 7.5, then the current c_2 value is recorded as the best known position, c_2^{best}, and the search is expanded by doubling the last step size, up to a maximum step length of $\triangle c_2^{\mathrm{max}}$. If the new step length would take the search up to or past the search limit, c_2^{limit}, such as for iteration 6 in the figure, then the step length is reduced to $\triangle c_2^{\mathrm{min}}$. This prevents unnecessary iterations trialling c_2 values past the search limit.

If p_1 does not meet p_1^{target} (lines 16-22), such as for iteration 4 in Figure 7.5, the current c_2 position is recorded as the search limit, c_2^{limit}, then the search is contracted by reducing the step size, $\triangle c_2$, to the minimum length, $\triangle c_2^{\mathrm{min}}$, and by reversing the search position to one step after the best known position, $c_2^{\mathrm{best}} + \triangle c_2$. If, for this second case, the step is already at its minimum length, then no further increase of c_2 is possible and the search terminates. The search will also terminate if the next search position meets or exceeds the search limit,

Algorithm 7.1 Procedure used to tune selector weight multiplier c_2

```
Require: $c_1$, $\triangle c_2^{\text{min}}$, $\triangle c_2^{\text{max}}$, $l^{\text{max}}$, training/test data, machine learning model type
 1: Initialise $c_2 := 0$, $c_2^{\text{best}} := 0$, $c_2^{\text{limit}} := \infty$, $\triangle c_2 := \triangle c_2^{\text{min}}$
 2: for $l := 1, l^{\text{max}}$ do
 3:     Create training dataset using weight multipliers $c_1$ and $c_2$
 4:     Use training dataset to train algorithm selector
 5:     Evaluate selector using supplied test data
 6:     Determine value for performance measure $p_1$
 7:     if $l = 1$ then
 8:         $p_1^{\text{target}} := p_1$
 9:     else if $p_1$ meets $p_1^{target}$ then
10:         $p_1^{\text{target}} := p_1$
11:         $c_2^{\text{best}} := c_2$
12:         $\triangle c_2 := \min(2 \times \triangle c_2, \triangle c_2^{\text{max}}))$
13:         if $c_2 \geqslant c_2^{\text{limit}}$ then
14:             $\triangle c_2 := \triangle c_2^{\text{min}}$
15:         end if
16:     else
17:         $c_2^{\text{limit}} := c_2$
18:         if $\triangle c_2 > \triangle c_2^{\text{min}}$ then
19:             $\triangle c_2 := \triangle c_2^{\text{min}}$
20:         else
21:             break
22:         end if
23:     end if
24:     $c_2 := c_2 + \triangle c_2$
25:     if $c_2 \geqslant c_2^{\text{limit}}$ then
26:         break
27:     end if
28: end for
29: return $c_2 := c_2^{\text{best}}$
```

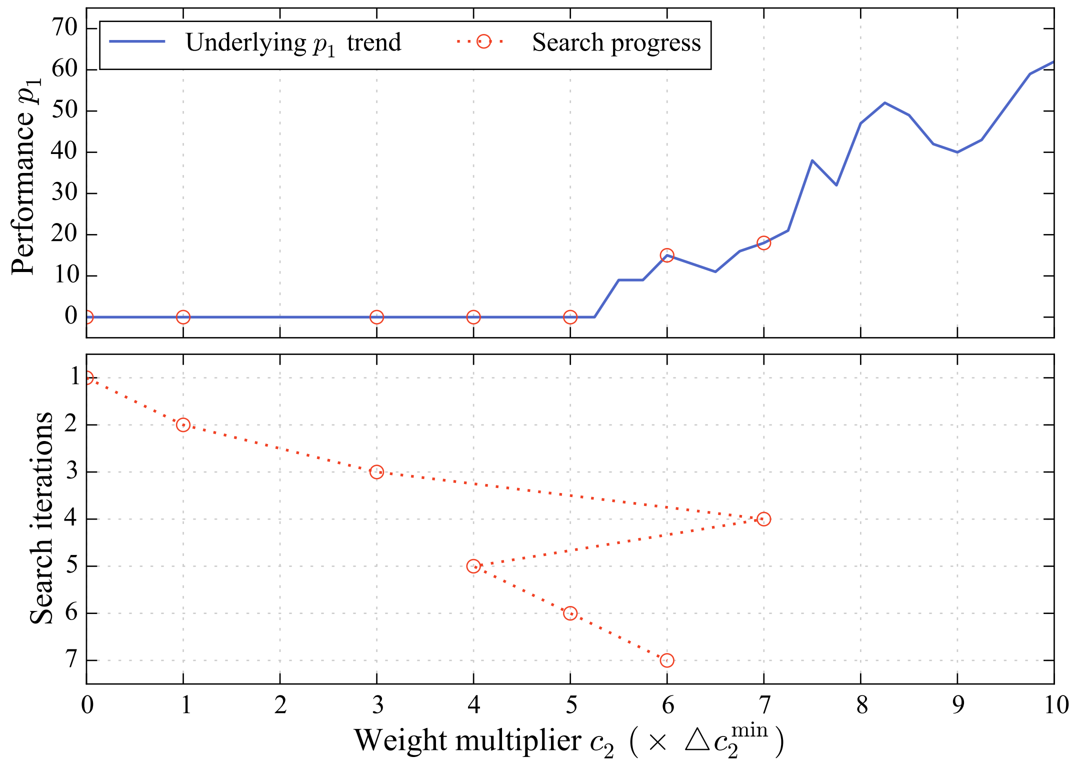


Fig. 7.5 Example of the operation of the procedure for tuning selector weight multiplier c_2

c_2^{limit}, or the iteration limit $l^{\max}$ is reached. In all cases, the procedure will return c_2^{best} on termination.

This selector design has a number of design choices. These are described below, along with the options investigated in this work:

1. Objectives: the method outlined above requires there to be two ordered objectives. In this work, two options are investigated for the objectives: (1) minimising the number of overloads whilst minimising the amount of curtailment, and (2) minimising the overload energy whilst minimising curtailment.

2. Algorithms: which algorithms are considered for selection. As with the other direct selectors, considering different sub-sets of the power flow management algorithms for selection can simplify the learning task and thus affect the performance of a selector. The following sets of algorithms are considered for selection for each system:

    - 33 kV meshed distribution system (4 options): (1) PFM-OPF and PFSF-TMA, (2) PFM-OPF and PFSF-LP, (3) PFM-OPF, PFSF-TMA, and PFSF-LP; plus (4) the set of all algorithms (including the baseline option of not applying any algorithm).

- IEEE 14-bus system (5 options): each of the pairs of algorithms that could remove all overloads were considered, namely: (1) PFM-CSP and PFSF-Egal, (2) PFM-CSP and PFSF-TMA, (3) PFM-CSP and PFM-OPF, and (4) PFM-OPF and PFSF-Egal; along with (5) the set of all algorithms.

- IEEE 57-bus system (3 options): (1) PFM-CSP and PFM-OPF, which is the pair of algorithms that minimises the number and energy of overloads, when compared with all other algorithm pairs for that system; (2) PFM-OPF, PFM-CSP, and PFSF-Egal, which is the set of algorithms that minimises the overload energy; and (3) the set of all algorithms.

3. Tuning parameters: the values of $\triangle c_2^{\min}$, $\triangle c_2^{\max}$, and l^{max} affect the search process in Algorithm 7.1 and may affect the performance of the resulting selector, particularly as the precision at which $\triangle c_2^{\min}$ is found depends on $\triangle c_2^{\min}$. However, these parameters are not as important as the other design choices so their values have been fixed in this work, in order to limit the scope of this work to a tractable number of design variants. The values used are: $\triangle c_2^{\min} = 1.0 \times 10^{-3}$, $\triangle c_2^{\max} = 10.0$, and $l^{max} = 100$.

   There is an additional design choice with respect to the tuning that is not explicitly shown in Algorithm 7.1, which is how to split the available data for selector training into training and test data during the tuning process. This is required as it is good practice in machine learning to use independent data sets for any testing during a tuning process (often referred to as a *validation* set) and for the final evaluation of a tuned model. In this work, a 2:1 split is used for the training and validation data during tuning; additionally, once the tuning process has been used to determine a c_2 value for a particular model, the model is then re-trained using all the training data available.

4. Comparison used in search: as seen in Figure 7.4, the value p_1 of the performance measure related to the first objective (the number of overloads in the figure) may not increase monotonically. Furthermore, as has been observed during the development of selectors for this work, p_1 may decrease before increasing as c_2 becomes larger. Because of the potential non-monotonicity of p_1, the choice of comparison used in line 9 of Algorithm 7.1 is critical for determining how quickly the tuning process will terminate, which could result in a c_2 value lower than could be obtained if the search process continued for additional steps. For this reason, two design options for the comparison are investigated in this work: (1) $p_1 = p_1^{\text{target}}$, and (2) $p_1 \leq p_1^{\text{target}}$.

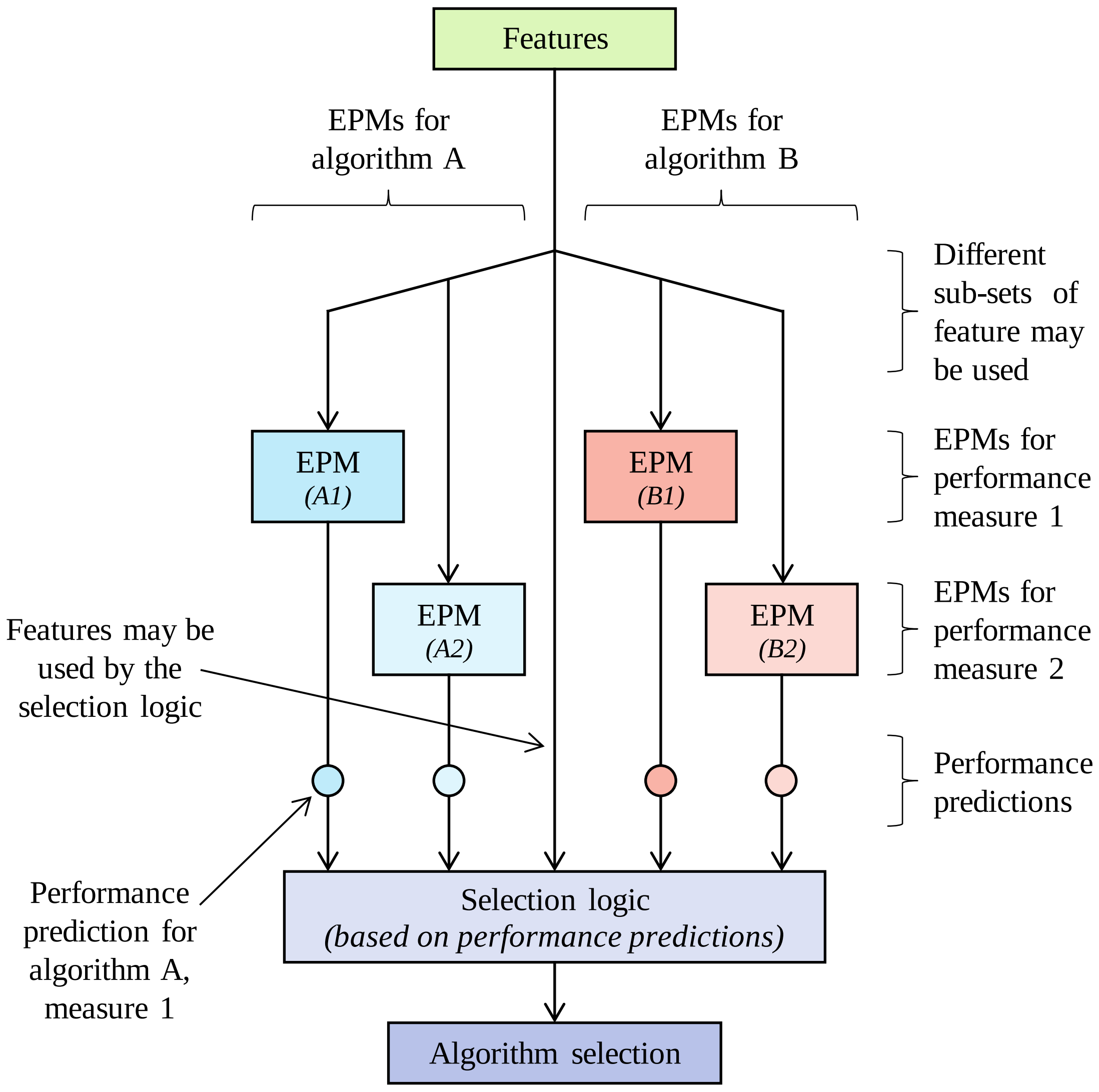

Features
EPMs for algorithm A
EPMs for algorithm B
Different sub-sets of feature may be used
EPM (A1)
EPM (B1)
EPMs for performance measure 1
EPM (A2)
EPM (B2)
EPMs for performance measure 2
Features may be used by the selection logic
Performance predictions
Performance prediction for algorithm A, measure 1
Selection logic (based on performance predictions)
Algorithm selection

Fig. 7.6 Example structure of an EPM-based algorithm selector

7.2 Preparation of training data for EPM-based selectors

Figure 7.6 illustrates the structure of an EPM-based algorithm selector. Within this structure there is an EPM for each combination of algorithm and performance measure. The EPMs provide performance predictions when given features as inputs, which for a particular EPM may be a sub-set of the features that are available. The performance predictions, perhaps supplemented by feature values, are used within the selection logic element to make an algorithm selection decision. Although the structure shown in the figure is for two algorithms (A and B) and two performance measures (1 and 2), it can be extended to an arbitrary number of algorithms and measures by increasing the number of EPMs.

An EPM-based selector has the following design choices:

1. Performance measures: the performance measures that the EPMs are used to predict depend upon the performance objectives considered. Where there are multiple objectives, the design choice is whether some of the performance measures should be ignored, which could help improve the performance of an EPM-based selector if the performance measures in question cannot be predicted accurately.

2. Algorithms: the design choice here is whether to predict the performance of, and subsequently select from within, the set of all available algorithms or a particular sub-set. If one algorithm's performance cannot be predicted accurately, which could confuse the selection logic, the overall performance of the EPM-based selector may be improved by ignoring that algorithm in the selector.

3. Features provided to EPMs: the design choice here is whether a common sub-set of features is supplied as inputs to every EPM, or if different sub-sets are provided to particular EPMs.

4. Features provided to selection logic: a choice between whether a sub-set of features are provided to supplement the performance predictions as inputs to the selection logic element, or if only the performance predictions are provided.

5. Machine learning models: which machine learning models and parameters are used for creating the EPMs, including whether the same type of model is used for all EPMs or if different models are used for particular EPMs; for example, the type of model used could be varied depending on which performance measure is being predicted, such as using a classifier to predict an integer-valued performance measure that takes on only a few values, or using a regression model to predict a real-valued performance measure.

6. Selection logic: the function of this element is to make selection decisions based on the performance predictions and the sub-set of features, if they are provided, so a variety of decision strategies could be employed. For example, one strategy would be simply to select the algorithm with the best predicted performance; while another would be to create a direct selector that accepts performance predictions as inputs, and to use that as the selection logic element.

Although there are only a handful of design choices, each may be associated with a large number of options and the effect of exploring different options for each choice is multiplicative. Therefore, investigating even a small number of options for each design choice can quickly become intractable. For example, if 10 different machine learning models were trialled for each EPM in Figure 7.6, it would result in $10^{2\times2} = 10,000$ different EPM-based selector designs to be evaluated. If the structure in the figure were extended to the five power flow management algorithms (as well as the option to not apply any control) used in this work, there would be 10^{12} possible selector designs, which is an intractable number of selectors to investigate.

Due to the high dimensionality associated with the different possible EPM-based selector designs, the number of design options has been restricted in this work in order to make a study of EPM-based selector designs tractable. However, the design options selected still allow for a range of designs to be examined, which increases the likelihood of there being some selectors that deliver effective performance. In particular, the following design options have been investigated:

1. Performance measures: EPMs for each performance measure have been produced, which have been combined to consider each of these sets of objectives: (1) minimising the number of overloads, (2) minimising the overload energy, (3) minimising the number of overloads whilst also minimising the curtailment, and (4) minimising the overload energy whilst also minimising the curtailment. For the 33 kV meshed distribution system, (1) and (2) have not been investigated, as there is no potential performance benefit per-state algorithm selection for either of the overload measures due to PFM-OPF being able to remove all overloads.

2. Algorithms: the algorithm combinations described in Section 7.1.1 have been used.

3. Features provided to EPMs: only one design option has been investigated, which is where the same set of features is supplied to all EPMs within a selector.

4. Features provided to selection logic: only one design option has been investigated, which is where only the performance predictions of the EPMs are provided.

5. Machine learning models: a number of options have been investigated, with EPMs being created using a number of different machine learning models (listed in Section 7.3). To reduce the number of possible designs investigated, the EPMs for a particular performance measure within an EPM were restricted to being only a single machine learning model type.

6. Selection logic: only one design option has been investigated, which is to select the algorithm with the best predicted performance, assessed against the objectives considered. In the event of a tie, when two or more algorithms give the best predicted

performance for a state, then the algorithm from the set that has the best overall performance on the training states is selected.

7.3 Machine learning models

7.3.1 Models used in this work

In pursuit of research objective 2, a broad selection of machine learning models have been investigated to create algorithm selectors for per-state selection of power flow management algorithms. The intention was not to perform an exhaustive evaluation of all possible machine learning models to find which was “best” for creating selectors for power flow management; rather, the intention was to perform an extensive evaluation – using models from the main model families – in order to ascertain if machine learning *can* be used to create effective selectors for power flow management.

The investigation of a broad selection of machine learning models was enabled by using the Java-based WEKA software [108] to train machine learning models to be used as algorithm selectors for power flow management. WEKA is well established in the machine learning research community and includes implementations of many popular state-of-the-art machine learning algorithms, including those that train models for classification or regression.

Many machine learning algorithms have a number of parameters that can be tuned to optimise the performance of the trained models; however, tuning involves an iterative process of training and testing different model parameterisations, which can require a significant amount of time. Therefore, in order to limit the machine learning task to a tractable size, no tuning has been performed and the models trained in this work have been trained using the default parameters from WEKA. However, there are a select few models whose parameters were known a priori to significantly affect the performance of the trained models, so a small number of parameter variations have been investigated for those models.

Table 7.1 lists the machine learning models investigated in this work. Strictly speaking, the table lists the *algorithms* used to train the models, but in the following discussion, the names of machine learning algorithms are used synonymously to refer to the models that they produce. The following information is presented in the table:

- WEKA class name: the name of the Java class of the model within WEKA.
- Model category: the family of models or other grouping to which the model belongs. A description of each category is provided in Section 7.3.2

- Type: if the model can be used for classification (“C”), regression (“R”), or both. All the direct selectors use classification models, while the EPM-based selectors use regression models for the EPMs with the exception of the EPMs that predict the number of overloads. As the number of overloads can be treated as a discrete class, both classification and regression models have been investigated to produce EPMs for that performance measure.

- Variations: whether multiple variations of a model, with different parameter values, have been investigated. Details of the variations used are given in Section 7.3.3.

- Ref.: reference describing the model and the algorithm used for learning.

In total, 55 models for classification have been investigated (38 base model types plus a number of variations), and 35 models for regression (24 base models plus variations).

7.3.2 Descriptions of machine learning model categories

Artificial neural networks

Artificial neural networks (ANNs) consist of interconnected layers of nodes – the neurons – that are analogous to biological neural networks. The output of a neuron is determined by an *activation* function, which is a function of the weighted outputs of the neurons it is connected to. The layers of neurons are arranged into a single *input* layer, which takes the features as input, followed by one or more *hidden* layers, with a final *output* layer, which provides the prediction of the model. For classification, there is one output neuron per class, whereas for regression there is usually a single output neuron with a linear activation function. Typically, the size of an ANN is set before training, and the training process is used to determine the strength (weights) of the connections between the neurons.

Both ANN models used in this work (`MLPRegressor` and `MultilayerPerceptron`) create a type of ANN known as a *multilayer perceptron*, which are feedforward networks with sigmoid activation functions.

Bayesian learners

These methods apply Bayes’ theorem to estimate the probability that an example belongs to a particular class. `NaiveBayes` uses the training data to learn conditional probability distributions for the features (given the class), whereas `BayesNet` learns a Bayesian network where the nodes represent different conditional probability distributions.

Table 7.1 Machine learning models used in this work

WEKA class name	Model category	Type	Variations	Ref.
`AdaBoostM1`	Ensemble	C	-	[109]
`ADTree`	Tree	C	-	[110]
`Bagging`	Ensemble	C	-	[111]
`BayesNet`	Bayesian	C	-	[108]
`BFTree`	Tree	C	-	[112]
`ConjunctiveRule`	Rule	C & R	-	[108]
`DecisionStump`	Tree	C & R	-	[108]
`DecisionTable`	Rule	C & R	-	[113]
`DTNB`	Rule	C	-	[114]
`FT`	Tree	C	-	[115]
`GaussianProcesses`	Other	R	-	[116]
`HoeffdingTree`	Tree	C	-	[117]
`HyperPipes`	Other	C	-	[108]
`IBk`	Lazy	C & R	Y	[118]
`IsotonicRegression`	Other	R	-	[108]
`J48`	Tree	C	Y	[13]
`J48graft`	Tree	C	Y	[119]
`JRip`	Rule	C	-	[120]
`KStar`	Lazy	C & R	-	[121]
`LADTree`	Tree	C	-	[122]
`LeastMedSq`	Linear regression	R	-	[123]
`LibSVM`	SVM	C & R	Y	[124]
`LinearRegression`	Linear regression	R	-	[108]
`LMT`	Tree	C	-	[115]
`Logistic`	Other	C	-	[125]
`LWL`	Lazy	C & R	-	[126]
`M5P`	Tree	R	-	[127]
`M5Rules`	Rule	R	-	[128]
`MLPRegressor`	Artificial neural network	R	-	[108]
`MultilayerPerceptron`	Artificial neural network	C & R	-	[108]
`NaiveBayes`	Bayesian	C	-	[129]
`NBTree`	Tree	C	-	[130]

(continued on next page)

Table 7.1 (continued from previous page)

WEKA class name	Model category	Type	Variations	Ref.
`OneR`	Rule	C	-	[131]
`PaceRegression`	Linear regression	R	-	[132]
`PART`	Rule	C	-	[133]
`RandomForest`	Ensemble	C & R	-	[134]
`RandomTree`	Tree	C & R	-	[108]
`RBFClassifier`	Other	C	-	[135]
`RBFNetwork`	Other	C & R	-	[135]
`RBFRegressor`	Other	R	-	[135]
`REPTree`	Tree	C & R	-	[108]
`Ridor`	Rule	C	-	[136]
`SGD`	Other	C	-	[108]
`SimpleCart`	Tree	C	-	[137]
`SimpleLinearRegression`	Linear regression	R	-	[108]
`SimpleLogistic`	Other	C	-	[138]
`SMO`	SVM	C	-	[139]
`SMOreg`	SVM	R	-	[140]
`ZeroR`	Rule	C & R	-	[108]

Ensembles

These methods (which are also referred to as learning *meta-algorithms*) combine the predictions of a group of other models – the *ensemble* – in order to make a prediction. Often, the predictive performance of an ensemble is more powerful than the individual models that the ensemble is comprised of.

The main difference between the ensemble methods is in how they train the constituent models and combine their predictions. For example, the `Bagging` method trains models on separate random sub-samples of the training data, whereas `AdaBoostM1` iteratively trains a sequence of models with training sets that are altered to accentuate misclassified examples. `Bagging` makes predictions by combining the outputs of the constituent models by voting (for classification) or averaging (for regression), whereas `AdaBoostM1` – which can only be used for classification – uses a weighted sum of the class probabilities from each model. The choice of which models are used in an ensemble is usually an adjustable parameter, although for some ensemble methods this choice is fixed, such as `RandomForest`, which only uses `RandomTree` models.

Lazy learners

Lazy learners essentially do no training at all, instead using the training set directly to make predictions. `IBk`, for example, is a nearest-neighbour model that predicts the class or value of an example based on the *k* training examples that are closest to it in the feature space.

Linear regression

These methods create linear functions of the feature values that are used to predict the value of an example, with the parameters of the function fitted to minimise error when the model is applied to the training data. One of the main differences between the methods listed in the table is how error is treated during training, for example `LinearRegression` uses the method of least squares (minimising the sum of the squared errors), while `LeastMedSq` minimises the median of the squared errors.

SVM-based learners

Support vector machines (SVMs) use *kernel* functions to project the feature space into a higher-dimensional space. A hyper-plane is then fitted to the higher-dimensional representation that achieves the maximum separation between examples of different classes (in the case of using a SVM for classification), or that most closely fits the training example target values (in the case of regression).

Rule learners

These are different methods that induce rules from the training data, where each rule represents a particular prediction and is activated if the feature values match those described in the rule. The main differences between the methods listed in the table is how single rules are learned and what post-processing of the set of rules take place, such as pruning rules from the set to prevent overfitting.

Tree learners

These models take the form of decisions trees, which are hierarchical structures consisting of decision nodes (branches), where feature values are used to determine which branch is followed, and terminal nodes (leaves), which determine the predicted output. The different tree learning methods listed in the table vary in how they determine which features are used at the branches – for example, `J48` uses the feature that results in the most information gain, while `RandomTree` chooses features at random – as well as varying in what form the leaves take – most simply label a leaf with the predicted output, although some methods use other models at their leaves, for example `LMT` uses logistic regression models.

Other learners

These methods are not from a single model family and use a variety of approaches. For example, `GaussianProcesses`, also known as *Kriging*, is a regression method that interpolates between known data points (from the training set) using a Gaussian process; while `Logistic` and `SimpleLogistic` implement logistic regression, which – despite its name – is a classification method that fits a logistic function of the weighted sum of feature values.

7.3.3 Model variations

For the following models, the stated parameter variations were investigated:

- `AdaBoostM1` and `Bagging`: five different models were used within these ensemble learners: `DecisionStump` (the default ensemble model within `AdaBoostM1`), `NaiveBayes`, `LogisticRegression`, `OneR`, and `REPTree` (the default ensemble model within `Bagging`).
- `IBk`: the following number of nearest neighbours were used: 1 (the default), 5, 10, 50, and 100.
- `J48` and `J48graft`: both pruned and unpruned trees were trained.
- `LibSVM`: for classification the default `C-SVC` training algorithm was used, while for regression both the `epsilon-SVR` and `nu-SVR` algorithms were used. Four different kernels were used (linear, polynomial, radial basis function, and sigmoid) for each training algorithm.

7.4 Feature construction and selection

It is vital that a machine learning model is provided with *features* as inputs that are predictive of the target output for each training and testing example. Some features can be informative and will relate to the targets while others may be uninformative, with no relationship to the targets, or redundant, with values that are strongly or perfectly correlated with the values of other features. If there are too few features used, important information may be missed that is necessary to make good predictions. Conversely, too many features can lead to overfitting, where a model becomes biased towards the examples in the training set and thus generalises poorly (when this happens, the model can accurately predict the targets values of training examples but performs poorly when applied to test examples). Furthermore, the number of features used will also affect the time and space required to train a model.

When machine learning is used to construct algorithm selectors, the features represent characteristics of the problems the algorithms are being applied to, which relate to a power system's state when selecting algorithms for power flow management. In this context, it is important that the features are predictive of which algorithm should be selected (in the case of direct selectors) or are predictive of individual algorithms' performance with respect to particular performance measures (in the case of EPM-based selectors).

In a review paper on the subject, Guyon and Elisseef [141] describe two main aspects to consider with relation to features: (1) feature construction, and (2) feature selection. These two aspects are discussed below, particularly in relation to using machine learning for algorithm selection.

7.4.1 Feature construction

Feature construction is concerned with what characteristics of the training and testing examples are represented. Domain knowledge can be used to determine features that are relevant to what the examples represent, so for algorithm selection for power flow management, the features will relate to the state of the power systems that the algorithms are being selected for. Feature construction also involves applying transformations to the base set of features. Transformations include normalisation and scaling (so that all features vary across a similar range, which many machine learning algorithms require), dimensionality reduction (where a higher-dimensional feature space is transformed into a lower-dimensional space, using techniques such as principal components analysis), and feature expansion (creating new features by applying functions to existing features; for instance, constructing features that are the products of other features). Although transformations do not increase the amount of information available to the machine learning models – in fact, some transformations such as dimensionality reduction may actually reduce the amount of information – they can reveal important relationships in the data, and thus help improve predictive performance.

7.4.2 Feature selection

Feature selection is concerned with determining a subset of a set of potential features that is either particularly predictive, particularly concise and therefore efficient in terms of the time and space requirements for training and testing a model, or that reveal more information about the process that generated the data.

Guyon and Elisseef [141] distinguish the following broad strategies for feature selection:

1. **Filter methods**: these are applied as a pre-processing step to determine potentially relevant feature sets before training a machine learning model. Each feature is ranked according to a particular criterion; for example, the correlation between a feature and the targets, or information theoretic criteria such as the mutual information between a feature and the targets. From the ranks, a subset of features can be determined.

   The main advantage of filter methods is that they are applied before the model is trained, so can therefore be efficient in terms of the time and space required for their application. They are also agnostic of the machine learning model used. However, filter methods may result in redundant features being selected (as relationships between different features may not be recognised), and they may ignore features that may be low ranking, and therefore appear irrelevant, but that could be highly predictive when combined with other features.

2. **Wrapper methods**: this is where candidate feature subsets are ranked by using them to train and then evaluate a machine learning model, which is treated as a black box. This typically involves splitting the training set in two, with one part being used for training a model, given a particular feature set, while the other part – the validation set – is used for evaluating the predictive performance of the trained model.

   The feature subsets are determined iteratively, either by adding features to a subset that is initially empty, based on what predictive performance they add (*forward selection*), or by removing features from a subset that initially contains all features, based on how little they add to predictive performance (*backward elimination*). Which feature is added or removed at each step is determined by a search algorithm, such as best-first, branch-and-bound, or genetic algorithms.

   The main advantages of wrapper methods are that they can help to avoid overfitting and that they consider how each variable affects the predictive performance of the trained model, thus better rejecting redundant features. The main disadvantage is the "brute force" nature of training models to evaluate candidate feature subsets, which can be expensive in terms of the time and space required.

3. **Embedded methods**: these are feature selection methods that form part of a particular machine learning algorithm. For example, the `J48` algorithm (WEKA's implementation of the C4.5 decision tree learning algorithm [13]) implicitly performs feature selection when it considers what feature to split on for each decision node in the tree, based on the information gain associated with each feature.

7.4.3 Features used in this work

For the power flow management algorithm selection application considered in this work, the training and test examples represent particular states of the case study power systems. The state space of each system consists of only a few variables, and these have been used as the feature set for machine learning. In particular, the feature set for each system comprises:

- The output level of each generator. Where two or more generators are scaled by the same factor (and therefore always have the same output level as each other), the output level of only one of the generators is included in the feature set, preventing duplication.
- The load level (scaling factor).

For the 33 kV meshed distribution system, the feature set comprises three variables, while the IEEE 14-bus system has five, and the IEEE 57-bus has four. These fixed feature sets are consistently used for all the machine learning algorithms considered, in order to allow fair comparison of the machine learning algorithms. The features, as they are the state variables, fully characterise each state so no information is lost but the information is at a high-level, which some models may not be able to use effectively. Although using a fixed feature set may restrict the predictive performance of some model types that could perform better with different features, performing feature selection for each machine learning algorithm considered would be prohibitively slow, so is not performed in this work.

7.5 Training data volume

In any machine learning application, a model trained with too few training examples will have high variance and will be unable to generalise well when applied to separate test examples. Conversely, increasing the number of training examples may not result in an improvement in prediction performance, and is associated with greater time and space requirements. The amount of predictive performance that can be attained from a particular amount of training data can be determined empirically; however, in the machine learning literature it is typical to use ratios of training to test data of between 60:40 and 80:20.

The typical ratios of training to test data are achieved in this work by generating additional power flow management algorithm performance data. This required testing the power flow management algorithms on an extra 30,000 states for each case study system, over and above the states described in Chapter 4.

For IEEE 14- and 57-bus systems, the additional states were generated randomly using the same methods as described in Sections 4.4.2 and 4.5.2, for each system respectively.

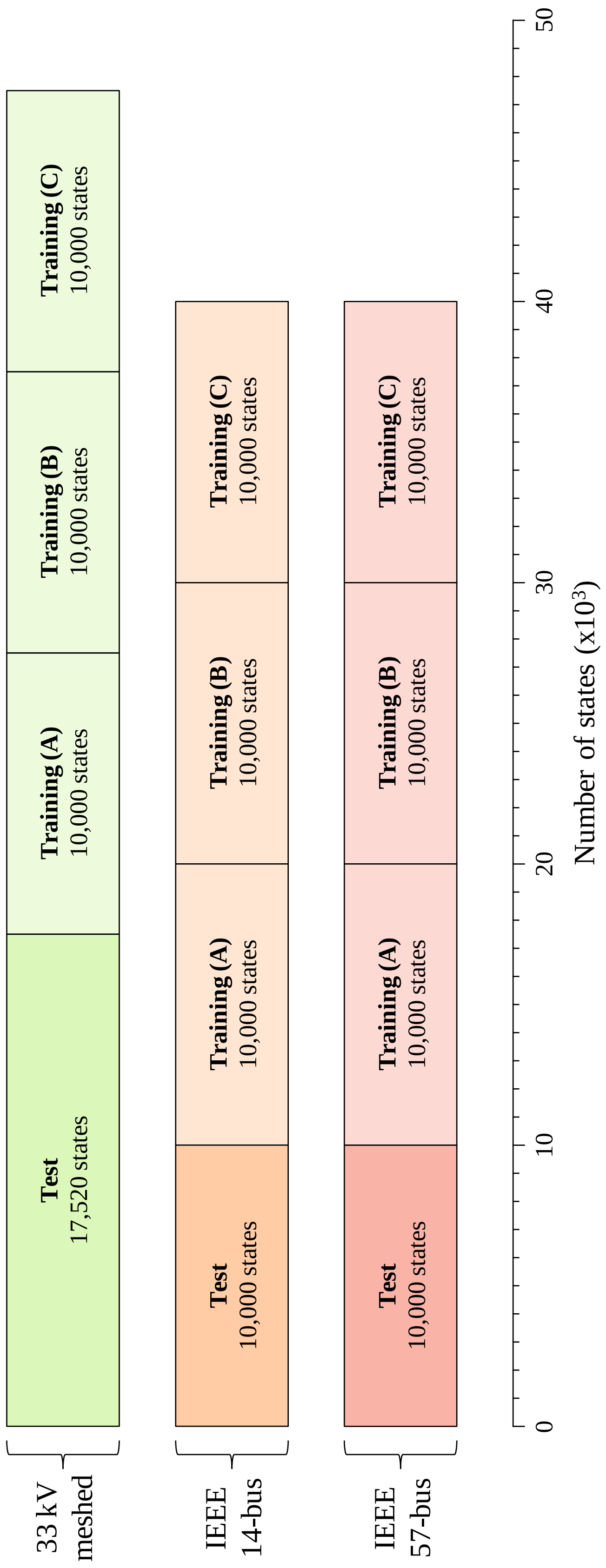


Fig. 7.7 Amount of power flow management algorithm performance data per system for use training and testing algorithm selectors

For the 33 kV meshed distribution system, the year-long profile data which was used to generate the states described in Section 4.3.2 had already been completely used to create the previously used to test the power flow management algorithms, so an alternative method was required to generate additional states. This was achieved by replacing each profile with random variables drawn from independent uniform distributions, similar to how states were generated for the other case study systems, which allowed for an arbitrary number of test states to be generated.

A summary of the performances of the power flow management algorithms when applied to the additional test states can be found in Appendix C.

Figure 7.7 illustrates, to scale, the amount of data available per case study system for training and testing machine learning algorithms as algorithm selectors. For each system, the data is divided into four splits:

1. The *test* split, being the states originally used for testing the power flow management algorithms as described in Chapter 4. For the 33 kV meshed distribution system, this split contains performance data for each power flow management on 17,520 states, while for the other systems there are 10,000 states in the test split.

2. The *training (A)* split, comprising the first 10,000 states of the additional states tested for each system.

3. The *training (B)* split, comprising the second 10,000 additional states.

4. The *training (C)* split, comprising the remaining 10,000 additional states.

In this work, each model has been trained with 30,000 states (all three training sets). For the direct selectors with tuned weights, the (A) and (B) sets were used for training during the tuning process, with the (C) sets used for validation. In addition, a small number of the models were re-trained using different training set sizes in order to investigate the effect of the training data volume on the performance of the selectors.

7.6 Conclusions

This chapter has summarised the different designs developed in this work for per-state algorithm selectors for power flow management, namely direct and EPM-based selectors. Specific design choices associated with each selector design have been described along with the design options investigated in this work. Additionally, design choices that are common across all of the designs have been described – the choice of machine learning model, features used, and the volume of training data – along with the options for these that have been investigated. The work in this chapter forms the basis for the next, Chapter 8, where the performance of each selector design variant is tested and analysed.

Chapter 8

Performance of algorithm selectors for power flow management

This chapter reports the results of implementing the direct and EPM-based selector designs described in Chapter 7. The performance of the selectors is evaluated in order to determine whether machine learning-based algorithm selectors can provide performance benefits for power flow management, in support of research objective 2.

The chapter begins with Section 8.1 with a reminder of the selector designs and introduces a number of conventions used throughout the chapter. An overview of the performance of all the selectors created in this work is then provided, in Section 8.2, followed in Section 8.3 by the performance of the most effective selectors created for each design variant.

The subsequent sections then analyse how particular aspects of the selectors' designs, as described in Chapter 7, affect the selectors' performance. Section 8.4 examines whether the objectives considered when creating the selectors are reflected in the selectors' performance. Section 8.5 examines whether selectors that consider larger set of algorithms are able to exploit the potential performance that the larger set can provide, or whether considering larger sets decreases the performance of the selectors. Section 8.6 compares the performance of selectors created using different machine learning model types, in order to identify which models are most promising for use in algorithm selectors for power systems control. Section 8.7 concentrates on weighted direct selectors, and investigates whether tuning the weights can provide performance benefits.

The chapter then closes with conclusions drawn about the performance of the selectors created for power flow management.

8.1 Preliminary remarks

This section provides a reminder of the selector design variants investigated (direct and EPM-based, as described in Chapter 7) and introduces a number of presentational conventions that are used throughout this chapter.

8.1.1 Direct selectors

Direct selectors (described in Section 7.1), use machine learning models to directly predict which algorithm is likely to have the best performance. Three designs were investigated:

- *Unweighted* direct selectors (whose design is described in Section 7.1.1), which were created using training examples without weights associated with them, so each training example was treated with equal importance during training.
- *Weighted* direct selectors (Section 7.1.2), which were created using training examples with weights. The weights were determined according to a particular single objective (minimising either the number or energy of overloads) so that training examples that provided a larger performance gain were prioritised.
- *Tuned* direct selectors (Section 7.1.3), which also had weighted training examples. However, the weights represent two different objectives, with the balance between the two objectives being set by a process of “tuning” the weights so a particular performance of the resultant selector was obtained.

Note that in some of the tables and figures in this chapter, “unweighted” and “weighted” are abbreviated to “unw.” and “wei.”, respectively, where concise presentation was necessary.

8.1.2 EPM-based selectors

EPM-based selectors use machine learning models to predict the performance of each algorithm, with the algorithm with the best predicted performance then selected. Their design is discussed in Section 7.2.

The EPM-based selectors in this work have either one or two sets of EPMs: the first set (subsequently referred to as the “primary” EPMs) predicts the performance of the power flow management algorithms with respect to either the number or energy of overloads; whereas the optional second set of EPMs (subsequently referred to as “secondary” EPMs) predicts curtailment performance.

Note that results are not presented for three of the machine learning model types listed in Section 7.3.1. For ADTree and SGD, no selectors were produced as these are binary classifiers, which could not be applied to predict either the number or energy of overloads as those performance measures could take on more than two values. GaussianProcesses has poor scalability to large numbers of training examples ($\mathscr{O} \sim N^3$, where N is the number of training examples [142]), which meant that training required more memory than was available on the machine used for this work.

8.1.3 Machine learning model types

When discussing the different machine learning model types, the WEKA class names listed in Section 7.3.1 are used. A number of the model types have several variants, and the following nomenclature has been used:

- For the ensemble learners AdaBoostM1 and Bagging, the name of the model type used within the ensembles is given in parentheses, for example: AdaBoostM1 (DecisionStump).
- For the nearest-neighbour IBk model, the number of nearest neighbours is given in parentheses, for example: IBk (10 NN).
- For J48 and J48graft, whether pruning was used is indicated in parentheses.
- For the LibSVM variants, the training algorithm and kernel are indicated in parentheses, for example: LibSVM (nu-SVR, poly), where “poly” is an abbreviated form of “polynomial” kernel.
- Although only a single variant of MultilayerPerceptron has been used in this work, it is referred to as MLP where concise presentation was needed.

8.1.4 Objectives

The names of the objectives have been abbreviated where concise presentation was necessary to: number of overloads (OL), overload energy (OE), and curtailment (CT).

8.2 Performance overview

This section provides an overview of the performance of the direct and EPM-based selectors developed for the 33 kV meshed distribution system and the IEEE 14- and 57-bus systems. The overview is provided via three means. Firstly, a table (Table 8.1) lists the numbers of selectors created for each selector design variant, the number of those that provide performance benefits, and the number of selectors that provide *statistically significant* performance benefits. The definition of "performance benefit" depends on the system and the performance measure considered, and is explained in the subsequent text for each system. Secondly, a series of plots (Figures 8.1 to 8.5) present the performance of the selectors graphically, alongside the performance of the oracles and the power flow management algorithms. The plots include general views (such as Figure 8.1a) of the entire performance space, showing the performance of all the selectors, and detailed views (such as Figure 8.1b) that show the performance of the selectors whose performance is close to the both the oracles and the most effective power flow management algorithms. Thirdly, the text provides an explanation of the table and figures, and draws out the key findings about the overall performance of the selectors.

Whether a performance benefit is statistically significant is assessed using the same method as used for comparing the performance of the power flow management algorithms and the oracles in Chapters 4 and 5 (described in Section 4.1.3). The performance of a selector is compared to that of the most effective algorithm for the performance measure of interest using a paired t-test, yielding a p-value. The p-values of all the comparisons of performance made for a particular system and performance measures of interest are then collected together (they are treated as a single family of hypotheses for each system) and the Benjamini-Hochberg procedure is applied to determine which represent statistically significant results, while correcting for making multiple comparisons. As in the previous chapters, a significance level of $\alpha = 0.05$ is used. Note the families of hypotheses in this chapter are considered to be separate to those in Chapters 4 and 5, as the hypotheses in those chapters were comparing the performance of all the power flow management algorithms (and oracles) for all performance measures, whereas the hypotheses in this chapter compare the performance of algorithm selectors to that of one or two power flow management algorithms for the performance measures of interest.

A summary of the overall performance of the direct and EPM-based selectors created for the 33 kV meshed distribution system is shown in the first group of four rows of Table 8.1. As described in Section 7.1.2, no (untuned) weighted selectors were created for this system as two objectives needed to be considered by the selectors. For this system, the performance benefit is when a selector applies less curtailment than PFM-OPF, while still removing all overloads (the same as PFM-OPF). So, for example, the table shows (in the first row) that 1.47% of the unweighted direct selectors apply statistically significantly less curtailment than PFM-OPF, while still removing all overloads.

Table 8.1 Summary of direct and EPM-based selectors giving performance benefits for each system and overload objective

System & (*overload objective*)	Selector design variant	Total selectors	Number of selectors giving a performance benefit		No. of selectors with statistically significant performance benefit	
33 kV meshed (*either*)	Unw. direct	1628	24	(1.47%)	24	(1.47%)
	Wei. direct	-	-	-	-	-
	Tuned direct	419	20	(4.77%)	15	(3.58%)
	EPM-based	21,175	186	(0.88%)	143	(0.68%)
IEEE 14-bus (*number*)	Unw. direct	2144	892	(41.60%)	738	(34.42%)
	Wei. direct	530	298	(56.23%)	196	(36.98%)
	Tuned direct	526	282	(53.61%)	195	(37.07%)
	EPM-based	21,175	5896	(27.84%)	2887	(13.63%)
IEEE 14-bus (*energy*)	Unw. direct	2144	612	(28.54%)	402	(18.75%)
	Wei. direct	530	222	(41.89%)	181	(34.15%)
	Tuned direct	526	216	(41.06%)	168	(31.94%)
	EPM-based	21,175	5316	(25.11%)	3165	(14.95%)
IEEE 57-bus (*number*)	Unw. direct	1288	1040	(80.75%)	1009	(78.34%)
	Wei. direct	301	165	(54.82%)	163	(54.15%)
	Tuned direct	292	159	(54.45%)	157	(53.77%)
	EPM-based	12,705	6443	(50.71%)	6301	(49.59%)
IEEE 57-bus (*energy*)	Unw. direct	1288	298	(23.14%)	172	(13.35%)
	Wei. direct	301	38	(12.62%)	27	(8.97%)
	Tuned direct	292	35	(11.99%)	26	(8.90%)
	EPM-based	12,705	1453	(11.44%)	1248	(9.82%)

Figure 8.1 plots the performance of every selector created for the 33 kV meshed distribution system, in terms of the number of overloads and the curtailment applied. The figure shows that some of the selectors always select the same algorithm, which is shown by the markers indicating the performance of selector overlapping with the marker of one of the power flow management algorithms. However, numerous markers of selectors' performance do not overlap the markers of the algorithms' performance, which shows that those selectors are selecting at least two different algorithms for a number of states.

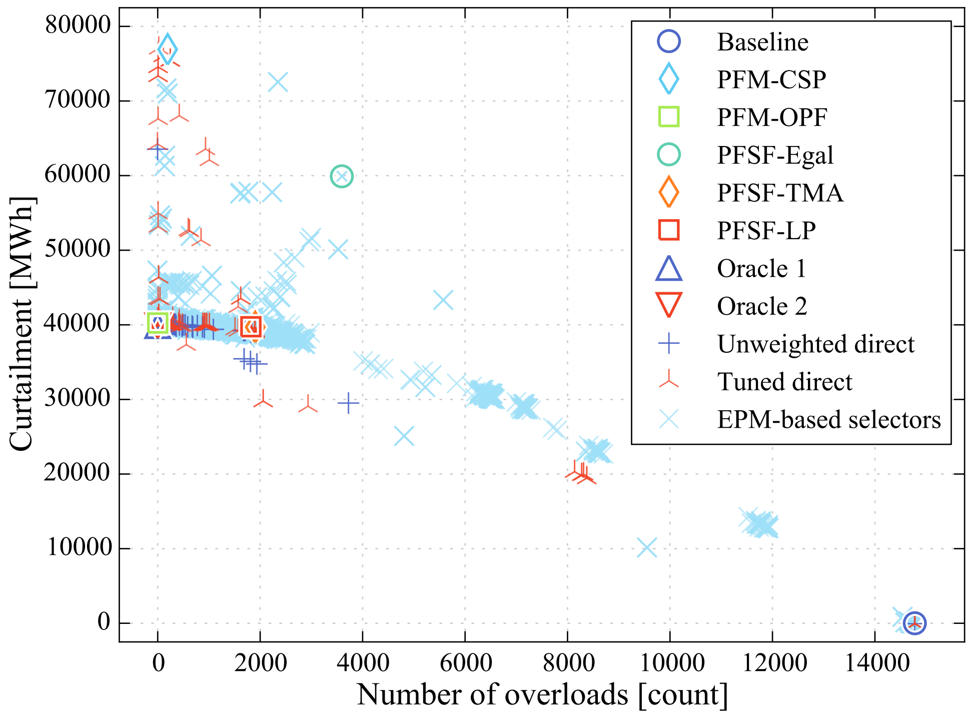
For the IEEE 14-bus system, the definition of "performance benefit" depends on which overload objective is considered. If minimising the number of overloads is considered (the second group of four rows in Table 8.1) then a selector delivers a performance benefit if it leaves fewer overloads than PFM-OPF. If minimising overload energy is considered (the third group of rows in Table 8.1) then there is a performance benefit if a selector leads to lower overload energy than PFSF-TMA. The performance of the selectors created for the IEEE 14-bus system is shown graphically in Figure 8.2, in terms of the number of overloads and the amount of curtailment, and in Figure 8.3, in terms of overload energy and curtailment.

For the IEEE 57-bus system, the same definitions of "performance benefit" are used as for the IEEE 14-bus system, however, the comparisons for both overload objectives are made relative to the performance of PFM-OPF. The fourth and fifth groups of four rows in Table 8.1 provide a summary of the selectors created for this system, according to the overload performance measure considered. The performance of the selectors for the IEEE 57-bus system is shown graphically in Figures 8.4 and 8.5 for the number and energy of overloads, respectively, against the amount of curtailment applied.

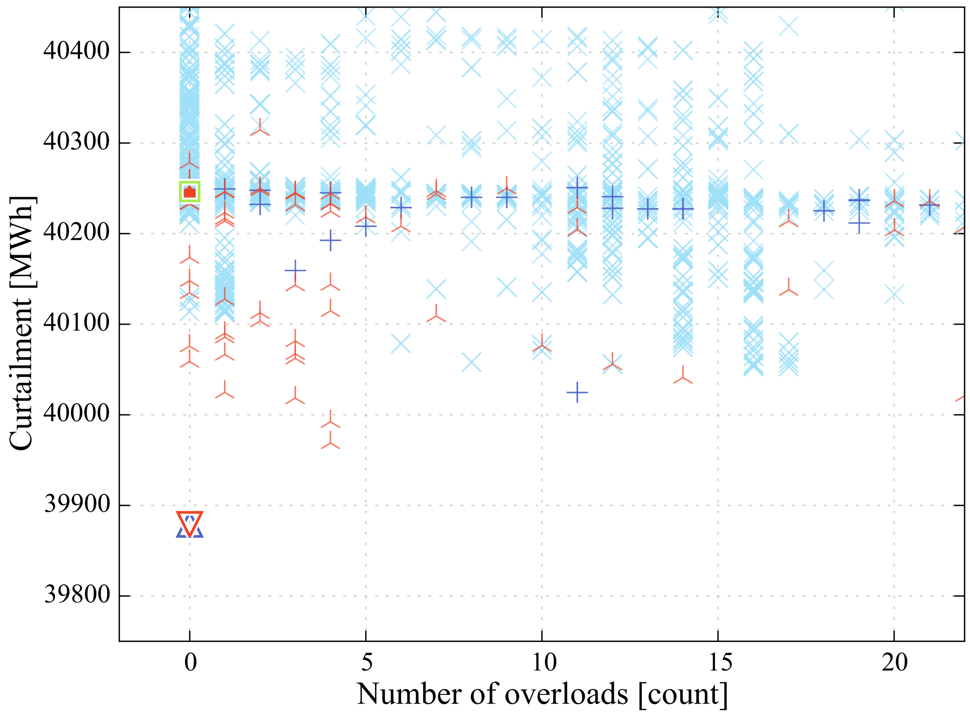
The results summarised in Table 8.1 show that each of the selector design variants considered is able to produce selectors that offer statistically significant performance benefits for each of the case study systems. There are variations in the proportion of selectors that give a performance benefit across the different systems, design variants and overload objectives considered. For example, for the 33 kV meshed distribution system, relatively few selectors provide performance benefits, which is likely due to the curtailment performance benefit also requiring algorithms to be selected that remove all overloads. Furthermore, for the IEEE 14- and 57-bus systems, each selector design variant is less likely to produce a selector that offers a performance benefit for overload energy, than for the number of overloads.

8.3 Most effective selectors

The previous section provided an overview of the performance of *all* the selectors created for each system, whereas this section concentrates only on the *most effective* selectors produced

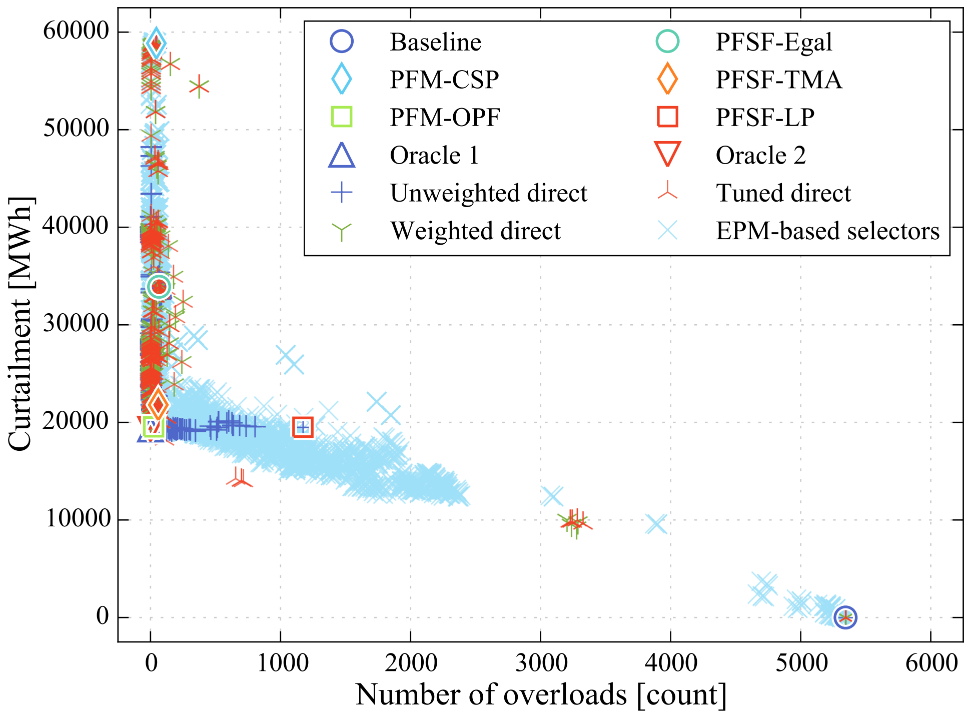


(a) View showing all selectors, power flow management algorithms and oracles

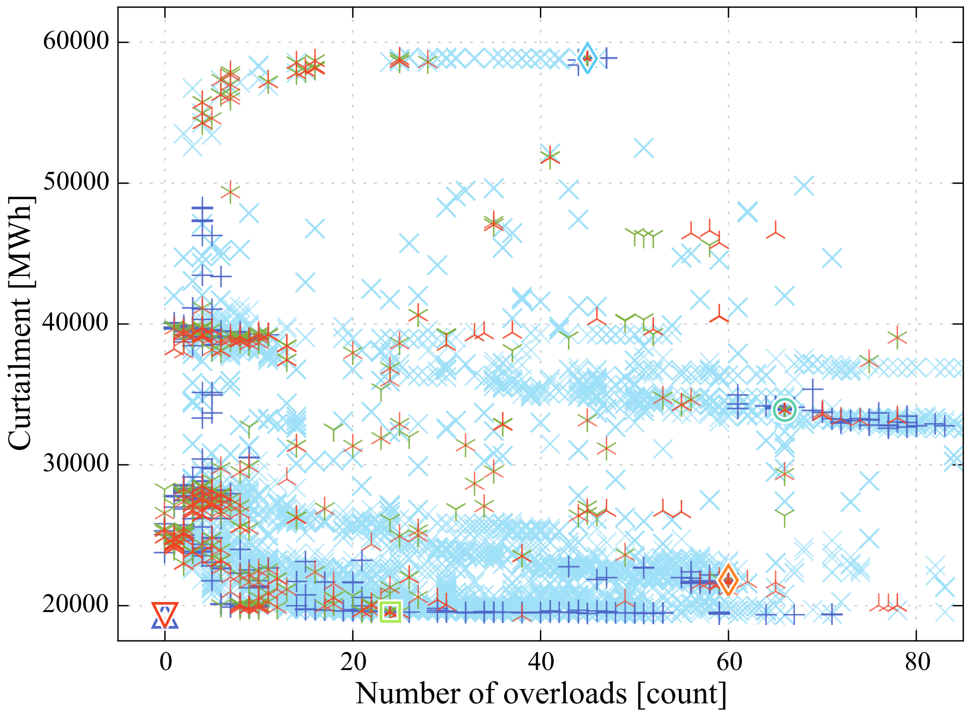


(b) Detailed view, including selectors that remove all overloads with less curtailment than PFM-OPF

Fig. 8.1 Performance of the direct and EPM-based selectors for the 33 kV meshed distribution system, with respect to the number of overloads and the amount of curtailment

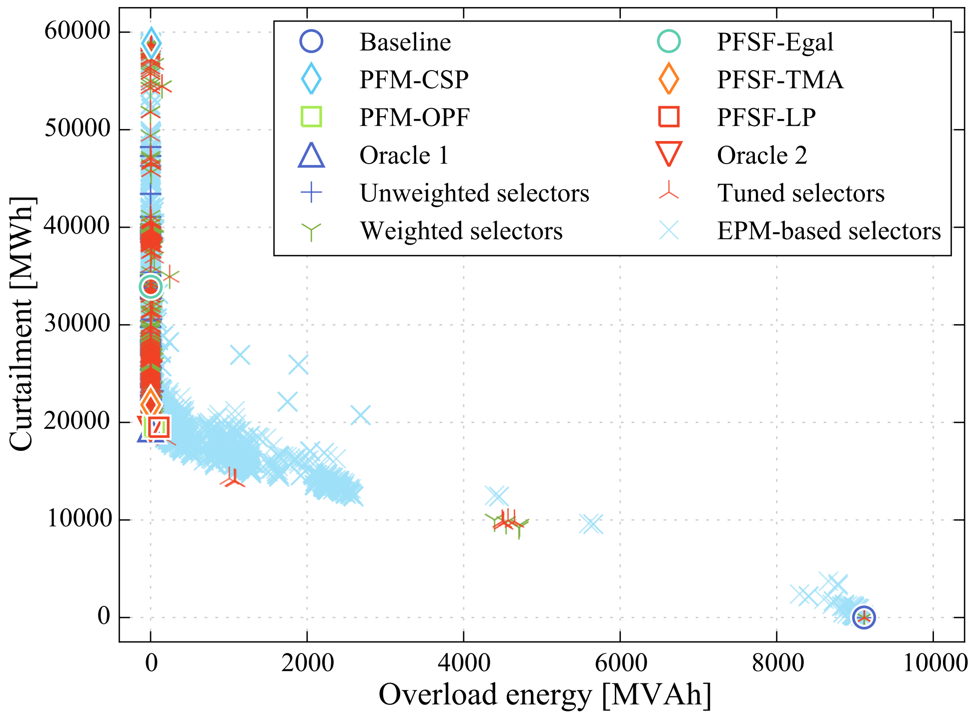


(a) View showing all selectors, power flow management algorithms and oracles

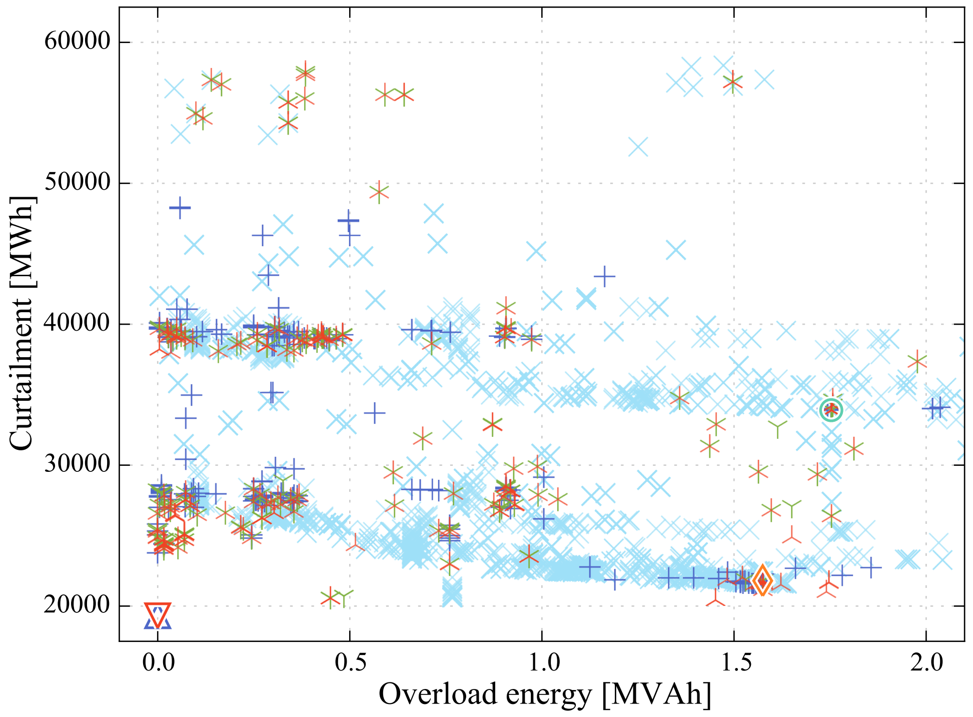


(b) Detailed view, including selectors that provide a performance benefit with respect to minimising the number of overloads

Fig. 8.2 Performance of the direct and EPM-based selectors for the IEEE 14-bus system, with respect to the number of overloads and the amount of curtailment

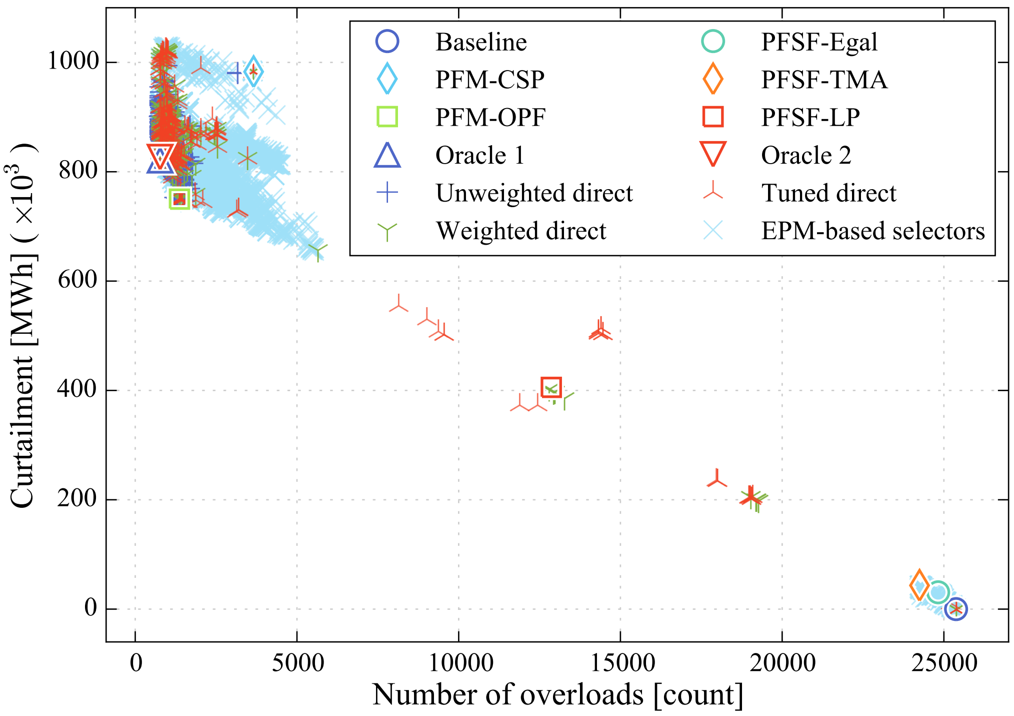


(a) View showing all selectors, power flow management algorithms and oracles

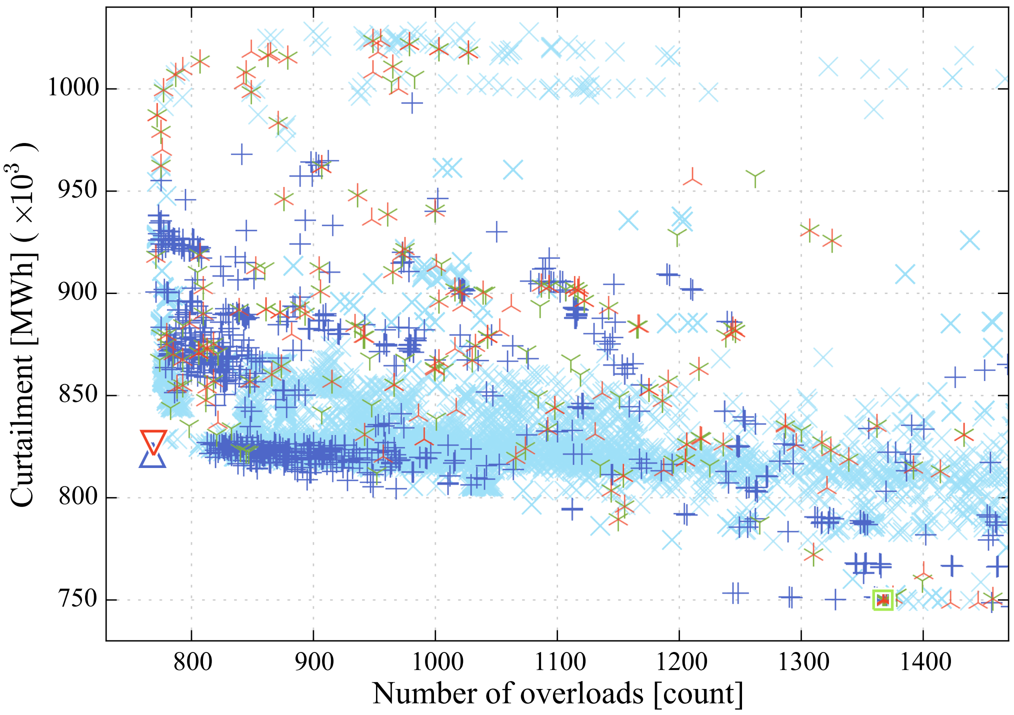


(b) Detailed view, including selectors that provide a performance benefit with respect to minimising overload energy

Fig. 8.3 Performance of the direct and EPM-based selectors for the IEEE 14-bus system, with respect to overload energy and the amount of curtailment

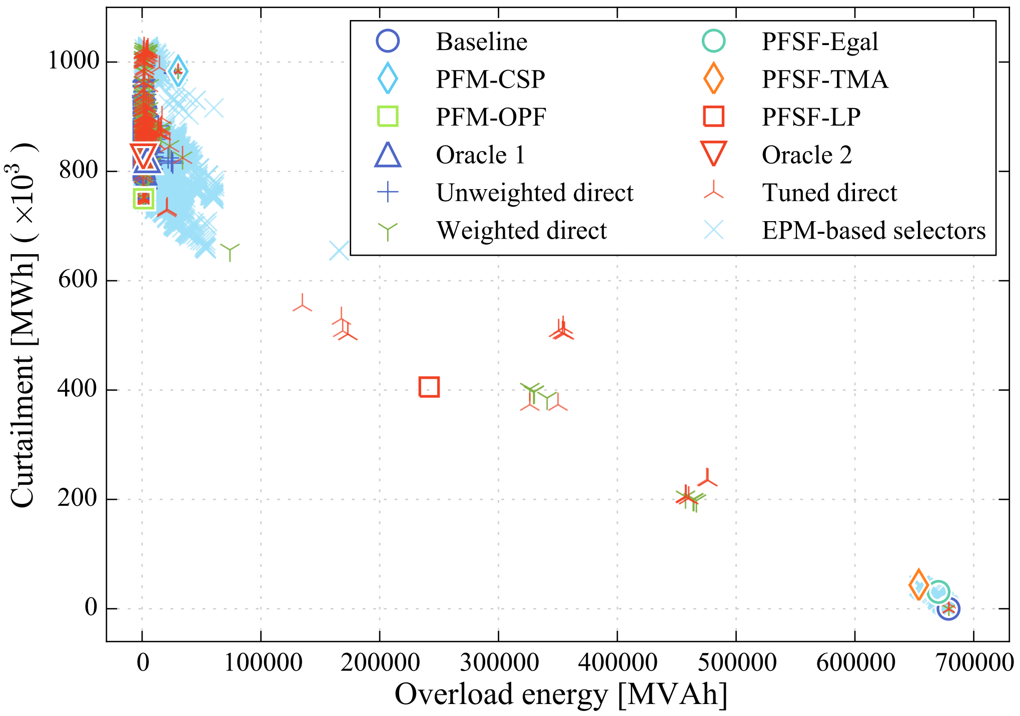


(a) View showing all selectors, power flow management algorithms and oracles

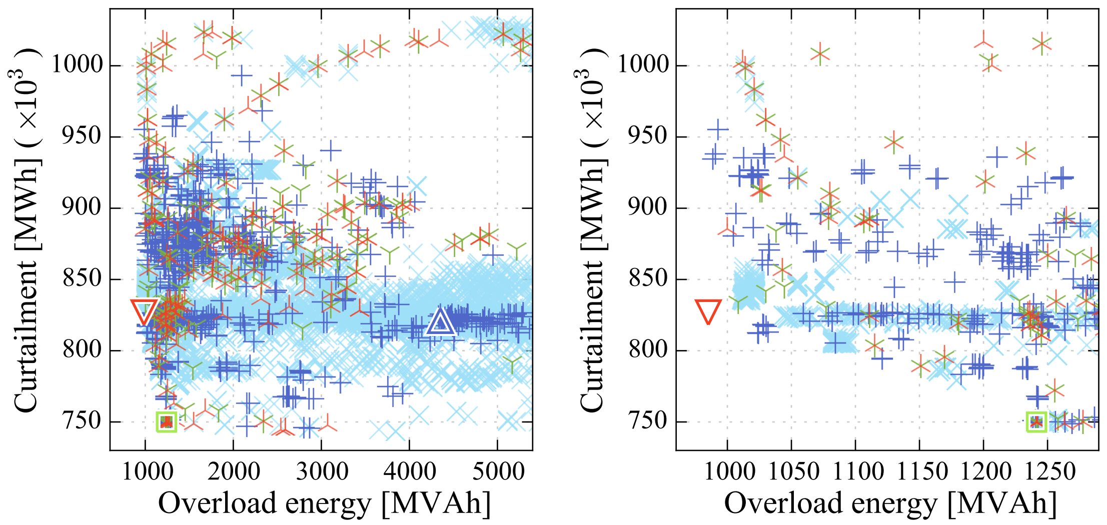


(b) Detailed view, including selectors that provide a performance benefit with respect to minimising the number of overloads, and thus approach the performance of the oracles

Fig. 8.4 Performance of the direct and EPM-based selectors for the IEEE 57-bus system, with respect to the number of overloads and the amount of curtailment



(a) View showing all selectors, power flow management algorithms and oracles



(b) Detailed view, including the selectors, both oracles and PFM-OPF

(c) Detailed view, including the selectors, oracle 2 and PFM-OPF

Fig. 8.5 Performance of the direct and EPM-based selectors for the IEEE 57-bus system, with respect to the overload energy and the amount of curtailment

for each design variant. This shows the extent of performance benefit that can be achieved from algorithm selectors using the designs considered in this work.

Table 8.2 lists the performance of the most effective selectors produced by each of the three direct selector design variants and for the EPM-based selectors. The table is subdivided by system, and by the overload objective considered when determining which selector is most effective. Each of the selectors shown is the most effective for that design variant with respect to minimising the stated overload objective whilst also minimising curtailment. The configurations of the selectors listed in Table 8.2 are described in Table 8.3 for the direct selectors, and Table 8.4 for the EPM-based selectors. The configuration tables include the model types, objectives sets, and algorithm sets used to create the selectors.

For the 33 kV meshed distribution system, the most effective selectors produced using all three of the selector designs considered (selectors (1) to (3)) are able to provide a performance benefit compared to PFM-OPF, in terms of minimising the number or energy of overloads, whilst minimising curtailment. Comparing the differences in curtailment performance for each of the selectors with PFM-OPF using paired t-tests yields $p < 0.001$. After applying the Benjamini-Hochberg procedure, these p-values are found to represent statistically significant differences in performance (note that for this system the p-value at the threshold between significant and non-significant results was found to be 0.046984). However, the tuned direct selector gives the larger performance benefit, and is able to close 51.04% of the curtailment performance gap between PFM-OPF and the oracles, although the remaining performance gap is still statistically significant ($p < 0.001$).

For the IEEE 14-bus system, the most effective selectors for all of the direct selector design variants are able to remove all overloads, thus matching the performance of both oracles with respect to the overload performance measures. The most effective EPM-based selector leaves a single overload, although comparing its performance for the number (or the energy) of overloads to either of the oracles using the paired t-test yields $p = 0.3173$. This does not represent a statistically significant difference in performance (the threshold p-value for this system after applying the Benjamini-Hochberg procedure was found to be 0.040743). All of the most effective selectors apply more curtailment than PFM-OPF, PFSF-TMA, and the two oracles.

For the IEEE 57-bus system, the most effective direct selectors with respect to minimising the number of overloads remove the same number of overloads (771), and close 99.50% of the performance gap between PFM-OPF and oracle 1. The most effective EPM-based selector leaves only one additional overload compared with the most effective direct selectors. Comparing the performance of each of the selectors and oracle 1 with respect to the number of overloads using paired t-test yields p-values > 0.083, so the differences are not statistically

Table 8.2 Performance of the most effective power flow management algorithms, selectors (direct and EPM-based) and oracles for each of the case study systems

System	Overload objective	Algorithm, selector or oracle	Number of overloads [count]	Overload energy [MVAh]	Curtailment [MWh]
33 kV meshed	Number or energy	PFM-OPF	0	0.00	40246.45
		Selector (1) – unweighted	0	0.00	40243.50
		Selector (2) – tuned	0	0.00	40058.93
		Selector (3) – EPM-based	0	0.00	40114.12
		Oracle 1 & 2	0	0.00	39879.06
IEEE 14-bus	Number	PFM-OPF	24	49.97	19544.04
		Selector (4) – unweighted	0	0.00	23775.17
		Selector (5) – weighted	0	0.00	24880.62
		Selector (6) – tuned	0	0.00	24880.62
		Selector (7) – EPM-based	1	< 0.01	42003.45
		Oracle 1	0	0.00	19311.40
	Energy	PFSF-TMA	60	1.57	21797.45
		Selector (4) – unweighted	0	0.00	23775.17
		Selector (5) – weighted	0	0.00	24880.62
		Selector (6) – tuned	0	0.00	24880.62
		Selector (7) – EPM-based	1	< 0.01	42003.45
		Oracle 2	0	0.00	19311.40
IEEE 57-bus	Number	PFM-OPF	1367	1241.44	749899.56
		Selector (8) – unweighted	771	1557.06	900733.95
		Selector (9) – weighted	771	1920.04	917846.93
		Selector (10) – tuned	771	1920.04	917846.93
		Selector (11) – EPM-based	772	2432.60	926645.90
		Oracle 1	768	4352.19	821087.30
	Energy	PFM-OPF	1367	1241.44	749899.56
		Selector (12) – unweighted	774	989.01	934569.31
		Selector (13) – weighted	833	1008.38	834089.75
		Selector (14) – tuned	799	1000.24	885244.98
		Selector (15) – EPM-based	842	1011.69	836116.32
		Oracle 2	769	985.14	826709.24

Table 8.3 Configurations of the most effective direct selectors

System	Ref	Variant	Model	Objectives	Algorithm set
33 kV meshed	(1)	Unw.	`LMT`	OL + CT	PFM-CSP, PFM-OPF
	(2)	Tuned	`MLP`	OL + CT	PFM-OPF, PFSF-TMA
IEEE 14-bus	(4)	Unw.	`AdaBoostM1` (`REPTree`)	OL	PFM-CSP, PFM-OPF
	(5)	Wei.	`FT`	OE	PFM-CSP, PFM-OPF
	(6)	Tuned	`FT`	OE + CT	PFM-CSP, PFM-OPF
IEEE 57-bus	(8)	Unw.	`MLP`	OL	All algorithms
	(9)	Wei.	`MLP`	OL	PFM-CSP, PFM-OPF
	(10)	Tuned	`MLP`	OL + CT	PFM-CSP, PFM-OPF
	(12)	Unw.	`MLP`	OL + OE	All algorithms
	(13)	Wei.	`IBk` (10 NN)	OE	PFM-CSP, -OPF, PFSF-Egal
	(14)	Tuned	`AdaBoost` (`REPTree`)	OE + CT	PFM-CSP, PFM-OPF

Table 8.4 Configurations of the most effective EPM-based selectors

System	Ref.	Primary EPM	Secondary EPM	Algorithm set
33 kV meshed	(3)	OL/OE: `IBk` (10 NN)	CT: `IBk` (1 NN)	PFM-OPF, PFSF-LP
IEEE 14-bus	(7)	OL: `RBFRegressor`	–	PFM-CSP, PFSF-Egal
IEEE 57-bus	(11)	OL: `RandomForest`	CT: `IBk` (1 NN)	All algorithms
	(15)	OE: `IBk` (5 NN)	–	All algorithms

significant (the threshold p-value for this system was found to be 0.048739). Note that selectors (9) and (10) are identical, as the tuned selector (10) has $c_2 = 0$.

The performance of the selectors that are most effective at minimising overload energy for the IEEE 57-bus system has more variation than for the selectors that are most effective at minimising the number of overloads. The unweighted selector (12) gives the best performance for overload energy, and closes 98.49% of the performance gap between PFM-OPF and oracle 2. Although this is close to the performance of the oracle, a paired t-test comparing the selector performance to that of oracle 2 yields a p-value < 0.001, so the performance difference is statistically significant. The tuned selector (14) performs slightly worse than the unweighted selector with respect to overload energy, followed by the weighted selector (13) and the EPM-based selector (15). Each of the selectors also removes more overloads than PFM-OPF, although with more curtailment applied.

Overall, the most effective direct selectors give better performance than the EPM-based selectors, with a tuned direct selector being the most effective for the 33 kV meshed distribution system, and unweighted direct selectors being the most effective with respect to both overload objectives on the other two case study systems. However, the performance differences between the most effective direct and EPM-based selectors are minimal in some

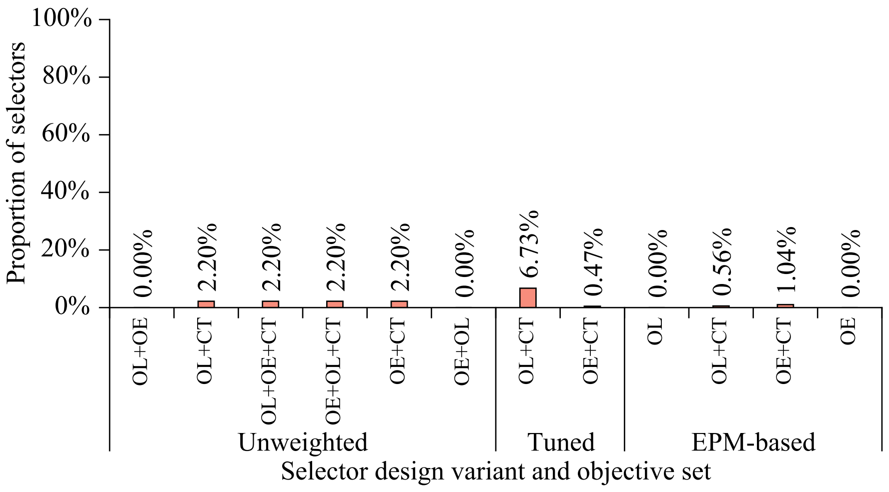


Fig. 8.6 Comparison of the proportion of effective selectors (for either overload objective) produced per objective set for the 33 kV meshed distribution system

cases. For example, for the IEEE 14-bus system, there is no statistically significant difference between the oracles and both the most effective direct and EPM-based selectors, with respect to minimising the number or energy of overloads. Furthermore, for the IEEE 57-bus system the most effective direct and EPM-based selectors differed by a single overload.

8.4 Relationship between objectives and performance

For all the selector design variants, different sets of objectives were considered when creating selectors. This section examines if the objectives considered by the selectors is reflected in their performance. This is achieved by comparing the proportion of effective selectors produced by each objective set.

8.4.1 33 kV meshed distribution system

Figure 8.6 shows the proportion of "effective" selectors produced by each objective set considered for the 33 kV meshed distribution system. No (untuned) weighted selectors were created for this system, so only results for the unweighted and tuned direct selectors are shown, along with results for the EPM-based selectors. For this system, "effective" selectors are those that remove all overloads and do so with statistically significant less curtailment than PFM-OPF.

As could be expected, only the selectors that have objective sets that consider minimising curtailment in addition to minimising overloads (number or energy) produce effective

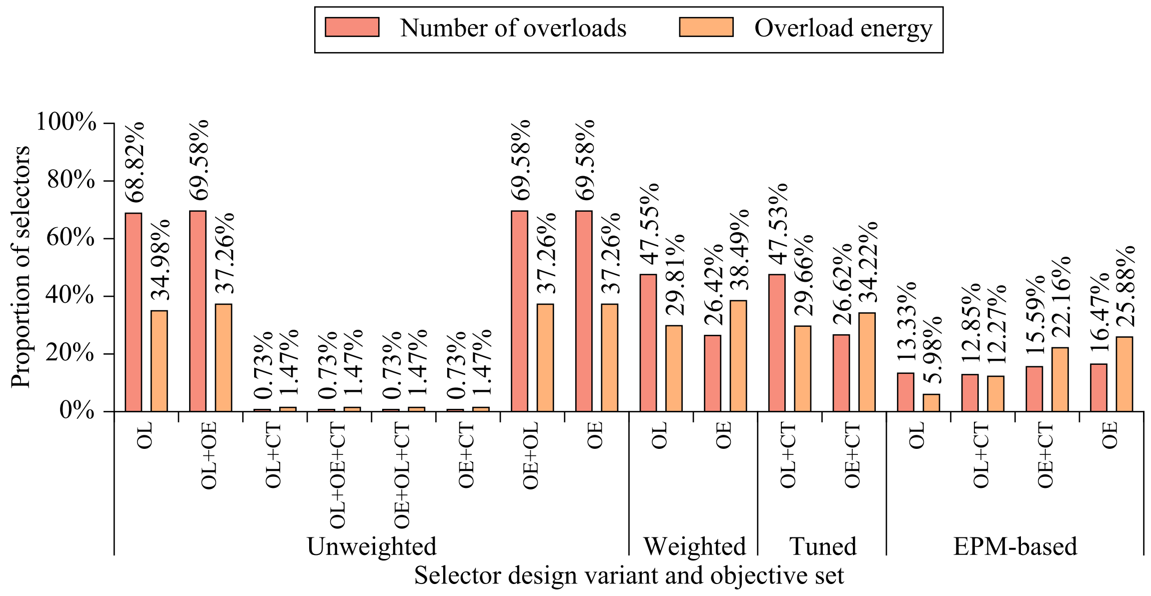


Fig. 8.7 Comparison of the proportion of effective selectors (for each of the overload objectives) produced per objective set for the IEEE 14-bus system

selectors for this system. For the unweighted selectors, the choice of overload objective considered does not affect the likelihood of producing an effective selector; however, for the tuned selectors, selectors that consider the number of overloads are more likely to be effective, whereas for the EPM-based selectors, those that consider overload energy (in the primary EPMs) are more likely to be effective.

8.4.2 IEEE 14-bus system

Figure 8.7 is similar to Figure 8.6, but examines the performance of all three direct selector design variants and the EPM-based selectors created for the IEEE 14-bus system. For this system, the "effective" selectors are those that either: leave statistically significantly fewer overloads in comparison to PFM-OPF, or lead to statistically significantly less overload energy in comparison to PFSF-TMA. The results presented in the figure are split by which overload objective is used to determine the effectiveness of the selectors. For example, the first pair of bars represent all the unweighted selectors created using minimisation of the number of overloads as the sole objective during training. The first bar of the pair represents the proportion of those selectors that were effective, if minimising the number of overloads was considered as the first objective for determining effectiveness; whereas the second bar represents the proportion of those selectors that were effective if minimising the overload energy was considered first.

For the unweighted selectors shown in Figure 8.7, when curtailment is considered in the objective set, then the proportion of effective selectors drops considerably. This is potentially due to the training sets for these selectors lacking differentiation between training examples where an algorithm should be selected for overload performance, and those where the difference is only for curtailment. For the remaining objective sets that contain only overload objectives, these have substantially more selectors that are effective. These objective sets typically have twice the number of selectors that are effective with respect to the number of overloads, than those that are effective with respect to overload energy, regardless of which overload objective is considered first. When overload energy is considered in the objective set, the proportion of selectors that are effective for that objective does increase, but there is still a bias towards minimising the number of overloads.

The weighted and tuned selectors both consider only two sets of objectives, and produce similar results. The objective sets that consider the number of overloads lead to a greater proportion of selectors that are effective against that performance measure, rather than being effective at minimising overload energy. Conversely, the objective sets that consider overload energy lead to greater proportions of selectors that are effective at minimising the energy, rather than the number, of overloads. This is contrary to the trends observed for the unweighted selectors – that minimising the number of overloads consistently outweighs minimising overload energy for the selectors that only consider overload objectives – and suggests that the performance of the weighted selectors is more strongly dependent on the overload objective considered when creating the selectors.

For the EPM-based selectors, which overload objective is considered by the selectors is reflected in the proportion that are effective for each overload objective, similar to the weighted and tuned selectors. When the selectors consider minimising the number of overloads (in the primary EPMs), a greater proportion of selectors are effective for the objective rather than for minimising overload energy. When the selectors consider minimising the overload energy, then the inverse is true, as well as the proportion of effective selectors increasing for both overload objectives.

8.4.3 IEEE 57-bus system

Figure 8.8 is similar to Figures 8.6 and 8.7, but shows results for all selector design variants created for the IEEE 57-bus system. The "effective" selectors for this system are those that can reduce either the number or energy of overloads compared with PFM-OPF. For the unweighted selectors similar trends are observed to those seen for the IEEE 14-bus system. Firstly, there is a reduction in the proportion of effective selectors when curtailment is considered in the objective set, although the reduction is less dramatic than that seen

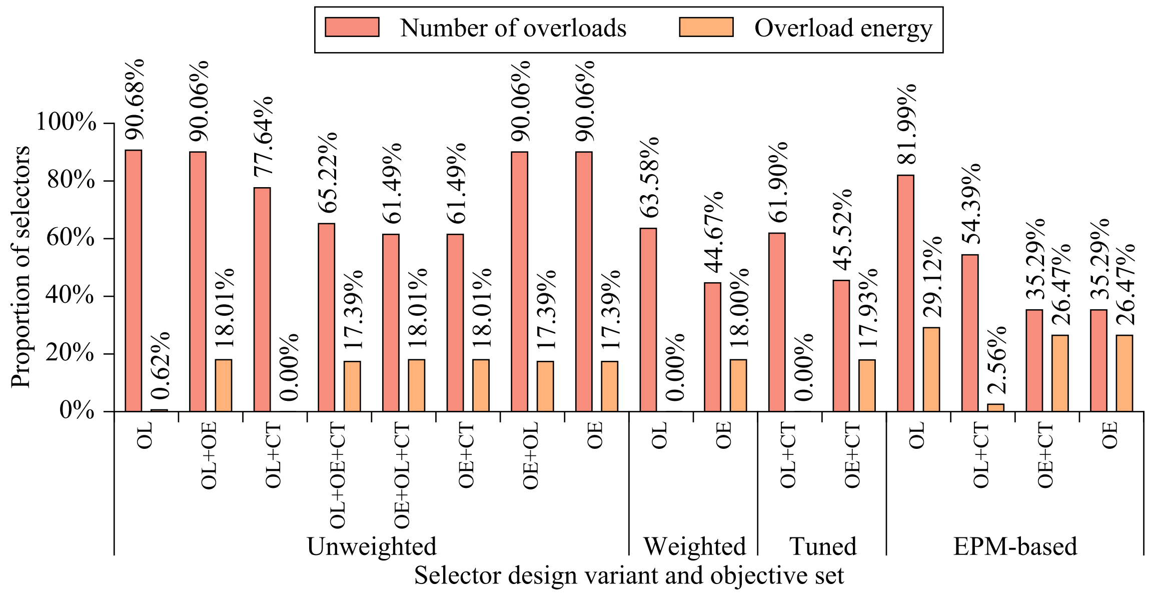


Fig. 8.8 Comparison of the proportion of effective selectors (for each of the overload objectives) produced per objective set for the IEEE 57-bus system

for the IEEE 14-bus system. Secondly, there is a bias towards minimising the number of overloads, regardless of what overload objectives are considered. Despite this, a dependence is observed between objective sets considering minimising overload energy and an increase in the proportion of selectors that are effective for that objective. For instance, only 0.62% of selectors that consider just the number overloads are effective with respect to minimising overload energy; however, at least 17.39% of selectors created for each objective set that considers overload energy are effective with respect to minimising overload energy.

For the weighted, tuned, and EPM-based selectors, there is a dependence between each overload objective and the proportion of selectors that are effective against that objective, similar to what was observed for the IEEE 14-bus system. This is particularly stark for the weighted and tuned selectors, as none of the selectors that consider minimising the number of overloads are effective for minimising the overload energy, while there are selectors that consider minimising overload energy that are effective against that objective. However, for all the selector design variants there is a bias towards producing more selectors that are effective at minimising the number, rather than the energy, of overloads.

For the EPM-based selectors, the effect of considering curtailment – that is, using secondary EPMs – depends on what overload objective is considered by the selector. For the selectors that consider minimising the number of overloads, also considering curtailment leads to a drop in the proportion of effective selectors, which is particularly severe for overload energy performance. However, when the selectors consider overload energy, the proportion of effective selectors is the same regardless of whether curtailment is also considered, suggesting that the secondary EPMs have no effect for those selectors.

8.4.4 Summary

These results have shown that the performance of the selectors does depend on the objectives considered when creating the selectors. For the 33 kV meshed distribution system, where effective performance required the minimisation of overloads, only the selectors that considered minimising curtailment were effective. For the IEEE 14- and 57-bus systems, selectors that considered minimising the number of overloads were more likely to be effective for that objective, whereas if the selectors considered minimising overload energy then there was a relative increase in the proportion of selector effective for that objective. However, there was a general bias towards selectors that were effective at minimising the number of overloads.

8.5 Effect of considering different algorithm sets

For each of the case study systems, selectors have been created that consider different sets of power flow management algorithms, with at least two different set sizes considered for each system. As explained in Section 5.4, considering a larger set of algorithms for selection compared with using a smaller set – where the smaller set is an exact sub-set of the larger set – will yield either the *same* or *improved* performance, if the optimal selection decisions are made. However, for machine learning-based selectors, the performance may *decline* if larger sets of algorithms are considered, as the larger sets may make the machine learning task more difficult and thus reduce the accuracy of the selectors produced.

8.5.1 Method

To investigate whether considering a larger set of algorithms leads to differences in selector performance, the performance of selectors with different algorithm sets (which were otherwise identical) were compared. Table 8.5 tabulates the results of these comparisons, which are subdivided according to: the system, the algorithm sets compared (typically several smaller sets were compared against a larger set containing all algorithms), the overload objective considered when determining which selectors are the most effective, and the selector design variant. The number of "configurations" is stated for each subdivision, which is the number of selector designs that were identical (such as using the same model type and objective set) apart from the algorithm sets considered. Some configurations have been omitted where selectors could not be created for one or more of the algorithm sets.

Table 8.5 Analysis of decline in selector performance when a larger set of power flow management algorithms is considered

System	Small algorithm sets	Large algorithm set	Overload objective	Selector design variant	No. of configurations	Configurations with performance decline from using large set	
33 kV meshed	Pairs	All	Number	Unw.	316	176	(55.70%)
				EPM	4235	3392	(80.09%)
			Energy	Unw.	316	173	(54.75%)
				EPM	4235	3392	(80.09%)
	Pairs	Trio	Number	Tuned	105	40	(38.10%)
			Energy	Tuned	105	39	(37.14%)
	Pairs & Trio	All	Number	Tuned	95	66	(69.47%)
			Energy	Tuned	95	66	(69.47%)
IEEE 14-bus	Pairs	All	Number	Unw.	384	380	(98.96%)
				Wei.	97	96	(98.97%)
				Tuned	93	93	(100.00%)
				EPM	4235	3988	(94.17%)
			Energy	Unw.	384	376	(97.92%)
				Wei.	97	96	(98.97%)
				Tuned	93	92	(98.92%)
				EPM	4235	4087	(96.51%)
IEEE 57-bus	Pair	Trio	Number	Unw.	424	200	(47.17%)
				Wei.	100	70	(70.00%)
				Tuned	96	71	(73.96%)
				EPM	4235	3367	(79.50%)
			Energy	Unw.	424	165	(38.92%)
				Wei.	100	59	(59.00%)
				Tuned	96	67	(69.79%)
				EPM	4235	3148	(74.33%)
	Pair & Trio	All	Number	Unw.	424	332	(78.30%)
				Wei.	97	85	(87.63%)
				Tuned	92	81	(88.04%)
				EPM	4235	3507	(82.81%)
			Energy	Unw.	424	323	(76.18%)
				Wei.	97	74	(76.29%)
				Tuned	92	75	(81.52%)
				EPM	4235	3405	(80.40%)

The last columns in Table 8.5 state the result of the comparison of the selectors that use different algorithm sets. To derive the values shown, the selectors from each configuration that considered the smaller sets of algorithms were compared, and the most effective of these determined, in terms of minimising the overload objective stated in the table, whilst also minimising curtailment. For each configuration, the most effective selector (from the selectors that considered the smaller algorithm sets) was then compared to the selector that considered the larger set, and it was recorded whether the selector with the larger set led to a decline in performance.

What constituted a "decline" in performance when going from the smaller to the larger algorithm sets depended on the overload objective considered. If the number of overloads was considered, the decline in performance was either: an increase in the number of overloads, or an increase in the overload energy whilst the number of overloads stayed constant, or increased curtailment with no change in overloads. Alternatively, if overload energy was considered, the decline in performance was either: an increase in the overload energy, or an increase in the number of overloads whilst the overload energy stayed constant, or increased curtailment with no change in overloads.

For example, the first row in Table 8.5 (33 kV meshed $\rightarrow$ Pairs $\rightarrow$ All $\rightarrow$ Number $\rightarrow$ Unw.) concerns the unweighted direct selectors created for the 33 kV meshed distribution system, and whether otherwise identical selectors (the "configurations') created for that system experience a decline in performance if a larger set of algorithms is considered by the selector during its creation. For this row, the decline in performance considers the number of overloads as the primary objective. Here, the smaller sets of algorithms are two pairs: PFM-OPF and PFSF-LP, or PFM-OPF and PFSF-TMA; while the larger set is the set of all power flow management algorithms. The table shows that out of the 316 unweighted direct selector configurations, 176 (55.70%) see a decline in performance when the selectors are created considering the set of all algorithms, rather than either one of the pairs.

Different algorithm sets are considered by the selectors for each system, with not all the same sets used by all the selector designs. The algorithm sets considered by each selector design variant are detailed in the relevant sections in Chapter 7: Section 7.1.1 for unweighted direct selectors, Section 7.1.2 for (untuned) weighted direct selectors, Section 7.1.3 for tuned weighted direct selectors, and Section 7.2 for EPM-based selectors.

8.5.2 Summary of results

Table 8.5 shows that most of the selector configurations are more likely to see a decline in performance if a larger set of algorithms is considered when creating the selectors. This is particularly apparent for the IEEE 14-bus system, where almost all of the configurations see a performance decline when the set of all algorithms is considered for selection rather than just a pair of algorithms.

For the IEEE 57-bus system, the proportion of configurations that see a performance decline is less when the larger algorithm set is the trio, compared with when the larger set is the set of all algorithms. For example, 47.17% of the unweighted selector combinations see a performance decline (when the number of overloads is considered) when the selectors are created using the trio, rather than the pair of algorithms. However, a larger proportion of the same combinations (78.30%) see a performance decline when the larger set of all algorithms is used to create the selectors. The same trend is seen for the other selector design variants on the IEEE 57-bus system, and for the tuned selectors on the 33 kV meshed distribution system, reinforcing the observation that the machine learning-based algorithm selectors tend to perform worse when larger sets of algorithms are considered.

Although considering larger sets of algorithms often leads to a decline in selector performance, that does not necessarily mean that effective selectors cannot be created that consider larger algorithm sets. For example, the most effective unweighted direct and EPM-based selectors for the IEEE 57-bus system (selectors (8) and (12) in Table 8.3, and selectors (11) and (15) in Table 8.4, respectively) consider the full set of algorithms.

8.6 Effect of different model types

Several machine learning model types were used to create the selectors, as described in Section 7.3.1. This section examines the performance of the selectors produced by each model type, in order to ascertain which model types lead to effective selectors. Direct and EPM-based selectors are treated in separate sections – Sections 8.6.1 and 8.6.2, respectively – as how the models are used is different for each selector design.

8.6.1 Effect of model type on direct selectors

Figure 8.9 is a matrix summarising the performance of the selectors created using each model type (listed in the columns) subdivided (in the rows) according to the system and selector design variant used for creating the selectors, and the overload objective used when comparing performance. Each cell in the matrix represents the selector created for a particular system, model type and selector design variant, which is most effective at minimising either the number or energy of overloads whilst also minimising curtailment. Each selector is then compared with the most effective power flow management algorithm for the objective considered, which is PFM-OPF for all comparisons except for minimising overload energy

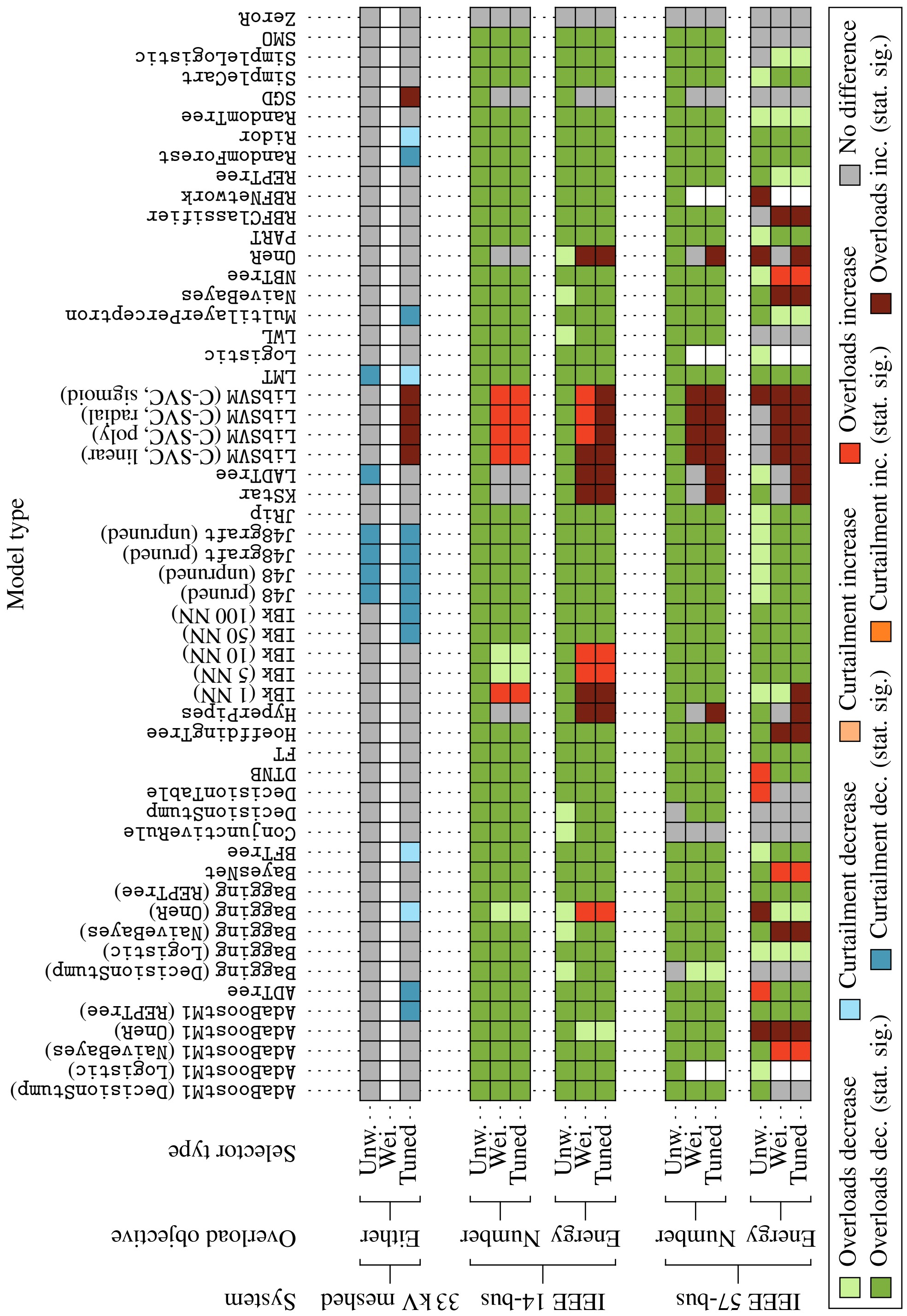


Fig. 8.9 Matrix comparing the performance of the most effective selectors created by each machine learning model type for each system and overload objective

for the IEEE 14-bus system, where PFSF-TMA is used for the comparisons as it is the most effective algorithm for that objective.

The colours of the cells are used to indicate the results of the comparison between the most effective selector and the most effective algorithm. Dark green (🟩) indicates that the selector gives a performance benefit in terms of reducing the number or energy of overloads in comparison to the most effective algorithm, and that the difference is statistically significant. Light green (🟩) similarly represents a reduction in the number or energy of overloads, but that the difference to the most effective algorithm is not statistically significant. Dark blue (🟦) indicates that a selector provides a performance benefit by applying less curtailment than the most effective algorithm whilst giving the same overload performance as the most effective selector, and with the difference in the amount of curtailment applied being statistically significant. Light blue (🟦) is similar, but indicates combinations where the reduction in curtailment is not statistically significant. Grey (⬜) indicates a selector that performs the same as the most effective algorithm, so therefore does not provide any performance benefit. Dark orange (🟧) indicates a selector that applies more curtailment than the most effective algorithm, but that gives the same overload performance, with the difference in the curtailment applied being statistically significant. Light orange (🟧) is similar, but where the difference is not statistically significant. Dark red (🟫) indicates selectors that result in an increase in the number or energy of overloads, where the difference to the most effective algorithm is statistically significant. Light red (🟥) indicates selectors that incur more overloads or overload energy, but where the difference to the most effective algorithm is not statistically significant. A blank cell indicates that no selectors were created for a system using a particular model and selector design variant.

As an example of how to interpret the matrix, consider the column for the model type `LADTree`. The cell in the first row of that column represents the most effective unweighted selector created using `LADTree` for the 33 kV meshed distribution system, with respect to minimising the number of overloads whilst also minimising curtailment. This selector has been compared to PFM-OPF, and, as indicated by the dark blue colour (🟦), the selector is able to produce a statistically significant reduction in curtailment, whilst achieving the same overload performance as PFM-OPF. The third cell down also represents a most effective selector for the 33 kV meshed distribution system, with respect to minimising the number of overloads whilst minimising curtailment, but the selector in this case – a tuned selector – just provides the same performance as PFM-OPF, indicated by the grey colour (⬜). Moving down to the tenth row, the cell there represents the most effective unweighted selector created using `LADTree` for the IEEE 14-bus system, with respect to minimising the overload energy whilst also minimising curtailment. This selector has been compared to PFSF-TMA, and, as indicated by the dark green colour (🟩), the selector is able to reduce the overload energy compared to PFSF-TMA by a statistically significant amount. The next cell down uses the same objectives, but represents the most effective weighted selector created using `LADTree` for the IEEE 14-bus system. This particular selector, as indicated by the dark red colour (🟥), results in statistically significantly more overload energy remaining than PFSF-TMA.

For the 33 kV meshed distribution system, only unweighted and tuned selectors were created, as two objectives needed to be considered but (untuned) weighted selectors were limited to consider one objective only (as described in Section 7.1.2). The majority of the model types produce most effective unweighted selectors that just select PFM-OPF for all states, so therefore give no performance gain for the 33 kV meshed distribution system. There are, however, 6 model types (all of them tree learners) that can offer a performance benefit when used to create unweighted selectors, in terms of reducing the amount of curtailment (by a statistically significant amount) compared to PFM-OPF while still achieving zero overloads. A larger number of model types (10) are able to produce tuned selectors that provide a statistically significant performance benefit.

With the exception of `ZeroR`, all the model types can create unweighted selectors for the IEEE 14-bus system that leave statistically significantly fewer overloads. The same model types also reduce the overload energy left by the most effective unweighted selectors, although for 7 of these the performance difference to PFSF-TMA is not statistically significant. Most of the model types also create weighted and tuned selectors that provide statistically significant performance benefits: 41 with respect to minimising the number of overloads, and 40 with respect to minimising overload energy.

Most model types (51 of the 55) can produce unweighted selectors for the IEEE 57-bus system that are able to reduce the number of overloads by a statistically significant amount compared to PFM-OPF, although there are fewer (20) that can reduce overload energy by a statistically significant amount. Similarly, for the weighted and tuned selectors, fewer model types produce selectors that provide a statistically significant performance benefit with respect to overload energy (20), that for the number of overloads (40).

8.6.2 Effect of model type on EPM-based selectors

The EPM-based selectors developed in this work contain up to two model types, one used for the primary EPMs and the other for the secondary EPMs (sometimes omitted). If selectors are looked at in isolation, it is difficult to determine which of the model types has the most influence on the performance of the selector. However, by comparing the performance of selectors that have the same primary EPMs, or the same secondary EPMs, patterns in performance can be observed that help illustrate which model types lead to better performance. In the following section, the performance of the most effective selectors for each combination of primary and secondary model type is visualised in order to reveal any patterns in performance.

33 kV meshed distribution system

Figure 8.10 is a matrix showing all combinations of model types used for creating EPM-based selectors for the 33 kV meshed distribution system. Each combination comprises a model type used for the primary EPM (either the number or energy of overloads – listed in the columns in the matrix) and those used for the secondary EPMs (for curtailment – listed in the rows), including where no secondary EPM is used (the first row). The performance of the most effective selector produced by each combination – in terms of minimising the number of overloads, whilst also minimising curtailment – is compared with the performance of PFM-OPF, which is the most effective algorithm overall for the 33 kV meshed distribution system and can remove all overloads. This is similar to the process used to create the model type comparison matrix for the direct selectors (Figure 8.9), and the same colours are used to indicate the results of the comparisons. In addition to showing a performance comparison for each combination, for each model type used for primary EPMs, a comparison of the most effective selector drawn from all combinations involving that model type is provided at the end of the columns. Similarly, a summary for each of the model types used for secondary EPMs is provided at the end of the rows.

As an example of how to interpret the matrix, taking the fifth column across (`AdaBoostM1` (`REPTree`)), it can be seen that when that model type is used solely as a primary EPM to predict the number or energy of overloads (first row), the combination is coloured grey (■), as the most effective of the selectors that are produced for this combination gives the same performance as PFM-OPF. Going down to the third row, where `DecisionStump` is also used to predict curtailment performance, the combination is coloured light blue (🟦), as the most effective selector produced does give improved performance compared with PFM-OPF, in terms of reducing the curtailment applied, but this performance difference is not statistically significant. Moving down to near the middle of the rows, where `LibSVM` (nu-SVR, poly) is used to predict curtailment performance, the combination is coloured dark blue (🟦) to indicate that the most effective selector produced for this combination applies less curtailment than PFM-OPF, and that the difference in performance is statistically significant.

The combinations that only consider a single objective (the first row) lead to selectors that either just select PFM-OPF, and thus match its performance, or achieve worse performance than PFM-OPF, such as most of the `LibSVM` variants, `M5P`, and `M5Rules`. When these single-objective selectors are extended to include EPMs to allow minimising curtailment to also be

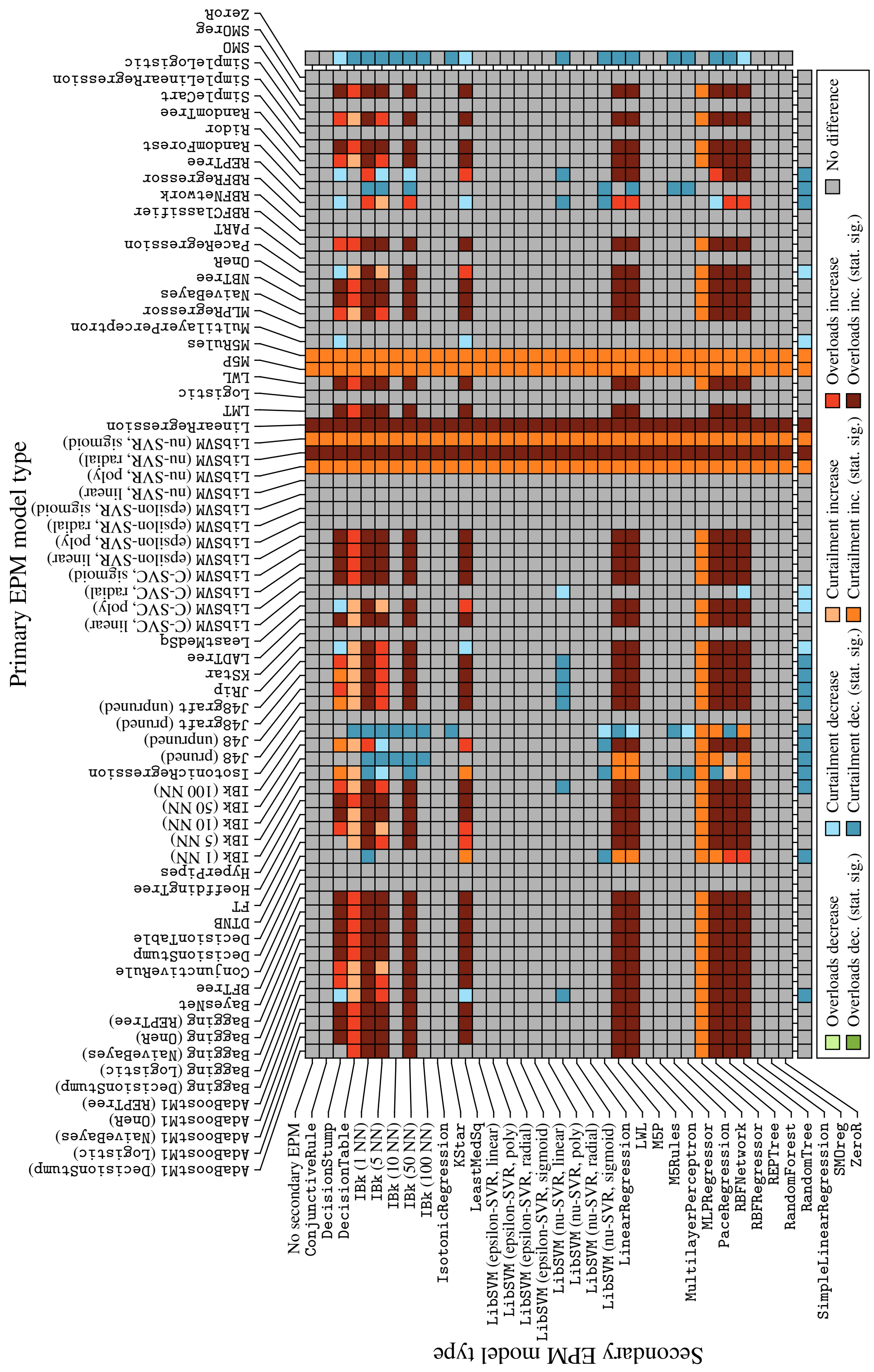


Fig. 8.10 Matrix comparing the performance of the most effective selectors created by each combination of primary and secondary EPM model types to PFM-OPF for the 33 kV meshed distribution system

considered, then the performance of the selectors produced may improve or decline. In some cases, such as for most of the `LibSVM` variants, the performance of the selectors does not change, which suggests that EPMs in those selectors for the first (overload) objectives always produce singleton selection sets, therefore no selection based on curtailment takes place.

The most effective selectors produced by the majority (1709, 68.77%) of the combinations simply always select PFM-OPF, and therefore provide no performance benefit. The next largest group, of 488 combinations (19.64%), produce selectors that incur some overloads, with the difference in performance compared with PFM-OPF being statistically significant for 428 (17.22%) of the combinations. The third largest group, of 227 combinations (9.13%), have most effective selectors that remove all overloads but with more curtailment than PFM-OPF, and for 201 (8.09%) combinations the difference is statistically significant. There are 61 (2.45%) combinations that produce selectors that can improve on the performance of PFM-OPF, by reducing the amount of curtailment applied, with 40 (1.61%) providing a difference that is statistically significant.

Of the 71 model types used for primary EPMs (the rows of the matrix), 14 of these produce selectors that can give a statistically significant performance benefit. These include all the variants of `IBk`, with `IBk` (50 NN) appearing the most frequently, being in 10 combinations that give statistically significant performance benefits. Of the 35 model types used for secondary EPMs (the columns), 15 result in combinations that provide a statistically significant performance benefit. Out of these, `LibSVM` (nu-SVR, poly) appears most frequently (in 8 combinations).

IEEE 14-bus system

Figure 8.11 shows all combinations of model types used for the IEEE 14-bus system. In this matrix, the most effective selector for each combination is that which minimises the number of overloads whilst also minimising curtailment. The performance of the most effective selectors is compared with PFM-OPF, with the outcome indicated by the colours of each combination in the figure (using the same colours as Figure 8.9).

The majority of combinations (1450, 58.35%) lead to selectors that remove more overloads than PFM-OPF, and for 865 (34.81%) combinations this performance improvement is statistically significant. The next largest group, comprising 510 (20.52%) combinations, leave the same number of overloads as PFM-OPF but with more curtailment. For 472 (18.99%) of the combinations the increase in curtailment is statistically significant. 243 (9.78%) of the combinations lead to an increase in the number of overloads, but none of these increases are statistically significant.

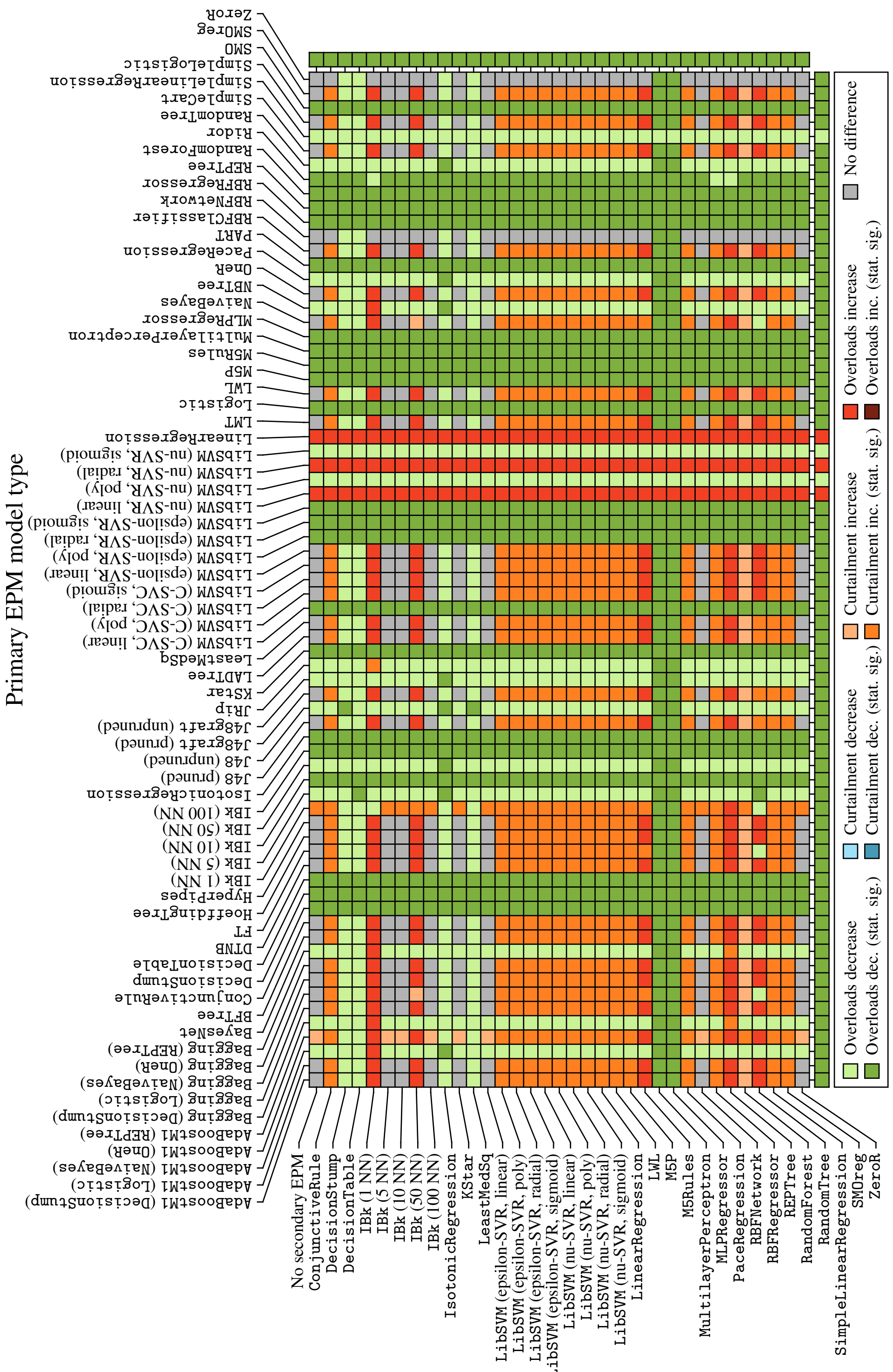


Fig. 8.11 Matrix comparing the performance of the most effective selectors (for minimising the number of overloads whilst minimising curtailment) created by each combination of primary and secondary EPM model types to PFM-OPF for the IEEE 14-bus system

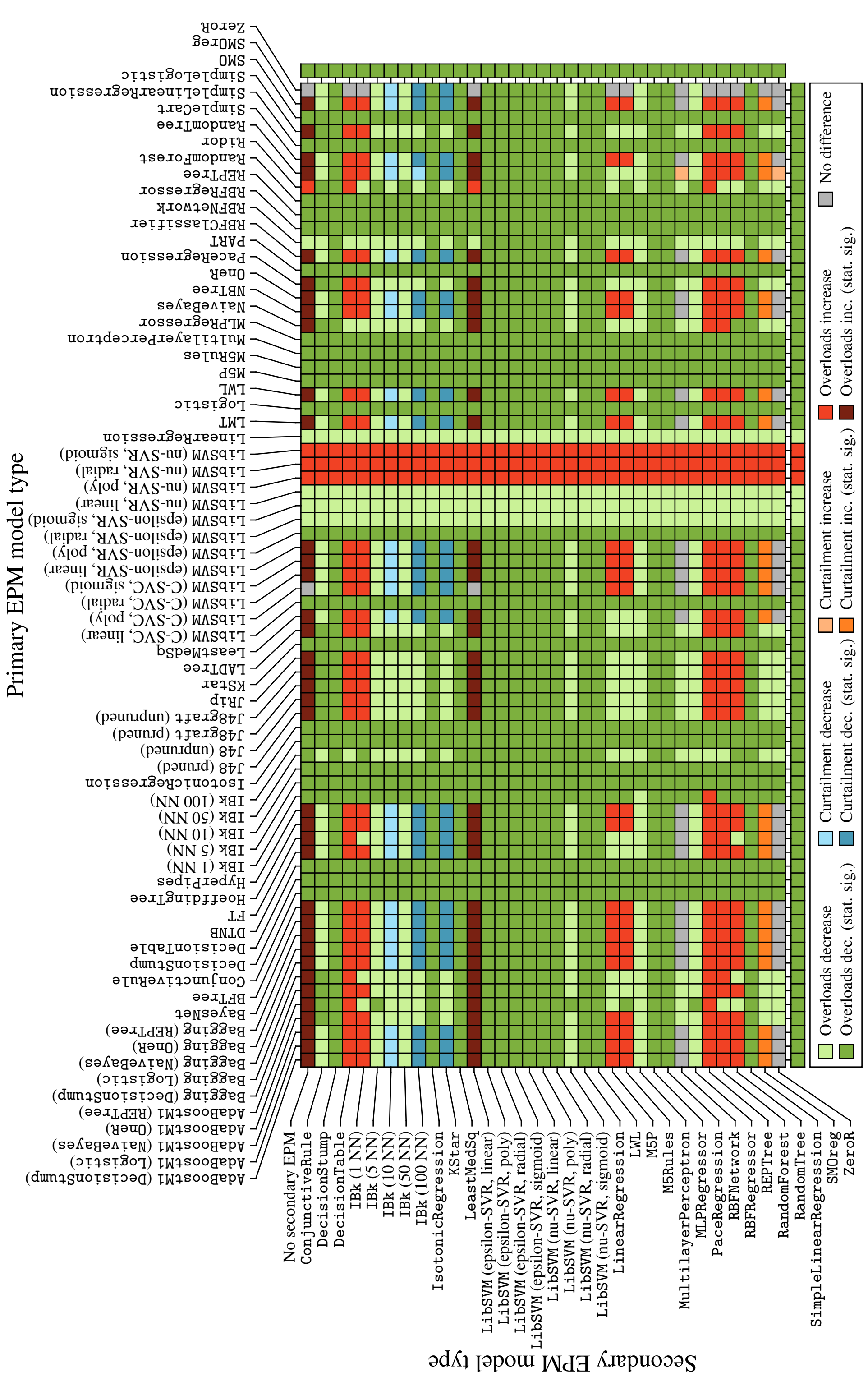


Fig. 8.12 Matrix comparing the performance of the most effective selectors (for minimising overload energy whilst minimising curtailment) created by each combination of primary and secondary EPM model types to PFSF-TMA for the IEEE 14-bus system

The performance of the most effective selectors produced by particular model types used to predict the number of overloads stays constant regardless of what model type is then used to predict the amount of curtailment. For example, when `LibSVM` (epsilon-SVR, linear) is used for the primary EPMs and is then combined with any other model type, the most effective selectors all leave 10 overloads remaining. Because the performance does not change, this suggests that within these selectors, the selection sets derived from the predicted overload performance are singletons, and therefore no further selection based on the predicted curtailment performance takes place. Note that, because the figure indicates the performance of each combination relative to PFM-OPF, rather than the absolute overload and curtailment performance values, it may mask where there are absolute difference in performance between combinations. In other words, the figure shows *if* there is a performance difference to PFM-OPF, but not *how much* a performance difference may be. For example, all the combinations that use `IBk` (100 NN) as the primary EPM consistently remove statistically significantly more overloads than PFM-OPF, however, the number of overloads removed varies: most of the combinations leave 6 overloads, but when used alone or combined with `LeastMedSq`, only 2 overloads are left.

Where the inclusion of secondary EPMs leads to changes in performance, this suggests that the selections using the primary EPM can lead to non-singleton selection sets, so further selection based on the predictions of the secondary EPM can take place. For most of the model types used for the secondary EPMs, there are trends in how the EPM will affect the performance of a combination. For example, for many of the combinations that use a `LibSVM` variant as a secondary EPM, the number of overloads does not change, compared to PFM-OPF, but there is a statistically significant increase the amount of curtailment applied. In some cases – such as for most of the combinations that use `MLPRegressor` or `MultilayerPerceptron` as secondary EPMs – the inclusion of a secondary EPM can lead to improved overload performance, even though the secondary EPM is not used to predict that performance measure. This suggests the primary EPM is not accurately predicting the number or energy of overloads – otherwise, it would not produce non-singleton selection sets containing algorithms with dissimilar overload performance – and that the algorithm that leaves fewer overloads is predicted to have less curtailment by the secondary EPM – either because it *does* have less curtailment, or the EPM is inaccurate and just *predicts* that the curtailment is less.

Figure 8.12 is similar to Figure 8.11, but the most effective selectors for each combination is that which minimises overload energy (rather than the number) whilst minimising curtailment. Furthermore, the performance of the selectors is compared to PFSF-TMA, which is the most effective algorithm when those objectives are considered.

The majority (1910, 76.86%) of combinations can reduce the total overload energy compared with PFSF-TMA, with a statistically significant difference for 1385 (55.73%) of the combinations. This is a larger number of combinations than those that could provide a performance benefit with respect to minimising the number of overloads. The next largest group in Figure 8.12, of 334 (13.44%) combinations, produce increases in overload energy, although there are only 74 (2.98%) combinations where the difference is statistically significant. The most effective selectors produced by the remaining combinations give the same performance as PFSF-TMA with respect to overload energy; 78 (3.14%) apply less curtailment, with the difference being statistically significant for 51 (2.05%) combinations; 62 (2.49%) apply the same amount of curtailment as PFSF-TMA; and 27 (1.09%) apply more curtailment, with the difference being statistically significant for 25 (1.01%) combinations.

IEEE 57-bus system

Figure 8.13 shows all combinations of model types used for the IEEE 57-bus system. The most effective selector for each combination, in terms of minimising the number of overloads whilst also minimising curtailment, is compared to the performance of PFM-OPF. The outcomes of these comparisons are indicated using the same colours as Figure 8.9.

As can be seen in Figure 8.13, the majority of combinations (1554, 62.54%) lead to selectors that can remove more overloads than PFM-OPF, with the performance difference being statistically significant for 1512 (60.85%) combinations. The next largest group, of 861 (34.65%) combinations, lead to a worsening of overload performance, and for 838 (33.72%) combinations the performance difference compared with PFM-OPF is statistically significant. 67 (2.70%) of the combinations give the same performance as PFM-OPF, while 3 (0.12%) lead to a (statistically significant) increase in the amount of curtailment applied, whilst keeping the same overload performance. No combinations have most effective selectors that leave the same number of overloads as PFM-OPF but with less curtailment.

Figure 8.14 compares the performance of the most effective selectors produced by each combination for the IEEE 57-bus system, with respect to minimising overload energy whilst also minimising curtailment, to the performance of PFM-OPF. In contrast to the matrix that considered minimising the number of overloads as the first objective (Figure 8.13), far fewer combinations (550, 22.13%) lead to selectors that provide a performance benefit with respect to minimising overload energy, with the difference to PFM-OPF being statistically significant for 467 (18.79%) combinations. In fact, the majority of combinations (1828, 73.56%) lead to an increase in overload energy, with the difference to PFM-OPF being statistically significant for 1742 (70.10%) combinations. The remaining 107 (4.31%) combinations achieve the same performance as PFM-OPF.

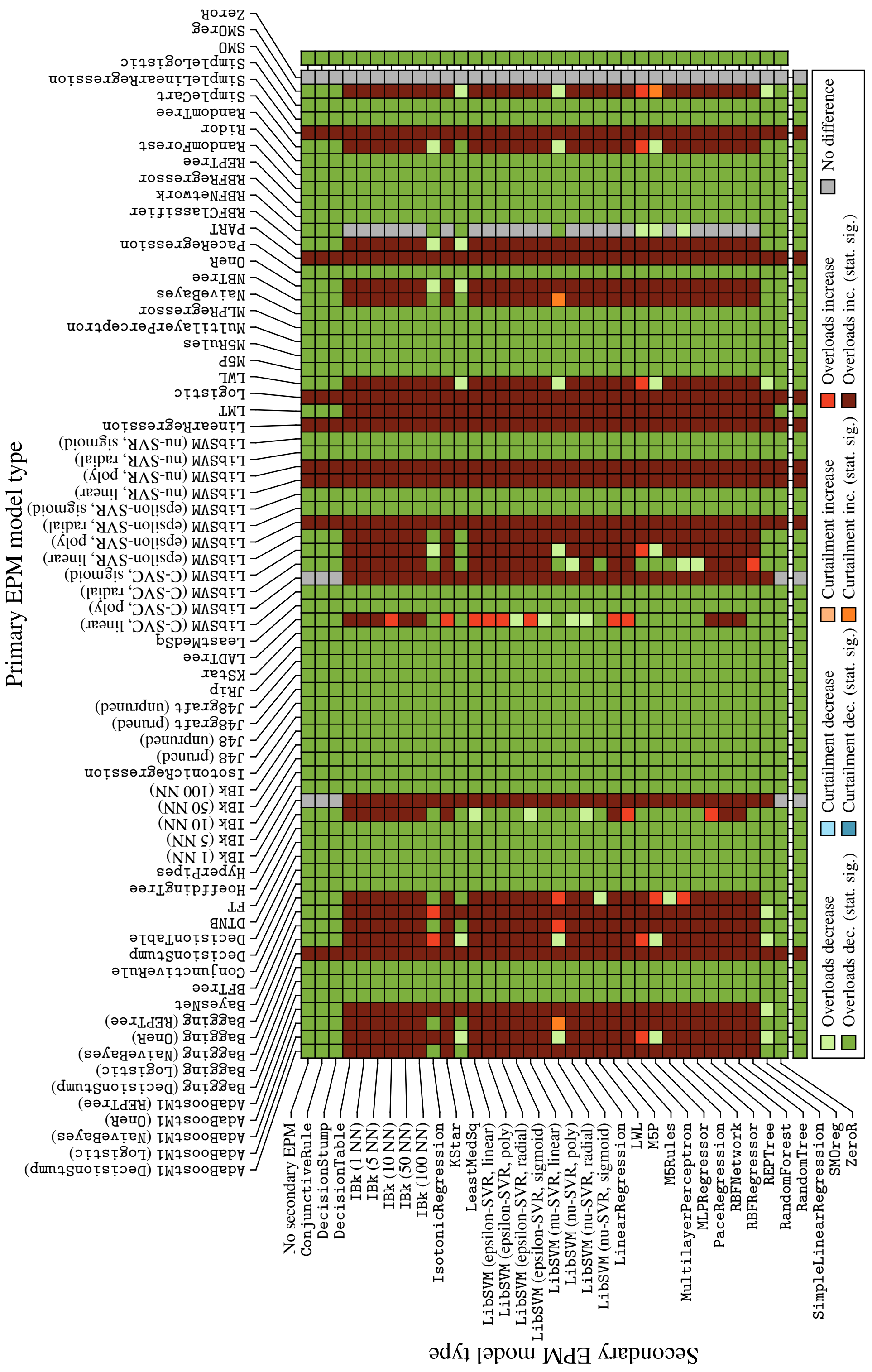


Fig. 8.13 Matrix comparing the performance of the most effective selectors (for minimising the number of overloads whilst minimising curtailment) created by each combination of primary and secondary EPM model types to PFM-OPF for the IEEE 57-bus system

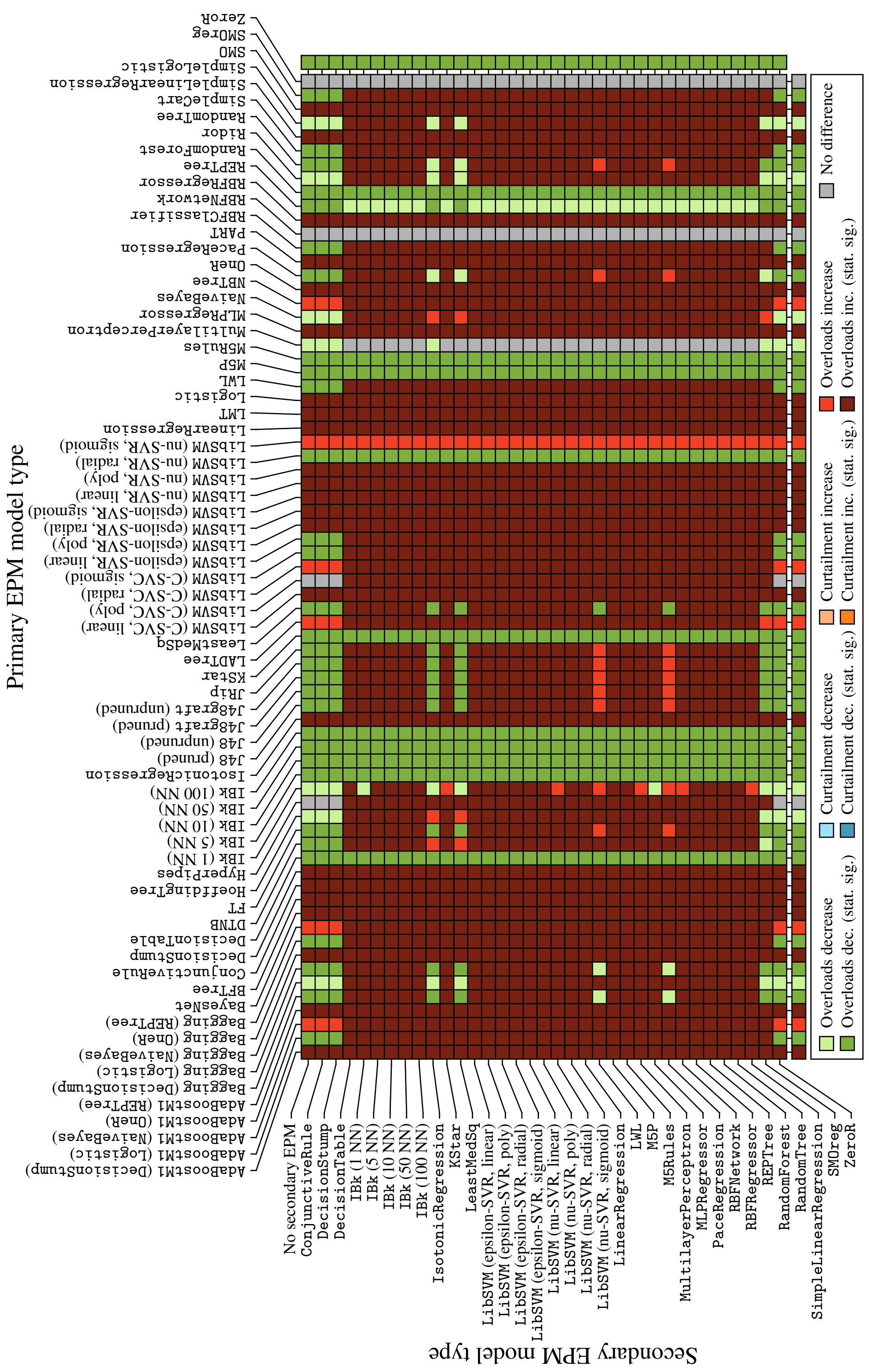


Fig. 8.14 Matrix comparing the performance of the most effective selectors (for minimising overload energy whilst minimising curtailment) created by each combination of primary and secondary EPM model types to PFM-OPF for the IEEE 57-bus system

8.6.3 Summary

The results presented in this section show that many model types are able to produce effective selectors when used within one or more of the selector design variants for one or more case study power systems. However, fewer model types lead to effective selectors in *all* of the systems, and these are listed in Table 8.6. For each of the selector design variants listed, a "Y" indicates that a model type produced selectors of that variant which were effective (for either of the overload objectives) in each of the case study systems. No differentiation is made between the three direct selector variants, but for the EPM-based selectors the table differentiates between where a model type was used for the primary or secondary EPMs.

The model types listed in Table 8.6 are considered the most promising of those tested in this work for creating algorithm selectors for power systems control, as they can produce selectors for each of the systems that provide statistically significant performance benefits. As well as producing effective selectors, some of the model types listed in the table also produce the *most effective* selectors for each of the systems, as all of the model types used in the most effective selectors (Section 8.3) are represented in the table.

It is important to note that the list of model types used in this work is not exhaustive: there are other model types, and other variants of the model types used. Furthermore, no optimisation of the models took place, as default parameters were used for all model types (with a few exceptions, as described in Section 7.3.1), and the same set of features and amount of training data were used for all models created for a particular case study system. It is possible that other model types than those listed in Table 8.6 could produce effective selectors, if different model types, parameters, features, or training set sizes were used.

8.7 Effect of tuning on direct weighted selectors

The results of the previous sections have generally shown that the tuned weighted selectors give broadly similar performance to the (untuned) weighted selectors. Many of the tuned selectors were found to have weight multipliers c_2 that were zero, and were therefore identical to weighted selectors with the same configuration. Therefore it was pertinent to investigate whether tuning could provide performance benefits beyond the performance that could be obtained by an (untuned) weighted selector of the same configuration.

Table 8.7 presents the results of comparing pairs of (untuned) weighted and tuned selectors, that had otherwise identical configurations (such as the model type and algorithm set). The values in the table count the number of selector pairs where the tuned selector leads to a particular difference in performance. For example, the first row shows that for the IEEE 14-bus system, there were 7 tuned selectors that could reduce the number of overloads

Table 8.6 Model types that created effective selectors for every case study system

Model type	Selector design variant / usage			
	Direct	EPM (primary)	EPM (secondary)	All
`AdaBoostM1` (`REPTree`)	Y	Y	–	–
`ADTree`	Y	–	–	–
`DecisionTable`	–	Y	Y	–
`IBk` (1 NN)	–	Y	Y	–
`IBk` (5 NN)	–	Y	Y	–
`IBk` (10 NN)	–	Y	Y	–
`IBk` (50 NN)	Y	Y	Y	Y
`IBk` (100 NN)	Y	Y	Y	Y
`J48` (pruned)	Y	Y	–	–
`J48` (unpruned)	Y	Y	–	–
`J48graft` (pruned)	Y	Y	–	–
`J48graft` (unpruned)	Y	Y	–	–
`KStar`	–	–	Y	–
`LADTree`	Y	–	–	–
`LibSVM` (nu-SVR, poly)	–	–	Y	–
`LinearRegression`	–	–	Y	–
`LMT`	Y	–	–	–
`M5P`	–	–	Y	–
`M5Rules`	–	–	Y	–
`MultilayerPerceptron`	Y	–	–	–
`PaceRegression`	–	–	Y	–
`RBFNetwork`	–	–	Y	–
`REPTree`	–	Y	Y	–
`RandomForest`	Y	Y	Y	Y
`RandomTree`	–	Y	–	–

Table 8.7 Performance differences from applying tuning to (untuned) weighted selectors

System	Overload objective	Overload change	Curtailment change		
			Decrease	Same	Increase
IEEE 14-bus	Number	Decrease	7	0	14
		Same	23	394	8
		Increase	60	0	20
	Energy	Decrease	7	0	17
		Same	20	394	7
		Increase	63	0	18
IEEE 57-bus	Number	Decrease	12	0	19
		Same	3	179	7
		Increase	32	0	39
	Energy	Decrease	15	0	20
		Same	2	179	4
		Increase	30	0	41

and the amount of curtailment compared with equivalent (untuned) weighted selectors; and that there were 14 tuned selectors that could reduce the number of overloads compared with equivalent weighted selectors, but with an increase in curtailment.

Table 8.7 shows that, at least for the IEEE 14- and 57-bus systems, the majority of the tuned selectors just provide the same performance as equivalent weighted selectors. This is not unexpected, however, performance gains come primarily from reducing the number or energy of overloads, whereas the tuning process is focussed on optimising curtailment performance. For both systems, if there is a performance change, it is most likely to be a decline in performance, with an increase in either the number or energy of overloads. Although it would seem that tuning tends to lead to no performance change, or a decline in performance, there are some tuned selectors that provide a performance improvement, either by reducing the number or energy of overloads, or by reducing the curtailment applied.

In addition to tuned selectors having zero weights, some were found to have arbitrarily high weights after reaching the iteration limit within the tuning process. Essentially, those selectors were insensitive to tuning, and several of the model types were found to be consistently insensitive and therefore not recommended for tuning: `HyperPipes`, `KStar`, `LADTree`, all the `LibSVM` variants, `OneR`, and `SGD`.

8.8 Conclusions

This chapter has shown that machine learning can be used to create effective algorithm selectors for power flow management for each of the three case study systems, answering research objective 2 in the affirmative. Effective selectors have been created using both direct and EPM-based designs that provide statistically significant performance benefits, and in some cases, match the performances of the oracles. Although the most effective selectors for each system were direct selectors, the performances provided by the most effective EPM-based selectors were not dissimilar, so there is no strong evidence to suggest that either design is most promising for power systems control applications.

The performance of the selectors created in this work is summarised below for each case study system:

- For the 33 kV meshed distribution system, selectors were created that applied statistically significantly less curtailment than PFM-OPF, while still removing all overloads. The most effective selector – a tuned selector using a `MultilayerPerceptron` model – was able to close 51.04% of the performance gap between PFM-OPF and the oracles.

- For the IEEE 14-bus system, selectors were created that could provide a statistically significant performance benefit by minimising the number and energy of overloads. Several selectors could remove all overloads, thus matching the performance of the oracles. The most effective selector – an unweighted selector using an `AdaBoostM1` (`REPTree`) model – applied the least curtailment whilst removing all overloads.

- For the IEEE 57-bus system, selectors were created that could provide a statistically significant performance benefit by minimising either the number or energy of overloads. The most effective selector at minimising the number of overloads – an unweighted selector using a `MultilayerPerceptron` model – could close 99.50% of the performance gap between PFM-OPF and oracle 1. In fact, the difference in the number of overloads between that selector and oracle 1 was not statistically significant. The most effective selector at minimising overload energy – another unweighted selector using a `MultilayerPerceptron` model – was able to close 98.49% of the performance gap between PFM-OPF and oracle 2 (although the performance difference was still statistically significant).

This chapter has also revealed the effect of some aspects of the selectors' designs:

- The objectives considered when creating the selectors were found to be reflected in how likely the resultant selectors were to be effective against each of the performance measures. For example, only the selectors that considered minimising curtailment were effective for that performance measure on the 33 kV meshed distribution system. For the IEEE 14- and 57-bus a dependence on the overload objective (number or energy) and the performance against each objective was observed, although there was a general bias towards minimising the number of overloads.

- Selectors that consider larger algorithm sets have worse performance than equivalent selectors that consider smaller sets, likely due to the increased complexity of the learning task if more algorithms are considered. This suggests smaller sets of algorithms should be considered when creating a selector.

- Which of the 74 machine learning model types and variants used in this work produced effective selectors was found to vary, depending on the system, the selector design variant, and the objectives used to assess performance. However, relatively few could produce effective selectors for each of the systems, and those that could (listed in Table 8.6) are considered the most promising for power systems control applications.

- Tuning the weights of weighted direct selectors delivered a performance benefit for the 33 kV meshed distribution system, as the equivalent untuned selectors – whose weights only represented overload performance – simply just selected PFM-OPF for all states. Furthermore, the most effective tuned selector had better performance than the most effective unweighted selector for that system. For the other two systems, the performance of the tuned selectors was often found to match – and sometimes be worse than – that of an equivalent untuned selector, although sometimes performance could improve with tuning. For those two systems, the overload objectives were paramount, so it is not an unexpected finding that the tuning process – which optimises the second objective, which was minimising curtailment in this work – did not offer much of a performance benefit for those systems. However, the example of tuning in this work shows tuning can be beneficial, depending on the system and objectives considered.

Chapter 9

Discussion

This chapter considers a few points relevant to the implementation of algorithm selection systems, based on the findings presented in previous chapters.

9.1 Implementation outline

The first step when considering implementing an algorithm selection system is to ascertain what the potential performance benefit could be, which can be achieved using offline simulations with several algorithms as described in Chapters 2 to 5. Once a potential benefit is established, then techniques such as those developed in this work (explained in Chapters 6 and 7) can be used to develop selectors, while also considering the points discussed below.

9.2 Selector design choices

The results of the previous chapter showed that which selector design (direct or EPM-based) and which model types produced effective selectors – and those that produced the most effective selectors – varied between the case study systems. As well as the model types varying, so did the model families, with the most effective model types representing artificial neural networks (`MultilayerPerceptron`), ensembles (`AdaBoostM1` (`REPTree`)), lazy learners (`IBk`), tree learners (`FT`, `LMT`, and `RandomForest`), and other approaches (`RBFRegressor`). Although there was no consensus on which model type produces the most effective selectors, the concise set of model configurations listed in Table 8.6 are recommended for preliminary evaluations of whether effective machine learning-based algorithm selectors can be created for a new application.

9.3 Data requirements

At its very essence, machine learning is concerned with finding relationships in data, so the availability of data is essential for creating and using machine learning-based algorithm selectors. The *quantity* of data available is important for training, while the *quality* of the data – in terms of what features it contains, and whether there is missing or erroneous data – is important for both training selectors, which can be a purely offline process, and the testing selectors of selectors, when they are used online to make algorithm selection decisions.

Data quantity

The amount of data (the number of states for per-state algorithm selection) required to train a machine learning model to achieve a particular level of predictive performance is not known *a priori*, if indeed the model is sufficiently complex to represent the underlying relationship in the data. The amount of data required can be expected to vary between machine learning model types, as well as depending on the features used, the system that the states describe, and the distribution of states during training and during use (testing).

Although the amount of data required is not known *a priori*, a general observation from machine learning is that larger quantities of training data tends to lead to machine learning models with improved predictive performance [143]. This relationship has been observed when using machine learning to create power flow management algorithm selectors, as can be seen in Figure 9.1. The figure shows the effect of changing the training set size for three different model types used to create unweighted direct selectors for the IEEE 14-bus system. The minimisation of overloads was the objective considered when creating the selectors, and it can be seen that the performance against this objective tends to improve as larger training set sizes are used. However, if the underlying relationship in the data is more complex than what the machine learning model can represent, then predictive performance will plateau, such as for the `OneR`-based selector in Figure 9.1. In this case, larger quantities of data will only lead to longer training times. Furthermore, there is the risk of *overtraining* when larger quantities of training data are used, where the model has good performance during training but does not generalise well, giving poor performance on the test set.

Related to the question of what quantity of data is *required* is that of whether such data is *available*. If the data comes from a real-world system, it may already be available, such as stored SCADA data, or may be collected specifically for creating algorithm selectors. It is expected that visibility within power systems will increase in the future [2], so the availability of data may not constrain the creation of algorithm selectors. However, direct selectors require that each of the algorithms is tested on exactly the same set of states, otherwise it is

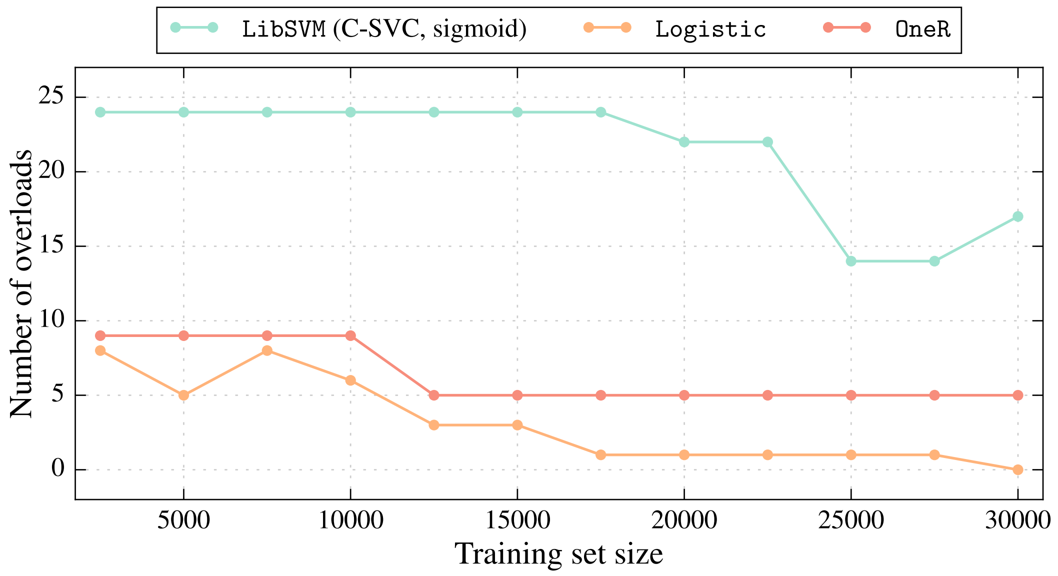


Fig. 9.1 Effect of training set size on overload performance of unweighted selectors created for the IEEE 14-bus system, with the selectors only considering minimising overloads

not possible to determine which algorithm is most effective for each state. As encountering exactly the same state twice in a real-world system is highly improbable, it would be difficult – if not impossible – to create a direct selector just from collected data.

An alternative to using data from a real-world system is to generate the data from simulations. This was the approach taken in this work, with the main constraints being processor time for the simulations and storage space for retaining results. As simulations rely on models, it is important that the model used can sufficiently approximate the behaviour of the real-world system and therefore will have a similar state space. Even if the state space of the model is close to that of the real-world system, the distribution of states in the simulations – and then subsequently used for creating an algorithm selector – may be different to the distribution of states encountered when using (testing) the selector. For example, the states used in this work to train the selectors for the 33 kV meshed distribution system were generated by drawing the state variables from independent uniform random distributions, so there was an unbiased distribution of states within the state space; however, the test states for that system were based on real-world data that was non-uniform, so the distribution of states was different to that used for training. However, this did not appear to prevent the creation of effective algorithm selectors.

It may be beneficial to have a different distribution of states for training than that found during the use of a selector. In a classification task there may be classes that have high costs, but appear infrequently in the test data, and it has been observed that a classifier trained using data which over-represents those classes may perform better than if training data was used that had a similar distribution of examples to the test set [144]. This could be expected to apply to classifiers used for algorithm selection, so for example, it could be beneficial for power flow management algorithm selection to skew the distribution of states in the training data to favour states where overloads occur and there are performance differences between the algorithms.

Data quality

For power flow management algorithm selection, the *features* used by the selectors represent characteristics of the system state that are predictive of algorithm performance, and are therefore helpful in making algorithm selection decisions. Features could include the branch loadings, busbar voltages, and switch statuses if the network topology is subject to change.

Although the feature sets used in this work fully describe the systems' states, and therefore fully characterise the problems that the power flow management algorithms are applied to, they do so at a high level. Because of this, some information that could potentially be relevant to selecting a power flow management algorithm – for example, the loadings of the circuits that can become overloaded – are only available by executing a load flow, which is a non-linear function of the state variables (contained in the feature sets). Such information could help improve the predictive performance of a direct selector or EPM used within a selector, as a number of the machine learning algorithms evaluated in this work do not consider non-linear relationships between the features. However, the effect of different feature sets has not been investigated in this work in order to limit the scope, although the features that were used did not prevent effective selectors from being produced.

A particular set of features may produce the most effective selector, but in an implementation of an algorithm selection system it would also be important to consider whether those features are available as measurements from the system that the algorithms are being selected for. It could be expected that some measurements would already be available, as required by the control algorithms, so these could be utilised for the algorithm selector. Another issue with a real-world implementation of an algorithm selection system is that the measurements would be subject to noise and failures, so the algorithm selectors – and indeed the control algorithms – would need to be robust to this. Creating models that are robust to noise is a common concern in machine learning, and there are existing techniques that could be applied, such as artificially adding noise to the training data [145].

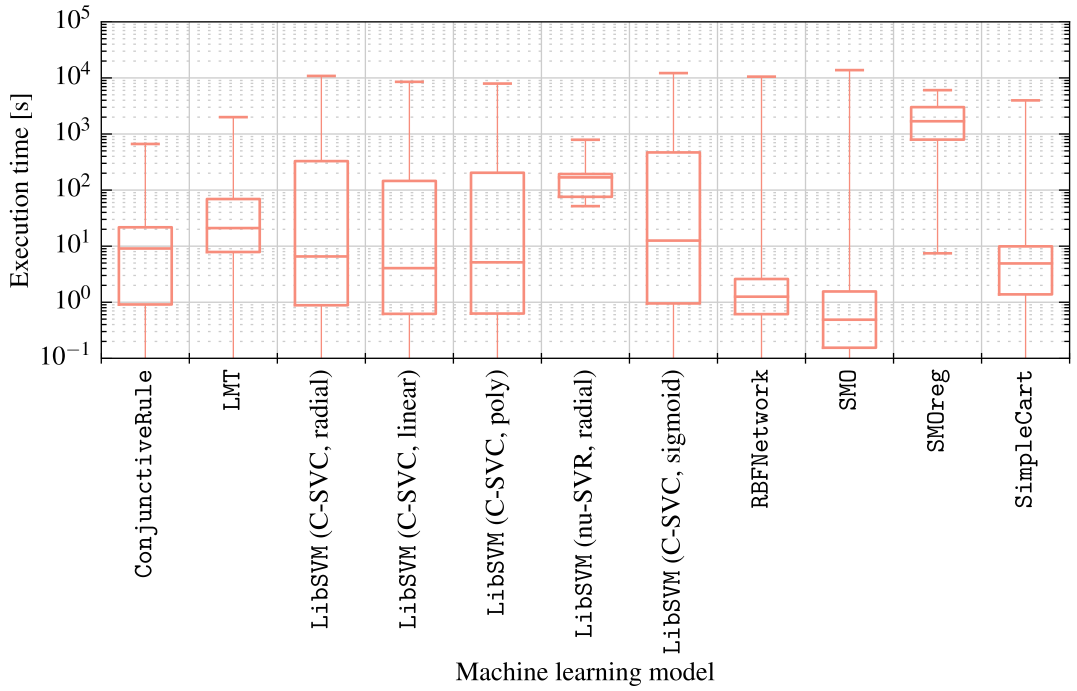


Fig. 9.2 Box plot of time taken to train the 10 machine learning model types that had the longest training time

9.4 Time requirements

Aside from the data required for an algorithm selector, another important factor to consider is the time requirements, both in terms of creating the selector, and in its use.

Time required to create a selector

The time required to create a selector can be split in to two main components. The first component is the time required to obtain the data to be used for training. If data needs to be generated synthetically, the time required will vary depending on such factors as the number of states simulated, the complexity of the power system studied (as that will influence the execution time for load flow), and the number of algorithms to be simulated. This could take considerable time – for example, generating all of the power flow management performance data used in this work (as described in Chapters 3 and 4) took around 2 weeks of computation – although this can be reduced through optimising the simulation code and using high-performance computing. This time is of course unnecessary if data already exists that can be used for training, but even then, there may still be time associated with preparing the data to make it suitable to be used for training selectors.

The second component of the time required to create a selector is that associated with training, once training data has been obtained. This varies according to the same factors as considered for the generation of training data, as well as varying with the machine learning model type used. The majority (62 of 74) of the model types and variants investigated in this work took less than 10 minutes for a single training run (here, the meaning of “run” depends on the selector design: for unweighted and weighted direct selectors, it is the single training pass associated with those designs; for tuned direct selectors, it is the re-training step at the end of the tuning process; and for EPM-based selectors, it is the training of a single EPM, rather than all EPMs in a selector). The training times for the remaining model types – with the exception of `GaussianProcesses`, for which no runs were completed (please refer to Section 8.1.2) – are shown in Figure 9.2. The figure is a box plot showing the range of training times for each model type, with the extent of the boxes indicating the lower and upper quartiles, the mid point of each box showing the median time, and the whiskers indicating the minimum and maximum. The figure shows that four of the model types can take over 10^4 seconds for a single training run, with `SMO` having the longest (13782 seconds, which is almost four hours).

Once the selector is implemented, if there are changes to the power system – thus altering the state space – then new training data will need to be obtained, and the selector re-trained. Similarly, if an additional algorithm need to be consider for selection, then new training will need to be obtained – but only for the one algorithm – and the selector re-trained. In this case, direct selectors would need to be completely retrained; however, for EPM-based selectors it would only be necessary to create new EPMs for the additional algorithm and then integrate that in to the selection logic. In this respect EPM-based selectors are more scalable.

Time required to use a selector

All of the model types take on average less than a second to make a prediction for a single state, with the exception of `KStar`, which can take up to 5.32 seconds on average. In fact, the majority (62 out of 74) model types take less than 0.01 seconds to make a prediction for a single state. Such short time requirements for generating algorithm selection decisions should be suitable for use online in real-time control applications [3, 31], although the time associated with receiving measurements from the power system (the features) would also need to be taken into account when implementing an algorithm selection system.

9.5 Financial assessment

This work has demonstrated that algorithm selection for power flow management can deliver statistically signification benefits, in terms of reducing overloads and curtailment. However, for any real-world implementation a cost-benefit analysis would need to be undertaken to ascertain whether algorithm selection would provide an overall financial benefit, compared with only using a single algorithm, or alternative options such as network reinforcement. The financial benefits and costs associated with an algorithm selection system would be project-specific, so are not evaluated here; however, the main factors that would influence the cost-benefit analysis are described below.

The main costs to consider would be:

- Development and validation: this would include obtaining and preparing training data, choosing a selector design, training the selector, testing the selector offline (validation), and wrapping the selector in a software environment ready for deployment.
- Processing hardware: the selector would require a computational platform to run on, which could be located locally on the network or remotely, such as in a control centre. Potentially, the computational platform could have no extra cost if the hardware required to support the control algorithms is used.
- Sensors and communications: these would be required to obtain the measurements used as features by the selector. The measurements already required by the control algorithms could be used, which would have little or no extra cost. If additional measurements would improve the performance of the selector, further analysis would be required to compare the costs of obtaining the extra measurements against the benefits of the performance improvement of the selector.
- Other: the algorithm selection system may have additional costs such as commissioning, training, maintenance, and the licence costs of each control algorithm that is considered for selection.

The main financial benefits to consider would be:

- Increased asset lifespan: a reduction in overloading of assets would increase their lifespan, thus deferring investment in replacements.
- Improved utilisation of existing capacity: using the control algorithms selected by an algorithm selection system could allow existing assets to be utilised more effectively,

which could defer, reduce or remove the need for investment in reinforcement to provide additional capacity.

- Reduced costs associated with curtailment: the generators would be expected to increase their revenue as they would be able to export more energy. Furthermore, the operator of the power system may also see a reduction in their costs, if they are otherwise required to reimburse the generators for curtailment.

Chapter 10

Conclusions

This chapter concludes this work. First, the research presented in this work is evaluated against the objectives that were outlined at the start in Chapter 1, and then the major contributions highlighted. Finally, an outlook for applying algorithm selection more generally within power systems is elaborated, along with some areas for future work to extend this research.

10.1 Evaluation against research objectives

At the start of this work, Section 1.3 listed the objectives for this research. The objectives are reiterated below (in *italics*), along with descriptions of the work that has been presented to address each objective.

1. *Examine if potential performance benefits for power systems control can be derived by selecting the algorithms on a per-state basis.*

   Overall, this objective has been met and answered in the affirmative. Power system modelling and extensive simulations (Chapters 2-4) were conducted to characterise the performance of several power flow management algorithms. Chapter 5 used these results to demonstrate there were statistically significant potential performance benefits if algorithms were optimally selected on a per-state basis for three of the four case study systems.

   This objective had the following sub-objectives:

   (a) *Identify a power systems control task that has characteristics shared with many other power systems control tasks, so the results for the one control task are likely to be generalisable to other tasks.*

Chapter 2 identified power flow management as a control task that shares a number of characteristics with other power system control tasks, such as considering similar control actions, applying across a range of scales within power systems, and that multiple algorithms are available to achieve the control task. Due to these shared characteristics, the findings of this work for power flow management are expected to apply more generally to other power system control tasks.

(b) *Implement and test several power systems control algorithms for the chosen control task, which represent diverse approaches to tackling the control task.*

In Chapter 2, a survey of existing power flow management algorithms identified five algorithms that were subsequently implemented. As described in the chapter, those algorithms represented diverse approaches, which included optimal power flow, constraint satisfaction, and linear programming. The algorithms were tested in a purpose-built test environment (described in Chapter 3)), with the results presented in Chapter 4.

(c) *Test the algorithms on power system models that represent different network designs.*

Four case study power systems were used in this work, which were diverse in a number of aspects of their designs: their topologies (radial and meshed), their voltage levels (transmission and distribution), their origins (UK and the US), and their scale (in terms of the number of buses, branches and generators). The case study systems were described in Chapter 4, along with the results of testing the power flow management algorithms within each system.

(d) *Simulate a wide range of conditions within the power system models, in order to exercise the performance of the power system control algorithms.*

The results presented in Chapter 4 were for at least 10,000 states simulated per system. For three of the four case study systems, the simulated states were distributed within the state spaces in a random and unbiased manner, so represented a wide range of conditions. For the remaining system, the values of the state variables were taken from real data, so although a range of states were simulated, the distribution of states followed that of what could be found in a real system.

2. *If the answer to research objective 1 is affirmative, the research shall investigate if algorithm selection systems (algorithm selectors) can be created to exploit the potential performance benefits for power systems control from per-state algorithm selection.*

As the answer to research objective 1 was in the affirmative, research was conducted to develop and evaluate algorithm selectors to perform per-state algorithm selection for power flow management. For the three case study systems that exhibited potential performance benefits from per-state algorithm selection, algorithm selectors were successfully created that could exploit some of the potential performance benefits and outperform the most effective algorithms by statistically significant amounts, as shown by the results presented in Chapter 8.

This objective included:

(a) *Identifying existing algorithm selection techniques that could be applicable to developing algorithm selectors for power systems control.*

A review of existing algorithm selection techniques in Chapter 6 identified machine learning-based approaches as the predominant way to create algorithm selectors, across a number of problem domains. The review also identified two main variants of the machine learning-based approach: the creation of direct selectors, or those based on empirical performance models.

(b) *Exploiting the existing algorithm selection techniques to create algorithm selectors for power systems control.*

Based on the findings of the review in Chapter 6, Chapter 7 describes how the existing algorithm selection techniques were adapted and extended for selecting power flow management algorithms, with the performance of the created selectors presented in Chapter 8.

Selectors were created based on the two main design variants (direct and EPM-based). For the three case study power systems for which selectors were created, the results show that both selector design variants could create effective algorithm selectors, for the power flow management algorithms, performance objectives, and machine learning model types considered in this work. However, the most effective direct selectors outperformed the most effective EPM-based selectors.

Additionally, the influence of other aspects of the selectors' design were examined. The machine learning model type used to create the selectors was found to have a significant influence on performance, although which model type produced the most effective selectors was found to vary between systems and selector design variants. Restricting the number of algorithms considered by a selector was found to be beneficial, as selectors that considered larger sets were more likely to perform worse than otherwise identical selectors that considered smaller sets. Finally, the results showed that the objectives considered by the selectors were reflected in their performance, demonstrating that algorithm selectors can be tailored to the objectives of interest.

(c) *Extending the existing techniques and, when necessary, developing new techniques to allow algorithm selection to be applied to power systems control.*

In addition to the adaptations necessary to exploit existing algorithm selection techniques for power flow management, an additional technique was developed that allowed two nested objectives to be considered (described in Section 7.1.3). The technique varied (tuned) the weights of the data used to create a direct selector so that two objectives could be balanced against one another. This was useful in particular for the 33 kV meshed distribution system, where the potential benefit of per-state algorithm selection was minimising curtailment, whilst also ensuring all overloads were removed.

10.2 Contributions

The main contributions of this work are as follows:

1. Large-scale testing of power flow management algorithms, including developing a test environment that allows testing of other power system control algorithms. Five power flow management algorithms that represented different approaches were tested across a range of conditions on four case study power system models, which had varying scales and topologies. Results presented for each algorithm included the number and energy of overloads, the amount of generator curtailment applied, and execution times.

2. Demonstrated that none of the algorithms tested provided the most effective performance for every state in each of the case study systems, for the particular performance objectives considered.

3. Demonstrated that if power flow management algorithms were optimally selected on a per-state basis, there were statistically significant potential performance benefits for three of the four case study systems.

   (a) For the 33 kV meshed distribution system, the potential performance benefit was a reduction in curtailment, while still removing all overloads.

   (b) For the IEEE 14-bus system, the potential performance benefits were a reduction in the number or energy of overloads, as well as a reduction in curtailment compared with the most effective algorithms at removing overloads.

   (c) For the IEEE 57-bus system, the potential performance benefits were a reduction in the number or energy of overloads.

4. Evaluated machine learning-based algorithm selectors for power flow management, including direct and EPM-based selector designs, by adapting and further developing existing algorithm selection techniques. The influence of a number of design choices was examined, including the choice of machine learning model type, the objective used by the selectors, and the sets of algorithms considered for selection. A total of 74 machine learning model types were evaluated, which is in excess of other studies that have used a large range of models to create algorithm selectors for other applications, such as [7] and [96].

5. Developed methods to create algorithm selectors that considered two nested objectives, using tuning of training set weights.

6. Demonstrated that machine learning-based algorithm selectors could realise some of the potential performance benefits offered by per-state algorithm selection.

   (a) For the 33 kV meshed distribution system, the most effective algorithm selectors applied statistically significantly less curtailment than the most effective power flow management algorithm, whilst still removing all overloads.

   (b) For the IEEE 14-bus system, the most effective selectors could reduce the number or energy of overloads by a statistically significant amount, in comparison to the most effective algorithms for each of those performance measures. Furthermore, some of the selectors were able to remove all overloads entirely, thus making the optimal selection decisions with respect to minimising the number and energy of overloads.

   (c) For the IEEE 57-bus system, the most effective selectors could provide statistically significant performance benefits in terms of reducing either the number or energy of overloads. In comparison to the number of overloads left if the optimal selection decisions were made to minimise that performance measure, one selector had such similar performance that the difference was not statistically significant.

10.3 Outlook and future work

This work has demonstrated the per-state algorithm selection for power flow management can not only *offer* a potential performance benefit, but that machine learning-based algorithm selectors can exploit and *deliver* some of that performance benefit. The case study power systems used to demonstrate this are diverse in a number of ways – so therefore offer the promise of generalising the findings to other power systems – but share one thing in common: they represent *existing* power systems.

In contrast to the systems examined in this work, future power systems are expected to be more complex, and also more uncertain [2]. The complexity will come from the number of active participants in the management of the system, as potentially millions of active devices are connected in. The uncertainty will come, for example, increasingly large amounts of intermittent generation injecting power throughout the system, and through shifts in the temporal and spatial patterns of load. In addition to these two factors, economic, social, environmental and other constraints will limit possibilities to reinforce the system. Therefore, the challenge of ensuring the system is kept within acceptable operating limits will not just rely on investing in assets, but increasingly on using algorithms to manage the system. In this context, while control algorithms will be more vital, the increasing complexity and uncertainty will reduce the likelihood that any single algorithm will always provide the most effective performance, and thus the potential for using – and benefiting from – algorithm selection could be greater. Although the increased complexity and uncertainty would also affect the selectors, they would be expected to scale as well as the algorithms they select, so if complexity or uncertainty meant no algorithm could be used, whether a selector would be able to make an appropriate selection decision would be irrelevant.

Based on these observations, it is suggested that the research presented in this work could be extended in the following ways:

- Applying the techniques developed in this work to other power systems and conditions within those systems, including system models that represent what would be expected for future power systems.

- Evaluating other power system control applications – such as frequency control, or power electronic converter control – to assess whether per-state algorithm selection can offer performance benefits. If potential benefits are found, then the techniques developed in this work could be adapted to exploit the performance benefits. Potentially, other tasks within the power systems domain where multiple algorithms are be applied could benefit from some form of fine-grained algorithm selection.

- Examine if machine learning techniques other than those examined in this work and variations of the machine learning processes – such as tailoring model parameters, the features, and the amount of training data used – can create algorithm selectors that are able to further exploit the potential performance benefits on offer.

- Investigating the performance of algorithm selectors for power systems control applications in a hardware-in-the-loop simulation, to validate their performance in a real-world setting.
- Developing algorithm selectors that are able to select a sequence of algorithms over a time horizon, as the selectors in this work can only provide an algorithm selection for a single time instant.
- Developing algorithm selectors for non-deterministic algorithms.

References

[1] J. E. King, S. C. E. Jupe, and P. C. Taylor, “Autonomic control algorithm selection in decentralised power systems: a voltage control case study,” in *22nd International Conference and Exhibition on Electricity Distribution (CIRED)*, Stockholm, 2013.

[2] S. D. J. McArthur, P. C. Taylor, G. W. Ault, J. E. King, D. Athanasiadis, V. D. Alimisis, and M. Czaplewski, “The Autonomic Power System - Network operation and control beyond smart grids,” in *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe, ISGT Europe*, Berlin, 2012, p. IEEE Power and Energy Society; Technische Universi. [Online]. Available: http://dx.doi.org/10.1109/ISGTEurope.2012.6465807

[3] D. Roberts, “Network Management Systems for Active Distribution Networks - A Feasibility Study,” DTI, Tech. Rep., 2004.

[4] D. Harel and Y. A. Feldman, *Algorithmics: The Spirit of Computing*. Addison Wesley, 2004. [Online]. Available: http://books.google.co.uk/books?id=txxLovFWkCUC

[5] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[6] M. Streeter, D. Golovin, and S. F. Smith, “Combining multiple heuristics online,” in *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, ser. AAAI’07. AAAI Press, 2007, pp. 1197–1203. [Online]. Available: http://dl.acm.org/citation.cfm?id=1619797.1619838

[7] L. Kotthoff, I. P. Gent, and I. Miguel, “A Preliminary Evaluation of Machine Learning in Algorithm Selection for Search Problems,” in *Fourth International Symposium on Combinatorial Search (SoCS-2011)*, 2011, pp. 84–91. [Online]. Available: https://www.aaai.org/ocs/index.php/SOCS/SOCS11/paper/viewFile/4006/4362

[8] J. Seo, M. Bakay, Y.-W. Chen, S. Hilmer, B. Shneiderman, and E. P. Hoffman, “Interactively optimizing signal-to-noise ratios in expression profiling: project-specific algorithm selection and detection p-value weighting in Affymetrix microarrays,” *Bioinformatics*, vol. 20, no. 16, pp. 2534–2544, Nov. 2004. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/20/16/2534.abstract

[9] W. Armstrong, P. Christen, E. McCreath, and A. P. Rendell, “Dynamic Algorithm Selection Using Reinforcement Learning,” in *International Workshop on Integrating AI and Data Mining (AIDM ’06)*, 2006, pp. 18–25.

[10] J. Yang and B. Jiu, “Algorithm selection: a quantitative approach,” *Trading*, vol. 2006, no. 1, pp. 26–34, 2006. [Online]. Available: http://www.iijournals.com/doi/abs/10.3905/tr.2006.664138

[11] M. Thomson, “Automatic voltage control relays and embedded generation. I,” *Power Engineering Journal*, vol. 14, no. 2, pp. 71–76, 2000.

[12] T. Xu, N. Wade, E. Davidson, P. C. Taylor, S. McArthur, and W. Garlick, “Case-based reasoning for coordinated voltage control on distribution networks,” *Electric Power Systems Research*, vol. 81, no. 12, pp. 2088–2098, Dec. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378779611001921http://linkinghub.elsevier.com/retrieve/pii/S0378779611001921

[13] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[14] S. Riccardo, “Architectures for distributed and hierarchical Model Predictive Control – A review,” *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, May 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959152409000353

[15] G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos, “Self-Organisation and Emergence in MAS : An Overview,” *Informatica*, vol. 30, pp. 45–54, 2006. [Online]. Available: http://www.informatica.si/PDF/30-1/03_Serugendo-Self-OrganisationandEmergencein...pdf

[16] G. Anderson, “Dynamics and Control of Electric Power Systems,” pp. 36–46, 2012. [Online]. Available: http://www.eeh.ee.ethz.ch/fileadmin/user_upload/eeh/studies/courses/power_system_dynamics_and_control/Documents/DynamicsPartI_lecture_notes_2012.pdf

[17] A. Pillay, S. Prabhakar Karthikeyan, and D. P. Kothari, “Congestion management in power systems – A review,” *International Journal of Electrical Power & Energy Systems*, vol. 70, no. 0, pp. 83–90, Sep. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061515000411

[18] Q. Zhou and J. W. Bialek, “Generation curtailment to manage voltage constraints in distribution networks,” *IET Generation, Transmission & Distribution*, vol. 1, no. 3, pp. 492–498, 2007.

[19] J. Carpentier, “Contribution a l’etude du dispaching economique,” *Bulletin de la Societe Francaise des Electriciens*, vol. 3, no. 8, pp. 431–447, 1962.

[20] S. Frank, I. Steponavice, and S. Rebennack, “Optimal power flow: a bibliographic survey I,” *Energy Systems*, vol. 3, no. 3, pp. 221–258, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1007/s12667-012-0056-yhttp://link.springer.com/10.1007/s12667-012-0056-y

[21] “IEEE Xplore - Search Results.” [Online]. Available: http://ieeexplore.ieee.org/search/searchresult.jsp?queryText%3D.QT.optimal+power+flow.QT.

[22] H. W. Dommel and W. F. Tinney, “Optimal Power Flow Solutions,” *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-87, no. 10, pp. 1866–1876, 1968.

[23] D. S. Kirschen and H. P. Van Meeteren, “MW/voltage control in a linear programming based optimal power flow,” *IEEE Transactions on Power Systems*, vol. 3, no. 2, pp. 481–489, 1988.

[24] S. Granville, “Optimal reactive dispatch through interior point methods,” *IEEE Transactions on Power Systems*, vol. 9, no. 1, pp. 136–146, 1994.

[25] L. L. Lai, J. T. Ma, R. Yokoyama, and M. Zhao, “Improved genetic algorithms for optimal power flow under both normal and contingent operation states,” *International Journal of Electrical Power & Energy Systems*, vol. 19, no. 5, pp. 287–292, Jun. 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061596000518

[26] A. G. Bakirtzis, P. N. Biskas, C. E. Zoumas, and V. Petridis, “Optimal power flow by enhanced genetic algorithm,” *IEEE Transactions on Power Systems*, vol. 17, no. 2, pp. 229–236, 2002.

[27] M. A. Abido, “Optimal power flow using particle swarm optimization,” *International Journal of Electrical Power & Energy Systems*, vol. 24, no. 7, pp. 563–571, Oct. 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061501000679

[28] J. Lavaei and S. H. Low, “Zero Duality Gap in Optimal Power Flow Problem,” *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 92–107, 2012.

[29] 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 *15th International Conference on Intelligent System Applications to Power Systems, 2009. ISAP ’09.*, 2009, pp. 1–7.

[30] M. J. Dolan, E. M. Davidson, I. Kockar, G. W. Ault, and S. D. J. McArthur, “Distribution Power Flow Management Utilizing an Online Optimal Power Flow Technique,” *IEEE Transactions on Power Systems*, vol. 27, no. 2, pp. 790–799, 2012.

[31] ——, “Reducing Distributed Generator Curtailment Through Active Power Flow Management,” *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 149–157, 2014.

[32] “PowerWorld » The visual approach to electric power systems.” [Online]. Available: http://www.powerworld.com/

[33] A. R. Ahmadi and T. C. Green, “Optimal power flow for autonomous regional active network management system,” in *Power & Energy Society General Meeting, 2009. PES ’09. IEEE*, 2009, pp. 1–7.

[34] S. Gill, I. Kockar, and G. W. Ault, “Dynamic Optimal Power Flow for Active Distribution Networks,” *IEEE Transactions on Power Systems*, vol. 29, no. 1, pp. 121–131, 2014.

[35] S. W. Alnaser and L. F. Ochoa, “Distribution network management system: An AC OPF approach,” in *2013 IEEE Power and Energy Society General Meeting (PES)*, 2013, pp. 1–5.

[36] "AIMMS." [Online]. Available: http://www.aimms.com/

[37] S. Jupe, P. C. Taylor, and A. Michiorri, "Coordinated output control of multiple distributed generation schemes," *IET Renewable Power Generation*, vol. 4, no. 3, pp. 283–297, 2010. [Online]. Available: http://link.aip.org/link/ISETCN/v4/i3/p283/s1&Agg=doi

[38] L. L. Grigsby, *Power Systems, Third Edition*, ser. The electric power engineering handbook. CRC Press, 2012. [Online]. Available: https://books.google.co.uk/books?id=h0jNBQAAQBAJ

[39] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation, and Control.* Wiley, 2012. [Online]. Available: http://books.google.co.uk/books?id=ItuC5oZ-16QC

[40] S. Jupe and P. C. Taylor, "Strategies for the control of multiple distributed generation schemes," in *20th International Conference and Exhibition on Electricity Distribution (CIRED)*, 2009, pp. 1–4.

[41] S. C. E. Jupe and P. C. Taylor, "Distributed generation output control for network power flow management," *IET Renewable Power Generation*, vol. 3, no. 4, pp. 371–386, 2009.

[42] L. Kane, G. Ault, and S. Gill, "An Assessment of Principles of Access for Wind Generation Curtailment in Active Network Management Schemes," in *22nd International Conference and Exhibition on Electricity Distribution (CIRED)*, no. 0237, 2013.

[43] C.-L. Chang and Y.-Y. Hsu, "Steady-state security control using a sensitivity-based approach," *Electric Power Systems Research*, vol. 18, no. 1, pp. 1–10, Jan. 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037877969090040A

[44] I. Skokljev, V. Maksimovic, and H. Weber, "Symbolic analysis congestion management," in *IEEE Power Tech*, vol. 2, Bologna, 2003.

[45] P. Wong, P. Albrecht, R. Allan, R. Billinton, Q. Chen, C. Fong, S. Haddad, W. Li, R. Mukerji, D. Patton, A. Schneider, M. Shahidehpour, and C. Singh, "The IEEE Reliability Test System-1996. A report prepared by the Reliability Test System Task Force of the Application of Probability Methods Subcommittee," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 1010–1020, 1999.

[46] M. J. Dolan, E. M. Davidson, G. W. Ault, K. R. W. Bell, and S. D. J. McArthur, "Distribution Power Flow Management Utilizing an Online Constraint Programming Method," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 798–805, 2013.

[47] T. Luo, M. J. Dolan, E. M. Davidson, and G. W. Ault, "Assessment of a New Constraint Satisfaction-Based Hybrid Distributed Control Technique for Power Flow Management in Distribution Networks with Generation and Demand Response," *IEEE Transactions on Smart Grid*, vol. 6, no. 1, pp. 271–278, 2015.

[48] R. A. F. Currie, G. W. Ault, C. E. T. Foote, and J. R. McDonald, "Active power-flow management utilising operating margins for the increased connection of distributed generation," *IET Generation, Transmission & Distribution,*, vol. 1, no. 1, pp. 197–202, 2007.

[49] "constraint 0.4.1 : Python Package Index." [Online]. Available: https://pypi.python.org/pypi/constraint/

[50] "PYPOWER 4.0.1 : Python Package Index." [Online]. Available: https://pypi.python.org/pypi/PYPOWER/4.0.1

[51] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2011.

[52] K. T. Fang, R. Li, and A. Sudjianto, *Design and Modeling for Computer Experiments*, ser. Chapman & Hall/CRC Computer Science & Data Analysis. CRC Press, 2005. [Online]. Available: https://books.google.co.uk/books?id=VGqbyIUVZw8C

[53] K. Hinkelmann and O. Kempthorne, *Design and Analysis of Experiments, Introduction to Experimental Design*, ser. Design and Analysis of Experiments. Wiley, 2007. [Online]. Available: https://books.google.co.uk/books?id=T3wWj2kVYZgC

[54] G. Andersson, "Modelling and analysis of electric power systems," 2008. [Online]. Available: http://www.eeh.ee.ethz.ch/uploads/tx_ethstudies/modelling_hs08_script_02.pdf

[55] F. Milano, *Power system modelling and scripting*. Springer, 2010. [Online]. Available: http://books.google.com/books?hl=en&lr=&id=MQu7IqoLrfYC&oi=fnd&pg=PR4&dq=Power+System+Modelling+and+Scripting&ots=ajftJtiBPa&sig=FPZPVxWvsx5FVFZkzGCaUvZB9w0

[56] R. Kuffel, J. Giesbrecht, T. Maguire, R. P. Wierckx, and P. McLaren, "RTDS-a fully digital power system simulator operating in real time," in *IEEE WESCANEX 95. Communications, Power, and Computing.*, vol. 2, 1995, pp. 300–305 vol.2.

[57] V. Alimisis, C. Piacentini, J. E. King, and P. C. Taylor, "Operation and Control Zones for Future Complex Power Systems," in *2013 IEEE Green Technologies Conference*, 2013, pp. 259–265.

[58] J. E. King, S. C. E. Jupe, and P. C. Taylor, "Performance evaluation of control algorithms for active distribution networks - the potential for algorithm selection," in *CIGRÉ Session*, Paris, 2014.

[59] J. King, S. Jupe, and P. Taylor, "The potential of network state-based algorithm selection to improve power flow management," in *2014 IEEE PES General Meeting | Conference & Exposition*, Washington DC, 2014, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6938918

[60] J. E. King, S. C. E. Jupe, and P. C. Taylor, "Network State-Based Algorithm Selection for Power Flow Management Using Machine Learning," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2657–2664, 2015.

[61] J. L. Rueda, D. G. Colome, and I. Erlich, “Assessment and Enhancement of Small Signal Stability Considering Uncertainties,” *IEEE Transactions on Power Systems*, vol. 24, no. 1, pp. 198–207, 2009.

[62] R. Preece and J. V. Milanovic, “Efficient Estimation of the Probability of Small-Disturbance Instability of Large Uncertain Power Systems,” *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1063–1072, 2016.

[63] W. Mendenhall, R. Beaver, and B. Beaver, *Introduction to Probability and Statistics*, ser. Available 2010 Titles Enhanced Web Assign Series. Cengage Learning, 2008. [Online]. Available: https://books.google.co.uk/books?id=-39d6IwtdPkC

[64] G. Argyrous, *Statistics for Research: With a Guide to SPSS*. SAGE Publications, 2011. [Online]. Available: https://books.google.co.uk/books?id=XYv_DIyN4YYC

[65] Y. Hochberg and A. C. Tamhane, *Multiple Comparison Procedures*. New York, NY, USA: John Wiley &amp; Sons, Inc., 1987.

[66] Y. Benjamini and Y. Hochberg, “Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995. [Online]. Available: http://www.jstor.org/stable/2346101

[67] C. Genovese, “False Discovery Rate Control,” in *Brain Mapping: An Encyclopedic Reference*, A. W. Toga, Ed. Elsevier Science, 2015, pp. 501–507. [Online]. Available: https://books.google.co.uk/books?id=ysucBAAAQBAJ

[68] O. J. Dunn, “Multiple Comparisons among Means,” *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, Mar. 1961. [Online]. Available: http://amstat.tandfonline.com/doi/abs/10.1080/01621459.1961.10482090

[69] S. Holm, “A Simple Sequentially Rejective Multiple Test Procedure,” *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979. [Online]. Available: http://www.jstor.org/stable/4615733

[70] “UK Power Networks - Long Term Development Statement.” [Online]. Available: http://www.ukpowernetworks.co.uk/internet/en/about-us/regulatory-information/long-term-development-statement.html

[71] “Long Term Development Statement - SP Energy Networks.” [Online]. Available: http://www.spenergynetworks.co.uk/pages/long_term_development_statement.asp

[72] “United Kingdom Generic Distribution System.” [Online]. Available: http://monaco.eee.strath.ac.uk/ukgds

[73] University of Washington, “Power Systems Test Case Archive,” 2013. [Online]. Available: http://www.ee.washington.edu/research/pstca/

[74] J. Rice, “The algorithm selection problem,” *Advances in Computers*, vol. 15, pp. 65–118, 1976. [Online]. Available: http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1098&context=cstech

[75] M. G. Lagoudakis and M. L. Littman, “Algorithm Selection using Reinforcement Learning,” in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML ’00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 511–518. [Online]. Available: http://dl.acm.org/citation.cfm?id=645529.657981

[76] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, “ParamILS: An Automatic Algorithm Configuration Framework,” *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306, Oct. 2009.

[77] S. Ali and K. A. Smith, “On learning algorithm selection for classification,” *Applied Soft Computing*, vol. 6, no. 2, pp. 119–138, Jan. 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494605000049

[78] C. E. Houstis, E. N. Houstis, M. Katzouraki, T. S. Papatheodorou, and J. R. Rice, “ATHENA: A Knowledge Base System for//ELLPACK,” 1990.

[79] C. Brodley, “Addressing the selective superiority problem: Automatic algorithm/model class selection,” in *10th International Conference on Machine Learning*, Amherst, MA, 1993, pp. 17–24. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.3831&rep=rep1&type=pdf

[80] J. Beck and E. Freuder, “Simple Rules for Low-Knowledge Algorithm Selection,” in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, ser. Lecture Notes in Computer Science, J.-C. Régin and M. Rueher, Eds. Springer Berlin Heidelberg, 2004, vol. 3011, pp. 50–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24664-0_4

[81] L. Lobjois and M. Lemaître, “Branch and bound algorithm selection by performance prediction,” in *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, ser. AAAI ’98/IAAI ’98. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1998, pp. 353–358. [Online]. Available: http://dl.acm.org/citation.cfm?id=295240.295633

[82] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2014.

[83] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[84] Microsoft, “Microsoft demos breakthrough in real-time translated conversations - The Official Microsoft Blog,” 2014. [Online]. Available: http://blogs.microsoft.com/blog/2014/05/27/microsoft-demos-breakthrough-in-real-time-translated-conversations/

[85] N. M. Ball and R. J. Brunner, “DATA MINING AND MACHINE LEARNING IN ASTRONOMY,” *International Journal of Modern Physics D*, vol. 19, no. 07, pp. 1049–1106, Jul. 2010. [Online]. Available: http://dx.doi.org/10.1142/S0218271810017160

[86] The Atlantic, “The Trick That Makes Google’s Self-Driving Cars Work,” 2014. [Online]. Available: http://www.theatlantic.com/technology/archive/2014/05/all-the-world-a-track-the-trick-that-makes-googles-self-driving-cars-work/370871/

[87] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0893608089900208

[88] J. A. Hartigan and M. A. Wong, “Algorithm AS 136: A k-means clustering algorithm,” *Applied statistics*, pp. 100–108, 1979.

[89] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[90] M. Gagliolo and J. Schmidhuber, “Learning dynamic algorithm portfolios,” *Annals of Mathematics and Artificial Intelligence*, vol. 47, no. 3-4, pp. 295–328, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10472-006-9036-z

[91] T. Carchrae and J. C. Beck, “Applying Machine Learning to Low-Knowledge Control of Optimization Algorithms,” *Computational Intelligence*, vol. 21, no. 4, pp. 372–387, 2005. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8640.2005.00278.x

[92] ——, “Low-Knowledge Algorithm Control,” in *Proceedings of the National Conference on Artificial Intelligence*, 2004, pp. 49–54. [Online]. Available: http://www.aaai.org/Papers/AAAI/2004/AAAI04-008.pdf

[93] N. Thomas, G. Tanase, O. Tkachyshyn, J. Perdue, N. M. Amato, and L. Rauchwerger, “A framework for adaptive algorithm selection in STAPL,” in *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, ser. PPoPP ’05. New York, NY, USA: ACM, 2005, pp. 277–288. [Online]. Available: http://doi.acm.org/10.1145/1065944.1065981

[94] K. A. Smith-Miles, “Towards insightful algorithm selection for optimisation using meta-learning concepts,” in *IEEE International Joint Conference on Neural Networks (IJCNN 2008)*, 2008, pp. 4118–4124.

[95] R. Ewald, J. Himmelspach, and A. M. Uhrmacher, “An Algorithm Selection Approach for Simulation Systems,” in *22nd Workshop on Principles of Advanced and Distributed Simulation (PADS ’08)*, 2008, pp. 91–98.

[96] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown, “Algorithm runtime prediction: Methods & evaluation,” *Artificial Intelligence*, vol. 206, pp. 79–111, Jan. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370213001082

[97] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, “SATzilla: Portfolio-based Algorithm Selection for SAT,” *Journal of Artificial Intelligence Research*, vol. 32, pp. 565–606, Jun. 2008.

[98] E. O’Mahony, E. Hebrard, and A. Holland, “Using case-based reasoning in an algorithm portfolio for constraint solving,” in *19th Irish Conference on AI*, 2008. [Online]. Available: http://4c.ucc.ie/~aholland/publications/cpHydra.pdf

[99] E. Fink, “How to Solve It Automatically: Selection Among Problem-Solving Methods,” in *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, Pittsburgh, 1998, pp. 128–136. [Online]. Available: http://www.aaai.org/Papers/AIPS/1998/AIPS98-016.pdf

[100] C. P. Gomes and B. Selman, “Algorithm portfolios,” *Artificial Intelligence*, vol. 126, no. 1–2, pp. 43–62, Feb. 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370200000813

[101] S. Kadioglu, Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann, “Algorithm selection and scheduling,” in *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming*, ser. CP’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 454–469. [Online]. Available: http://dl.acm.org/citation.cfm?id=2041160.2041198

[102] L. Kotthoff, “Hybrid Regression-Classification Models for Algorithm Selection,” in *20th European Conference on Artificial Intelligence*, Montpellier, France, 2012. [Online]. Available: http://4c.ucc.ie/~larsko/papers/stacking-crc.pdfhttps://www.haiti.cs.uni-potsdam.de/proceedings/ECAI2012/content/ecai/ecai2012083.pdf

[103] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier, “Meta-learning by landmarking various learning algorithms,” in *Proceedings of the 17th International Conference on Machine Learning, ICML’2000.* Bristol, UK: University of Bristol, 2000.

[104] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: http://dx.doi.org/10.1038/nature1453910.1038/nature14539

[105] M. Matijaš, J. A. K. Suykens, and S. Krajcar, “Load forecasting using a multivariate meta-learning system,” *Expert Systems with Applications*, vol. 40, no. 11, pp. 4427–4437, Sep. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741300078X

[106] M. Graff, R. Peña, A. Medina, and H. J. Escalante, “Wind speed forecasting using a portfolio of forecasters,” *Renewable Energy*, vol. 68, no. 0, pp. 550–559, Aug. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148114001323

[107] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes.* lulu.com, 2011. [Online]. Available: http://www.cleveralgorithms.com/

[108] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: an update,” *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[109] Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm,” in *Thirteenth International Conference on Machine Learning.* San Francisco: Morgan Kaufmann, 1996, pp. 148–156.

[110] Y. Freund and L. Mason, “The alternating decision tree learning algorithm,” in *Proceeding of the Sixteenth International Conference on Machine Learning*, Bled, Slovenia, 1999, pp. 124–133.

[111] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[112] H. Shi, “Best-first decision tree learning,” Master’s thesis, University of Waikato, Hamilton, NZ, 2007.

[113] R. Kohavi, “The Power of Decision Tables,” in *8th European Conference on Machine Learning*. Springer, 1995, pp. 174–189.

[114] M. Hall and E. Frank, “Combining Naive Bayes and Decision Tables,” in *Proceedings of the 21st Florida Artificial Intelligence Society Conference (FLAIRS)*. AAAI press, 2008, pp. 318–319.

[115] N. Landwehr, M. Hall, and E. Frank, “Logistic Model Trees,” *Machine Learning*, vol. 95, no. 1-2, pp. 161–205, 2005.

[116] D. J. C. MacKay, *Introduction to Gaussian processes*, 1998.

[117] G. Hulten, L. Spencer, and P. Domingos, “Mining time-changing data streams,” in *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. ACM Press, 2001, pp. 97–106.

[118] D. Aha and D. Kibler, “Instance-based learning algorithms,” *Machine Learning*, vol. 6, pp. 37–66, 1991.

[119] G. Webb, “Decision Tree Grafting From the All-Tests-But-One Partition.” San Francisco, CA: Morgan Kaufmann, 1999.

[120] W. W. Cohen, “Fast Effective Rule Induction,” in *Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995, pp. 115–123.

[121] J. G. Cleary and L. E. Trigg, “K*: An Instance-based Learner Using an Entropic Distance Measure,” in *12th International Conference on Machine Learning*, 1995, pp. 108–114.

[122] G. Holmes, B. Pfahringer, R. Kirkby, E. Frank, and M. Hall, “Multiclass alternating decision trees,” in *ECML*. Springer, 2001, pp. 161–172.

[123] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*, 1987.

[124] C.-C. Chang and C.-J. Lin, “LIBSVM - A Library for Support Vector Machines,” 2001. [Online]. Available: http://www.csie.ntu.edu.tw/$\delimiter"026E30F$~cjlin/libsvm/

[125] S. le Cessie and J. C. van Houwelingen, “Ridge Estimators in Logistic Regression,” *Applied Statistics*, vol. 41, no. 1, pp. 191–201, 1992.

[126] C. Atkeson, A. Moore, and S. Schaal, “Locally weighted learning,” *AI Review*, 1996.

[127] Y. Wang and I. H. Witten, “Induction of model trees for predicting continuous classes,” in *Poster papers of the 9th European Conference on Machine Learning*. Springer, 1997.

[128] G. Holmes, M. Hall, and E. Frank, “Generating Rule Sets from Model Trees,” in *Twelfth Australian Joint Conference on Artificial Intelligence*. Springer, 1999, pp. 1–12.

[129] G. H. John and P. Langley, “Estimating Continuous Distributions in Bayesian Classifiers,” in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, ser. UAI’95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 338–345. [Online]. Available: http://dl.acm.org/citation.cfm?id=2074158.2074196

[130] R. Kohavi, “Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid,” in *Second International Conference on Knoledge Discovery and Data Mining*, 1996, pp. 202–207.

[131] R. C. Holte, “Very Simple Classification Rules Perform Well on Most Commonly Used Datasets,” *Machine Learning*, vol. 11, no. 1, pp. 63–90, 1993. [Online]. Available: http://dx.doi.org/10.1023/A:1022631118932

[132] Y. Wang and I. H. Witten, “Modeling for optimal probability prediction,” in *Proceedings of the Nineteenth International Conference in Machine Learning*, Sydney, Australia, 2002, pp. 650–657.

[133] E. Frank and I. H. Witten, “Generating Accurate Rule Sets Without Global Optimization,” in *Fifteenth International Conference on Machine Learning*, J. Shavlik, Ed. Morgan Kaufmann, 1998, pp. 144–151.

[134] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[135] E. Frank, “Fully supervised training of Gaussian radial basis function networks in WEKA,” Tech. Rep., 2014.

[136] B. R. Gaines and P. Compton, “Induction of Ripple-down Rules Applied to Modeling Large Databases,” *J. Intell. Inf. Syst.*, vol. 5, no. 3, pp. 211–228, Nov. 1995. [Online]. Available: http://dx.doi.org/10.1007/BF00962234

[137] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, California: Wadsworth International Group, 1984.

[138] M. Sumner, E. Frank, and M. Hall, “Speeding up Logistic Model Tree Induction,” in *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*. Springer, 2005, pp. 675–683.

[139] J. Platt, “Fast Training of Support Vector Machines using Sequential Minimal Optimization,” in *Advances in Kernel Methods - Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola, Eds. MIT Press, 1998. [Online]. Available: http://research.microsoft.com/$\delimiter"026E30F$~jplatt/smo.html

[140] S. K. Shevade, S. S. Keerthi, C. Bhattacharyya, and K. R. K. Murthy, “Improvements to the SMO Algorithm for SVM Regression,” in *IEEE Transactions on Neural Networks*, 1999.

[141] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944968

[142] W. Zhuo, Prabhat, C. Paciorek, C. Kaufman, and W. Bethel, “Parallel Kriging Analysis for Large Spatial Datasets,” in *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, 2011, pp. 38–44.

[143] M. Banko and E. Brill, “Scaling to Very Very Large Corpora for Natural Language Disambiguation,” in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ser. ACL ’01. Stroudsburg, PA, USA: Association for Computational Linguistics, 2001, pp. 26–33. [Online]. Available: http://dx.doi.org/10.3115/1073012.1073017

[144] N. Chawla and K. Bowyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, Jun. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622407.1622416http://arxiv.org/abs/1106.1813

[145] L. Holmstrom and P. Koistinen, “Using additive noise in back-propagation training,” *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 24–38, 1992.

Appendix A

Data model for power system model (`SysModel`)

Table A.1 System data model

Type Name	Parameter Name	Parameter Type	Python Data Type	Comment
`brk`	`branch_list`	Static	`list`	List of cct, tx and dcx that the breaker can trip.
`brk`	`bus`	Static	`str`	Bus (name) that the breaker is associated with.
`brk`	`enabled`	Dynamic	`bool`	Enables control by an algorithm.
`brk`	`name`	ID	`str`	Bus name, must be unique.
`brk`	`trip`	Control	`bool`	Whether the breaker is tripped.
`brk`	`type`	Static	`str`	Unit type tag/comment.
`bus`	`geo_x`	Static	`float`	Geographical longitude.
`bus`	`geo_y`	Static	`float`	Geographical latitude.
`bus`	`name`	ID	`str`	Bus name, must be unique.
`bus`	`type`	Static	`str`	Unit type tag/comment.
`bus`	`volts_deg`	Measurement	`float`	Voltage angle, in degrees.
`bus`	`volts_nom`	Static	`float`	Nominal bus voltage, in kV.
`bus`	`volts_pu`	Measurement	`float`	Voltage magnitude, in pu.
`cct`	`_status`	Internal	`bool`	Circuit status (in/out).
`cct`	`bus_from`	Static	`str`	Bus that circuit/transformer is from.
`cct`	`bus_to`	Static	`str`	Bus that circuit/transformer is to.
`cct`	`elec_b`	Static	`float`	Charging susceptance, in pu.
`cct`	`elec_r`	Static	`float`	Resistance, in pu.
`cct`	`elec_x`	Static	`float`	Reactance, in pu.
`cct`	`name`	ID	`str`	Circuit/transformer name, must be unique.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
cct	p_from	Measurement	float		Real power at ’from’ end towards ’to’ end.
cct	p_to	Measurement	float		Real power at ’to’ end towards ’from’ end.
cct	q_from	Measurement	float		Reactive power at ’from’ end towards ’to’ end.
cct	q_to	Measurement	float		Reactive power at ’to’ end towards ’from’ end.
cct	rating_dynamic	Dynamic	float		Dynamic rating, in MVA.
cct	rating_static	Static	float		Static rating, in MVA.
cct	s_from	Measurement	float		Apparent power magnitude at ’from’ end towards ’to’ end.
cct	s_to	Measurement	float		Apparent power magnitude at ’to’ end towards ’from’ end.
cct	type	Static	str		Unit type tag/comment.
dcx	_p_from	Internal	float		Calculated real power out of ’from’ end.
dcx	_p_from_last	Internal	float		Last real power out of ’from’ end.
dcx	_p_to	Internal	float		Calculated real power out of ’to’ end.
dcx	_p_to_last	Internal	float		Last real power out of ’to’ end.
dcx	_q_from	Internal	float		Calculated reactive power out of ’from’ end.
dcx	_q_from_last	Internal	float		Last reactive power out of ’from’ end.
dcx	_q_from_max	Internal	float		Calculated maximum reactive power out of ’from’ end.
dcx	_q_from_min	Internal	float		Calculated minimum reactive power out of ’from’ end.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
dcx	_q_to	Internal	float		Calculated reactive power out of 'to' end.
dcx	_q_to_last	Internal	float		Last reactive power out of 'to' end.
dcx	_q_to_max	Internal	float		Calculated maximum reactive power out of 'to' end.
dcx	_q_to_min	Internal	float		Calculated minimum reactive power out of 'to' end.
dcx	_status	Internal	bool		Circuit status (in/out).
dcx	_v_from	Internal	float		Calculated voltage at 'from' end, 0 in Q/PF mode.
dcx	_v_to	Internal	float		Calculated voltage at 'to' end, 0 in Q/PF mode.
dcx	bus_from	Static	str		Bus that circuit/transformer is from.
dcx	bus_to	Static	str		Bus that circuit/transformer is to.
dcx	control_at_from	Dynamic	bool		Toggles if 'from' end controls power transfer.
dcx	efficiency	Static	float		Real power transfer efficiency of the DC link.
dcx	enabled	Dynamic	bool		Toggles whether algorithms can have control.
dcx	mode_q_from	Mode	str		Mode selected for reactive power at 'from' end.
dcx	mode_q_from_list	Modelist	list		Modes available for reactive power at 'from' end.
dcx	mode_q_to	Mode	str		Mode selected for reactive power at 'to' end.
dcx	mode_q_to_list	Modelist	list		Modes available for reactive power at 'to' end.
dcx	name	ID	str		Circuit/transformer name, must be unique.
dcx	p_cmd	Control	float		Commanded real power transfer in 'to' direction. Negative values are calculated at the 'to' end towards the 'from' end.
dcx	p_from	Measurement	float		Real power at 'from' end towards 'to' end.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
dcx	p_ramp	Static	float		Maximum real power ramp rate of the DC link.
dcx	p_to	Measurement	float		Real power at ’to’ end towards ’from’ end.
dcx	pf_from_cmd	Control	float		Commanded power factor at ’from’ end.
dcx	pf_from_max	Static	float		Maximum power factor (in pf mode) at ’from’ end.
dcx	pf_from_min	Static	float		Minimum power factor (in pf mode) at ’from’ end.
dcx	pf_to_cmd	Control	float		Commanded power factor at ’to’ end.
dcx	pf_to_max	Static	float		Maximum power factor (in pf mode) at ’to’ end.
dcx	pf_to_min	Static	float		Minimum power factor (in pf mode) at ’to’ end.
dcx	q_from	Measurement	float		Reactive power at ’from’ end towards ’to’ end.
dcx	q_from_cmd	Control	float		Commanded reactive power out of ’from’ end.
dcx	q_from_max	Static	float		Maximum reactive power out of ’from’ end.
dcx	q_from_min	Static	float		Minimum reactive power out of ’from’ end.
dcx	q_from_ramp	Static	float		Maximum reactive power ramp rate at ’from’ end.
dcx	q_to	Measurement	float		Reactive power at ’to’ end towards ’from’ end.
dcx	q_to_cmd	Control	float		Commanded reactive power out of ’to’ end.
dcx	q_to_max	Static	float		Maximum reactive power out of ’to’ end.
dcx	q_to_min	Static	float		Minimum reactive power out of ’to’ end.
dcx	q_to_ramp	Static	float		Maximum reactive power ramp rate at ’to’ end.
dcx	rating_dynamic	Dynamic	float		Dynamic rating, in MW.
dcx	rating_static	Static	float		Static rating, in MW.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
dcx	s_from	Measurement	float		Apparent power magnitude at ’from’ end towards ’to’ end.
dcx	s_from_max	Static	float		Maximum apparent power of ’from’ end.
dcx	s_to	Measurement	float		Apparent power magnitude at ’to’ end towards ’from’ end.
dcx	s_to_max	Static	float		Maximum apparent power of ’to’ end.
dcx	type	Static	str		Unit type tag/comment.
dcx	v_from_cmd	Control	float		Commanded voltage at ’from’ end.
dcx	v_from_max	Static	float		Maximum voltage at ’from’ end.
dcx	v_from_min	Static	float		Minimum voltage at ’from’ end.
dcx	v_to_cmd	Control	float		Commanded voltage at ’to’ end.
dcx	v_to_max	Static	float		Maximum voltage at ’to’ end.
dcx	v_to_min	Static	float		Minimum voltage at ’to’ end.
gen	_p	Internal	float		Calculated real power output.
gen	_p_last	Internal	float		Last real power output.
gen	_q	Internal	float		Calculated reactive power output.
gen	_q_last	Internal	float		Last reactive power output.
gen	_q_max	Internal	float		Calculated maximum reactive power output.
gen	_q_min	Internal	float		Calculated minimum reactive power output.
gen	_v	Internal	float		Calculated voltage of unit, 0 in Q/PF mode.
gen	bus	Static	str		Bus (name) that the unit is attached to.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
gen	enabled	Dynamic	bool		Enables control by an algorithm.
gen	mode_p	Mode	str		Mode selected for real power.
gen	mode_p_list	Modelist	list		Modes available for real power.
gen	mode_q	Mode	str		Mode selected for reactive power.
gen	mode_q_list	Modelist	list		Modes available for reactive power.
gen	name	ID	str		Generator name, must be unique.
gen	p	Measurement	float		Measured real power output of unit.
gen	p_adj_dn	Dynamic	float		How much the desired real power can be reduced.
gen	p_adj_up	Dynamic	float		How much the desired real power can be increased.
gen	p_cap_max	Control	float		Algorithm-set maximum real power in CAP mode.
gen	p_cap_min	Control	float		Algorithm-set minimum real power in CAP mode.
gen	p_cmd	Control	float		Algorithm-set real power target in SET mode.
gen	p_des	Dynamic	float		The desired real power output of the unit.
gen	p_max	Static	float		Maximum real power output.
gen	p_min	Static	float		Minimum real power output.
gen	p_ramp	Static	float		Maximum real power ramp rate.
gen	pf_cmd	Control	float		Commanded power factor (export).
gen	pf_max	Static	float		Maximum power factor (in pf mode).
gen	pf_min	Static	float		Minimum power factor (in pf mode).
gen	q	Measurement	float		Measured reactive power output of unit.
gen	q_cmd	Control	float		Commanded reactive power output.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
gen	q_max	Static	float		Maximum reactive power output.
gen	q_min	Static	float		Minimum reactive power output.
gen	q_ramp	Static	float		Maximum reactive power ramp rate.
gen	s_max	Static	float		Maximum apparent power of unit.
gen	type	Static	str		Unit type tag/comment.
gen	v_cmd	Control	float		Commanded voltage at output.
gen	v_max	Static	float		Maximum voltage of output.
gen	v_min	Static	float		Minimum voltage of output.
ld	_p	Internal	float		Calculated real power of load.
ld	_q	Internal	float		Calculated reactive power of load.
ld	bus	Static	str		Bus (name) that the load is attached to.
ld	dsr_p	Dynamic	float		Real power that can be shed through DSR.
ld	dsr_q	Dynamic	float		Reactive power that can be shed through DSR.
ld	enabled	Dynamic	bool		Enables control by an algorithm.
ld	mode_dsr	Mode	str		Mode selected for DSR (OFF or SHED).
ld	mode_dsr_list	Modelist	list		Modes available for DSR (OFF or SHED).
ld	name	ID	str		Load name, must be unique.
ld	p	Measurement	float		Measured real power demand of load.
ld	p_max	Static	float		Maximum real power that a load can take on.
ld	p_now	Dynamic	float		Current real power demanded by load.
ld	q	Measurement	float		Measured reactive power demand of load.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
`ld`	`q_now`	Dynamic	`float`		Current reactive power demanded by load.
`ld`	`type`	Static	`str`		Load type tag/comment.
`qbk`	`_q`	Internal	`float`		Calculated reactive power of bank.
`qbk`	`_q_step_last`	Internal	`int`		Last reactive power step setting.
`qbk`	`bus`	Static	`str`		Bus (name) that the bank is attached to.
`qbk`	`enabled`	Dynamic	`bool`		Enables control by an algorithm.
`qbk`	`name`	ID	`str`		Capacitor/reactor bank name, must be unique.
`qbk`	`p`	Measurement	`float`		Measured real power output of unit.
`qbk`	`q`	Measurement	`float`		Measured reactive power output of unit.
`qbk`	`q_step`	Dynamic	`int`		Current reactive power step setting, 0 is 'off' step.
`qbk`	`q_step_cmd`	Control	`int`		Target number of reactive power steps active.
`qbk`	`q_step_countdown`	Dynamic	`int`		Step switching operation delay remaining.
`qbk`	`q_step_delay`	Static	`int`		Restricts the number of step switching operations: +n delays operations for n intervals, -n restricts operations to n within an interval, 0 is unrestricted.
`qbk`	`q_step_size`	Static	`float`		MVAr of each reactive power step, +ve for reactor.
`qbk`	`q_steps`	Static	`int`		Number of reactive power steps available. This includes the 'off' step so needs to be 2 or more.
`qbk`	`type`	Static	`str`		Load type tag/comment.
`slk`	`bus_name`	ID	`str`		Bus name of slack, must be unique.
`slk`	`p`	Measurement	`float`		Real power import at slack.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
slk	q	Measurement	float		Reactive power import at slack.
slk	volts_deg	Dynamic	float		Voltage angle setpoint of slack, in degrees.
slk	volts_pu	Dynamic	float		Voltage magnitude setpoint of slack, in pu.
sto	_p	Internal	float		Calculated real power output.
sto	_p_last	Internal	float		Last real power output.
sto	_q	Internal	float		Calculated reactive power output.
sto	_q_last	Internal	float		Last reactive power output.
sto	_q_max	Internal	float		Calculated maximum reactive power output.
sto	_q_min	Internal	float		Calculated minimum reactive power output.
sto	_v	Internal	float		Calculated voltage of unit, 0 in Q/PF mode.
sto	bus	Static	str		Bus (name) that the unit is attached to.
sto	capacity	Static	float		Storage capacity, in MW x intervals.
sto	efficiency	Static	float		Charge/discharge efficiency.
sto	enabled	Dynamic	bool		Enables control by an algorithm.
sto	mode_p	Mode	str		Mode selected for real power.
sto	mode_p_list	Modelist	list		Modes available for real power.
sto	mode_q	Mode	str		Mode selected for reactive power.
sto	mode_q_list	Modelist	list		Modes available for reactive power.
sto	name	ID	str		Storage unit name, must be unique.
sto	p	Measurement	float		Measured real power output of unit.
sto	p_adj_dn	Dynamic	float		How much the desired real power can be reduced.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
sto	p_adj_up	Dynamic	float		How much the desired real power can be increased.
sto	p_cap_max	Control	float		Algorithm-set maximum real power in CAP mode.
sto	p_cap_min	Control	float		Algorithm-set minimum real power in CAP mode.
sto	p_cmd	Control	float		Algorithm-set real power target in SET mode.
sto	p_des	Dynamic	float		The desired real power output of the unit.
sto	p_max	Static	float		Maximum real power output.
sto	p_min	Static	float		Minimum real power output.
sto	p_ramp	Static	float		Maximum real power ramp rate.
sto	pf_cmd	Control	float		Commanded power factor (export).
sto	pf_max	Static	float		Maximum power factor (in pf mode).
sto	pf_min	Static	float		Minimum power factor (in pf mode).
sto	q	Measurement	float		Measured reactive power output of unit.
sto	q_cmd	Control	float		Commanded reactive power output.
sto	q_max	Static	float		Maximum reactive power output.
sto	q_min	Static	float		Minimum reactive power output.
sto	q_ramp	Static	float		Maximum reactive power ramp rate.
sto	s_max	Static	float		Maximum apparent power of unit.
sto	soc	Dynamic	float		State of charge.
sto	type	Static	str		Unit type tag/comment.
sto	v_cmd	Control	float		Commanded voltage at output.
sto	v_max	Static	float		Maximum voltage of output.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
sto	v_min	Static	float		Minimum voltage of output.
sys	comments	Static	str		Comments about the network.
sys	name	ID	str		System name.
sys	version	Static	str		Network/system version.
tx	_shift_angle	Internal	float		Calculated phase shift angle.
tx	_shift_last	Internal	int		Last phase shift tap value.
tx	_status	Internal	bool		Circuit status (in/out).
tx	_tap_last	Internal	int		Last tap value.
tx	_tap_ratio	Internal	float		Calculated transformer ratio.
tx	bus_from	Static	str		Bus that circuit/transformer is from.
tx	bus_to	Static	str		Bus that circuit/transformer is to.
tx	control_at_from	Dynamic	bool		Toggles whether ’from’ end controls the tx.
tx	elec_b	Static	float		Charging susceptance, in pu.
tx	elec_r	Static	float		Resistance, in pu.
tx	elec_x	Static	float		Reactance, in pu.
tx	enabled	Dynamic	bool		Toggles whether an algorithm can control the tx.
tx	name	ID	str		Circuit/transformer name, must be unique.
tx	p_from	Measurement	float		Real power at ’from’ end towards ’to’ end.
tx	p_to	Measurement	float		Real power at ’to’ end towards ’from’ end.
tx	q_from	Measurement	float		Reactive power at ’from’ end towards ’to’ end.
tx	q_to	Measurement	float		Reactive power at ’to’ end towards ’from’ end.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
tx	rating_dynamic	Dynamic	float		Dynamic rating, in MVA.
tx	rating_static	Static	float		Static rating, in MVA.
tx	s_from	Measurement	float		Apparent power magnitude at ’from’ end towards ’to’ end.
tx	s_to	Measurement	float		Apparent power magnitude at ’to’ end towards ’from’ end.
tx	shift	Dynamic	int		Current phase shift tap value.
tx	shift_cmd	Control	int		Current shift tap setting command, starts at 0.
tx	shift_countdown	Dynamic	int		Shift operation delay remaining.
tx	shift_delay	Static	int		Restricts the number of shift tapping operations: +n delays operations for n intervals, -n restricts operations to n within an interval, 0 is unrestricted.
tx	shift_max	Static	float		Maximum phase shift, in degrees.
tx	shift_min	Static	float		Minimum phase shift, in degrees.
tx	shift_steps	Static	int		Number of phase shift tap steps, 1 as a minimum.
tx	tap	Dynamic	int		Current tap value.
tx	tap_cmd	Control	int		Current tap setting command, starts at 0.
tx	tap_countdown	Dynamic	int		Tap operation delay remaining.
tx	tap_delay	Static	int		Restricts the number of tap operations: +n delays operations for n intervals, -n restricts operations to n within an interval, 0 is unrestricted.

(continued on next page)

Table A.1 (continued from previous page)

Type Name	Parameter Name	Parameter Type	Python Type	Data	Comment
`tx`	`tap_max`	Static	`float`		Maximum tap, % away from nominal.
`tx`	`tap_min`	Static	`float`		Minimum tap, % away from nominal.
`tx`	`tap_steps`	Static	`int`		Number of tap steps, 1 as a minimum.
`tx`	`type`	Static	`str`		Unit type tag/comment.
`zn`	`buses`	Dynamic	`list`		List of buses within zone.
`zn`	`id`	ID	`str`		Zone id, must be unique.

Appendix B

Case study system data

B.1 11 kV radial distribution system

Table B.1 Bus data for 11 kV radial distribution system

Name	Nominal voltage [kV]
Bus 1	33.000
Bus 10	11.000
Bus 2	11.000
Bus 3	11.000
Bus 4	11.000
Bus 5	11.000
Bus 6	11.000
Bus 7	11.000
Bus 8	11.000
Bus 9	11.000

Table B.2 Circuit data for 11 kV radial distribution system

Name	R [pu]	X [pu]	B [pu]	Rating [MVA]
Circuit 2-3	0.17385	0.08455	0.00000	7.62
Circuit 3-4	0.00097	0.00052	0.00000	7.43
Circuit 4-5	0.05190	0.02782	0.00000	7.43
Circuit 5-6	0.00339	0.00165	0.00000	2.00
Circuit 6-7	0.03916	0.01904	0.00000	7.62

(continued on next page)

Table B.2 (continued from previous page)

Name	R [pu]	X [pu]	B [pu]	Rating [MVA]
Circuit 7-8	0.00068	0.00033	0.00000	7.62
Circuit 8-9	0.01612	0.00784	0.00000	1.50
Circuit 9-10	0.29930	0.08139	0.00000	5.14

Table B.3 Transformer data for 11 kV radial distribution system

Name	R [pu]	X [pu]	Rating [MVA]	Tap min	Tap max	Tap steps
Transformer 1-2	0.00000	1.00000	16.00	-10.00%	10.00%	21

Table B.4 Load data for 11 kV radial distribution system

Name	Bus	P [MW]	Q [MVAr]
Load 3	Bus 3	0.28	0.14
Load 4	Bus 4	0.52	0.25
Load 5	Bus 5	0.36	0.17
Load 7	Bus 7	0.81	0.39
Load 8	Bus 8	0.81	0.39
Load 9	Bus 9	0.38	0.20

Table B.5 Generator data for 11 kV radial distribution system

Name	Bus	P max [MW]	Q min [MVAr]	Q max [MVAr]	Voltage [pu]
Generator 10	Bus 10	2.00	0.00	0.00	1.00000
Generator 6	Bus 6	1.60	0.00	0.00	1.00000

B.2 33 kV meshed distribution system

Table B.6 Bus data for 33 kV meshed distribution system

Name	Nominal voltage [kV]
Bus 1	275.000
Bus 10	132.000
Bus 11	132.000

(continued on next page)

Table B.6 (continued from previous page)

Name	Nominal voltage [kV]
Bus 12	33.000
Bus 13	33.000
Bus 14	33.000
Bus 15	11.000
Bus 16	11.000
Bus 17	0.690
Bus 18	33.000
Bus 19	33.000
Bus 2	132.000
Bus 20	33.000
Bus 21	33.000
Bus 22	33.000
Bus 23	33.000
Bus 24	33.000
Bus 25	33.000
Bus 26	33.000
Bus 27	33.000
Bus 3	132.000
Bus 4	132.000
Bus 5	132.000
Bus 6	132.000
Bus 7	132.000
Bus 8	132.000
Bus 9	132.000

Table B.7 Circuit data for 33 kV meshed distribution system

Name	R [pu]	X [pu]	B [pu]	Rating [MVA]
Circuit 10-11	0.01080	0.02484	0.00660	88.80
Circuit 13-14	0.24530	0.48450	0.00000	7.00
Circuit 13-19	0.21940	0.43340	0.00000	12.00
Circuit 14-18	0.11644	0.19593	0.00302	16.40
Circuit 16-15	0.21548	0.09862	0.00000	100.00
Circuit 19-18	0.03949	0.04220	0.00331	16.80

(continued on next page)

Table B.7 (continued from previous page)

Name	R [pu]	X [pu]	B [pu]	Rating [MVA]
Circuit 19-20	0.01470	0.02390	0.00000	16.80
Circuit 19-21	0.01120	0.01730	0.00000	16.48
Circuit 19-22	0.13620	0.18800	0.00000	17.60
Circuit 19-25	0.24960	0.47190	0.00000	16.48
Circuit 2-3	0.00918	0.03197	0.01373	80.00
Circuit 2-4	0.02225	0.09060	0.03959	80.00
Circuit 22-13	0.20291	0.39960	0.00884	10.00
Circuit 23-22	0.34455	0.66949	0.00000	16.48
Circuit 23-24	0.25578	0.50288	0.00000	17.20
Circuit 24-25	0.30772	0.60641	0.00000	13.44
Circuit 25-26	0.08620	0.16440	0.00000	16.48
Circuit 27-26	0.24060	0.47470	0.00000	16.88
Circuit 3-4	0.01465	0.06606	0.02671	80.00
Circuit 4-5 (1)	0.00959	0.04221	0.01556	80.00
Circuit 4-5 (2)	0.01710	0.04541	0.00893	80.00
Circuit 5-6	0.02064	0.04875	0.01426	71.20
Circuit 5-8	0.02340	0.05528	0.01093	80.00
Circuit 6-7	0.03499	0.08265	0.01634	88.80
Circuit 8-10	0.04407	0.10408	0.02058	80.00
Circuit 8-9	0.00102	0.00241	0.00048	80.00

Table B.8 Transformer data for 33 kV meshed distribution system

Name	R [pu]	X [pu]	Rating [MVA]	Tap min	Tap max	Tap steps
Transformer 1-2 (1)	0.00167	0.08333	384.00	-15.00%	15.00%	19
Transformer 1-2 (2)	0.00167	0.08333	384.00	-15.00%	15.00%	19
Transformer 1-2 (3)	0.00167	0.08333	100.00	-15.00%	15.00%	19
Transformer 1-2 (4)	0.00167	0.08333	192.00	-15.00%	15.00%	19
Transformer 10-13	0.01430	0.28266	60.00	-20.00%	10.00%	19
Transformer 11-12	0.01083	0.25000	72.00	-20.00%	10.00%	19
Transformer 14-15	0.13330	1.33333	100.00	-10.00%	10.00%	17
Transformer 17-16	0.08330	0.83350	100.00	0.00%	0.00%	1
Transformer 7-19	0.01593	0.27760	60.00	-20.00%	10.00%	19

Table B.9 Load data for 33 kV meshed distribution system

Name	Bus	P [MW]	Q [MVAr]
Load 14	Bus 14	3.59	0.51
Load 18	Bus 18	7.98	7.97
Load 19	Bus 19	7.01	1.09
Load 20	Bus 20	4.88	1.20
Load 21	Bus 21	7.21	2.23
Load 22	Bus 22	9.62	0.97
Load 24	Bus 24	3.56	0.94
Load 25	Bus 25	4.09	0.97
Load 26	Bus 26	4.96	0.64
Load 27	Bus 27	4.41	0.43

Table B.10 Generator data for 33 kV meshed distribution system

Name	Bus	P max [MW]	Q min [MVAr]	Q max [MVAr]	Voltage [pu]
Generator 12	Bus 12	58.00	0.00	0.00	1.00000
Generator 13	Bus 13	44.40	0.00	0.00	1.00000
Generator 17	Bus 17	9.30	0.00	0.00	1.00000
Generator 23	Bus 23	10.20	0.00	0.00	1.00000

B.3 IEEE 14-bus system

Table B.11 Bus data for IEEE 14-bus system

Name	Nominal voltage [kV]
Bus 1	1.000
Bus 10	1.000
Bus 11	1.000
Bus 12	1.000
Bus 13	1.000
Bus 14	1.000
Bus 2	1.000
Bus 3	1.000

(continued on next page)

Table B.11 (continued from previous page)

Name	Nominal voltage [kV]
Bus 4	1.000
Bus 5	1.000
Bus 6	1.000
Bus 7	1.000
Bus 8	1.000
Bus 9	1.000

Table B.12 Circuit data for IEEE 14-bus system

Name	R [pu]	X [pu]	B [pu]	Rating [MVA]
Circuit 1-2	0.01938	0.05917	0.05280	250.00
Circuit 1-5	0.05403	0.22304	0.04920	100.00
Circuit 10-11	0.08205	0.19207	0.00000	10.00
Circuit 12-13	0.22092	0.19988	0.00000	10.00
Circuit 13-14	0.17093	0.34802	0.00000	10.00
Circuit 2-3	0.04699	0.19797	0.04380	100.00
Circuit 2-4	0.05811	0.17632	0.03400	75.00
Circuit 2-5	0.05695	0.17388	0.03460	50.00
Circuit 3-4	0.06701	0.17103	0.01280	30.00
Circuit 4-5	0.01335	0.04211	0.00000	75.00
Circuit 6-11	0.09498	0.19890	0.00000	10.00
Circuit 6-12	0.12291	0.25581	0.00000	15.00
Circuit 6-13	0.06615	0.13027	0.00000	25.00
Circuit 9-10	0.03181	0.08450	0.00000	10.00
Circuit 9-14	0.12711	0.27038	0.00000	15.00

Table B.13 Transformer data for IEEE 14-bus system

Name	R [pu]	X [pu]	Rating [MVA]	Tap min	Tap max	Tap steps
Transformer 4-7	0.00000	0.20912	50.00	-2.20%	-2.20%	1
Transformer 4-9	0.00000	0.55618	25.00	-3.10%	-3.10%	1
Transformer 5-6	0.00000	0.25202	75.00	-6.80%	-6.80%	1
Transformer 7-8	0.00000	0.17615	25.00	0.00%	0.00%	1

(continued on next page)

Table B.13 (continued from previous page)

Name	R [pu]	X [pu]	Rating [MVA]	Tap min	Tap max	Tap steps
Transformer 7-9	0.00000	0.11001	50.00	0.00%	0.00%	1

Table B.14 Load data for IEEE 14-bus system

Name	Bus	P [MW]	Q [MVAr]
Load 10	Bus 10	9.00	5.80
Load 11	Bus 11	3.50	1.80
Load 12	Bus 12	6.10	1.60
Load 13	Bus 13	13.50	5.80
Load 14	Bus 14	14.90	5.00
Load 2	Bus 2	21.70	12.70
Load 3	Bus 3	94.20	19.00
Load 4	Bus 4	47.80	-3.90
Load 5	Bus 5	7.60	1.60
Load 6	Bus 6	11.20	7.50
Load 9	Bus 9	29.50	16.60

Table B.15 Generator data for IEEE 14-bus system

Name	Bus	P max [MW]	Q min [MVAr]	Q max [MVAr]	Voltage [pu]
Generator 10	Bus 10	10.00	-5.00	5.00	1.00000
Generator 11	Bus 11	20.00	-10.00	10.00	1.00000
Generator 12	Bus 12	20.00	-10.00	10.00	1.00000
Generator 14	Bus 14	30.00	-15.00	15.00	1.00000

B.4 IEEE 57-bus system

Table B.16 Bus data for IEEE 57-bus system

Name	Nominal voltage [kV]
Bus 1	1.000
Bus 10	1.000
Bus 11	1.000

(continued on next page)

Table B.16 (continued from previous page)

Name	Nominal voltage [kV]
Bus 12	1.000
Bus 13	1.000
Bus 14	1.000
Bus 15	1.000
Bus 16	1.000
Bus 17	1.000
Bus 18	1.000
Bus 19	1.000
Bus 2	1.000
Bus 20	1.000
Bus 21	1.000
Bus 22	1.000
Bus 23	1.000
Bus 24	1.000
Bus 25	1.000
Bus 26	1.000
Bus 27	1.000
Bus 28	1.000
Bus 29	1.000
Bus 3	1.000
Bus 30	1.000
Bus 31	1.000
Bus 32	1.000
Bus 33	1.000
Bus 34	1.000
Bus 35	1.000
Bus 36	1.000
Bus 37	1.000
Bus 38	1.000
Bus 39	1.000
Bus 4	1.000
Bus 40	1.000
Bus 41	1.000

(continued on next page)

Table B.16 (continued from previous page)

Name	Nominal voltage [kV]
Bus 42	1.000
Bus 43	1.000
Bus 44	1.000
Bus 45	1.000
Bus 46	1.000
Bus 47	1.000
Bus 48	1.000
Bus 49	1.000
Bus 5	1.000
Bus 50	1.000
Bus 51	1.000
Bus 52	1.000
Bus 53	1.000
Bus 54	1.000
Bus 55	1.000
Bus 56	1.000
Bus 57	1.000
Bus 6	1.000
Bus 7	1.000
Bus 8	1.000
Bus 9	1.000

Table B.17 Circuit data for IEEE 57-bus system

Name	R [pu]	X [pu]	B [pu]	Rating [MVA]
Circuit 1-15	0.01780	0.09100	0.09880	580.00
Circuit 1-16	0.04540	0.20600	0.05460	300.00
Circuit 1-17	0.02380	0.10800	0.02860	320.00
Circuit 1-2	0.00830	0.02800	0.12900	440.00
Circuit 10-12	0.02770	0.12620	0.03280	60.00
Circuit 11-13	0.02230	0.07320	0.01880	120.00
Circuit 12-13	0.01780	0.05800	0.06040	160.00
Circuit 12-16	0.01800	0.08130	0.02160	250.00
Circuit 12-17	0.03970	0.17900	0.04760	270.00

(continued on next page)

Table B.17 (continued from previous page)

Name	R [pu]	X [pu]	B [pu]	Rating [MVA]
Circuit 13-14	0.01320	0.04340	0.01100	170.00
Circuit 13-15	0.02690	0.08690	0.02300	230.00
Circuit 14-15	0.01710	0.05470	0.01480	240.00
Circuit 18-19	0.46100	0.68500	0.00000	7.00
Circuit 19-20	0.28300	0.43400	0.00000	3.00
Circuit 2-3	0.02980	0.08500	0.08180	430.00
Circuit 21-22	0.07360	0.11700	0.00000	10.00
Circuit 22-23	0.00990	0.01520	0.00000	40.00
Circuit 22-38	0.01920	0.02950	0.00000	40.00
Circuit 23-24	0.16600	0.25600	0.00840	40.00
Circuit 25-30	0.13500	0.20200	0.00000	10.00
Circuit 26-27	0.16500	0.25400	0.00000	40.00
Circuit 27-28	0.06180	0.09540	0.00000	25.00
Circuit 28-29	0.04180	0.05870	0.00000	25.00
Circuit 29-52	0.14420	0.18700	0.00000	30.00
Circuit 3-15	0.01620	0.05300	0.05440	70.00
Circuit 3-4	0.01120	0.03660	0.03800	270.00
Circuit 30-31	0.32600	0.49700	0.00000	3.50
Circuit 31-32	0.50700	0.75500	0.00000	10.00
Circuit 32-33	0.03920	0.03600	0.00000	10.00
Circuit 34-35	0.05200	0.07800	0.00320	20.00
Circuit 35-36	0.04300	0.05370	0.00160	20.00
Circuit 36-37	0.02900	0.03660	0.00000	30.00
Circuit 36-40	0.03000	0.04660	0.00000	10.00
Circuit 37-38	0.06510	0.10090	0.00200	40.00
Circuit 37-39	0.02390	0.03790	0.00000	10.00
Circuit 38-44	0.02890	0.05850	0.00200	70.00
Circuit 38-48	0.03120	0.04820	0.00000	40.00
Circuit 38-49	0.11500	0.17700	0.00300	20.00
Circuit 4-5	0.06250	0.13200	0.02580	100.00
Circuit 4-6	0.04300	0.14800	0.03480	140.00
Circuit 41-42	0.20700	0.35200	0.00000	10.00
Circuit 44-45	0.06240	0.12420	0.00400	90.00

(continued on next page)

Table B.17 (continued from previous page)

Name	R [pu]	X [pu]	B [pu]	Rating [MVA]
Circuit 46-47	0.02300	0.06800	0.00320	80.00
Circuit 47-48	0.01820	0.02330	0.00000	40.00
Circuit 48-49	0.08340	0.12900	0.00480	30.00
Circuit 49-50	0.08010	0.12800	0.00000	40.00
Circuit 5-6	0.03020	0.06410	0.01240	90.00
Circuit 50-51	0.13860	0.22000	0.00000	25.00
Circuit 52-53	0.07620	0.09840	0.00000	15.00
Circuit 53-54	0.18780	0.23200	0.00000	20.00
Circuit 54-55	0.17320	0.22650	0.00000	30.00
Circuit 56-41	0.55300	0.54900	0.00000	5.00
Circuit 56-42	0.21250	0.35400	0.00000	10.00
Circuit 57-56	0.17400	0.26000	0.00000	3.00
Circuit 6-7	0.02000	0.10200	0.02760	80.00
Circuit 6-8	0.03390	0.17300	0.04700	70.00
Circuit 7-8	0.01390	0.07120	0.01940	70.00
Circuit 8-9	0.00990	0.05050	0.05480	150.00
Circuit 9-10	0.03690	0.16790	0.04400	40.00
Circuit 9-11	0.02580	0.08480	0.02180	95.00
Circuit 9-12	0.06480	0.29500	0.07720	40.00
Circuit 9-13	0.04810	0.15800	0.04060	100.00

Table B.18 Transformer data for IEEE 57-bus system

Name	R [pu]	X [pu]	Rating [MVA]	Tap min	Tap max	Tap steps
Transformer 10-51	0.00000	0.07120	50.00	-7.00%	-7.00%	1
Transformer 11-41	0.00000	0.74900	20.00	-4.50%	-4.50%	1
Transformer 11-43	0.00000	0.15300	20.00	-4.20%	-4.20%	1
Transformer 13-49	0.00000	0.19100	60.00	-10.50%	-10.50%	1
Transformer 14-46	0.00000	0.07350	80.00	-10.00%	-10.00%	1
Transformer 15-45	0.00000	0.10420	90.00	-4.50%	-4.50%	1
Transformer 21-20	0.00000	0.77670	10.00	4.30%	4.30%	1
Transformer 24-25 (1)	0.00000	1.18200	10.00	0.00%	0.00%	1
Transformer 24-25 (2)	0.00000	1.23000	10.00	0.00%	0.00%	1
Transformer 24-26	0.00000	0.04730	40.00	4.30%	4.30%	1

(continued on next page)

Table B.18 (continued from previous page)

Name	R [pu]	X [pu]	Rating [MVA]	Tap min	Tap max	Tap steps
Transformer 34-32	0.00000	0.95300	20.00	-2.50%	-2.50%	1
Transformer 39-57	0.00000	1.35500	10.00	-2.00%	-2.00%	1
Transformer 4-18 (1)	0.00000	0.55500	20.00	-3.00%	-3.00%	1
Transformer 4-18 (2)	0.00000	0.43000	30.00	-2.20%	-2.20%	1
Transformer 40-56	0.00000	1.19500	10.00	-4.20%	-4.20%	1
Transformer 41-43	0.00000	0.04730	20.00	0.00%	0.00%	1
Transformer 7-29	0.00000	0.06480	80.00	-3.30%	-3.30%	1
Transformer 9-55	0.00000	0.12050	40.00	-6.00%	-6.00%	1

Table B.19 Load data for IEEE 57-bus system

Name	Bus	P [MW]	Q [MVAr]
Load 10	Bus 10	5.00	2.00
Load 12	Bus 12	377.00	24.00
Load 13	Bus 13	18.00	2.30
Load 14	Bus 14	10.50	5.30
Load 15	Bus 15	22.00	5.00
Load 16	Bus 16	43.00	3.00
Load 17	Bus 17	42.00	8.00
Load 18	Bus 18	27.20	9.80
Load 19	Bus 19	3.30	0.60
Load 2	Bus 2	3.00	88.00
Load 20	Bus 20	2.30	1.00
Load 23	Bus 23	6.30	2.10
Load 25	Bus 25	6.30	3.20
Load 27	Bus 27	9.30	0.50
Load 28	Bus 28	4.60	2.30
Load 29	Bus 29	17.00	2.60
Load 3	Bus 3	41.00	21.00
Load 30	Bus 30	3.60	1.80
Load 31	Bus 31	5.80	2.90
Load 32	Bus 32	1.60	0.80
Load 33	Bus 33	3.80	1.90
Load 35	Bus 35	6.00	3.00

(continued on next page)

Table B.19 (continued from previous page)

Name	Bus	P [MW]	Q [MVAr]
Load 38	Bus 38	14.00	7.00
Load 41	Bus 41	6.30	3.00
Load 42	Bus 42	7.10	4.40
Load 43	Bus 43	2.00	1.00
Load 44	Bus 44	12.00	1.80
Load 47	Bus 47	29.70	11.60
Load 49	Bus 49	18.00	8.50
Load 5	Bus 5	13.00	4.00
Load 50	Bus 50	21.00	10.50
Load 51	Bus 51	18.00	5.30
Load 52	Bus 52	4.90	2.20
Load 53	Bus 53	20.00	10.00
Load 54	Bus 54	4.10	1.40
Load 55	Bus 55	6.80	3.40
Load 56	Bus 56	7.60	2.20
Load 57	Bus 57	6.70	2.00
Load 6	Bus 6	75.00	2.00
Load 8	Bus 8	150.00	22.00
Load 9	Bus 9	121.00	26.00

Table B.20 Generator data for IEEE 57-bus system

Name	Bus	P max [MW]	Q min [MVAr]	Q max [MVAr]	Voltage [pu]
Generator 12	Bus 12	410.00	-500.00	500.00	1.01500
Generator 2	Bus 2	0.00	-500.00	500.00	1.01000
Generator 3	Bus 3	140.00	-500.00	500.00	0.98500
Generator 6	Bus 6	0.00	-500.00	500.00	0.98000
Generator 8	Bus 8	550.00	-500.00	500.00	1.00500
Generator 9	Bus 9	0.00	-500.00	500.00	0.98000

Appendix C

Additional power flow management algorithm performance data

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	3344	0	2	0	2	0	0
E T L	0	0	0	0	0	0	0	0
E **T** L	0	0	0	0	0	0	0	0
E T **L**	0	0	0	0	0	0	0	0
E **T** L	0	0	0	0	0	0	0	0
E T **L**	0	0	0	0	0	0	0	0
E **T** **L**	0	0	0	0	0	0	0	0
E **T** **L**	0	0	0	0	0	0	0	0

(a) Number of overloads [count]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	871.95	0.00	0.10	0.00	0.10	0.00	0.00
E T L	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E **T** L	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E T **L**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E **T** L	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E T **L**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E **T** **L**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E **T** **L**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

(b) Magnitude of overloads [MVAh]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	0.00	2121.40	769.94	2121.40	769.94	771.46	771.46
E T L	856.20	855.40	855.40	770.08	855.40	770.08	770.08	770.08
E **T** L	891.01	890.34	882.49	770.08	882.49	770.08	770.08	770.08
E T **L**	773.33	772.66	772.66	770.08	772.66	770.08	770.08	770.08
E **T** L	776.81	776.14	776.14	770.08	776.14	770.08	770.08	770.08
E T **L**	773.33	772.66	772.66	770.08	772.66	770.08	770.08	770.08
E **T** **L**	773.29	772.62	772.62	770.08	772.62	770.08	770.08	770.08
E **T** **L**	773.29	772.62	772.62	770.08	772.62	770.08	770.08	770.08

(c) Total curtailment applied [MWh]

Fig. C.1 Comparison of the performance of oracle 1 for the 11 kV radial distribution system with different algorithm combinations considered for selection

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	3344	0	2	0	2	0	0
E T L	0	0	0	0	0	0	0	0
E **T** L	0	0	0	0	0	0	0	0
E T **L**	0	0	0	0	0	0	0	0
E **T** L	0	0	0	0	0	0	0	0
E T **L**	0	0	0	0	0	0	0	0
E **T** **L**	0	0	0	0	0	0	0	0
E **T** **L**	0	0	0	0	0	0	0	0

(a) Number of overloads [count]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	871.95	0.00	0.10	0.00	0.10	0.00	0.00
E T L	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E **T** L	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E T **L**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E **T** L	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E T **L**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E **T** **L**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E **T** **L**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

(b) Magnitude of overloads [MVAh]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	0.00	2121.40	769.94	2121.40	769.94	771.46	771.46
E T L	856.20	855.40	855.40	770.08	855.40	770.08	770.08	770.08
E **T** L	891.01	890.34	882.49	770.08	882.49	770.08	770.08	770.08
E T **L**	773.33	772.66	772.66	770.08	772.66	770.08	770.08	770.08
E **T** L	776.81	776.14	776.14	770.08	776.14	770.08	770.08	770.08
E T **L**	773.33	772.66	772.66	770.08	772.66	770.08	770.08	770.08
E **T** **L**	773.29	772.62	772.62	770.08	772.62	770.08	770.08	770.08
E **T** **L**	773.29	772.62	772.62	770.08	772.62	770.08	770.08	770.08

(c) Total curtailment applied [MWh]

Fig. C.2 Comparison of the performance of oracle 2 for the 11 kV radial distribution system with different algorithm combinations considered for selection

			B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E	T	L	-	14779	194	0	194	0	0	0
E	T	L	3594	3594	42	0	42	0	0	0
E	**T**	L	1903	1903	13	0	13	0	0	0
E	T	**L**	1819	1819	13	0	13	0	0	0
E	**T**	L	1903	1903	13	0	13	0	0	0
E	T	**L**	1819	1819	13	0	13	0	0	0
E	**T**	**L**	1819	1819	13	0	13	0	0	0
E	**T**	**L**	1819	1819	13	0	13	0	0	0

(a) Number of overloads [count]

			B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E	T	L	-	9445.74	10.56	0.00	12.73	0.00	0.00	0.00
E	T	L	78.25	224.40	1.44	0.00	3.59	0.00	0.00	0.00
E	**T**	L	40.41	119.24	0.73	0.00	2.33	0.00	0.00	0.00
E	T	**L**	36.48	115.31	0.73	0.00	2.33	0.00	0.00	0.00
E	**T**	L	40.41	119.24	0.73	0.00	2.33	0.00	0.00	0.00
E	T	**L**	37.52	116.34	0.73	0.00	2.33	0.00	0.00	0.00
E	**T**	**L**	38.31	117.14	0.73	0.00	2.33	0.00	0.00	0.00
E	**T**	**L**	38.31	117.14	0.73	0.00	2.33	0.00	0.00	0.00

(b) Magnitude of overloads [MVAh]

			B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E	T	L	-	0.00	76910.75	40246.45	76894.78	40246.45	40230.24	40230.24
E	T	L	59905.56	57677.23	65971.97	40166.77	65956.11	40166.77	40150.83	40150.83
E	**T**	L	39694.49	39098.60	53549.92	39908.24	53538.15	39908.24	39893.69	39893.69
E	T	**L**	39743.66	39147.76	53214.32	39895.97	53202.56	39895.97	39881.42	39881.42
E	**T**	L	39694.49	39098.60	53549.92	39908.24	53538.15	39908.24	39893.69	39893.69
E	T	**L**	39737.32	39141.42	53214.32	39895.97	53202.55	39895.97	39881.42	39881.42
E	**T**	**L**	39727.48	39131.59	53211.91	39893.60	53200.14	39893.60	39879.06	39879.06
E	**T**	**L**	39727.48	39131.59	53211.91	39893.60	53200.14	39893.60	39879.06	39879.06

(c) Total curtailment applied [MWh]

Fig. C.3 Comparison of the performance of oracle 1 for the 33 kV meshed distribution system with different algorithm combinations considered for selection

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	14779	194	0	194	0	0	0
E T L	3594	3594	42	0	42	0	0	0
E **T** L	1903	1903	13	0	13	0	0	0
E T **L**	1819	1819	13	0	13	0	0	0
E **T** L	1903	1903	13	0	13	0	0	0
E T **L**	1819	1819	13	0	13	0	0	0
E **T** **L**	1819	1819	13	0	13	0	0	0
E **T** **L**	1819	1819	13	0	13	0	0	0

(a) Number of overloads [count]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	9445.74	10.56	0.00	10.56	0.00	0.00	0.00
E T L	78.25	78.25	0.47	0.00	0.47	0.00	0.00	0.00
E **T** L	40.41	40.41	0.11	0.00	0.11	0.00	0.00	0.00
E T **L**	36.48	36.48	0.11	0.00	0.11	0.00	0.00	0.00
E **T** L	40.37	40.37	0.11	0.00	0.11	0.00	0.00	0.00
E T **L**	36.47	36.47	0.11	0.00	0.11	0.00	0.00	0.00
E **T** **L**	36.48	36.48	0.11	0.00	0.11	0.00	0.00	0.00
E **T** **L**	36.47	36.47	0.11	0.00	0.11	0.00	0.00	0.00

(b) Magnitude of overloads [MVAh]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	0.00	76910.75	40246.45	76910.75	40246.45	40230.24	40230.24
E T L	59905.56	59892.74	66096.69	40166.77	66096.69	40166.77	40150.83	40150.83
E **T** L	39694.49	39686.60	53554.47	39908.24	53554.47	39908.24	39893.69	39893.69
E T **L**	39743.66	39735.77	53218.87	39895.97	53218.87	39895.97	39881.42	39881.42
E **T** L	39802.81	39794.92	53554.47	39908.24	53554.47	39908.24	39893.69	39893.69
E T **L**	39744.51	39736.62	53218.87	39895.97	53218.87	39895.97	39881.42	39881.42
E **T** **L**	39741.25	39733.36	53216.46	39893.60	53216.46	39893.60	39879.06	39879.06
E **T** **L**	39742.10	39734.21	53216.46	39893.60	53216.46	39893.60	39879.06	39879.06

(c) Total curtailment applied [MWh]

Fig. C.4 Comparison of the performance of oracle 2 for the 33 kV meshed distribution system with different algorithm combinations considered for selection

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	5345	45	24	45	24	0	0
E T L	66	66	0	0	0	0	0	0
E **T** L	60	60	0	1	0	1	0	0
E T **L**	1172	1169	16	1	16	1	0	0
E **T** L	42	42	0	0	0	0	0	0
E T **L**	46	46	0	0	0	0	0	0
E **T** **L**	45	45	0	0	0	0	0	0
E **T** **L**	42	42	0	0	0	0	0	0

(a) Number of overloads [count]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	9116.25	11.01	49.97	92.60	49.97	0.00	0.00
E T L	1.75	6.17	0.00	0.00	0.00	0.00	0.00	0.00
E **T** L	1.57	8.54	0.00	0.01	0.00	0.01	0.00	0.00
E T **L**	105.55	2175.08	0.90	0.01	29.51	0.01	0.00	0.00
E **T** L	1.21	4.37	0.00	0.00	0.00	0.00	0.00	0.00
E T **L**	1.30	4.75	0.00	0.00	0.00	0.00	0.00	0.00
E **T** **L**	1.29	4.46	0.00	0.00	0.00	0.00	0.00	0.00
E **T** **L**	1.21	4.37	0.00	0.00	0.00	0.00	0.00	0.00

(b) Magnitude of overloads [MVAh]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	0.00	58855.10	19544.04	58598.26	19544.04	19863.94	19863.94
E T L	33900.57	33855.96	33156.06	19652.35	33156.06	19652.35	19645.63	19645.63
E **T** L	21797.45	21735.54	22204.86	19544.02	22204.86	19544.02	19529.00	19529.00
E T **L**	19503.16	16188.67	32475.44	19471.93	32429.97	19471.93	19466.91	19466.91
E **T** L	21150.71	21124.05	21582.88	19445.80	21582.88	19445.80	19439.74	19439.74
E T **L**	24424.79	24396.11	24509.66	19416.65	24509.66	19416.65	19410.59	19410.59
E **T** **L**	20655.58	20629.02	21106.55	19372.05	21106.55	19372.05	19363.73	19363.73
E **T** **L**	20388.98	20362.42	20849.00	19317.46	20849.00	19317.46	19311.40	19311.40

(c) Total curtailment applied [MWh]

Fig. C.5 Comparison of the performance of oracle 1 for the IEEE 14-bus system with different algorithm combinations considered for selection

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	5345	45	24	45	24	0	0
E T L	66	66	0	0	0	0	0	0
E **T** L	60	60	0	1	0	1	0	0
E T **L**	1172	1172	16	1	16	1	0	0
E **T** L	42	42	0	0	0	0	0	0
E T **L**	46	46	0	0	0	0	0	0
E **T** **L**	45	45	0	0	0	0	0	0
E **T** **L**	42	42	0	0	0	0	0	0

(a) Number of overloads [count]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	9116.25	11.01	49.97	11.01	49.97	0.00	0.00
E T L	1.75	1.75	0.00	0.00	0.00	0.00	0.00	0.00
E **T** L	1.57	1.57	0.00	0.01	0.00	0.01	0.00	0.00
E T **L**	105.55	105.55	0.55	0.01	0.55	0.01	0.00	0.00
E **T** L	1.16	1.16	0.00	0.00	0.00	0.00	0.00	0.00
E T **L**	1.26	1.26	0.00	0.00	0.00	0.00	0.00	0.00
E **T** **L**	1.25	1.25	0.00	0.00	0.00	0.00	0.00	0.00
E **T** **L**	1.16	1.16	0.00	0.00	0.00	0.00	0.00	0.00

(b) Magnitude of overloads [MVAh]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	0.00	58855.10	19544.04	58855.10	19544.04	19863.94	19863.94
E T L	33900.57	33889.36	33156.06	19652.35	33156.06	19652.35	19645.63	19645.63
E **T** L	21797.45	21793.06	22204.86	19544.02	22204.86	19544.02	19529.00	19529.00
E T **L**	19503.16	19498.88	32497.32	19471.93	32497.32	19471.93	19466.91	19466.91
E **T** L	21151.11	21146.72	21582.88	19445.80	21582.88	19445.80	19439.74	19439.74
E T **L**	24426.80	24422.51	24509.66	19416.65	24509.66	19416.65	19410.59	19410.59
E **T** **L**	20655.89	20651.61	21106.55	19372.05	21106.55	19372.05	19363.73	19363.73
E **T** **L**	20389.40	20385.11	20849.00	19317.46	20849.00	19317.46	19311.40	19311.40

(c) Total curtailment applied [MWh]

Fig. C.6 Comparison of the performance of oracle 2 for the IEEE 14-bus system with different algorithm combinations considered for selection

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	25377	3660	1367	3660	1367	768	768
E T L	24823	24823	3332	1367	3332	1367	768	768
E **T** L	24247	24246	3363	1367	3363	1367	768	768
E T **L**	12863	12860	3411	1367	3411	1367	768	768
E **T** L	24138	24138	3278	1367	3278	1367	768	768
E T **L**	12630	12630	3300	1367	3300	1367	768	768
E **T** **L**	12330	12329	3342	1367	3342	1367	768	768
E **T** **L**	12260	12260	3274	1367	3274	1367	768	768

(a) Number of overloads [count]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	678836.34	30389.77	1241.44	33971.65	1258.29	4352.19	4352.19
E T L	670366.33	672813.97	29762.04	1241.44	30233.15	1258.29	4352.19	4352.19
E **T** L	653798.21	659729.93	28731.76	1241.44	31325.87	1258.29	4352.19	4352.19
E T **L**	241625.85	257912.76	35238.65	1241.44	37764.98	1258.29	4352.19	4352.19
E **T** L	655545.49	657868.24	29481.80	1241.44	29953.45	1258.29	4352.19	4352.19
E T **L**	254935.47	256246.22	35949.43	1241.44	36365.57	1258.29	4352.19	4352.19
E **T** **L**	246650.62	249934.92	35272.47	1241.44	37196.98	1258.29	4352.19	4352.19
E **T** **L**	247791.31	249102.07	35756.55	1241.44	36172.68	1258.29	4352.19	4352.19

(b) Magnitude of overloads [MVAh]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	0.00	982967.03	749899.56	975409.24	749868.60	822355.45	822355.45
E T L	30780.23	22468.60	983543.98	749893.01	982710.51	749862.06	822348.91	822348.91
E **T** L	43312.05	33183.32	967312.15	748622.97	962925.43	748592.01	821111.80	821111.80
E T **L**	405575.30	384360.23	974953.65	748989.57	970722.55	748958.62	821450.33	821450.33
E **T** L	41612.79	37567.73	967215.58	748622.97	966412.40	748592.01	821111.80	821111.80
E T **L**	393904.95	391578.92	967797.68	748989.57	967088.52	748958.62	821450.33	821450.33
E **T** **L**	403351.97	397717.90	956545.37	748596.65	953292.44	748565.69	821087.30	821087.30
E **T** **L**	402806.21	400480.19	956923.44	748596.65	956214.28	748565.69	821087.30	821087.30

(c) Total curtailment applied [MWh]

Fig. C.7 Comparison of the performance of oracle 1 for the IEEE 57-bus system with different algorithm combinations considered for selection

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	25377	3660	1367	3660	1367	769	769
E T L	24823	24823	3332	1367	3332	1367	769	769
E **T** L	24247	24247	3365	1367	3365	1367	769	769
E T **L**	12863	12863	3414	1367	3414	1367	769	769
E **T** L	24140	24140	3280	1367	3280	1367	769	769
E T **L**	12631	12631	3302	1367	3302	1367	769	769
E **T** **L**	12331	12331	3346	1367	3346	1367	769	769
E **T** **L**	12263	12263	3278	1367	3278	1367	769	769

(a) Number of overloads [count]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	678836.34	30389.77	1241.44	30385.38	1241.44	986.91	986.91
E T L	670366.33	670366.33	24974.82	1239.67	24974.82	1239.67	985.14	985.14
E **T** L	653798.21	653798.21	24421.55	1241.44	24421.55	1241.44	986.91	986.91
E T **L**	241625.85	241625.85	26967.29	1241.44	26967.29	1241.44	986.91	986.91
E **T** L	652049.04	652049.04	23296.29	1239.67	23296.29	1239.67	985.14	985.14
E T **L**	238886.58	238886.58	24768.12	1239.67	24768.12	1239.67	985.14	985.14
E **T** **L**	232140.18	232140.18	23782.87	1241.44	23782.87	1241.44	986.91	986.91
E **T** **L**	231512.94	231512.94	23155.66	1239.67	23155.66	1239.67	985.14	985.14

(b) Magnitude of overloads [MVAh]

	B C O	**B** C O	B **C** O	B C **O**	**B** **C** O	**B** C **O**	B **C** **O**	**B** **C** **O**
E T L	-	0.00	982967.03	749899.56	981644.69	749899.56	827822.94	827822.94
E T L	30780.23	30765.98	995253.47	750047.46	995253.47	750047.46	827970.85	827970.85
E **T** L	43312.05	43305.44	974117.95	748622.97	974117.95	748622.97	826579.29	826579.29
E T **L**	405575.30	405571.69	988936.13	748989.57	988936.13	748989.57	826917.82	826917.82
E **T** L	48519.55	48512.94	978436.37	748777.42	978436.37	748777.42	826733.74	826733.74
E T **L**	417997.22	417993.61	989173.76	749144.03	989173.76	749144.03	827072.27	827072.27
E **T** **L**	423089.93	423086.32	975305.54	748596.65	975305.54	748596.65	826554.79	826554.79
E **T** **L**	426519.49	426515.88	978687.63	748751.10	978687.63	748751.10	826709.24	826709.24

(c) Total curtailment applied [MWh]

Fig. C.8 Comparison of the performance of oracle 2 for the IEEE 57-bus system with different algorithm combinations considered for selection