UNIVERSITÀ DEGLI STUDI DI PISA

Computer Science Department

Ph.D. Dissertation

ENHANCED POWER GRID EVALUATION THROUGH EFFICIENT STOCHASTIC MODEL-BASED ANALYSIS

Supervisors:
Dr. Felicita Di Giandomenico
Prof. Marco Danelutto

Candidate:
Giulio Masetti

Ph.D. term 2015-2018

To Francesca, my parents and my sister.

"*Sigmund Freud once stated that there are two things everyone believe they can do: psychoanalysis and horse riding, but only the horse can rebel. There is a third thing everyone believe they can do: simulation. Not even in this case there is a horse that rebels, but simulation can have serious human and social consequences.*"

Giuseppe Iazeolla, Principi e metodi di simulazione discreta [1]

Acknowledgements

I would like to thank first and foremost my advisor Dr. Felicita Di Giandomenico for the continuous supporting of my studies and research, for being always proactive, for the immeasurable passion she puts in everything she does, including guiding my work.

A special word of thanks goes for Dr. Silvano Chiaradonna that from the beginning of my Ph.D. inspired me with his work and his point of view on research and life in general. Paraphrasing Albert Einstein, Silvano taught me that "we cannot solve our problems with the same thinking we used when we created them".

I am deeply grateful to Simone Dutto, not only for having had the honor to work with him on the developments of Power Flow Equations solutions and optimizations presented in this thesis, but also for the uncountable number of suggestions he gave me on other topics and for discussing with me about mathematics, music and philosophy.

A great word of thank goes for Prof. Marco Danelutto, my co-advisor, for supporting me both in scientific matters related to research lines discussed in the thesis, and more practical aspects such as providing the computer for running a subset of the experiments.

Discussing technical questions is of course an important activity, but life and science are much more involving than that. I had the opportunity and the great honor to work at the SEDC lab at ISTI-CNR in Pisa, where I found a second family, a group of distinguished researchers that was willing to teach me how to navigate in the vast sea of science. I want to thank Dr. Antonella Bertolino, Dr. Eda Marchetti, Antonello Calabró and Dr. Francesca Lonetti for their kindness and for supporting me in all the steps of my journey.

I also want to thank Dr. Tommaso Piccioli and Enrico Fantini from ISTI-CNR for the technical support they provided to coupe with computer failures happened in a critical phase during the hot Italian summer.

A special thank goes for Prof. Pierpaolo Degano that introduced me to the Ph.D. program of the University of Pisa and always guided me. I spent three years working at both ISTI-CNR and the Computer Science

Department of the University of Pisa, sharing experiences and emotions with beautiful people. In particular, I want to thank my colleague Giovanna Broccia for supporting me along our journey through the Ph.D., and all the technical and administrative staff of the Department.

I want to express my deep gratitude to Prof. William H. Sanders, Dr. Brett Feddersen, Dr. Ken J. Keefe, Dr. Jenny A. Applequist and all the guys at the PERFORM lab in Urbana-Champaign. They welcomed me like a family, supporting my work, sharing with me their experiences and though me how much friendship and kindness can overcome cultural and linguistic barriers. I spent with them six months that I will never forget. Among the people I met in Urbana-Champaign, a special word of thanks goes for Dr. Thomas N. McGeary. It was a pleasure to share with him not only physical space but also time, thinkings, emotions and music. I have found a friend.

The meaning of our thoughts and words is sometimes not clear, especially to ourselves, until we receive feedbacks from people we care about. My work is indeed a central aspect of this thesis, but the way it was thought, organized, described, written, explained and exemplified is mainly due to the interactions I had with passionate and kind people. Thus, in addition to the already mentioned members of the internal evaluation committee, I want to thank Prof. Fabio Gadducci. The external referees Prof. Katinka Wolter and Prof. Hans P. Schwefel provided great suggestions to improve my thesis; I am grateful to them for the time they spent in reviewing my work. To help is not only the ability of making it easier for someone to do something, but also sharing experiences.

Feedbacks from different communities have also contributed to improve the quality of my work, and then of this thesis. In particular, I want to thank Prof. Kishor S. Trivedi, Dr. Leonardo Robol, Dr. Gianpaolo Coro, Prof. Paolo Lollini, Prof. Marcello Cinque and Dr. Catello Di Martino.

Last but not the least, I would like to thank Francesca, my girlfriend, my parents and my sister for supporting me spiritually throughout writing this thesis and walking side by side across life. Words cannot express how grateful I am to them, I would never have been able to complete my work without the constant patience and help they gifted me.

Contents

1 Introduction **1**
- 1.1 Motivations . . . 1
- 1.2 Contributions and structure of the thesis . . . 2
- 1.3 Publications . . . 6

2 Context **9**
- 2.1 Electrical grid context . . . 9
- 2.2 Dependability aspects of Smart Grids . . . 13
- 2.3 Interdependencies and types of failure . . . 15
- 2.4 Model-based analysis . . . 17
  - 2.4.1 Role of SG analysis and evaluation . . . 20
- 2.5 The Möbius modeling environment . . . 21

3 Basic Smart Grid model **27**
- 3.1 Logical architecture of the Smart Grid . . . 28
- 3.2 Major assumptions . . . 34
- 3.3 Smart Grid state . . . 35
  - 3.3.1 Electrical state definition . . . 36
  - 3.3.2 Monitoring and Control System logic . . . 39
- 3.4 Composed and atomic models . . . 41
- 3.5 Smart Grid topology modeling . . . 47
- 3.6 Definition of failure/attack scenarios . . . 49
  - 3.6.1 Impact of MV control capabilities on LV level . . . 50
  - 3.6.2 Impact of ICT failure/attack on EI . . . 53
  - 3.6.3 Impact of failure/attack in presence of FL . . . 60
- 3.7 Achievements and open problems . . . 64
- 3.8 Summary . . . 69

4 Compositional approaches **71**
- 4.1 Related work . . . 73
- 4.2 Addressed systems' logical architecture . . . 75

4.2.1 System state definition 76
4.2.2 Component interactions definition 77
4.2.3 Component actions definition 79
4.2.4 Overall logical component definition 79
4.3 Example: a load-sharing system 80
4.3.1 System description 80
4.3.2 Logical architecture of the example 83
4.4 Theoretical discussion 84
4.4.1 Formalism and modeling features 84
4.4.2 Non-anonymous replication 87
4.4.3 *State-Sharing Replication* 89
4.4.4 *Channel-Sharing Replication* 91
4.4.5 *Dependency-Aware Replication* 96
4.5 Implementation in Möbius 98
4.5.1 *State-Sharing Replication* 98
4.5.2 *Channel-Sharing Replication* 100
4.5.3 *Dependency-Aware Replication* 103
4.6 Model a load-sharing systems 109
4.6.1 SAN models 110
4.6.2 Simulation-based model 111
4.7 New proposals for Möbius 122
4.7.1 Anonymous Rep in Möbius 122
4.7.2 Improving the anonymous Rep 124
4.7.3 Comparison between anonymous Rep variants 125
4.8 Summary 128

5 State estimation **129**
5.1 Power Flow Equations 130
5.2 Available solution methods 134
5.2.1 Real-based strategies 135
5.2.2 Complex-based strategies 137
5.2.3 Discussion 139
5.3 Exploring Wirtinger calculus 139
5.4 Comparison results 145
5.4.1 Computational analysis 145
5.4.2 Convergence rate 149
5.5 Summary 152

6 Enhanced Smart Grid model **159**
6.1 Model architecture 160
6.2 Component modeling 161
6.2.1 Description of ESTATE_SAN 162
6.2.2 Description of AGC_SAN 167
6.2.3 Description of BUS_SAN and BRANCH_SAN 168
6.2.4 Description of CB_SAN and OLTC_SAN 169
6.2.5 Description of DG_SAN and FL_SAN 170
6.2.6 Description of HG_SAN 172
6.3 Performability measures 173
6.4 Case study 175
6.5 Comparison between basic and enhanced 178
6.6 Further analyzes on the enhanced model 178
6.7 Summary 181

7 Conclusions and outlook **183**

8 List of acronyms and symbols **187**

List of Figures

1.1 Schematic picture of research lines. 4

2.1 Existing architecture of distribution grid (a) and foreseen Smart Grid (SG) architecture (b). Figure inspired by [2]. 12

3.1 Schematic representation of a small portion of a real-world electrical distribution grid. 29
3.2 Logical scheme of how nodes and arcs of an electrical grid are represented. 31
3.3 Structure of the logical node at Medium Voltage (MV) level. . 32
3.4 Structure of the logical node at Low Voltage (LV) level. 32
3.5 Complete picture of the SG logical architecture. 33
3.6 Composed model representing the overall SG 42
3.7 Composed models representing the SG at MV (a) and LV (b). 42
3.8 Composed model representing the MV-EI. 42
3.9 SAN model MV_ESTATE_SAN. 43
3.10 Composed template models representing a generic node (a) and a generic arc (b) of Electrical Infrastructure (EI) at MV. . 43
3.11 Composed model representing the template model for MV-MCS. 44
3.12 SAN model MVGC_SAN. 45
3.13 SAN model MV_DS_SAN. 46
3.14 SAN model MV_PL_SAN. 49
3.15 Voltage on each bus in absence of control, one simulation run. 51
3.16 Voltage on buses *B2*, *B7*, *B48* and *B51* with control functionalities and failure of MV Distributed Generator (DG) control on buses *B2* and *B9*, for one simulation run. 51
3.17 Available and injected active power generated by MV DG on buses *B2* (*PVP1*) and *B9* (*WP1*), for one simulation run. . . 52
3.18 Mean voltage on buses *B48* vs the *10min*-mean value of $V_{B48}(t)$ is not within *10%* of the nominal voltage, for 128 simulation runs. 52

3.19 Expected available power P^{av} from Distributed Energy Resource (DER) and constant power demand P^{dem}. 54
3.20 Probabilities that V_i is out of bound, both undervoltage $UV_i(t)$ and overvoltage $OV_i(t)$ in absence of control and failures. . . . 56
3.21 Voltage of buses $B4$, $B7$ and $B11$ when full control is applied: DER power curtailment and operative On Load Tap Changer (OLTC), in absence of failures. 57
3.22 The curtailment of renewable active power generated by PVP and WP when full control is applied. 57
3.23 The OLTC tap values for each time t when full control is applied. 58
3.24 Probabilities that V_{B11} is out of bound, both undervoltage $UV_{B11}(t)$ and overvoltage $OV_{B11}(t)$, for two different values of timing failure (10 min and 20 min) and for the failure of the wind power plant (WP) control device. 58
3.25 Probabilities that V_{B11} is out of bound, both undervoltage $UV_{B11}(t)$ and overvoltage $OV_{B11}(t)$, for two different values of timing failure (10 min and 20 min), in conjunction with the failure of the WP control device. 59
3.26 Probability that the grid voltage requirement $MV1$ is rejected, for all buses in the grid, for two different values of timing failure (10 min and 20 min), when failures of the control device of WP occur and when they do not occur. 60
3.27 Probabilities that V_{B7} is out of bounds, both undervoltage $UV_{B7}(t)$ and overvoltage $OV_{B7}(t)$, when the OLTC fails and there is no load control over bus $B2$ and buses from $B4$ to $B11$. Only bus $B3$ has an operating load control (INDUSTRY load CTRL). 61
3.28 Expected unsatisfied power demand $UD_{B7}(t)$ for bus $B7$ at each instant of time t, when the OLTC fails and the only operative load control is on $B3$ (INDUSTRY load CTRL). . . 61
3.29 Expected available power P^{av} and variable power demand P^{dem}. 62
3.30 Probabilities $OV_i(t)$ and $-UV_i(t)$, for each bus of the grid, with variable power demand, in absence of control and failures. 63
3.31 Voltage of buses $B4$, $B7$ and $B11$ when full control is applied: DER power curtailment and operative OLTC, in absence of failures. 63
3.32 Probabilities that V_{B7} is out of bounds, both undervoltage $UV_{B7}(t)$ and overvoltage $OV_{B7}(t)$ when the OLTC fails and the load on bus $B3$ is either flexible (INDUSTRY Load CTRL) or inflexible. 64

3.33 Expected unsatisfied power demand $UD_{B7}(t)$ for bus $B7$ at each instant of time t when the OLTC fails and the load on bus $B3$ is either flexible (INDUSTRY Load CTRL) or inflexible. 65

4.1 Example of two read/write access topologies $\mathcal{T}^0$ and $\mathcal{T}^1$ (a) for a generic component C^g with two State Variables (SVs), gray SV^0 and white SV^1, and $n = 3$ (b). Dashed lines represent read/write access of C_i to dependency-aware SVs. 78
4.2 Continuation of the example depicted in Figure 4.1: topology of interactions $\mathcal{T}$ (a) and diagram with interations among specific components (b). 78
4.3 Example described in Section 4.3. The topology of interactions among components is represented in (a), the starting configuration of the system is depicted in (b), the load redispatching after a failure of $\texttt{Node}_2$ is represented in (c), the failure, and consequent reboot, of $\texttt{Node}_0$ after $T_{A,0}^{\text{threshold}}$ time units is depicted in (d). 82
4.4 *State-Sharing Replication* (*SSRep*) approach: composed *Rep* model M^{sys} (a) and template Stochastic Activity Network (SAN) model I initializing the index of the replicas (b). 99
4.5 *SSRep* approach: template SAN model M defining the generic component for the case study described in Section 4.3. 99
4.6 *Channel-Sharing Replication* (*CSRep*) approach: composed *Rep* model (a) and SAN models *WRITECHANNEL* (b) and *READCHANNEL* (c) used respectively to write and to read the shared data into the channel. 101
4.7 *CSRep* approach: template SAN model M defining the generic component for the case study described in Section 4.3. 102
4.8 Architecture of $\mathcal{D}^{perl}$. 104
4.9 Example of Topology XML file. 105
4.10 *Dependency-Aware Replication* (*DARep*) approach: composed *Join* model generated from the template M^{darep} for the case study described in Section 4.3. Notice that the figure represents only a portion of the composed model and has being generated by the *DARep* script. 106
4.11 *DARep* approach: SAN model *SANSANDAREP0* generated from the template M^{darep} defining the generic component for the case study described in Section 4.3 with $n = 100$ and $\delta = 74$. Notice that the figure represents only a portion of the composed model and has being generated by the *DARep* script. 107

4.12 *DARep* approach: a snapshot of the "Define Node Join Dialog" of the Möbius tool for the composed model *Msys* generated from the template M^{darep} for the case study described in Section 4.3 with $n = 100$ and $\delta = 74$. Notice that the figure has been generated by the *DARep* script. 107
4.13 CPU time for the initialization phase of *DARep* and *CSRep* relative to *SSRep* where δ varies. Bars on the left of the n value refer to *CSRep*, on the right to *DARep*. 116
4.14 initialization CPU time for *SSRep* at increasing of δ. 117
4.15 initialization CPU time for *CSRep* at increasing of δ. 117
4.16 initialization CPU time for *DARep* at increasing of δ. 118
4.17 batch time for *SSRep* at increasing of δ. 119
4.18 batch time for *CSRep* at increasing of δ. 120
4.19 batch time for *DARep* at increasing of δ. 121
4.20 Comparison of time complexity for the solver initialization phase in the current version of Möbius and for our new algorithm as the number n of workers increases. 127

5.1 Convergence behavior for case6470rte [3] with random initial voltages: the effect of changing σ on the number of converged runs (on the left) and the mean of data concerning singular values of the Jacobian when the strategies do not converge (on the right). 151
5.2 Convergence behavior for case2869pegase [3] with random initial voltages: the effect of changing σ on the number of converged runs (on the left) and the mean of data concerning singular values of the Jacobian when the strategies do not converge (on the right). 152
5.3 Convergence behavior for case2383wp [3] with random initial voltages: the effect of changing σ on the number of converged runs (on the left) and the mean of data concerning singular values of the Jacobian when the strategies do not converge (on the right). 153

6.1 Enhanced SG model architecture. 160
6.2 Example of Möbius *Join* produced by *DARep* for the enhanced SG model. Only a portion of the *Join* is depicted because it comprises hundreds of submodels. 161
6.3 ESTATE_SAN. 163
6.4 ESTATESANSANDAREP0. Here the Dependency-Aware places are superimposed. 164

6.5 AGC_SAN. 167
6.6 BRANCH_SAN. 168
6.7 BUS_SAN. 169
6.8 CB_SAN. 170
6.9 OLTC_SAN. 170
6.10 DG_SAN. 171
6.11 FL_SAN. 172
6.12 HG_SAN. 173
6.13 Structure of the EI for the scenario in which there are two areas. This picture has been produced using STAC [4]. 176
6.14 Active power profiles. 177

List of Tables

2.1 List of differences between the existing EI and the SG according to [2]. Features directly related to this thesis are in bold. 11

3.1 MV line parameters for the grid depicted in Figure 3.2. 54
3.2 Load parameters. 55
3.3 CPU times for initialization of the basic model as the number of buses n increases. The case studies employed are detailed in Section 6.5. 66
3.4 Number of CTRL calls, Power-Flow Equations (PFEs) subroutine calls, and among them how many do not converge, as n increases, for a simulation time of $24h$. 67

4.1 τ_{init} for the *SSRep* approach. 114
4.2 τ_{init} for the *CSRep* approach. 115
4.3 τ_{init} for the *DARep* approach. 115
4.4 $\Delta\tau(1000)$ for the *SSRep* approach. 115
4.5 $\Delta\tau(1000)$ for the *CSRep* approach. 116
4.6 $\Delta\tau(1000)$ for the *DARep* approach. 116
4.7 ${}^{D}_{S}\mathcal{C}\tau(k)$ between the *DARep* and *SSRep* when $\delta = 10$. 119
4.8 ${}^{C}_{S}\mathcal{C}\tau(k)$ between the *CSRep* and *SSRep* when $\delta = 10$. 120
4.9 ${}^{D}_{C}\mathcal{C}\tau(k)$ between the *DARep* and *CSRep* when $\delta = 10$. 120
4.10 CPU times for initialization as n increases. 126
4.11 CPU times for 1000 batches as n increases. 127

5.1 Theoretical estimations of the loads required for **F** and **J** at each step of each strategy 146
5.2 Means of the flops required by the linear solver at each step of each strategy. 147
5.3 Elapsed time in ms for evaluating **F**, **J** and solving the linear system (sol) at each step of each strategy 155

5.4 Total number of converged runs and mean of required steps to convergence with different initial voltages 156
5.5 Total number of converged runs and mean of required steps to convergence with different levels of loads 157
5.6 Total number of converged runs and mean of required steps to convergence with different R/X ratios 158

6.1 CPU initialization and mean batches time as n increases. . . . 179
6.2 Number of CTRL calls, PFEs subroutine calls, and among them how many do not converge, as n increases, for a simulation time of $24h$. 179
6.3 Convergence rate of *recovery block* ($\mathrm{NR}^{\mathrm{PC}}$,$\mathrm{NR}^{\mathrm{RC}}$,$\mathrm{NR}^{\mathrm{NV}}$) calls, and mean number of blackouts, as n increases. 180

Chapter 1

Introduction

1.1 Motivations

The electrical infrastructure can be considered nowadays as a meta-critical infrastructure: in fact it is the basis for almost all the critical infrastructures a modern nation can have, such as water, oil, gas and transportation. This implies that its correct operation is a fundamental requirement to the correct operation of the critical infrastructures that depend on it. To allow pervasive control and monitoring towards resilience and performance enhancements, the Smart Grid is emerging as a convergence of information and communication technology with power system engineering. In particular, the ever increasing level of distributed energy resources penetration calls for more and more sophisticated monitoring and control facilities.

So, studying the influence of distributed energy resources, of new control policies and ICT on the dependability of distribution grids offers valuable insights on how to improve the design of Smart Grids. In addition to standard dependability measures such as reliability and availability, among greatly relevant measures specifically defined for electrical distribution systems there are the voltage quality and the energy required, but not supplied, by the distribution system.

A popular approach to assess electrical distribution specific measures, in presence of failures or attacks to the ICT system and/or to the electric infrastructure, is the stochastic model-based analysis. Although several studies have been already proposed, the research in this context still faces a number of challenges, mainly due to the need: i) to consider both the ICT subsystem and the controlled electrical infrastructure, to properly account for (inter)dependencies through which operations (and failures/attacks) propagate; ii) to model and analyze the SG components at a sufficiently detailed

level of abstraction, targeting realistic representation of their structure and behavior in view of promoting accuracy of the assessment itself. Both nominal and a variety of faulty behaviors are to be investigated, since the interest is on assessing resilience and quality related attributes; iii) to tackle realistic segments of SG in terms of topology size, to make the evaluation study of real interest to stakeholders involved in the field.

To cope with all these needs results in huge and complex models, to be typically defined in a modular fashion and requiring sophisticated compositional operators. Moreover, model solution through simulation-based evaluation becomes unavoidable in presence of non-Markovian behavior of the involved components, thus preventing the use of analytical approaches.

Given the above premises, the stochastic model-based analysis of realistic SG topologies is a research area where further investigations and enhancements are highly desirable. In this context, this thesis offers contributions in the direction of promoting efficient evaluation of SG in realistic scenarios from a resilience perspective.

1.2 Contributions and structure of the thesis

Starting from a preliminary development of a model-based framework for SG evaluation already available at the ISTI SEDC Lab in Pisa, the work progressed in the following major directions, as depicted in Figure 1.1:

- finalization of a basic SG evaluation framework, carried on taking into account the challenges described above, especially in terms of detailed representation of structure and behavior of electric components and how they are impacted by the behavior of cyber components, as well as (inter)dependencies and failure phenomena. This was a valuable effort to start exercising the framework on a variety of grid topologies and failure scenarios, which helped greatly to understand strengths but also limitations of the current solution and pointing out directions for refinement steps. In particular, efficiency was identified as the major obstacle to deal with grid topologies representative of realistic regional segments, so further advancements concentrated on this aspect;

- compositional operators adopted in the modular development of models were identified as a major weakness from efficiency viewpoint, and therefore as a research area where available solutions need improvements. In particular, new non-anonymous replication techniques for constructing the system model starting from template models of its components have been formalized, implemented and tested within the

Möbius modeling framework. Among the available possibilities, the best from the point of view of performance has been selected to replace the one adopted in the basic model;

- state estimation computations were also identified as frequently executed operations and so another promising aspect for performance improvement. Several methods, all based on the Newton-Raphson method, to estimate the state of the electrical infrastructure have been investigated and among those the best approach has been selected;

- control logic, implemented as the solution of an optimization problem, requires several calls to the electrical state estimation process and then a fine tuning, aimed at minimizing the computational cost, has been carried out.

The benefits in efficiency brought by the newly developed solutions have been quantified through a comparison study between the basic evaluation framework and the enhanced one equipped with such solutions.

The thesis is then structured as follows. In Chapter 2, an overview of the context of work is presented. In particular, the electrical grid context is briefly discussed in Section 2.1 where the challenges and opportunities offered by the Smart Grid (SG) are discussed. The focus is on distribution level, where the SG evolution is taking place and the increasing penetration of distributed generators poses relevant challenges to the model-based analysis. Dependability analysis for SG is described in Section 2.2 putting the stress on stochastic modeling, in particular exploiting the *performability* analysis. The key concept of interdependencies among system components is detailed in Section 2.3, where the fact that interconnections promote interdependencies is explained in the context of cyber-physical system. Cascading and escalating failures occur in presence of interdependencies and model-based analysis is well suited to address their description, characterizing the way failures spread among components. Model-based analysis is then discussed in more details in Section 2.4, followed by a brief discussion on its role in particular for SGs in Section 2.4.1. Section 2.5 is devoted to the description of the main formalism employed all over the thesis, i.e., the Stochastic Activity Network (SAN) formalism, and the modeling environment in which the analysis has been carried out: the Möbius modeling tool [5].

Chapter 3 is devoted to the description of a basic SG model. This model was almost complete when the work presented in this thesis started, and served as the starting point for the investigations here developed. In particular, the logical architecture of the SG is detailed in Section 3.1 where

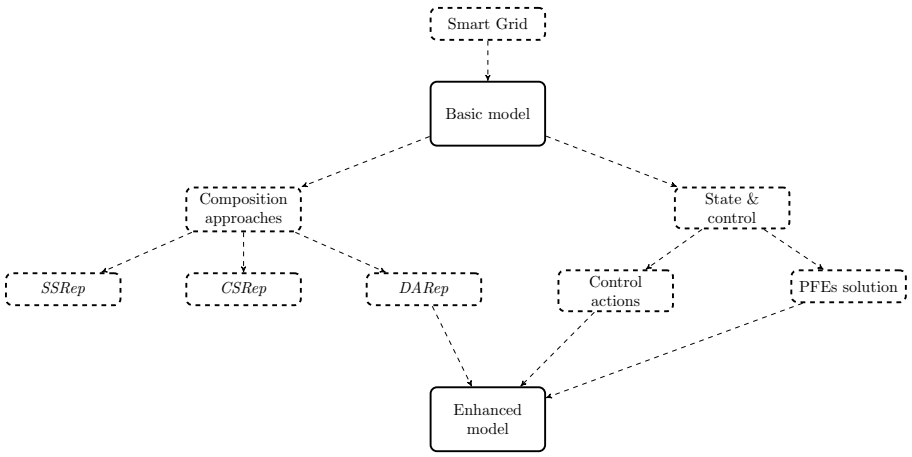


Figure 1.1: Schematic picture of research lines.

a description of electrical nodes, called Bus-Bars (buses), arcs and electrical components attached to them is offered. Major assumptions about SG modeling are stated in Section 3.2, the focus being on failures, time and interdependencies. Choosing the right level of abstraction is always challenging, and this section presents the point of view adopted both in the basic and enhanced models. Section 3.3.1 details the state estimation process and describes how control facilities are modeled. The basic SG model architecture is described in Sections 3.4 and 3.5, focusing in particular on the way interactions among components are modeled.

The original contribution of this thesis starts in Section 3.6, where failure and attack scenarios are defined and implemented within the basic model. Three different scenarios, involving different levels of the EI hierarchical structure, are presented. In Section 3.7 achievements and problems related to the basic SG model are discussed. There are two main obstacles to the application of the model to the study of large electrical distribution grids: the model architecture employs a state of the art modeling approach that introduces a relevant overhead, and the method used to estimate the EI state is inefficient. The observations of this section guided the developments of the next chapters and justify the design and implementation of a new, enhanced, SG model. As depicted in Figure 1.1, in this thesis two main research lines have been investigated: the architectural issues of the basic model have been formalized and overcome proposing two new submodels compositional approaches, as detailed in Chapter 4, while the EI state estimation inefficiencies are studied in Chapter 5.

Chapter 4, as already mentioned, introduces a formal description of the

compositional approach employed in the basic SG model, called *State-Sharing Replication* (*SSRep*), and proposes two new approaches, called *Channel-Sharing Replication* (*CSRep*) and *Dependency-Aware Replication* (*DARep*). In particular, after a brief review of the literature in Section 4.1, the logical architecture of the systems touched by the analysis is discussed. Notice that the submodels compositional approaches discussed in this thesis can be applied to a great variety of systems, including the SGs, and then a general description is offered. Both abstract, in Section 4.4, and implementation oriented, in Section 4.5, points of view are considered. A representative case study, not related to the SG, is presented in Section 4.6. The case study, a load-sharing system, is instrumental to test effectiveness and efficiency of the proposed compositional approaches. Numerical experiments are presented in Section 4.6.2 in order to verify the theoretical predictions stated within the chapter and help choosing the right approach for the enhanced SG model design. In addition, new proposals for the Möbius modeling tool are presented in Section 4.7.

Chapter 5 presents the abstract formalization of the power flowing in the electrical grid. In Section 5.1 the Power-Flow Equations (PFEs) are presented as the means to estimate the EI state, discussing their representation power and the issues the modeler has to overcome to solve them. A review of available solution methods is presented in Section 5.2, where solution strategies are divided according to the fact that they are based on real numbers or complex numbers. A new formalization, and consequent solution strategy, is presented in Section 5.3. The reformulation is the result of collecting together ideas coming from both the electrical engineering community and the mathematical community, in particular the use of Wirtinger calculus and the identification of symmetries that naturally emerge from the derivation. All the solutions strategies are variants of the Newton-Raphson method (NR) method and then comparisons among them are carried out in Section 5.4 studying two characteristic aspects: computational cost, in Section 5.4.1, and convergence behaviors, in Section 5.4.2.

Chapter 6 presents the enhanced SG model. In particular, the new model architecture is discussed in Section 6.1 and the component template models are presented in Section 6.2. In particular, a template model has a special role in this context: the one responsible for both the system state estimation and control policy application (Section 6.2.1). Here, the problems related to the PFEs solution and the optimization problem that implements control functionalities, highlighted in Section 3.7, are addressed. A case study is discussed in Section 6.4, where a set of electrical grids of increasing size is presented. Numerical experiments have been carried out in order to test the effectiveness and efficiency of the enhanced model in comparison with the

basic model. Results are presented in Section 6.5. Finally, conclusions and direction for further investigations are drawn in Chapter 7.

1.3 Publications

During the PhD period (November 1, 2015 - October 31, 2018) Giulio Masetti has co-authored the following papers:

- [6, 7]: Proceedings of the Smart Energy Grid Engineering conference and the Latin-American Symposium on Dependable Computing, related to the study of failure and attacks scenarios for the basic model described in Chapter 3. In particular, Section 3.6.1 of this thesis is based on developments presented in [6], and Sections 3.6.2 and 3.6.3 on developments in [7].

- [8]: American Institute of Physics Conference Proceedings, Numerical Computations: Theory and Algorithm conference, related to the study of Inexact Newton-Krylov method properties. The basic model of Chapter 3 employed the Inexact Newton-Krylov method to perform key computations and then some of its peculiar details, such as the impact of the equations ordering on convergence, were studied.

- [9–11]: fast abstract and student forum contributions presented at, and included in Proceedings of, European Dependable Computing Conference and the conference on Dependable Systems and Networks. In particular, in [9] the idea of developing a new non-anonymous model composition operator was presented for the first time, the enhanced model described in Chapter 6 was sketched in [10] and improvements of Möbius, detailed in Section 4.7.2, were discussed in [11].

- [12]: Proceedings of the European Workshop on Performance Engineering. In particular, [12] presents the definition of a new compositional operator, called *Channel-Sharing Replication* (*CSRep*). This paper is at the basis of Section 4.5.2 of this thesis.

- [13]: Proceedings of the IEEE International Symposium on Software Reliability Engineering. In particular, the implementation of the new model compositional operator called *Dependency-Aware Replication* (*DARep*) is described and tested against the approach employed for the basic model of Chapter 3, called *State-Sharing Replication* (*SSRep*). The implementation, with minor changes, served as the starting point

for the work discussed in Section 4.5.3 of this thesis. The case study was a preliminary version of the one discussed in Section 4.6.

- [14]: article submitted to the IEEE Transactions on Reliability journal. This paper collects all the investigations regarding model compositional approaches presented in this thesis and describes them introducing a new formal notation. The structure and content of Chapter 4 is based on the work presented in the article.

- [15]: Proceedings of the MODELSWARD conference, Special Session on domAin specific Model-based AppRoaches to vErificaTion and validaTiOn. Physical systems whose behavior is characterized by moving parts constrained by a phase relation are addressed. This is typically the case of AC electrical grids. System state is often modeled with the help of complex non-holomorphic equations and in this paper, a general purpose approach to solve those equations is presented.

- [16]: article published on the IEEE Transactions on Power Systems journal. Chapter 5 of the thesis is almost entirely based on this paper. In particular, several variants of the NR method have been investigated in the context of PFEs solution. Computational aspects and convergence behaviors are analyzed.

- [17, 18]: article and technical report regarding the re-formulation of performability measure in terms of matrix functions. Even if not directly related to the developments presented in the thesis, the mathematical investigations reported in these papers open new possibilities for what concerns numerical analysis based solutions of large composed models.

Chapter 2

Context

2.1 Electrical grid context

The Electrical Infrastructures (EIs) of Western countries have been developed during the last century and an half to meet demands of ever changing societies: more power is needed to support the industrialization evolution, greater electrical grids are needed to sustain the power request/supply and reach costumers sited far from big cities, hierarchical structure of grid management is needed to integrate different services and providers (public and private), diversification of power resources, and recently the proliferation of green resources, is needed to face an evolving market and decrease the CO_2 footprint. It is not in the scope of this section to describe the evolution of the EI or to provide a detailed picture of the state of art; instead the objective is to identify the fundamental characteristics that promote the adoption of a new, ground breaking, paradigm: the Smart Grid (SG).

The physical part of a modern country EI can be described considering three levels of abstraction: the High Voltage (HV) grid, the Medium Voltage (MV) grid and the Low Voltage (LV) grid. Grids of all levels comprise two kinds of elements: nodes, called Bus-Bar (bus), and electrical lines. Grid buses and lines can be considered as logical elements to which electrical components can be attached to. In particular, power generators and power consumers (called loads) can be attached to each bus.

The HV grid is the backbone that supports the power flowing: covers long distances, connects several regions within the country and can be interconnected to other countries HV grids. In a typical HV grid there are restrictions to the type of components that can be attached to its buses, in fact there are buses with attached only generators and other buses with attached only loads, so that the power flow is *unidirectional*: from the first

kind to the second kind of buses. The goal of the HV grid is to transmit power from source to consumption, and then, without considering too much details, in this thesis it will be also called *transmission grid* [19].

Each load of a HV grid can be a single entity that requires huge quantity of power, e.g., a big industry, or the abstraction of an entire MV grid. Similarly, each load of a MV grid can be a single entity or an entire LV grid. In other words, the physical part of an EI can be represented as a matrioska of electrical grids. The three levels (high, medium and low) reflect not only the level of abstraction used to represent the EI but also physical characteristics: long electrical lines, designed to work at high voltage, within HV grids, medium lines and medium voltages in MV grids and short lines at low voltages within LV grids [20].

The MV was originally designed to receive power from only one source, and in fact it is seen at the HV level as an unique load. The MV distributes the received power to its loads, that in turn can be medium sized industries, agricultural or commercial facilities, or LV grids. MV and LV grids are here called *distribution grids* [20]. The LV grid dispatches power to small commercial activities, agriculture facilities and houses, often following the street networks.

The existing EI is primarily powered by conventional carbon-based or nuclear fuels, so that only about one-third of the energy is actually converted in electricity without recovering the waste heat, and almost 8% of its output is lost along transmission lines. Moreover, about 20% of its generation capacity exists only to meet peak demand, i.e., it is in use only 5% of the time [2].

Thus, a new scenario has been promoted to – at least partially – overcome the inefficiencies of the existing EI: a combination of new architecture design, vast employment of Information and Communication Technology (ICT), at all levels, and DERs penetration, namely the SG, which characteristics are briefly listed in Table 2.1. During the last thirty years, an enormous amount of investments all over the world has been devoted not only to update the physical part of EIs, for instance introducing digitally controlled electrical components instead of electromechanical ones, but also to cover the infrastructure with a dense ICT layer, capable to enhance the monitoring and control facilities. The transmission grid has been equipped with sensors, ad hoc networks and actuators, thus obtaining a self-monitoring cyber-physical system and a pervasive control capability.

A similar process has been started for the distribution and LV grids but the specific characteristics of these grids pose overwhelming challenges, and then ask for innovative solutions. First, the number of sensors and actuators required to control the electrical components that compose all the MVs grid abstracted away at the HV level is huge. For the LV grids the prob-

Existing Grid	Smart Grid
electromechanical	digital
one-way communication	**two-way communication**
centralized generation	**distributed generation**
hierarchical	**network**
few sensors	sensors throughout
blind	**self-monitoring**
manual restoration	self-healing
failures and blackouts	**adaptive** and islanding
manual check/test	remote check/test
limited control	**pervasive control**
few costumer choices	many costumer choices

Table 2.1: List of differences between the existing EI and the SG according to [2]. Features directly related to this thesis are in bold.

lem is even more challenging. In addition, it has been considered infeasible to design ad hoc communication networks to connect all these devices and then mixed solutions are required, so that challenging reliability, availability and security issues have to be addressed. Second, during the last years two disruptive revolutions took place: open (and deregulated) market and renewable energy resources penetration. Nowadays, at least in Europe, the electricity bill for the end consumer usually involves four components: transmission service, provided by an operator, distribution service, provided by a different operator, power generation, provided by another company, and national or local taxes [21]. What once was a vertical and centrally orchestrated business, from energy sources to costumers, now is governed by a decentralized and open market, guided by competition and regulated by different entities at different levels (deregulation). The switch to open market influenced both transmission and distribution, but took place at the beginning of the ICT introduction at MV level and then has a greater impact on the latter: on the one hand, it busts up the interest of new players, on the other hand it is no more possible to follow the procedure for the introduction of ICT at HV level and new investments strategies are needed. The green energy movement, primarily aimed at reducing the CO_2 footprint of ever growing industrial societies, together with the aim of reducing dependencies of a country from foreign countries diversifying the local production of electricity, has determined the rapid penetration of renewable energy resources at both transmission and distribution levels.

At distribution level, distributed generators (DGs), as photovoltaic and

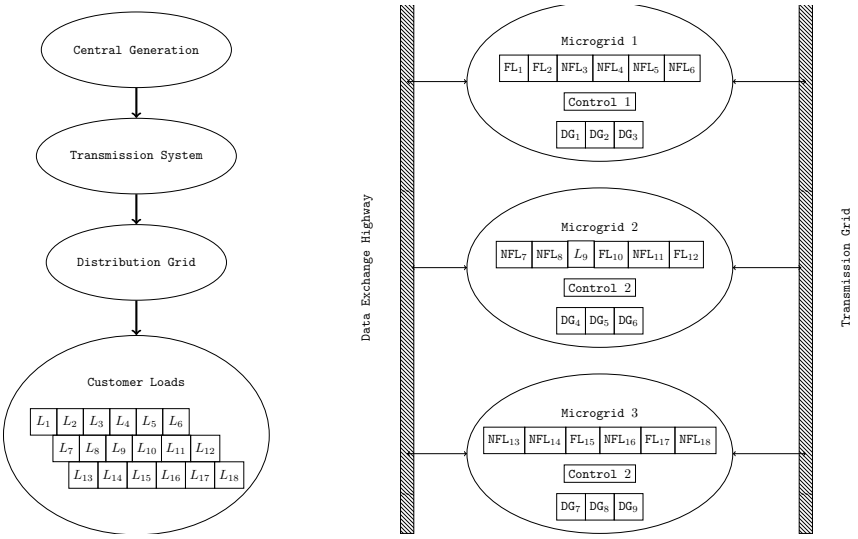


Figure 2.1: Existing architecture of distribution grid (a) and foreseen SG architecture (b). Figure inspired by [2].

wind, non renewable generators, as those fueled by natural gas, energy storage and other distributed source of energy are all classified as Distributed Energy Resource (DER) [2]. During time, the amount of power provided by DERs can change, and then the entire state of the EI at MV level can change accordingly. Every load in the context of this thesis is considered *controllable*, in the sense that it can be manipulated by the manager of its grid segment, and the only distinction among loads is between non-flexible load (NFL), whose manager can only allow them to consume all the required power or disconnect them, and flexible load (FL) whose manager can decide to supply them a fraction of the required power. From the point of view of the manager, NFLs are treated as binary entities whereas FLs are considered as continuous entities.

The growing penetration of DERs, the fast evolution of ICT and the presence of new players in the energy market, let domain experts to foresee a disruptive change of the EI architecture: from monolitic and vertically structured to modular and distributed. In the existing architecture, depicted in Figure 2.1a, a centralized control can influence the transmission grid and the data exchange with the distribution level is limited. The distribution level control capability comprises facilities to allow or stop the power flowing in specific lines (switches and breakers), change the voltage ratio at the ends of some lines (transformers with adjustable ratio) and balance the reac-

tive power flowing in some bus (capacitor banks). The loads at distribution level communicate with the Medium Voltage Grid Control (MVGC) only to transmit consumption data. Not considering electrical components protections, i.e., devices capable to rapidly (milliseconds) disconnect an electrical component from the grid in case anomalies are detected, all the logic of the system is concentrated within the MVGC.

In Figure 2.1b is instead depicted the architecture os a SG, where the distribution level is divided in interacting *Microgrids*, each having monitoring and control facilities that exchange data and can influence the behaviour of local DGs, FLs, NFLs and other components such as variable transformers and capacitor banks.

This is the high level reference architecture adopted in the thesis, further elaborated in Chapter 3.

Among the actions Microgrid control system is equipped with, *generation curtailment* and *load shedding* are of particular importance for this thesis. Whenever a DG is able to produce a certain amount of power, the microgrid Monitoring and Control System (MCS) can decide to allow the injection of only a fraction of the amount; this action is called curtailment. Similarly, if a FL demands a certain amount of power the microgrid MCS can decide to allow the dispatch of only a fraction of the amount; this is called shedding. In this way, the MCS can dynamically change the request of power to transmission level and neighbour microgrids, balancing the internal power production and consumption.

2.2 Dependability aspects of Smart Grids

When conceiving the design of a system, its characterization is given in terms of *functional specification*, i.e., a description of what the system is intended for, and a *non-functional specification*, i.e., a description of how well the system is supposed to provide its intended service. If the system complies with its functional and non-functional specification, then the system provides a *proper* service; otherwise, the provided service is *improper*. The transition from proper to improper service is referred to as a *failure*. *Service restoration*, on the other hand, is a transition from incorrect to correct service. The (part of the) system state that generates a failure is called *error*, and the hypothesised cause of the error is called a *fault*. The cause-effect relation between faults, errors and failures is known as the *Fault* $\rightarrow$ *Error* $\rightarrow$ *Failure* chain.

Dependability of a computing system is defined as the ability to deliver service that can justifiably be trusted [22]. Therefore, the building of a

dependable system requires mechanisms that break the *Fault* → *Error* → *Failure* chain, so reducing the number of failures and their severity. These]mechanisms can be grouped in four main categories [23]: fault prevention, fault tolerance, fault removal, fault forecasting. *Fault prevention* aims at eliminating all possible faults; in real-world systems, this is generally accomplished with design and implementation choices that reduce the probability of a system failure to an acceptably low value. Fault prevention techniques can be applied during all phases of the system lifecycle. *Fault tolerance* expects failures to occur and deals with mechanisms suitable to compensate failures with proper counteractions. Fault tolerance can be carried out via *error detection* (identification of errors), and *system recovery* (elimination of identified errors and prevention of a re-activation of their causes). *Fault removal* is intended to detect and remove faults introduced during all phases of the lifecycle of a computing system. During the development phase, fault removal consists in *verification* (checking system properties), *diagnosis* (identification of the fault), *correction* (fault removal). *Fault forecasting* deals with estimating present and future faults. Techniques for fault forecasting can be either *qualitative*, i.e., they identify, classify and rank events that possibly lead to service failures, or *quantitative*, i.e., they provide a probabilistic estimation of the occurrence of service failures.

The Smart Grid (SG) context addressed in this thesis fully belongs to the category of dependability-critical systems, since it is a critical infrastructure offering services from which our society strongly depends on, including a large number of other critical infrastructures. As such, several national and European projects have addressed dependability analysis of SG. In particular, the following projects constitute the kernel of knowledge at the basis of the work presented in this thesis: IRIIS [24], CRUTIAL [25], TCIPG [26], PIAFARA [27] and SmartC2Net [28]. Especially, the intriguing implications of interdependencies (better addressed in Section 2.3), existing intra power grid infrastructure subsystems and inter SGs and other critical infrastructures, have been investigated. An overview of research in infrastructure interdependency modeling and analysis carried on through model-based analysis is in [29–32].

To contribute to enhance dependability of SG, the research activity presented in this work focused on fault forecasting techniques, and in particular on the analysis of the impact of failures affecting either the grid infrastructure or the smart control functionalities on the ability of the electrical power system to provide correct service. Detailed description of the failure model are provided in Section 3.2.

The dependability of a system is generally measured against the following attributes: *availability*, *reliability*, *safety*, *integrity*, *maintainability*. *Avail-*

ability is defined as the readiness for correct service and is generally computed as the ratio between the up-time of the system to the duration of the considered time period. *Reliability* is defined as the continuity of correct service and is typically expressed by using the notions of mean time between failures (MTBF) and mean time to recover (MTTR) for repairable systems, and with mean time to failure (MTTF) for non-repairable systems. *Safety* is the absence of catastrophic consequences. This attribute is a special case of reliability: a safe state, in this case, can be either a state in which a proper service is provided, or a state where an improper service is provided due to non-catastrophic failures. *Integrity* is defined as the absence of improper system state alterations. *Maintainability* is the ability to undergo modifications and repairs. From these basic attributes, more sophisticated ones can be derived, e.g. *Performability*, defined as the ability of a system to accomplish its intended services in the presence of faults over a specified period of time [33]. Performability allows to evaluate different application requirements and to assess dependability-related attributes in terms of risk versus benefit.

In the context of SGs, performability-related indicators look the most appropriate to assess the quality of provided service in critical scenarios characterized by failure events, where degradation can be experienced at the level of both voltage and supplied energy, as form of partial or total blackouts. Deeper discussion on addressed dependability-related indicators is in Chapter 3.

2.3 Interdependencies and types of failure

As extensively discussed in [34], there is a consensus in the literature on critical infrastructures that interdependency analyzes are of paramount importance to improve the resilience, survivability and security of these vital systems. An interdependency is a bidirectional relationship between two infrastructures through which the state of each infrastructure influences or is correlated to the state of the other [35]. Infrastructure interdependencies can be categorized according to various dimensions in order to facilitate their identification, understanding and analysis. Six dimensions have been identified in [35]. They include: a) the type of interdependencies (physical, cyber, geographic, and logical), b) the infrastructure environment (technical, business, political, Legal, etc.), c) the couplings among the infrastructures and their effects on their response behavior (loose or tight, inflexible or adaptive), and d) the infrastructure characteristics (organizational, operational, temporal, spatial), e) the state of operation (normal, stressed, emergency, repair) and f) the degree to which the infrastructures are coupled, the type of failure

affecting the infrastructures (common-cause, cascading, escalating).

Interdependencies increase the vulnerability of the corresponding infrastructures as they give rise to multiple error propagation channels from one infrastructure to another that increase their exposure to accidental as well as to malicious threats. Consequently, the impact of infrastructure components failures and their severity can be exacerbated and are generally much higher and more difficult to foresee, compared to failures confined to single infrastructures. Just looking at the history of past major power grid blackouts, they have been initiated by a single event (or multiple related events such as an equipment failure of the power grid that is not properly handled by the SCADA system) that gradually leads to cascading outages and eventual to the collapse of the entire system [36].

Three types of failures are of particular interest when analyzing interdependent infrastructures: 1) cascading failures, 2) escalating failures, and 3) common cause failures.

- Cascading failures occur when a disruption in one infrastructure causes the failure of one or more component(s) in a second infrastructure.

- Escalating failures occur when an existing failure in one infrastructure exacerbates an independent disruption in another infrastructure, increasing its severity or the time for recovery and restoration from this failure.

- Common cause failures occur when two or more infrastructures are affected simultaneously because of some common cause.

Of course, besides analyzing the types of failures, it is important to understand the different causes that might lead to the occurrence of such failures. As discussed in [23], faults and their sources are very diverse. They can be classified according to different criteria: the phase of creation (development vs. operational faults), the system boundaries (internal vs. external faults), their phenomenological cause (natural vs. human-made faults), the dimension (hardware vs. software faults), the persistence (permanent vs. transient faults), the objective of the developer or the humans interacting with the system (malicious vs. nonmalicious faults), their intent (deliberate vs. nondeliberate faults), or their capability (accidental vs. incompetence faults). Knowing the cause of the failure, proper measures could be taken at level of the system controlling the infrastructure so as to prevent future occurrence of the same fault or at least mitigate its effects on the system.

In previous decades, accidental threats were basically the only real threats facing infrastructure, especially natural disasters, which tend to be localized

to one region and have a fixed and, at times, predictable duration. Until the bombing of the Murrah Federal Building in Oklahoma City in 1994, few paid attention to malicious acts targeting these critical components [37]. In more recent years, preparation for Y2K (2000), fall-out from post-9/11 events, and a series of blackouts of the power systems experienced both in US and Europe have all reinforced the evidence of how fragile these systems are or can become. This awareness has triggered a number of initiatives, on both national and international scales, to protect critical infrastructures from all hazards, both natural and man-made disasters and terrorism, whether a cyber-related threat or large-scale physical attack. Hurricanes, earthquakes, fires, and train crashes can impact infrastructure just as much as a premeditated act aimed at disrupting services or harming the people. An overview of most visible documented attacks over the last several years is provided in [38].

In this thesis, the interest is on both accidental faults and intentional attacks. However, rather than on the fault/attack perspective, the study focuses on the effect of such faults/attacks on the SG component(s) affected by them, that is on the generated failure(s). In fact, the aim of the study is to analyze the extent of a failure, once it happens, favored by propagation through existing dependencies, as well as the severity of the failure in hampering correct SG operation.

2.4 Model-based analysis

Addressing the analysis and evaluation of cyber-physical systems, as the smart distribution grid is, poses a number of challenging issues, including:

- complexity and scalability, because of the characteristics of SGs in terms of largeness, multiplicity of interactions and types of interdependencies involved;
- ability to integrate in the evaluation framework the effects of both accidental and malicious threats;
- ability to reproduce both structural aspects and temporal behaviors in a context characterized by the dynamics of involved components where events take place;
- ability to represent events that take place at different time scales, e.g., two or three orders of magnitude between ICT and EI component failure rates;

- ability to represent component states of both discrete (e.g., related to control actions, component failure status) and continuous (e.g., related to the electrical infrastructure) nature;
- both deterministic and stochastic events have to be considered to evaluate the measure of interest: the control policy comprises both scheduled actions and events reactions;
- even considering only the EI part of the system state, there is no closed formula to express the system state, i.e., at any given moment in time the state can only be estimated through an iterative procedure;
- there are many components interconnected through a predefined topology and interdependencies among them have a great impact on the measures of interest.

Major pursued approaches in this crucial and difficult task span both model-based evaluation and experimental techniques. In this thesis, a model-based approach is adopted.

Model-based evaluation is commonly used to support the analysis of dependable computer systems in all the phases of the system life cycle [39–41]. During the design phase, models allow to evaluate various alternatives. In assessing an already built system, they constitute a means for providing insights into specific aspects and for suggesting solutions for future releases. The modeling also allows to analyze the effects of system maintenance options and of possible changes or upgrades of the system configuration. Moreover, sensitivity analysis of the models is very useful in performing bottleneck analysis and optimizations. Various methods and tools for evaluations have been developed which provide support to the analyst, during the phases of definition and evaluation of the models. Combinatorial methods, model checking, state-based stochastic methods and discrete-event simulation are major representative approaches in this area; an overview is provided in [39, 40].

Model-based evaluation is generally cheap for manufacturers and has proven to be useful and versatile in all the phases of the system life cycle [41]. A model is an abstraction of a system "that highlights the important features of the system organization and provides ways of quantifying its properties neglecting all those details that are relevant for the actual implementation, but that are marginal for the objective of the study" [42]. Several types of models are currently used in practice. The most appropriate type of model depends upon the complexity of the system, the specific aspects to be studied, the attributes to be evaluated, the accuracy required, and the resources available for the study.

Simple combinatorial techniques [43] are not adequate to deal with the complex interdependencies of critical infrastructures, therefore they are not appropriate in this context. State-based stochastic methods are instead adequate to deal with complexity and interdependencies; the main disadvantage of this category is the well-known state-space explosion problem since the dimension of the state space grows exponentially with the number of parts. This problem has triggered many studies, and significant results have been achieved in the last years. The two general approaches for dealing with the state explosion problem are largeness avoidance and largeness tolerance [40]. Largeness avoidance techniques try to circumvent the generation of large models using, for example focusing on Markovian stochastic modeling [43, 44], state truncation methods, state lumping techniques [45, 46], hierarchical model solution methods [47, 48], model decomposition and approximate solution techniques.

However, these techniques may not be sufficient as the resulting model may still be large. Thus, largeness tolerance techniques are needed to provide practical modeling support to facilitate the generation of large state-space models through the use of structured model composition approaches. The basic idea is to build the system model from the composition of submodels describing system components and their interactions. Generic rules are defined for the elaboration of the submodels and their interconnection.

Usually, higher-level modeling formalisms such as Stochastic Petri Nets (SPNs) [43] and their extensions, are used to support these approaches and generate automatically the Markov chain. It is worth noting that the two categories of techniques (largeness avoidance and largeness tolerance) are complementary and, most of the time, both of them are used when detailed and large dependability models need to be generated and processed, putting more emphasis on one or the other. A number of modeling approaches based on largeness avoidance and largeness tolerance have appeared in the literature. They span: i) compositional modeling approaches, both at level of defining suitable composition operators to build models from a set of building blocks (particularly helpful when the modeled system exhibits symmetries), as well as defining composition rules that allow structuring the modeled system into different abstraction layers with a model associated to each level; ii) decomposition/aggregation modeling approaches, where the overall model is decoupled in simpler and more tractable sub-models, and the measures obtained from the solution of the sub-models are then aggregated to compute those concerning the overall model; iii) derivation of dependability models from high-level specification, e.g. from UML design [49]. Other examples of higher-level formalisms include Performance Evaluation Process Algebra [50, 51] and Stochastic Fault Trees [52], and formalisms capable to tackle

semi-Markov and non Markovian stochastic models, such as the Stochastic Activity Networks (SANs) [53].

Rather than generate and analyze the entire state space, the simulation technique samples many paths, independently of each other, through the state space and analyzes and evaluates the system only through them [1, 54]. The problem in simulation is how to ensure the statistical quality of the estimates. Especially, assuring that the estimator be unbiased and have low variance are the major issues especially when the measure under analysis is a small probability, as it is for typical dependability indicators (e.g., unreliability or unavailability). In fact, in case of analysis of rare events, applying standard simulation techniques may pose the problem that the statistical significance in the estimation of the target measures becomes very poor. As for the analytical approach, the accuracy of the obtained evaluation depends on the assumptions of the analyzed system as well as on the behavior of the environment, and on the simulation parameters. Anyway, discrete-event simulation is one of the most commonly used modeling techniques in practice, especially for highly complex systems, for which analytical solution is generally precluded (e.g., to overcome the exponential distribution for events occurrences, which is usually implied by the analytical solution).

In this thesis model-based analysis is exploited through SAN as formalism to express the system architecture and stochastic phenomena.

For model solution, discrete-event simulation is adopted because, as discussed in Section 2.4.1, interdependencies between EI and ICT in the SG require detailed component models that interact in complex ways, and the aforementioned largeness tolerance and avoidance techniques rely on unsatisfied assumptions.

2.4.1 Role of SG analysis and evaluation

In general, analysis and evaluation are required to identify vulnerabilities, accounting for interdependencies and interoperabilities among composing subsystems, so to understand what specific assets of the addressed critical system are utmost critical and need to be protected the most. Following this analysis, steps can be taken to mitigate the identified vulnerabilities, in an order that reflects the assessed level of criticality. In fact, if a single-point vulnerability or critical node of failure is damaged, entire systems, networks, or assets will be affected regardless of other protective measures in place to prevent disruption of services. Since it is unfeasible to protect all aspects of a critical infrastructure, as SGs are, assessment of the impact of different failure events allows for the prioritization of assets and resources based on their estimated criticality. In the SG context [55], this kind of analysis

is even more crucial, given its pervasiveness as a backbone to many other critical infrastructures. As an example, if an electric substation is damaged and the electricity goes out, railroad operations are hampered and the flow of area traffic is impacted, causing a decreased movement of commodities and potential complications for emergency services. Thus, that electric substation must be protected not only for the Energy Sector, but also for the safeguarding of other sectors infrastructure.

In this thesis, analysis of SG through stochastic model-based approach is conducted, offering a rich and sophisticated modeling framework exploitable to: i) track the SG state evolution in response to failure events (due to either accidental faults or attacks), useful to trace the propagation of failure and the triggering of related phenomena along time; ii) identify criticality of involved components and their potential to lead to blackouts when affected by a variety of failure events, in terms of their impact on dependability-related indicators. Advances in ICT systems have increased these interdependencies, enhancing sector operations but creating additional vulnerabilities and source of failures. Errors can be either *detected*, i.e., their presence is pointed out with proper messages and signals, or *latent*, they are present but not detected. A fault is *active* if it leads to an error, otherwise it is *dormant*. A computing system is defined *robust* when it is able to provide a proper service even in situations that exceed its specification [56]. A computing system operates in a *degraded mode* when a subset of the implemented services fails but the provided service is still acceptable since it is compliant with the specification of the service. In addition to that, due to the hierarchical topology of its assets, the existing electricity grid suffers from domino-effect failures.

2.5 The Möbius modeling environment

To implement and evaluate the approaches presented in this paper, the Möbius modeling framework [5] and its supporting tool Möbius [57] have been used. Among the formalisms available in Möbius, our models are defined using the Stochastic Activity Network (SAN) formalism [53], a stochastic extension of Petri nets based on the following primitives: *places*, *activities*, *arcs*, *input gates* and *output gates*.

Places in SANs have the same interpretation as in Petri Nets: they hold tokens, the number of tokens in a place is called the marking of that place, and the marking of the SAN is the vector containing the marking of all the places. Places are distinguished in *Plain* and *extended* places (represented as blue and orange circles, respectively). Plain places represent short types of the programming language C++, whereas extended places represent primitive

data types (like short, float, double) including also structures and arrays of primitive data types or of extended place types.

There are two types of *activities*, timed and instantaneous (represented by thick and thin blue bars, respectively). *Timed activities* are used to represent the time to complete a task (an operation or a communication), while *instantaneous activities* are used to abstract times deemed insignificant with respect to the measures of interest. Uncertainty about the length of the time represented by a timed activity is described by a continuous probability distribution function, called the *activity time distribution function*, that can be a generally distributed random variables, and each distribution can depend on the marking of the network. Activities can have cases (indicated with small circles on activities). Cases are used to represent uncertainty about the action taken upon completion of an activity.

Gates connect activities and places. *Input gates* (represented by red triangles oriented to the left) are connected to one or more places and one single activity. They have a predicate, a Boolean function of the markings of the connected places that controls the enabling of the activity, and an output function that defines the marking changes of the state of the system when an activity completes. When the predicate is true, the gate holds. *Output gates* (represented by black triangles oriented to the right) are connected to one or more places and the output side of an activity. If the activity has more than one case, output gates are connected to a single case. Output gates have only an output function, which defines the marking changes of the state of the system when an activity completes. Gate functions (both for input and output gates) provide flexibility in defining how the markings of connected places change when the time represented by an activity expires.

Arcs in SANs are default gates, defined to duplicate the behavior of arcs in Petri nets. Thus, arcs are directed. Each arc connects a single place and a single activity. The arc is an input arc if it is drawn from a place to an activity. An output arc is drawn from an activity to a place. An input arc holds if there is at least one token in the connected place. The function of an input arc removes a token from the connected place, while the function of an output arc adds a token to the connected place. An activity is thus enabled only when i) all of its input gates hold, and ii) all of its input arcs hold.

In the following we give some further details on two particularly important aspects: the execution of a timed activity, and the measure specifications.

- Completion rules: When an activity becomes enabled, it is activated, and the time between activation and the scheduled completion of an activity, called the *activity time*, is sampled from a specified distribution. Upon completion of an activity, the following events take place:

i) if the activity has cases, a case is (probabilistically) chosen; ii) the functions of all the connected input gates are executed; iii) tokens are removed from places connected by input arcs; iv) the functions of all the output gates connected to the chosen case are executed; v) tokens are added to places that are connected by output arcs to the chosen case. An activity is aborted when the SAN moves into a new stable marking in which at least one input gate no longer holds.

- Measures definition: Upon completing the model of the system, a modeler has to specify the measures in terms of the model. In the SAN modeling framework, the measures are specified in terms of reward variables [53]. Let $R(s)$ be the reward associated to state s, and let $C(a)$ be the reward earned upon completion of transition a. If $\{X_t, t > 0\}$ is the modeled stochastic process and $\mathbb{S}$ the set of all possible states, a reward variable collected at an instant of time conventionally denoted by V_t is informally defined as

$$\sum_s R(s)\,\mathbb{P}(X_t = s) + \sum_a C(a)\,I_t^a, \tag{2.1}$$

where I_t^a is the indicator of the event that a was the activity that completed to bring the SAN into the marking observed at time t. The steady-state reward can be obtained as $V_{t\to\infty}$. In the case in which the reward variable is evaluated considering an interval of time $[t, t+l]$, then the accumulated reward is related both to the number of times each activity completes and to the time spent in a particular marking during the interval. More precisely,

$$Y_{[t,t+l]} = \sum_s R(s)\,J^s_{[t,t+l]} + \sum_a C(a)\,N^a_{[t,t+l]}, \tag{2.2}$$

where $J^s_{[t,t+l]}$ is a random variable representing the total time that the SAN is in the marking s during $[t, t+l]$ and $N^a_{[t,t+l]}$ is a random variable representing the number of completions of activity a during [t,t+l].

Möbius provides an infrastructure to support multiple interacting modeling formalisms and solvers. Reporting from the Möbius description at [58], the main features of the tool include:

- multiple modeling languages, based on either graphical or textual representations. Supported model types include stochastic extensions to Petri nets, Markov chains and extensions, and stochastic process algebras. Models are constructed with the right level of detail, and customized to the specific behavior of the system of interest;

- hierarchical modeling paradigm. Models are built from the ground up. First specify the behavior of individual components, and then combine the components to create a model of the complete system. It is easy to combine components in multiple ways to examine alternative system designs;

- customized measures of system properties, with ability to construct detailed expressions that measure the exact information desired about the system (e.g., reliability, availability, performance, and security). Measurements can be conducted at specific time points, over periods of time, or when the system reaches steady state;

- study the behavior of the system under a variety of operating conditions. The functionality of the system can be defined as model input parameters, and then the behavior of the system can be automatically studied across wide ranges of input parameter values to determine safe operating ranges, to determine important system constraints, and to study system behaviors that could be difficult to measure experimentally with prototypes;

- distributed discrete-event simulation. The tool evaluates the custom measures using efficient simulation algorithms to repeatedly execute the system, either on the local machine or in a distributed fashion across a cluster of machines, and gather statistical results of the measures;

- numerical solution techniques. Exact solutions can be calculated for many classes of models, and advances in state-space computation and generation techniques make it possible to solve models with tens of millions of states. Previously, such models could be solved only by simulation.

In Möbius, submodels can be composed hierarchically using operators defined at level of Abstract Functional Interface (AFI) [5,59] by sharing State Variables (SVs) or actions. In particular, the *Join* and *Rep* state-sharing compositional operators [33] are supported, respectively, to bring together two or more submodels or to automatically construct identical replicas of a submodel.

All the formalisms and solvers supported by Möbius are based on, and defined in terms of, C++ code, embracing the object oriented programming paradigm. Thus, the tool supports external C++ data structures, statically defined at compilation time, and the linking of external C++ libraries.

The Möbius modeling framework offers the facilities necessary to implement the considered approaches to model replication of non-anonymous

components addressed in this paper. The modeler can define, through a Graphical User Interface, the C++ code of submodels, composed models, reward structures (performance variables) and studies (assignments of actual values to models parameters). The Möbius GUI produces also an xml file for each submodel, composed model, etc. To the purpose of actually realize one of the compositional operator described in Section 4.5.3, several auxiliary scripts that work on Möbius' xml files have been implemented through Perl [60] programs.

Chapter 3

Basic Smart Grid model

In this chapter, a basic modeling framework, based on SAN [53], to quantitatively assess representative indicators of the resilience and quality of service of the distribution grid is presented. The model design and preliminary studies were carried out within the EU SmartC2Net [28] project and are the starting point of the current work. A description of the model core concepts is offered in Sections 3.1, 3.2, 3.3.1 and 3.4, whereas the contributions of this thesis, mainly consisting in exercising the developed model in relevant case studies and model adjustments, are presented in Section 3.6. The following considerations were and still are at the basis of both the basic and enhanced SG models:

- the objective of the analysis is to investigate on the impact of failures, affecting one or more components of the smart grid, on indicators representative of the quality of the delivered service. As such, the interest is not on understanding the effectiveness of the individual components/-functionalities, for which a detailed representation in the models would be required. Therefore, the first step towards this modeling effort has been to identify the logical structure of the system components under analysis and related environment conditions, specified at a suitable abstraction level to allow relevant investigations on the impact of failures and interdependencies, without impairing model representation and solution. Moreover, the tools adopted for the development of the analysis are extremely versatile and allow the programming of the actions of interest in the system inside the tools primitives, thus updates are very easy;

- the definition and development of the framework follow a modular and compositional approach, to allow wide adaptability and customization towards a wide variety of potential grid scenarios. Rather general and

extended definition of the system components have been adopted, to promote reuse of the developed framework beyond the studies carried on in this thesis. Specifically, models are equipped with a number of parameters to account for as much as possible aspects of the system components/failure events potentially of interest. Then, when focusing on a specific scenario, some of such parameters could conveniently be put at values that make them neutral to the analysis purpose, thus representing a situation where the related captured aspects are actually absent. For example, if the analysis purpose concentrates on failures affecting one infrastructure only (ICT or electrical grid), failures affecting the other one can be easily neutralized by assigning zero to the parameters representing the relative failure events;

- metrics appropriate to be assessed through the developed modeling framework are not limited to those exercised in this thesis. These last are certainly key indicators, considering the stated purpose of the analysis, but not exhaustive of all the metrics that could be evaluated through the developed framework.

A description of the challenges encountered when dealing with larger and more complex SGs is presented in Section 3.7 as a motivation for the developments undertaken in Chapters 4 to 6.

3.1 Logical architecture of the Smart Grid

In this section, the main logical components of the SG to be considered in the modeling framework and their relationships are described, as derived from the description of the architecture [61,62] developed by the EU SmartC2Net project [28]. The focus is on SG at level of electrical distribution systems.

The considered SG is logically structured in two cooperating parts: the MCS based on the ICT and the EI. The EI represents the electrical infrastructure, which is responsible for generating electric power at medium and low voltage, and for transporting towards the final users the electric power received from the high voltage transmission system or generated at medium and low voltage. An example of (small) electrical distribution grid is depicted in Figure 3.1. The building blocks are nodes, arcs and electrical components. Nodes represent buses (horizontal bars in the figure) and can host different electrical components according to the level they belong to. In particular, power generators are represented here as circles labeled with "G". An arrow is drawn from the generator to the bus it inject power to. Generators belongs to different categories and, when needed, additional information are

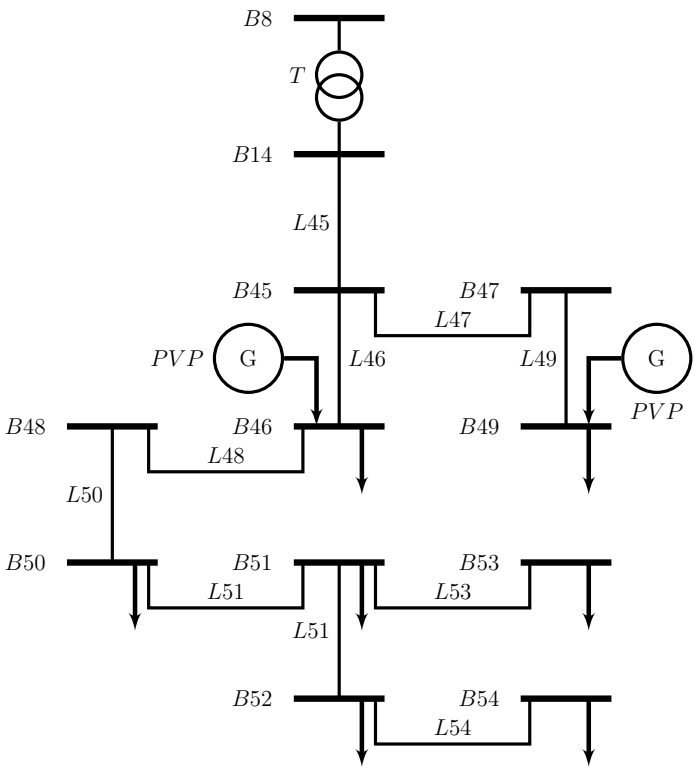


Figure 3.1: Schematic representation of a small portion of a real-world electrical distribution grid.

reported, as for instance the case of a photovoltaic generator attached to buses $B46$ and $B49$ in Figure 3.1. Loads, instead, are represented as arrows from the bus downwards. Arcs represent electrical branches together with switches (SW) and other control facilities. In order to represent both the backbone of the electrical grid and the electrical components attached to it, a more abstract, logical, picture is often considered. In particular, a bipartite graph is drawn where two kinds of elements are connected: nodes and arcs, represented as boxes in Figure 3.2. Each box comprises key elements, such as buses together with the set of electrical component attached to it (node box), and Power Line (PL) (arc box). Further details are depicted in Figures 3.3 and 3.4, where the following components, belonging to either MV or LV (or both) level(s), are represented:

- power stations (generators), i.e., sources of energy for the distribution network. Bulk Generators (BGs) are classic non-dispersed generator that can generate electrical energy in bulk quantity, Flexible Generators (FGs) are generators that offer flexibility in the power profile, in the sense that they can readily respond to unexpected changes;

- Distributed Generators (DGs) are DERs placed at different level of the grid, such as MV and LV;

- Load stations (loads), i.e., equipments absorbing energy from the distribution network;

- Distributed Storage units (DSes), i.e., electrical equipments that can be considered alternatively (flexible) generator or (flexible) load, depending on the state of the power grid;

- electrical components, such as capacitor banks (CBs) or on-load tap changers (OLTCs), capable of affecting the quality of voltages and powers flowing within the grid;

- substations, i.e., an assembly of electrical equipments that allows the routing and control of electricity across the network. They generally have protections and control equipments that can remotely manipulate the behavior of CBs and OLTCs. Primary and secondary substations reduce the voltage to a value suitable for medium and low voltage level, respectively;

- power lines (e.g., overhead lines and cables) connecting stations and substations.

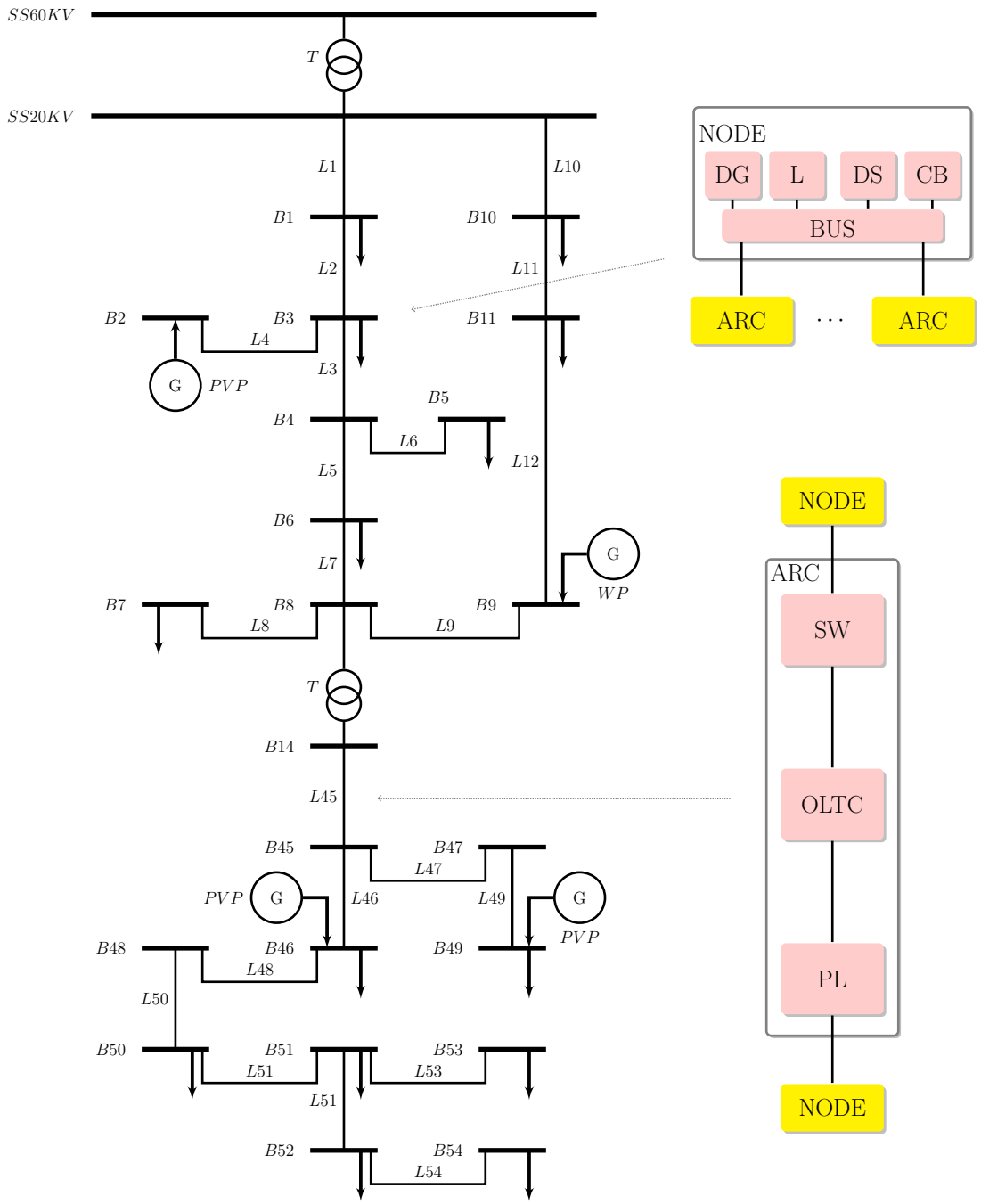


Figure 3.2: Logical scheme of how nodes and arcs of an electrical grid are represented.

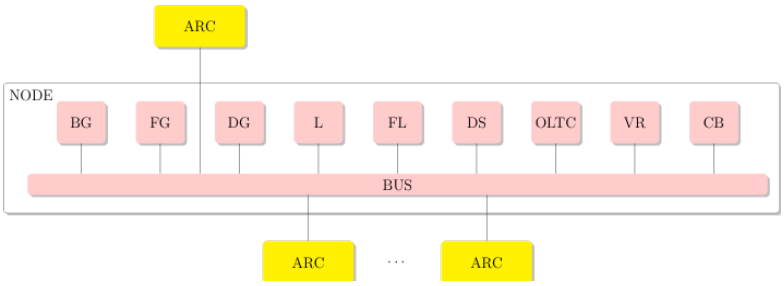


Figure 3.3: Structure of the logical node at MV level.

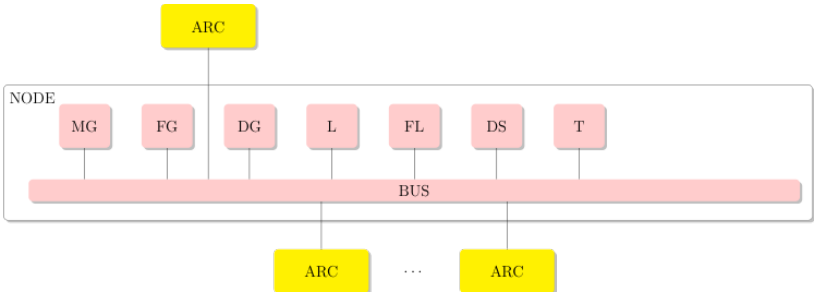


Figure 3.4: Structure of the logical node at LV level.

The topology of EI is typically a radial graph or, for redundancy purposes, a partially meshed graph. The important role of the topology in determining model improvements will be discussed in Section 3.7 and Chapter 4.

The MCS monitors and controls the physical parameters of EI, and is hierarchically structured in three main logical components, as shown in Fig. 3.5. In Figure 3.5, a complete picture of the SG logical architecture is depicted. Both EI and ICT are hierarchically structured to follow the HV, MV and LV partition of the EI. Transformers, represented as crossing circumferences, are placed between two boundaries in the EI to guarantee the voltage step from HV to MV and from MV to LV.

The ICT infrastructure can be described, at high level, as comprising three types of components, interconnected among each others: a group of MCSs, one for each level of the electrical grid, the communication infrastructure (represented by WAN, AN and Internet in Figure 3.5) and a set of Local Controllers (LCs) placed on the electrical components. The LCs send and receive data to the MCSs and implement the logic of the actuators responsible for applying commands coming from the MCSs. At medium voltage, one MVGC is associated to each different primary substation. MVGC monitors (in order to diagnose faults) and controls (to actuate appropriate set-points when needed) the assigned area, i.e., all the components located

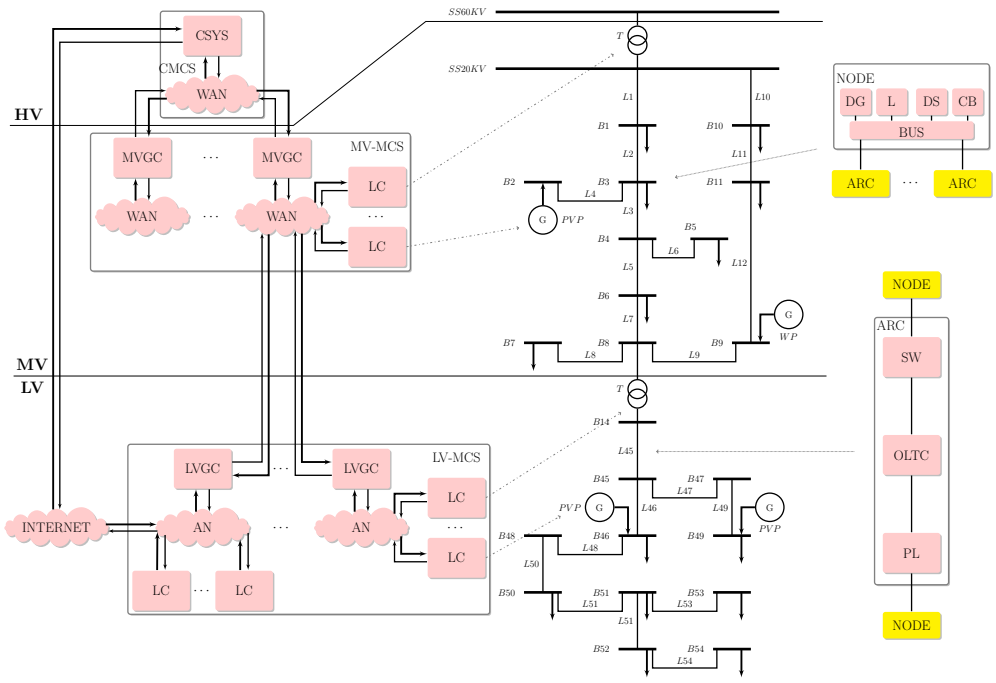


Figure 3.5: Complete picture of the SG logical architecture.

along the feeders emanating from the primary substation that can be remotely controlled by an operator. At low voltage, one Low Voltage Grid Control (LVGC) is associated to each different secondary substation. LVGC monitors and controls all the low-voltage electrical components connected to the secondary substation. Since MVGC and LVGC are not directly connected to the controlled components, the set-points are put in operation through the pertinent LC.

In the envisioned near future, in presence of high penetration of ICT at the distribution level, MVGC and LVGC will be able to influence the behavior of not only CBs and OLTCs but also DGs and FLs, so these components are explicitly addressed in the model.

To keep the modeling framework manageable and also considering the objective of the analysis (that is, assessing the impact of failures), all the control operations performed by MCS on EI are not represented in detail within the model. Rather, a simplified model is considered where only the effects on the distribution grid of the mitigation methods to cope with EI malfunctions, namely generation redispatch, load shedding, grid reconfiguration or voltage-var control are accounted for [2]. MVGC and LVGC actions are characterized by:

- an activation condition, specifying the events that enable the control reaction: disruption-faults, intermittency of renewable sources, power demand variation, etc.,

- a control strategy that defines new set-points, based on redispatch (varying the generated power), load shedding (varying the load demand), voltage-var control, load balancing, power loss reduction and line overload reduction,

- a reaction delay, representing the computation and application time needed by the control to apply new set-points.

3.2 Major assumptions

Major assumptions adopted in the modeling framework relate three aspects [62, 63]: failures and time. The focus of the SG analysis is on the impact of effects of faults, that is, the failures they induce, on the ability of the grid to deliver correct service, therefore a failure model instead of a fault model is considered.

The failure model assumed for EI and MCS is based both on the accidental and malicious faults (i.e., intentional attacks). Accidental failures are considered for both EI and MCS, whereas, malicious faults are only considered for MCS. Accidental failures of the electrical components can be transient, and in this case they last only for a certain time interval after which the component resumes correct operation, or permanent, that is they lasts until the failed components are repaired. A failure of an electrical component may affect in principle both the EI infrastructure (namely, the electrical quantities such as voltage, or the grid topology) and the MCS (by undermining the efficiency). In terms of the effect they produce, electrical components' failures can be summarized in:

i) "stuck at" failures, when the numerical value determined by the presence/absence of a given electrical component, such as the P^{inj} for a DG, is supposed to vary along time but, due to failure, it remains constant until the failure is recovered,

ii) "range failures", when an electrical component characterized by a range of states (such as an OLTC whose tap in normal conditions can assume values in $[-10, 10] \cap \mathbb{Z}$) is forced to operate only within a subrange of states (for instance the set $[1, 10] \cap \mathbb{N}$).

Sensors without actuators are considered perfect, so they are not directly addressed as a cause of failure.

Failures of components in MCS (including the communication networks) can be again transient or permanent. In addition, here we consider both accidental and malicious source for the MCS failures. In terms of the effects they produce, MCS failures can be summarized in:

i) content failures affecting the content of the service output,

ii) timing failures, when the timing of output delivery deviates from the time requested in correct conditions (early, late or omission timing failures).

Content malicious failures cause fake messages (e.g., for the Wide Area Network (WAN)) and fake set-points (for MVGC). Timing malicious failures cause delayed or lost messages, and delayed or lost set-points. This is a relatively comprehensive failure model, capable to address a variety of real-world scenarios. However, the framework allows extension towards a more refined failure model, at the price of possibly making the overall analysis and solution more complex and difficult.

Timing aspects are accounted for in the overall modeling framework. In particular:

- control operations have an associated time (≥ 0) within which they are accomplished (e.g., calculation of the set-points by the MVGC controls, as well as actuation time by the LC components);
- communication networks are characterized by a transmission time (again, ≥ 0), usually included within time bounds to recognize when a delay is experienced;
- repair/recovery times are also considered for both EI and control/communication components;
- time to failures for EI and MCS components (for these last, distinguished in time to accidental or malicious failures).

3.3 Smart Grid state

Due to the strong interconnection between EI and MCS, a failure in EI propagates to MCS and vice versa, possibly resulting in cascading or escalating failure. Formally, the EI and ICT can be considered as two separate infrastructure, and interdependency is a bidirectional relationship between them

through which the state of each infrastructure influences or is correlated to the state of the other. Thus, the next two subsections are devoted to the description of how EI can influence the state of ICT, and vice versa.

3.3.1 Electrical state definition

The state of EI is represented using mixed-integer values:

- the discrete values representing the components of EI that are connected to or disconnected from the electrical network described by the predefined topology,
- the continuous quantities associated to each component of EI: active power (P), reactive power (Q), voltage magnitude (V) and voltage phase (δ),
- the discrete or continuous quantities representing the current settings of the electrical control devices, like OLTC or CBs.

The electrical quantities representing the power $S_i = P_i + \jmath \mathrm{Q}_i \in \mathbb{C}$, the voltage magnitude and angle $V_i, \delta_i \in \mathbb{R}$ associated to each bus i are related by the Power-Flow Equations (PFEs):

$$\begin{aligned} P_i &= V_i \sum_{j=1}^{n} V_j \cdot |Y_{ij}^{\text{bus}}| \cos(\theta_{ij} - \delta_i + \delta_j), \\ Q_i &= -V_i \sum_{j=1}^{n} V_j \cdot |Y_{ij}^{\text{bus}}| \sin(\theta_{ij} - \delta_i + \delta_j), \end{aligned} \tag{3.1}$$

where $|Y_{ij}^{\text{bus}}|$ and θ_{ij} are the amplitude and phase angle of the ij–th entry of the admittance bus matrix, respectively [64].

In the following, vectors of homogeneous quantities within the grid will be indicated in bold, e.g., $\mathbf{S} \in \mathbb{C}^n$ is the vector of powers. Further details about the PFEs will be provided in Chapter 5, but it is important to notice that, at the level of abstraction considered here, the physics of electrical grids is governed by a set of $2n$ nonlinear equations, where n is the number of buses, that have closed formula solution only for $n = 2$. So, in realistic scenarios, it is only possible to estimate the state of the EI. In particular, the PFEs are numerically solved, as discussed in Section 3.7. After having solved the PFEs, obtaining the currents $\mathbf{I}$ flowing within the grid is a straightforward computation. A single-phase representation has been selected for the AC electrical grid. This is a trade-off between the need to tackle non-linear

phenomena and the complexity of the analysis. In particular, the selected per-unit system is the standard for single-phase: base complex power and base complex voltage are specified, base current is deduced.

At every time instant t, the impact of the components directly linked to the i-th bus can be represented by the following equations:

$$\begin{aligned} P_i &= \mathbb{1}_{i\in\mathrm{DG}} P_i^{\mathrm{inj}} - \mathbb{1}_{i\in\mathrm{FL}} P_i^{\mathrm{dis}} - \mathbb{1}_{i\in\mathrm{NFL}} P_i^{\mathrm{load}}, \\ \mathrm{Q}_i &= \mathbb{1}_{i\in\mathrm{DG}} \mathrm{Q}_i^{\mathrm{inj}} - \mathbb{1}_{i\in\mathrm{FL}} \mathrm{Q}_i^{\mathrm{dis}} - \mathbb{1}_{i\in\mathrm{NFL}} \mathrm{Q}_i^{\mathrm{load}} + \mathbb{1}_{i\in\mathrm{CB}} \mathrm{Q}_{cb} \cdot \mathrm{bank}_i, \\ V_j &= V_i \cdot f(\mathrm{tap}_i) \text{ if there is an OLTC on line } i-j, \end{aligned} \tag{3.2}$$

where

- $i \in$ DG if a distributed generator is attached to bus i, and $\mathbb{1}_{i\in\mathrm{DG}}$ is equal to 1 only if $i \in$ DG. Similarly for flexible loads, non-flexible loads, capacitor banks and on-load tap changers,

- $P_i^{\mathrm{av}}(t)$ is the stochastic process defining the maximum amount of active power that a DG can inject within the grid at time t. modeling the *available* active power P^{av} is particularly challenging for DERs because it is highly influenced by the weather (see Figure 3.19),

- $P_i^{\mathrm{inj}}(t)$ is the active power actually injected at time t. If neither MVGC nor LVGC have control capabilities on the DG attached to bus i then $P_i^{\mathrm{inj}}(t) = P_i^{\mathrm{av}}(t)$, otherwise $P_i^{\mathrm{inj}}(t)$ can be chosen to meet Quality of Service (QoS) requirements,

- For every time instant t, $0 \leq P_i^{\mathrm{inj}}(t) \leq P_i^{\mathrm{av}}(t)$,

- following the definitions of FL and NFL presented in Section 2.1, $P_i^{\mathrm{load}}(t)$ is the stochastic process defining the active power required by the NFL attached to bus i, and $0 \leq P_i^{\mathrm{dis}}(t) \leq P_i^{\mathrm{dem}}(t)$, where $P_i^{\mathrm{dem}}(t)$ is the stochastic process defining the maximum amount of active power that a FL can *demand* at time t. Both $P_i^{\mathrm{load}}(t)$ and $P_i^{\mathrm{dem}}(t)$ are modeled on the basis of past costumers power consumptions,

- $P_i^{\mathrm{dis}}(t)$ is the active power actually *dispatched* to the FL at bus i. If neither MVGC nor LVGC have control capabilities on the FL attached to bus i then $P_i^{\mathrm{dis}}(t) = P_i^{\mathrm{dem}}(t)$, otherwise $P_i^{\mathrm{dis}}(t)$ can be chosen to meet QoS requirements,

- $\mathrm{Q}_i^{\mathrm{inj}}, \mathrm{Q}_i^{\mathrm{dis}}$ and $\mathrm{Q}_i^{\mathrm{load}}$ are defined in terms of $P_i^{\mathrm{inj}}, P_i^{\mathrm{dis}}$ and P_i^{load} following

a standard expression involving the constant power factor (ρ), namely:

$$\begin{aligned} \mathrm{Q}_i^{\mathrm{inj}} &= \frac{\sqrt{1-\rho^2}}{\rho} P_i^{\mathrm{inj}}, \\ \mathrm{Q}_i^{\mathrm{dis}} &= \frac{\sqrt{1-\rho^2}}{\rho} P_i^{\mathrm{dis}}, \\ \mathrm{Q}_i^{\mathrm{load}} &= \frac{\sqrt{1-\rho^2}}{\rho} P_i^{\mathrm{load}}, \end{aligned} \tag{3.3}$$

- a CB comprises several banks, each capable to inject an amount of reactive power equal to Q_{cb}. The number of banks actually working at bus i is $\mathrm{bank}_i \in \mathbb{N}$. The control capabilities of MVGC or LVGC on the DG are represented by the possibility of changing bank_i,

- an OLTC is a transformer, i.e., an electrical component capable to impose a voltage drop through the ends of an electrical line, with variable factor determined by the function f of $\mathrm{tap}_i \in \mathbb{N}$. When correctly functioning, MVGC and LVGC have always the ability to change tap_i for OLTC under their control. More details about f can be found in [2].

As already mentioned, $P_i^{\mathrm{av}}(t)$ is a stochastic process, defined by the modeler. In particular, among several alternatives discussed in the literature, in this thesis the following processes have been selected:

- given a deterministic function $f(t) \in \mathbb{R}$, at every time instant $t = 300 \cdot h$, for $h = 0, \ldots, 288$, a random value p, uniformly distributed on a fixed interval, is selected and $P_i^{\mathrm{av}}(t)$ is defined as

$$P_i^{\mathrm{av}}(t) = f(t) + p.$$

- given an array of values $f = [f_0, \ldots, f_{287}]$, a random time t is selected uniformly in the interval $[0, 300)$ and $P_i^{\mathrm{av}}(t)$ is defined as

$$P_i^{\mathrm{av}}(h \cdot 300 + t) = f(h).$$

Thus, in both cases the time scale is 300 seconds, i.e., 5 minutes, but in the former case time is fixed and power is randomly selected, whereas in the latter case the power is fixed and the time of change is random.

3.3.2 Monitoring and Control System logic

Algorithm 1 describes the sequence of events and actions characterizing the interaction between EI and MCS for what concerns the voltage and current QoS management. An event loop captures new values of powers, voltages or currents within the grid. Whenever there are out of bound values, the MCS switches to the *critical state*: if the problem can be solved applying a predefined control policy then the critical state is removed and the MCS switches back to the *waiting* state, otherwise the out of bound component is disconnected from the grid, the MCS switches to the *waiting* state. Both MV and LV components can be disconnected. Notice that disconnecting an electrical component from the grid always produces new values of voltages and currents. Being Equation (3.1) nonlinear, a new value of V_i can imply V_j out of bound for $j \neq i$ quite far from i. Notice also that Algorithm 1 can promote cascading effects on the grid, eventually producing a blackout. A key aspect of the model is that Step 2 in Algorithm 1 can involve all the components included in "node" or in "arc" where a random event, that changes the state of the grid, occurs. Steps 3 and 4 result in a new electrical state of EI, derived, without control actions, by solving the system of $2n$ equations (3.1) through the Newton–Raphson method [65]. Steps 6 is performed by the MVGC (or LVGC) affected by the events on the node i. Considering control variables as known quantities, new set points are calculated by MCS by searching the best combination of values of the control variables for which the unknown quantities satisfy the required electrical bounds (no over voltage, no under voltage and no overloaded power line). Thus MCS solves an Optimal Power Flow (OPF) problem, that is a non linear and non convex optimization problem, with mixed control variables, where the objective function is described in Equation (3.4) and the nonlinear constraint are the equations provided by (3.1). Different control functions are in place to accomplish the overall control task. The MV voltage control function implemented in the framework can be based on OLTCs, on CBs, on the reactive power of DGs or on the curtailment of active power available on the DGs that should be minimized in order to maximize the use of the renewable energy. Taking the control cost and power curtailments into consideration, the objective function is decided

to be:

$$\begin{aligned}\min \sum_{i\in\text{buses}} \Big(\; & w_{\text{bound}}|V^{\text{ref}} - V_i| \\ & + w_{\text{curt}}\,\mathbb{1}_{i\in\text{DG}}\Big(P_i^{\text{av}} - P_i^{\text{inj}}\Big) \\ & + w_{\text{shed}}\,\mathbb{1}_{i\in\text{FL}}\Big(P_i^{\text{dem}} - P_i^{\text{dis}}\Big) \\ & + w_{\text{bank}}\,\mathbb{1}_{i\in\text{CB}}|\text{bank}_i^{\text{nominal}} - \text{bank}_i| \\ & + w_{\text{tap}}\,\mathbb{1}_{i\in\text{OLTC}}|\text{tap}_i^{\text{nominal}} - \text{tap}_i|\Big)\end{aligned} \tag{3.4}$$

where, V^{ref} is the reference value for voltages. In the electrical community V^{ref} is often expressed as 1 per unit (p.u.), where typical values of p.u. in Europe are $20kV$ for nodes at MV and $220V$ for nodes at LV. The reduction of the injected power, i.e., determining P_i^{inj}, is called *curtailment* and the reduction of dispatched power, i.e., determining P_i^{dis}, is called *load shadding*. The MCS control policies are modeled selecting different values for the weights

$$w_{\text{bound}} \geq w_{\text{curt}} \geq w_{\text{shed}} \geq w_{\text{bank}} \geq w_{\text{tap}}$$

and the reference point for $\text{bank}^{\text{nominal}}$ and $\text{tap}^{\text{nominal}}$. More details on the objective function can be found in [66]. Here, it is sufficient to notice that weights can model priorities among control actions. For instance, if w_{curt} and w_{shed} are chosen such that $\frac{w_{\text{curt}}}{w_{\text{shed}}} \geq 10^3$, then power curtailment is supposed to have a greater priority then load shedding. Notice though that priorities does not imply the presence of a total ordering among control actions, and in fact partial orders can be considered. The objective function detailed in Equation (3.4) stresses the distance between reference value and candidate value, but different choices are also possible. For instance, the actual voltage bounds can be included (see Section 6.2.1). The point here is that the modeler has the freedom to select the best optimization problem with respect to the specific context of application and the intended analysis.

Step 7 of Algorithm 1 is triggered by MVGC (or LVGC), but involves the components WAN (or Access Network (AN)) and LC. Step 9 occurs when MCS cannot find a set point that removes the critical state caused by the event occurred at step 2. It can involve all the components included in node or in arc and triggers new values at next step 2.

Algorithm 1: Sequence of events and actions characterizing the interation between EI and MCS in absence of failures of MCS.

Data:
$\mathbf{P} = (P_1, \ldots, P_n), \mathbf{Q} = (Q_1, \ldots, Q_n), \mathbf{V} = (V_1, \ldots, V_n), \boldsymbol{\delta} = (\delta_1, \ldots, \delta_n), \mathbf{I} = (I_1, \ldots, I_m)$
$\mathbf{V}^{min}, \mathbf{V}^{max}$: bounds for voltage
$\mathbf{I}^{min}, \mathbf{I}^{max}$: bounds for current flow through lines
$\mathbf{u}^*$: set points for the control variables

1 **loop**
2 **wait until** *new values for* P_i, Q_i, V_i or δ_i
3 $\mathbf{P}, \mathbf{Q}, \mathbf{V}, \boldsymbol{\delta} \leftarrow$ solutions of Equation (3.1), based on the new values at bus i
4 $\mathbf{I} \leftarrow$ current flows through the power lines
5 **if** $\exists j \mid V_j < V_j^{min}$ *or* $V_j > V_j^{max}$ *or* $I_j < I_j^{min}$ *or* $I_j > I_j^{max}$ **then** `// critical state`
6 $\mathbf{u}^* \leftarrow$ solutions of the OPF described by Equation (3.4), based on the new values at bus i
7 $\mathbf{P}, \mathbf{Q}, \mathbf{V}, \boldsymbol{\delta} \leftarrow$ solutions of Equation (3.1), based on $\mathbf{u}^*$ and on the new values at bus i
8 **if** $\exists j \mid V_j < V_j^{min}$ *or* $V_j > V_j^{max}$ *or* $I_j < I_j^{min}$ *or* $I_j > I_j^{max}$ **then** `// critical state not removed`
9 components at node j disconnect from the network triggering at next step new values for P_j, Q_j, V_j or δ_j

3.4 Composed and atomic models

The overall model is implemented by means of the modeling and solution tool Möbius [57], briefly described in Section 2.5. The atomic models, characterizing the components behavior, are defined using the SAN formalism [53]. The composed models determine the architecture of the overall model joining the atomic models and defining the information exchange among them.

Describing the overall model from top to bottom, the logical architecture of the SG (depicted in Figure 3.5) is reflected within the composed model of Figure 3.6 that *Join* the MV and LV levels. The models MV_M and LV_M are defined composing the submodels representing MCS and EI at MV and LV level, respectively, as shown in Figure 3.7. They represent different smart grid configurations, being input parameters: both the electrical grid topology and the components associated to each node or line of the grid. The topology

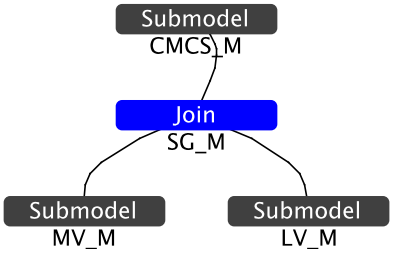


Figure 3.6: Composed model representing the overall SG

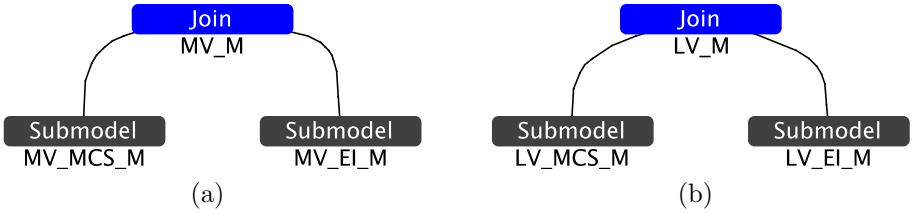


Figure 3.7: Composed models representing the SG at MV (a) and LV (b).

definition is a key aspect of the model and will be detailed in Section 3.5.

The composed model MV_EI_M, depicted in Figure 3.8, is defined composing three submodels. The atomic SAN model MV_EI_INIT_SAN triggers at time 0 the initialization of the model, i.e., loads the static parameters describing the case study under analysis and assigns the correct values to each place of the atomic SAN models. The submodel MV_N1AN2_Ms represents all the arcs of EI at MV with the associated starting and ending nodes. This submodel is devoted to represent the SG topology.

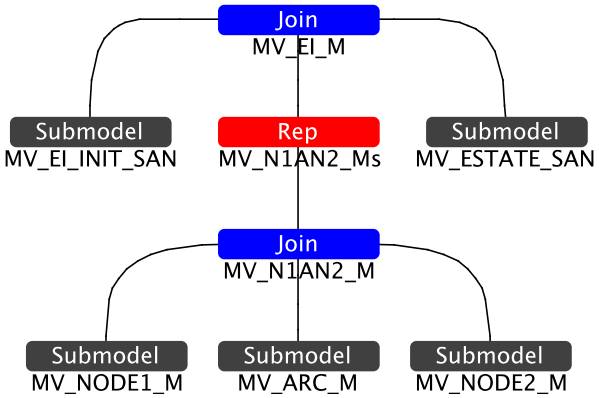


Figure 3.8: Composed model representing the MV-EI.

The atomic SAN model MV_ESTATE_SAN, shown in Figure 3.9, repre-

sents the steps 3 and 4 in Algorithm 1, i.e., the changes of the electrical state of EI based on the evaluation of the PFEs, in the output gate *NewES*, directly triggered by failures or other events occurring in EI (including the variation of the power generated by DGs, the variation of the power demand on loads or the variation of the power injected or absorbed by storage units). Output gate *updateRew* and linked places are used to obtain complex dependability or QoS measures.

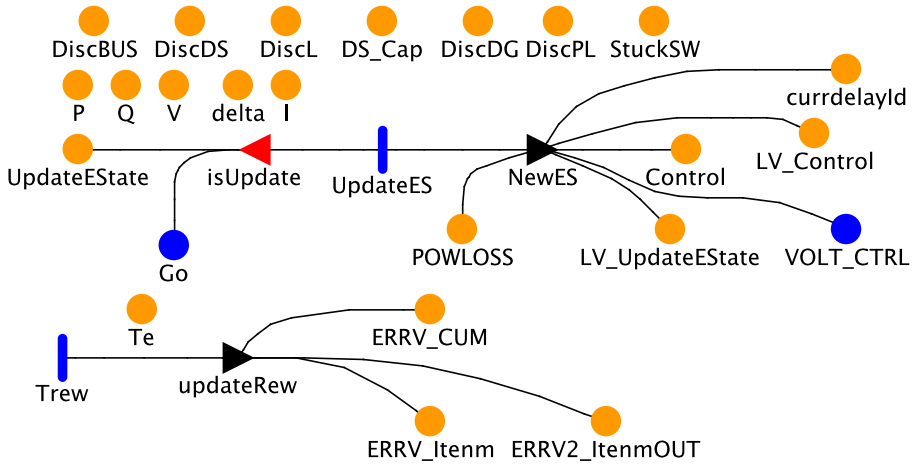


Figure 3.9: SAN model MV_ESTATE_SAN.

The template models MV_NODE1_M and MV_ARC_M, shown in Figure 3.10, are defined by composing the generic atomic SAN models representing the generic components that can be linked to each bus or node, respectively. Actions performed in steps 2 and 9 of Algorithm 1 are modeled through the atomic SANs shown in Figure 3.10.

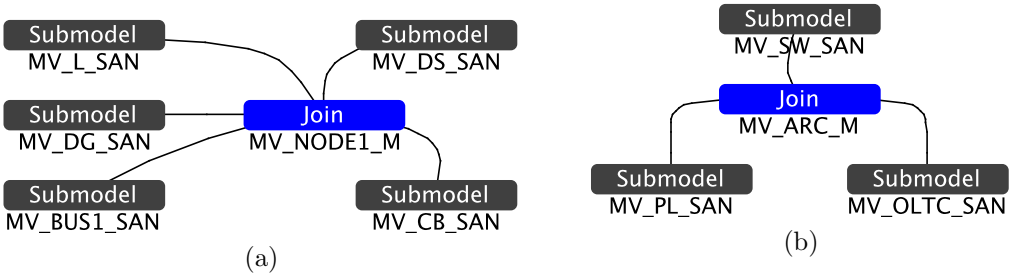


Figure 3.10: Composed template models representing a generic node (a) and a generic arc (b) of EI at MV.

The composed model MV_MCS_M, depicted in Figure 3.11, is defined composing four submodels. The model MV_LC_Ms represents all the MV

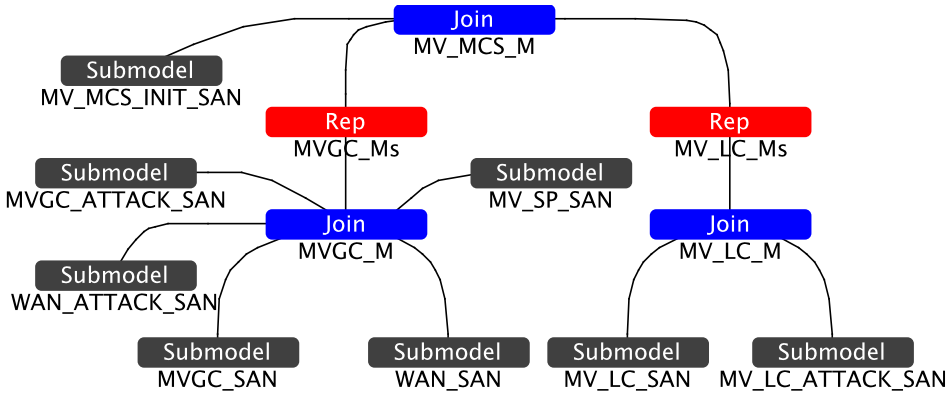


Figure 3.11: Composed model representing the template model for MV-MCS.

LCs. The model MVGCMs represents all the MVGC with the associated WAN shown in Figure 3.5. It is defined using the *rep* operator by replicating for each MVGC the template submodel MVGC_M. The template model MVGC_M represents a generic component MVGC with the associated generic WAN and it is composed by joining five template submodels. The atomic SAN models MVGC_ATTACK_SAN and WAN_ATTACK_SAN are the templates representing the malicious failures of MVGC and WAN, respectively, caused by an attack to these components. The atomic SAN model MV_SP_SAN is the template representing the changes of the state of EI at the actuation by LCs of the new set points derived by MVGC. It models the step 7, where new set points are actuated, based on the status of the components modeled in WAN_SAN and MV_LC_SAN.

SAN MVGC_SAN, as detailed in Figure 3.12, is a template model representing the occurrence of transient or permanent accidental failures (activity *AccidentalFailure*), the recovery actions (activities *TAccFRecovery* and *PAccFRecovery*), the step 5 of Algorithm 1, where the activation condition is checked, and the step 6, where the OPF problem is solved (in the gate *SetPoints*). Both these activities are based on the state of the WAN and LC defined in WAN_SAN and MV_LC_SAN, respectively.

SAN MV_DS_SAN, shown in Figure 3.13, is a template model representing for a generic DG: the generation of active and reactive power (gate *WeatherChange* and activity *WPChange*), the generation forecast and error (gates *isWPset* and *ForecastError*, activities *WP_NextSchedT* and *WPForecastChange*), the occurrence of failures and the recovery actions (at bottom and top right of the *SAN*). Further details can be found in [62].

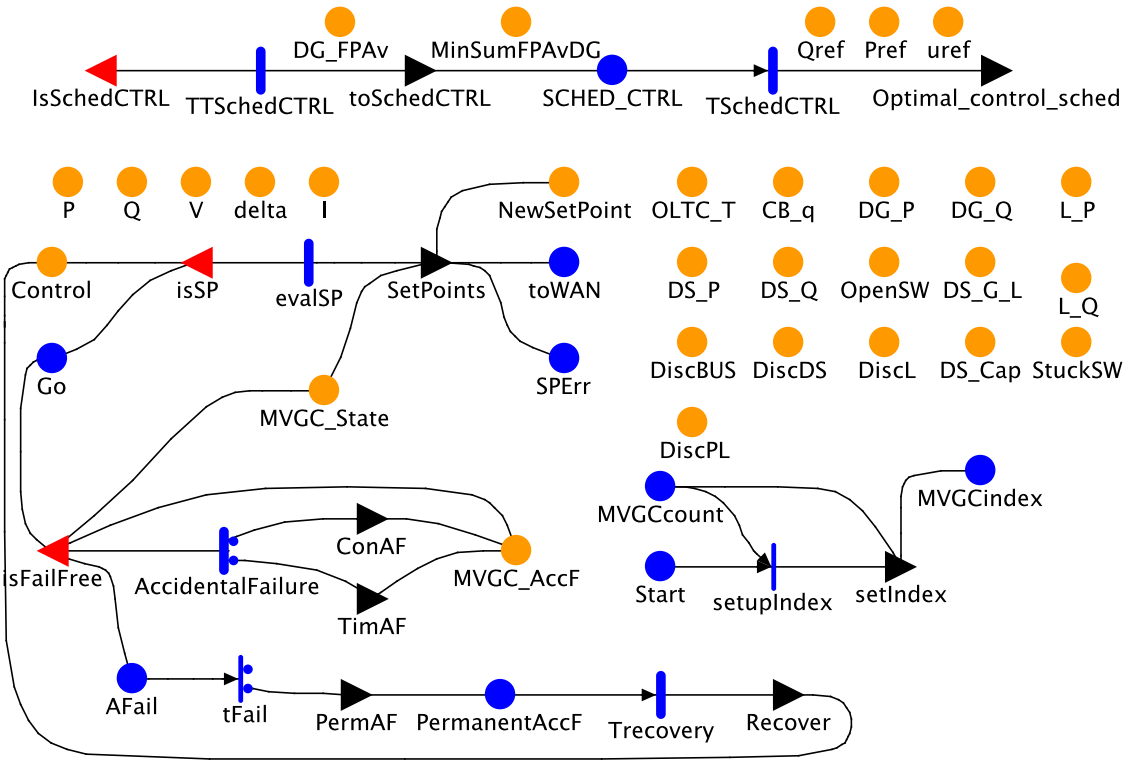


Figure 3.12: SAN model MVGC_SAN.

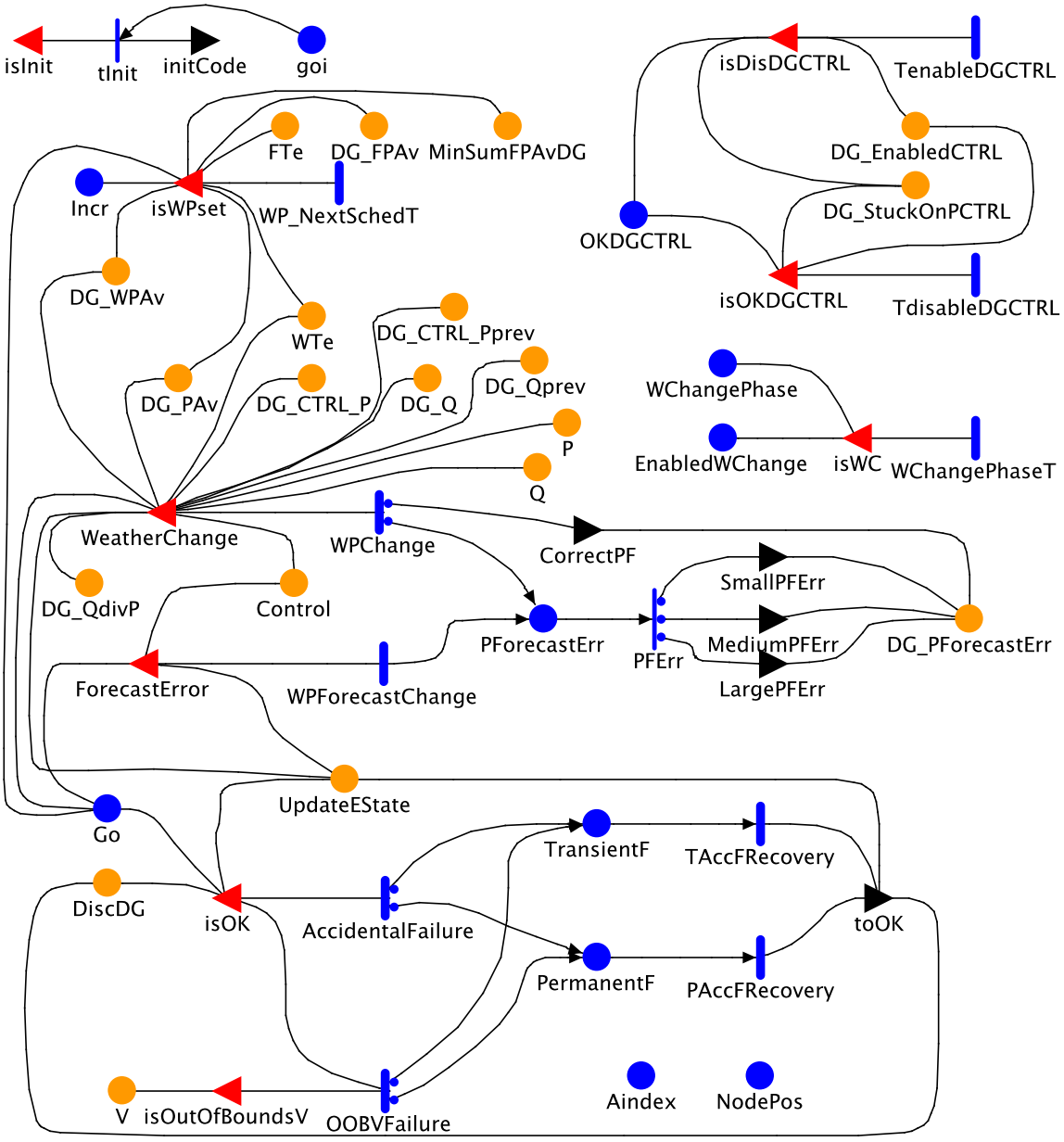


Figure 3.13: SAN model MV_DS_SAN.

3.5 Smart Grid topology modeling

The SG topology model MV_N1AN2_Ms is designed following the *generic modeling* approach: the modeler defines a set of template atomic and composed models that, at the beginning of the model solution, are instantiated to represent a specific case study. In other words, the model of a specific case study is not defined directly but through a template and a collection of static data that describe how to fill the template.

In particular, instead of manually designing a SAN model for each node and each arc of the SG, a template SAN node and template SAN arc are designed; at the beginning of the solution, the template models are replicated and each replica reads the static information about the specific electrical component it has to represent. In this way, each replica will behave differently and the SG topology is represented by the interaction of the instantiated replicas. In Figure 3.8 more details are depicted:

- a template model, called MV_N1AN2_M, is defined to represent a generic node-arc-node relation that does not depend on the specific case study under analysis,
- MV_N1AN2_M is a composed model and comprises three composed template submodels: MV_NODE1_M, MV_ARC_M and MV_NODE2_M, each representing a generic node or arc,
- MV_N1AN2_M is replicated using the *Rep* operator provided by Möbius,
- during the solution, each replica of MV_ARC_M is instantiated to a specific arc of the SG, and similarly MV_NODE1_M and MV_NODE2_M are instantiated to specific nodes,
- duplicated submodels are joined to form a unique submodel in the overall model.

As an example, consider the EI depicted in Figure 3.1. At the beginning of the model solution, 24 replicas of MV_N1AN2_M are created. When the replicas are instantiated, information about the specific arcs and nodes are taken into account: for instance, it will be possible to distinguish $B4 - L5 - B6$ from $B6 - L7 - B8$. Within the *Rep* of MV_N1AN2_Ms, duplicated submodels are joined to form an unique submodel: for instance, the $B6$ that is MV_NODE2_M for $L5$ and the $B6$ that is MV_NODE1_M for $L6$ are joined to deal with $B4 - L5 - B6 - L7 - B8$.

An important problem emerges from the description of generic modeling in Möbius: as discussed in Section 2.5, the *Rep* operator provided by Möbius

can produce only identical (anonymous) replicas but, to deal with the SG topology, each template replica needs to read information from static data and assigns to itself a specific role, representing a specific electrical component. The solution employed within this basic model is based on a race among SAN activities where each replica competes with the others to obtain an unique index. In particular, referring to the right bottom portion of Figure 3.14, the core idea can be summed up in the following points:

- the atomic model MV_PL_SAN is replicated n_{arcs} times, being part of the template model MV_ARC_M depicted in Figure 3.10b, the place *Acount* is shared among all the replicas, an the places *Start* and *Aindex* are local to each replica,
- at the beginning of the initialization phase, the place *Acount* contains n_{arcs} tokens, the place *Start* contains only one token and the number of tokens in place *Aindex* is irrelevant,
- when the initialization phase starts, all the *setupIndex* activities are enabled and can fire only once, because *Start* has only one token, and then there is a race among *setupIndex* activities for consuming the content of *Acount*,
- whenever one of the *setupIndex* activities fires it consume a token of *Acount* and, through the output gate *setIndex*, copy the content of *Acount* within *Aindex*,
- in this way the first replica in which *setupIndex* fires assign $n_{\text{arcs}} - 1$ to *Aindex*, the second assign $n_{\text{arcs}} - 2$, and so on, until the last replica assigns 0; in this way, at the end of the race, each replica will have an unique value within its *Aindex*, spanning from 0 to $n_{\text{arcs}} - 1$,
- the activities *setupIndex* are instantaneous and then the race has no impact on the measures defined on the model.

After having obtained a unique index, the replica is no more anonymous (in fact it is identified by its index) and can read information from a static data structure, e.g., a vector of records, and assign to itself only the portion of data relevant to behave like the electrical component it represents.

A very important aspect of the general modeling approach is the way template replicas interact. Within template SAN models, the symbols P, Q, V and *Delta* represent extended places of type array, that means they are arrays of places of size n, where n is the number of buses in the grid. These extended places are shared among *all* the replicas of MV_PL_SAN so

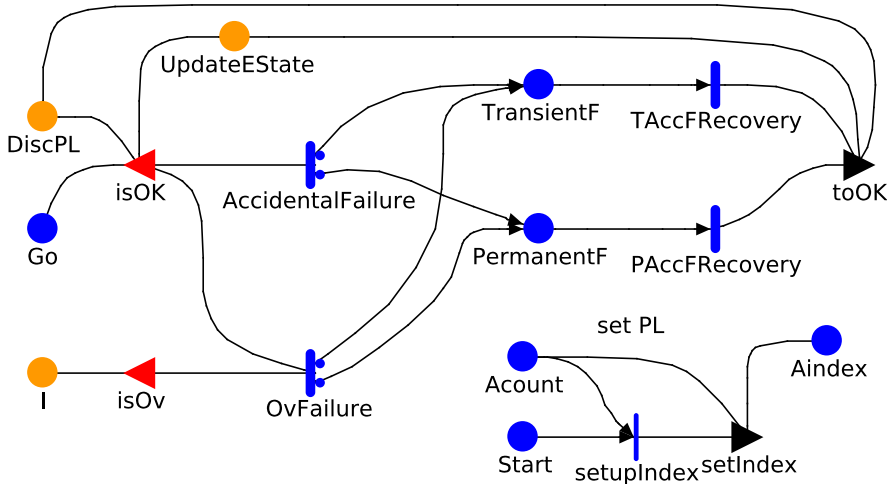


Figure 3.14: SAN model MV_PL_SAN.

that each replica – knowing its index – can access their value. In addition the extended places are shared also with the MV_ESTATE_SAN SAN, that keeps them in sync with the values produced by solving the PFEs, the MVGC_SAN SAN, that update them according to the solution of the optimization problem described by Equation (3.4).

In total, in the model there are 12 extended places shared among all the replicas of MV_PL_SAN and other SANs. The extended places maintain information about electrical values, such as P, and failure/working status of EI or ICT components, such as *DiscDG*.

This architectural choice, called in the rest of the thesis *State-Sharing Replication* (*SSRep*) approach, will be detailed in Section 4.4.3.

3.6 Definition of failure/attack scenarios

The contribution of this thesis within the context of the basic SG model consists in defining realistic scenarios of failure/attack and evaluating their impact on relevant performability measures, identified in Sections 2.2 and 2.5 as the most appropriate one to assess the quality of provided service in the SG context. Which measures are then evaluated is indicated when dealing with the specific scenarios. In order to do that, a real-world grid based on a small part of a Danish distribution network, with parameters provided by the Danish DSO HEF [61] has been considered as the starting point for several scenarios design. The grid, shown in Fig. 3.5, is composed of *11* MV buses and *13* LV buses, connected by *24* power lines.

3.6.1 Impact of MV control capabilities on LV level

In this scenario, all loads are considered constant and inflexible and there are 4 DGs: 2 at MV and 2 at LV. Only the DGs at MV level are controlled by the MVGC. The MVGC can also control the tap of the OLTC placed on the transformer between the HV and MV.

Here, only the failure of the MV control devices local to the photovoltaic power plant (PVP) and the WP at time *11:00* for an interval of *2* hours is considered. Being the control disconnected, all the produced power at PVP and WP is injected in the grid, i.e., during the failure $P_2^{\text{av}} = P_2^{\text{inj}}$ and $P_9^{\text{av}} = P_9^{\text{inj}}$. An observation window of *24* hours has been adopted.

A variety of measures of interest to final customers, service providers and operators can be defined. Here, only the following ones are reported:

- the voltage $V_i(t)$ on bus i at time t;
- P_i^{av} and P_i^{inj} on bus i at time t, i.e., a measure of the curtailment performed on the DGs;
- probability that the voltage requirement $MV1$ is fulfilled. In particular, the probability that the *10min*-mean value of V_i, for every bus i, is not within *10%* of the nominal voltage for *99%* of the time (European standard EN50160 [67]).

Fig. 3.15 shows that, in absence of control, the voltage of some LV nodes is out of bound. Fig. 3.16 shows the positive impact of the MV control on the voltage of those nodes that in Fig. 3.15 have the worst behavior. In fact, the voltage is always within *10%* of the nominal voltage, except for the period of time between *11:00* and *13:00*, when no curtailment is performed and all the MV DG available power is injected.

The effect of the absence of curtailment of available active power due to the failure of MV control devices on DGs in the time interval [11:00, 13:00] is illustrated in Fig. 3.17.

Fig. 3.18 shows an example of statistical analysis resulting from 128 simulation runs. At the top of the figure is depicted the mean of $V_{48}(t)$ and at the bottom the probability that the *10min*-mean value of $V_{48}(t)$ is not within *10%* of the nominal voltage. Interestingly, it can be observed that, even if the mean voltage over time is always in bound, the probability that the voltage is out of bound may be greater than 0 over time. Notice that a relevant role of the SG dynamics is played by the OLTC. In fact, during the day of analysis the tap changes considerably, in particular explaining the undervoltage depicted in Figure 3.15.

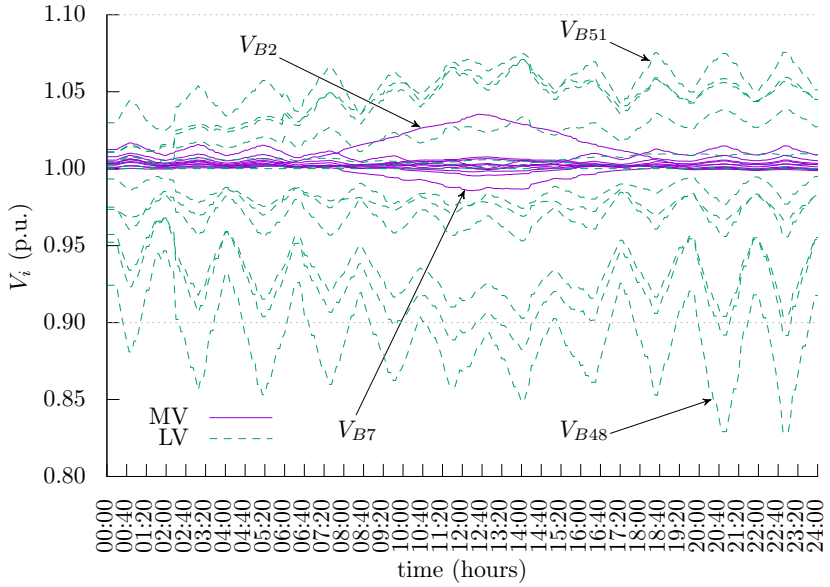


Figure 3.15: Voltage on each bus in absence of control, one simulation run.

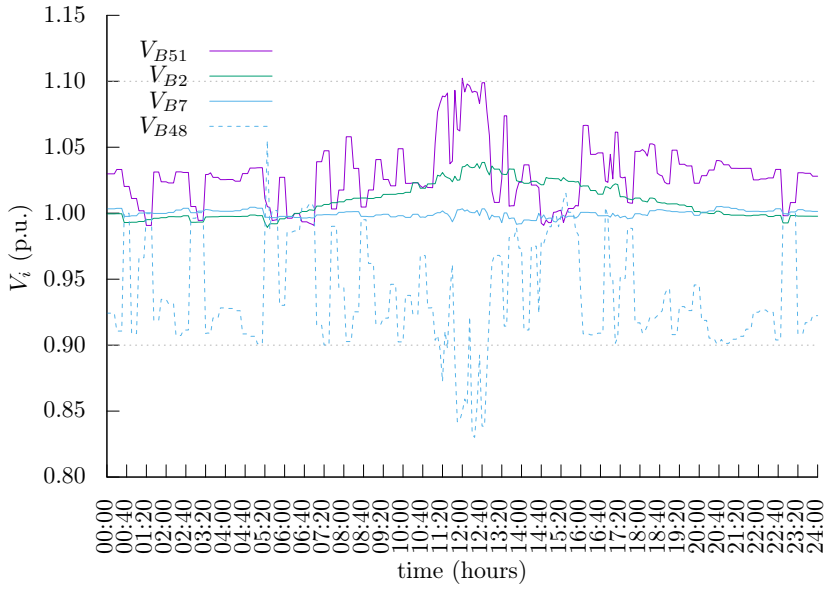


Figure 3.16: Voltage on buses *B2*, *B7*, *B48* and *B51* with control functionalities and failure of MV DG control on buses *B2* and *B9*, for one simulation run.

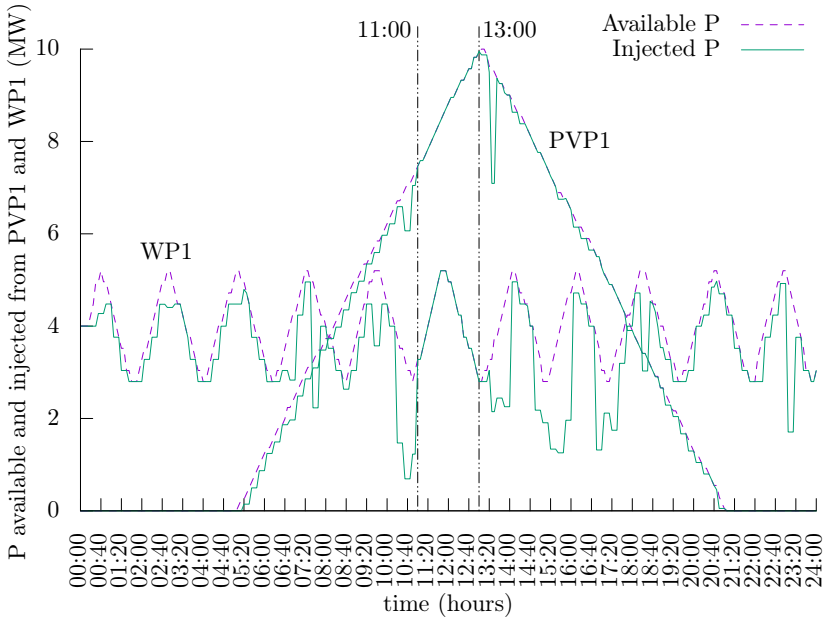


Figure 3.17: Available and injected active power generated by MV DG on buses *B2* (*PVP1*) and *B9* (*WP1*), for one simulation run.

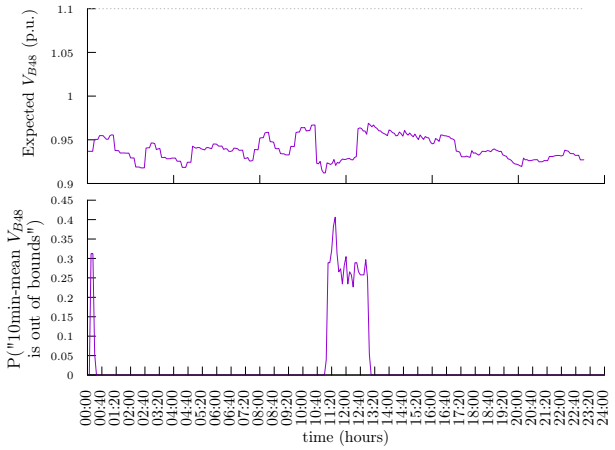


Figure 3.18: Mean voltage on buses *B48* vs the *10min*-mean value of $V_{B48}(t)$ is not within *10%* of the nominal voltage, for 128 simulation runs.

3.6.2 Impact of ICT failure/attack on EI

In this scenario only the MV level is considered and the interest is on failures affecting the cyber infrastructure responsible for the distribution grid control; specifically, three types of failure are considered:

- timing failure, resulting in delayed/omitted application of (part of) the control actions;
- control device failure, resulting in an incomplete application of the control actions. Specifically, the failure of control devices local to the distributed energy resources is tackled, leading to lack of control on the produced power and unavailability to perform curtailment of production to assure energy balancing;
- OLTC failure, potentially resulting in unsuccessful voltage control since OLTC constitutes a major device through which voltage regulation is performed.

Source for the considered failures can be in general either an intentional attack (e.g., to the communication network) or an accidental failure (e.g., a computer crash).

Given the interest in the voltage control functionality and its ability to promote resilient grid operation through fulfillment of voltage requirements, the following indicators have been evaluated:

1. the voltage $V_i(t)$ on bus i measured at each time instant t within the considered analysis period;
2. the probability that the value of $V_i(t)$ is out of bound of the nominal voltage: either undervoltage $UV_i(t)$ or overvoltage $OV_i(t)$;
3. the probability $\widetilde{P_i^{MV1}}$ that voltage requirement $MV1$ is not met on bus i (in order to simplify the analysis the requirement has been evaluated over the considered analysis interval of 24 hours);
4. the average unsatisfied power demand $UD_i(t)$ on load i at each instant of time t.

Metrics 2) and 3) are representative of the degree of reliability of the smart grid in delivering its service, while metric 4) expresses the effectiveness of the analyzed voltage control functionality in satisfying customers expectations.

The line parameters are shown in Table 3.1. Two DERs, a PVP and a WP, are linked to buses $B2$ and $B9$, respectively. The values of the active

Power line	**Resistance (Ohm/km)**	**Reactance (Ohm/km)**	**Length (km)**
L1,L2	0.10	0.10	1
L3,L5,L7	0.13	0.09	5
L4,L9	0.13	0.09	10
L6,L8	0.32	0.15	5

Table 3.1: MV line parameters for the grid depicted in Figure 3.2.

power P generated for each PVP and WP is modeled by a stochastic process, representing the value of P, and the associated Q, at each instant of time t, where the time between two consecutive updates is a random variable with distribution uniform and parameters $(0.5\mu, 1.5\mu)$, where $\mu = 6$ min is the mean. The expected values obtained from our model are shown in Figure 3.19. Note that different power profiles can be accounted for in the proposed modeling framework, by properly varying the distribution function and related parameters associated to the adopted stochastic process. Five

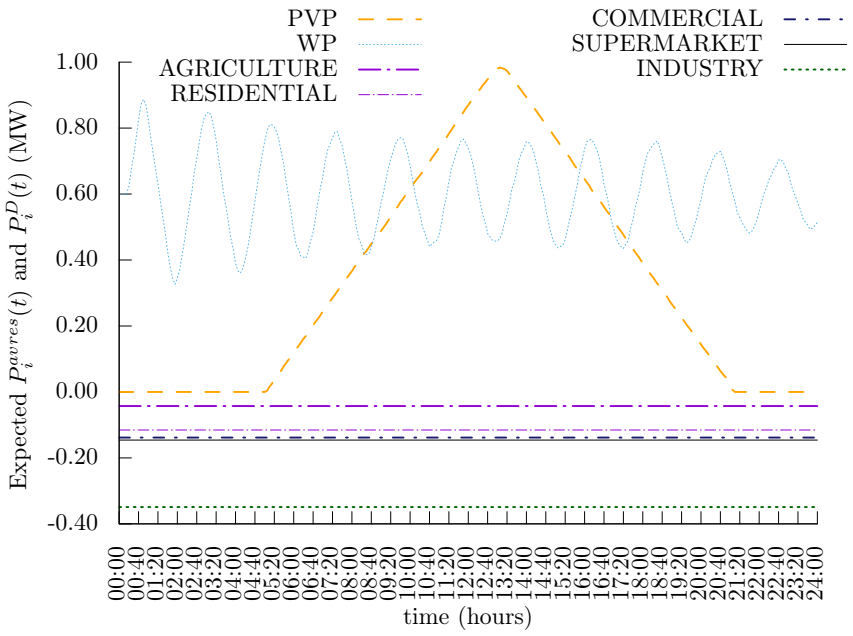


Figure 3.19: Expected available power P^{av} from DER and constant power demand P^{dem}.

loads are considered, representing different consumer types: Industry, Agriculture, Commercial, Supermarket and Residential. An OLTC is linked to

buses $B1$ and $B2$, with rating 50 MVA, 60/20 kV, resistance 3 Ohm, reactance 13 Ohm, 20 tap values, from -10 to 10, and voltage per tap 1.25%.

Failures occurrence in presence of constant power demand and inflexible loads. This scenario concentrates on the impact of the three failure types introduced on representative indicators of voltage control effectiveness. All loads are considered inflexible, i.e., no controlled variation of loads is considered. The active power generated by DER and the constant active power demand for each load are shown in Figure 3.19. The constant values of demand are detailed in Table 3.2. Such adopted values are congruent with the power generation profile in Figure 3.19, preserving the proportion among the different types of loads as in the real setting where the considered portion of MV grid operates [66].

Consumer	**P (MW)**	**Q (MW)**
AGRICULTURE	-0.428539	-0.207551
COMMERCIAL	-1.382439	-0.454386
INDUSTRY	-3.490970	-1.690750
RESIDENTIAL	-1.153281	-0.289040
SUPERMARKET	-1.459757	-0.479799

Table 3.2: Load parameters.

For the timing failure, two values are considered: 10 minutes (delay=10 min) and 20 minutes (delay=20 min), and 0 minutes which represents the nominal fault-free condition (default). For the failure of the control device used to actuate the set-points of WP, two faults occur at 2:00 and 13:30, lasting for 60 minutes, leading to the injection of all available power generated by WP to the grid for the whole fault duration. Thus no power curtailment on WP is possible from 2:00 until 3:00, and from 13:30 until 14:30.

For the failure of the OLTC T, a fault occurring over all the analysis period (24 hours) is considered, reducing the range of the tap values to $(0, 10)$.

Although they do not refer to any specific real world setting, the considered failure parameters are adequate ones for the performed analyzes; of course, they are model parameters that can be easily replaced, to explore other scenarios. An observation window of 24 hours has been adopted. Each numerical result is obtained as the average value over the execution of 32 simulation runs (batches) or as a result of a single simulation run.

Figure 3.20 shows the probabilities that the value of $V_i(t)$ is out of bound of the nominal voltage $OV_i(t)$ and $-UV_i(t)$, for each bus of the grid (except slack bus $B1$, for which the voltage is always 1) in absence of control and without failures. Negative values $-UV_i(t)$ are considered for the under voltage probabilities in order to increase the readability of the figure.

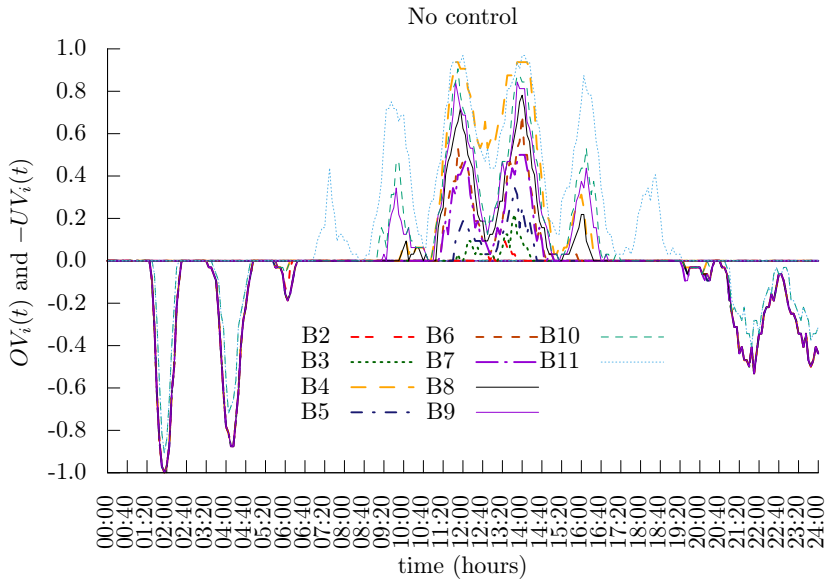


Figure 3.20: Probabilities that V_i is out of bound, both undervoltage $UV_i(t)$ and overvoltage $OV_i(t)$ in absence of control and failures.

Figure 3.21 shows the voltage of buses $B4$, $B7$ and $B11$ when full control is applied, obtained for a single simulation run. Buses $B7$ and $B11$ have the extreme voltage values and are both in bounds, thus the considered control strategies are able to overcame the critical scenario depicted in Figure 3.20.

When full control is applied, the cost of the control is shown in Figures 3.22 and 3.23 in terms of, respectively, i) the curtailment of renewable active power generated by PVP and WP at each instant of time, being P_i^{inj} is the power injected by the DER i, and ii) the tap values of the OLTC at each instant of time.

Figures 3.24 and 3.25 show the impact of different failures on the ability of the control to satisfy the voltage requirements over the bus $B11$. With only the timing delay, out of bounds probability is small for undervoltage, zero for overvoltage, and with only the WP control device failure the probability of both under and over voltage is zero.

In Figure 3.26 the focus is on the voltage requirement $MV1$ for every bus of the grid. Notice the difference between bus $B11$ and buses from $B2$ to $B10$: in presence of timing failure, at increasing the delay the probability of $MV1$ rejection increases for all buses, but in presence of multiple failures (timing delay and control device on WP) the probability of $MV1$ rejection does not increases for buses from $B2$ to $B10$.

Figures 3.27 and 3.28 show the importance of the load control when the

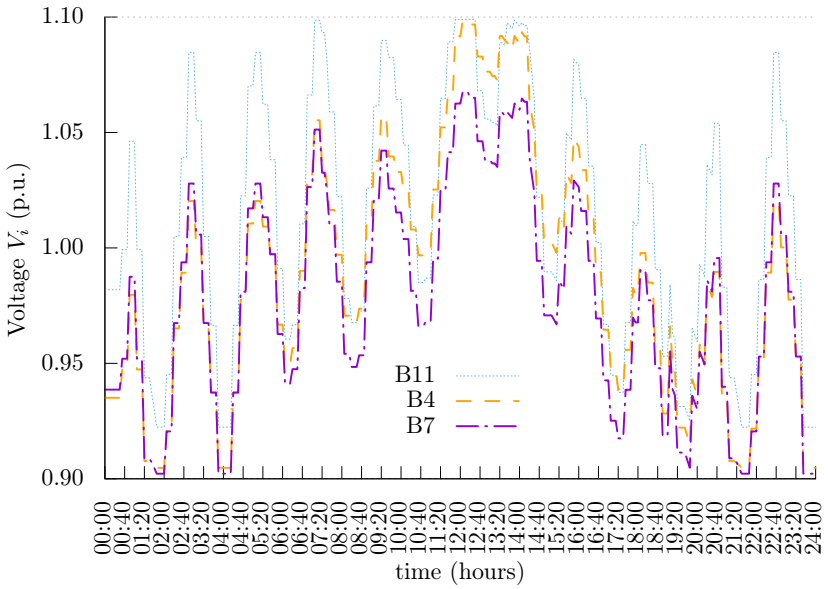


Figure 3.21: Voltage of buses $B4$, $B7$ and $B11$ when full control is applied: DER power curtailment and operative OLTC, in absence of failures.

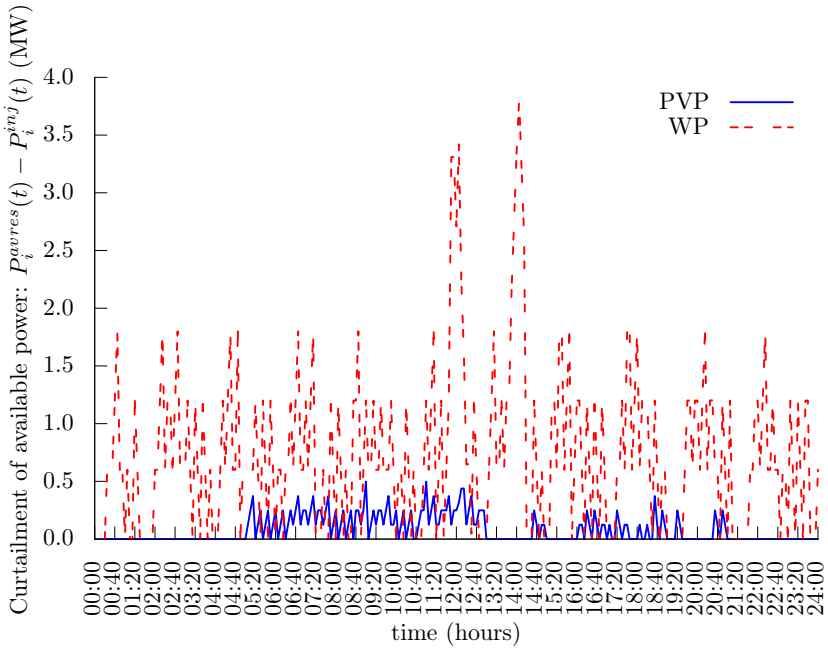


Figure 3.22: The curtailment of renewable active power generated by PVP and WP when full control is applied.

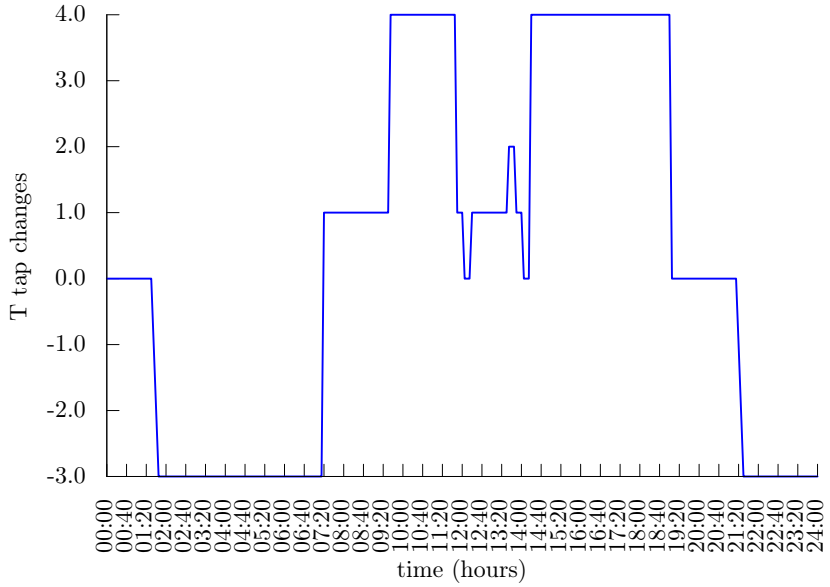


Figure 3.23: The OLTC tap values for each time t when full control is applied.

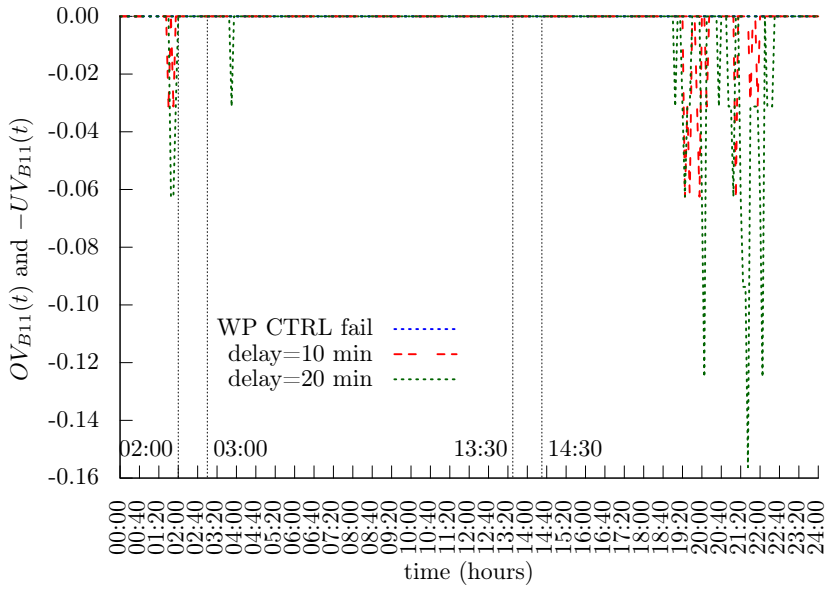


Figure 3.24: Probabilities that V_{B11} is out of bound, both undervoltage $UV_{B11}(t)$ and overvoltage $OV_{B11}(t)$, for two different values of timing failure (10 min and 20 min) and for the failure of the WP control device.

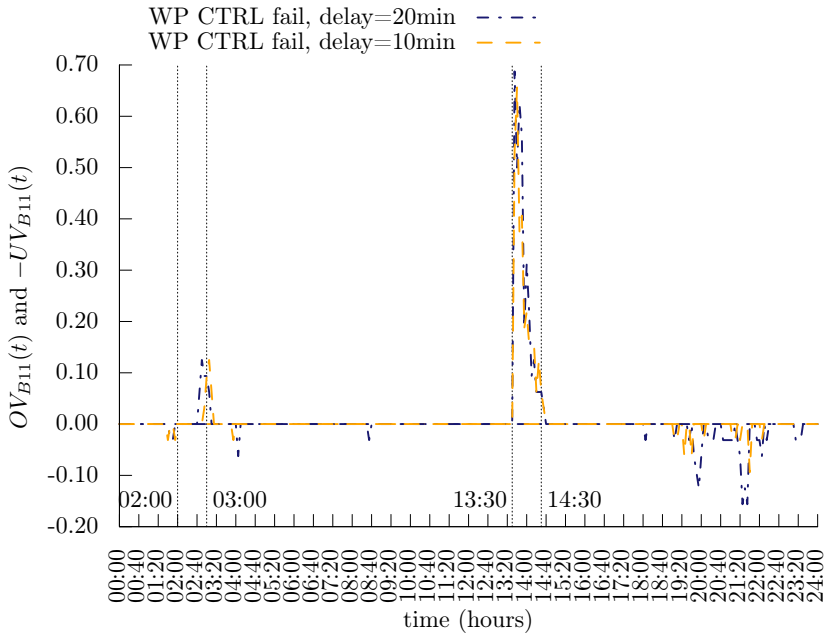


Figure 3.25: Probabilities that V_{B11} is out of bound, both undervoltage $UV_{B11}(t)$ and overvoltage $OV_{B11}(t)$, for two different values of timing failure (10 min and 20 min), in conjunction with the failure of the WP control device.

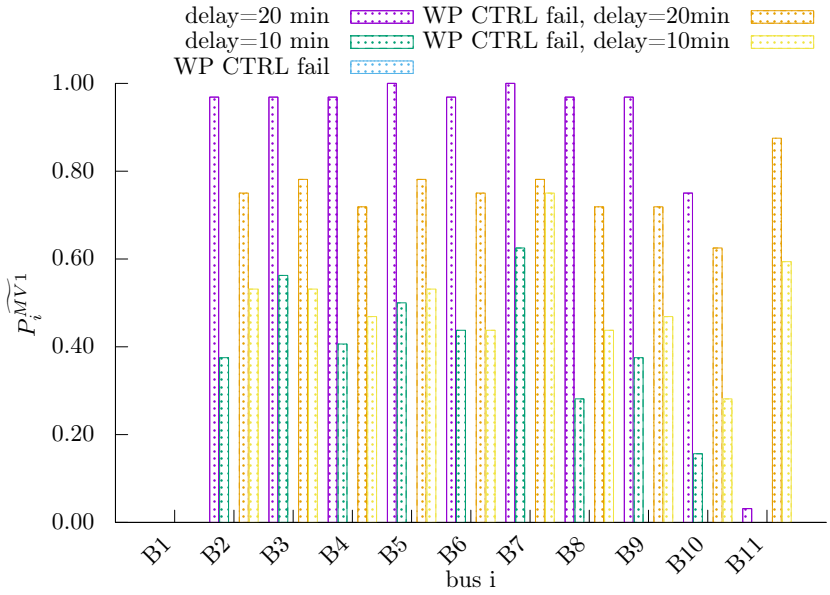


Figure 3.26: Probability that the grid voltage requirement $MV1$ is rejected, for all buses in the grid, for two different values of timing failure (10 min and 20 min), when failures of the control device of WP occur and when they do not occur.

failure of the OLTC T occurs. In fact, as shown in Figure 3.27, in absence of load control, the probability of under voltage on bus $B7$ is about 1 (-1 in the Figure). Instead, when the load control over bus $B3$ works the probability of under/over voltage at bus $B7$ is zero, but the voltage stability comes with unsatisfied power demand $UD(t)$ of bus $B3$ as depicted in Figure 3.28. In this scenario, bus $B3$ can reach a blackout.

3.6.3 Impact of failure/attack in presence of FL

In this scenario, variable power demand, considering flexible and inflexible loads is considered. Specifically, the load INDUSTRY on bus $B3$ is considered flexible, i.e., it has an operative load control (INDUSTRY load CTRL). The other loads are inflexible. This scenario concentrates on how voltage control functionalities increase the demand satisfaction if compared to the absence of control. Flexible load, based on controlled load variation, is a typical feature exploited by control operators in conjunction with DERs power curtailment. The active power generated by DER and the variable active power demand for each load are shown in Figure 3.29. As for other model parameters considered in the performed analyzes, a variety of alternative variable demand profiles

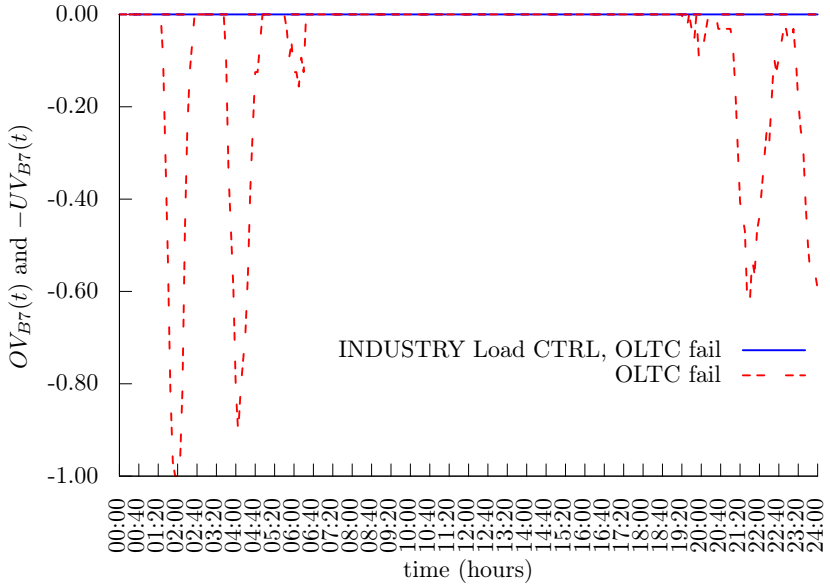


Figure 3.27: Probabilities that V_{B7} is out of bounds, both undervoltage $UV_{B7}(t)$ and overvoltage $OV_{B7}(t)$, when the OLTC fails and there is no load control over bus $B2$ and buses from $B4$ to $B11$. Only bus $B3$ has an operating load control (INDUSTRY load CTRL).

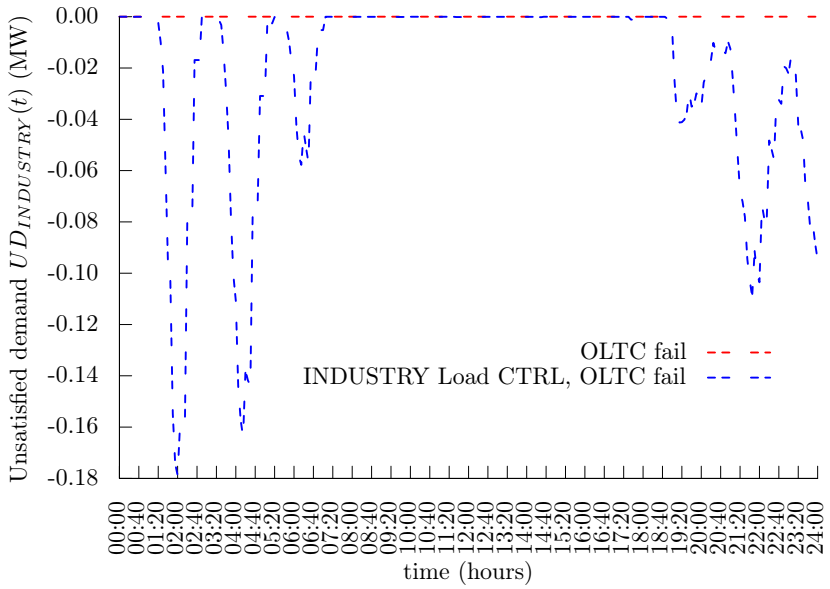


Figure 3.28: Expected unsatisfied power demand $UD_{B7}(t)$ for bus $B7$ at each instant of time t, when the OLTC fails and the only operative load control is on $B3$ (INDUSTRY load CTRL).

can be easily accommodated.

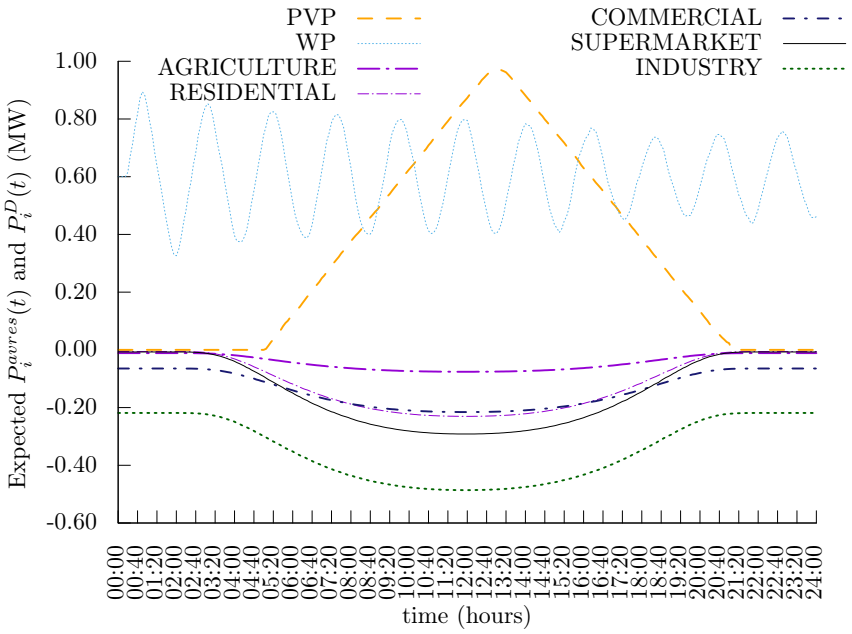


Figure 3.29: Expected available power P^{av} and variable power demand P^{dem}.

An observation window of 24 hours has been adopted. Each numerical result is obtained as the average value over the execution of 32 simulation runs (batches) or as a result of a single simulation run.

Figure 3.30 shows the measures $OV_i(t)$ and $-UV_i(t)$, for each bus of the grid (except slack bus $B1$ for which the voltage is always 1), with variable power demand, in absence of control and failures.

Figure 3.31 shows voltage values of buses $B4$, $B7$ and $B11$ when full control is applied, for a single simulation run. Voltage control guarantees voltage requirements, also in absence of load flexibility.

Figures 3.32 and 3.33 show the importance of the load control when the failure of the OLTC T occurs. In fact, as shown in Figure 3.32, in absence of load control (i.e., when all loads are inflexible), the failure of the OLTC causes undervoltage on bus $B7$ with a probability of about 0.9 (-0.9 in the Figure, curve labeled "OLTC fail"). Instead, when the load control over bus $B3$ works, i.e., the load on bus $B3$ is flexible (curve "INDUSTRY Load CTRL, OLTC fail'), then the probability of under/over voltage at bus $B7$ is zero. However, the ability to satisfy the voltage requirements comes with unsatisfied power demand $UD(t)$ of bus $B3$ as depicted in Figure 3.33. In this scenario, this loss of power is not considered a blackout, being the load on bus $B3$ a flexible load. Thus load flexibility can improve substantially

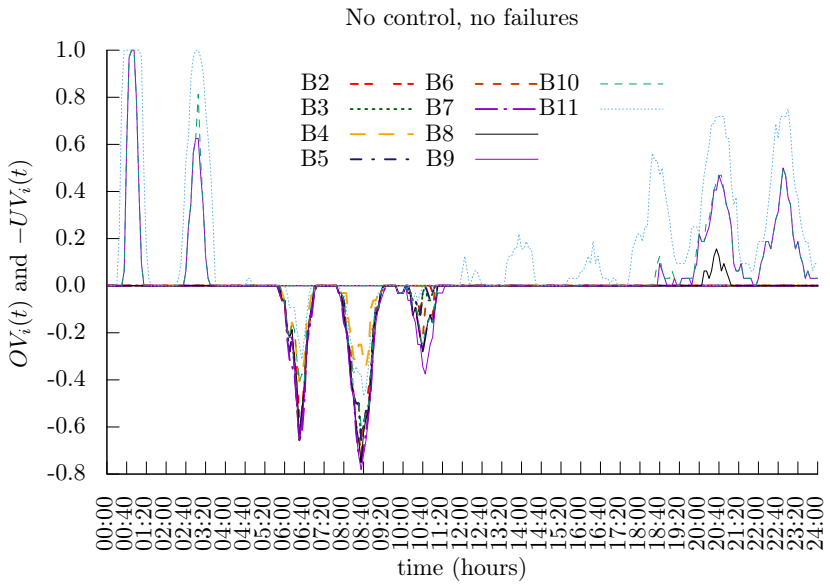


Figure 3.30: Probabilities $OV_i(t)$ and $-UV_i(t)$, for each bus of the grid, with variable power demand, in absence of control and failures.

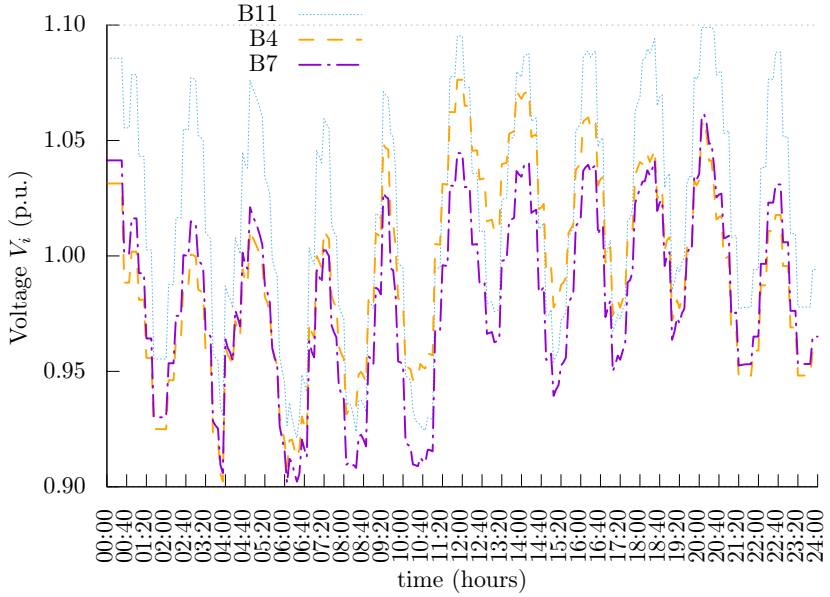


Figure 3.31: Voltage of buses $B4$, $B7$ and $B11$ when full control is applied: DER power curtailment and operative OLTC, in absence of failures.

the voltage quality of the considered electrical system. Notice that the non-

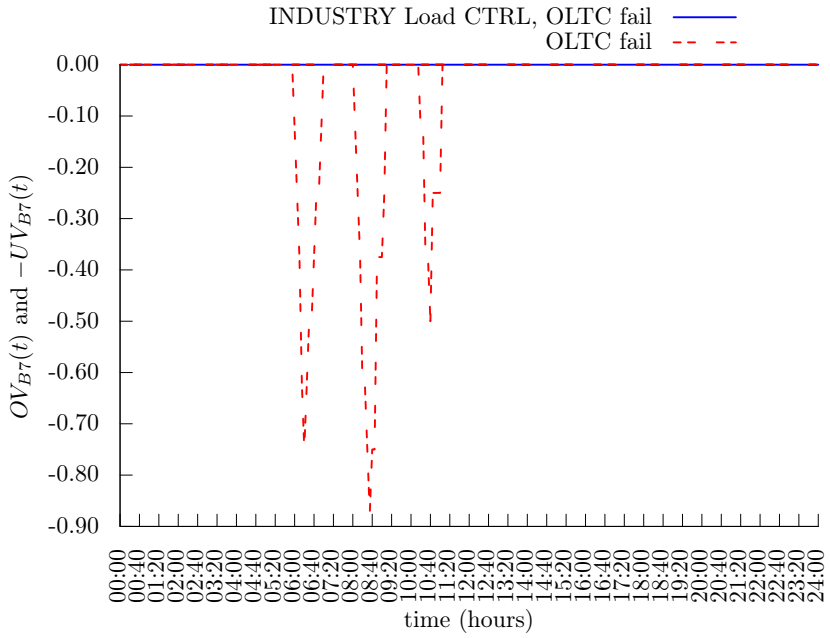


Figure 3.32: Probabilities that V_{B7} is out of bounds, both undervoltage $UV_{B7}(t)$ and overvoltage $OV_{B7}(t)$ when the OLTC fails and the load on bus $B3$ is either flexible (INDUSTRY Load CTRL) or inflexible.

linearity of the PFEs determines relevant contributions of control actions not only to the behavior of local electrical components but also on components that are far distant.

3.7 Achievements and open problems

The model that has been just described can be exploited to perform a vast range of analyzes, producing relevant information about the behavior of a SG under many different scenarios. The main issue, though, is that it cannot be used to model large EI. Primarily, the reason is that the model has to be able to represent the entire EI and consider all the possible interactions among ICT and EI components. Larger distribution grids also imply increased interactions, in addition to the higher number of components to account for.

Notice that the non-linearity of the PFEs, clearly visible for instance within the scenario discussed in Section 3.6.3, drastically reduces the ability of modeling subsets of a large EI separately and then combining the results of

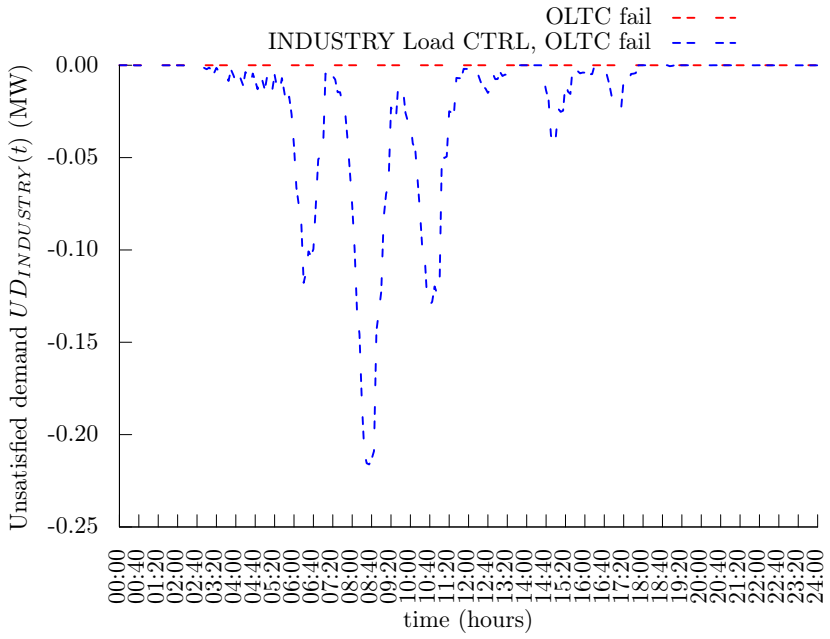


Figure 3.33: Expected unsatisfied power demand $UD_{B7}(t)$ for bus $B7$ at each instant of time t when the OLTC fails and the load on bus $B3$ is either flexible (INDUSTRY Load CTRL) or inflexible.

the individual analyzes. To deal with large distribution grids, the model has to be able to represent the entire EI and consider all the possible interactions among ICT and EI components. As learned from exercising the basic model

Table 3.3: CPU times for initialization of the basic model as the number of buses n increases. The case studies employed are detailed in Section 6.5.

n	t_{init} (s)
33	118.84
66	1854.39
99	8792.48
132	25771.30
165	60796.60
198	NaN

on the analyzed scenarios, there are two main obstacles that contrast the model scalability: model architecture and EI state management. Thus, the focus of the rest of the thesis is on studying alternative efficient ways for designing the model architecture and for the EI management.

The weak point in the architecture that is responsible for the huge initialization time visible in Table 3.3, obtained on a computer equipped with a Intel Q9550 CPU (4 cores, freq. 2.83GHz), 8Gb of RAM running an up to date GNU/linux OS, is the way template submodels are designed and how they work. On one hand, the principles discussed in Section 3.5 allow the definition of a generic model that can be easily instantiated to represent a specific case study, but on the other hand the way this is done involves too many shared SAN places so that it becomes infeasible when the EI dimension grows. In particular, for electrical grids with more than 165 buses it was not possible to wait until the initialization phase ended, *NaN* in Table 3.3, and then no measure was evaluated. Being the topology representation at the very core of the model, detailed investigations about generic modeling approaches have been carried out (see Chapter 4) and a complete re-design of the model, using a different approach, have been performed. Not only the investigations have clarified why there are inefficiencies when following the ideas presented in Section 3.5, from now on called *State-Sharing Replication* (*SSRep*) approach, but also have guided the definition of two new approaches, in the following named *Channel-Sharing Replication* (*CSRep*) and *Dependency-Aware Replication* (*DARep*), and a proposal for changing the way Möbius performs State Variables sharing, discussed in Section 4.7.2.

A key aspect of the analysis presented in Chapter 4 is the fact that many cyber-physical systems, e.g., the SG, present topologies characterized by a large number of components that are weakly interconnected among each others. It turned out that the *SSRep* approach is completely blind to the degree of interconnections among components so that assuming each component is always connected to all the other components. This is the primary source of inefficiency during the initialization phase and has also consequences on the simulation time.

Another source of inefficiency is related to how the EI state estimation is performed. A new state estimation is required, as shown in Algorithm 1, whenever there are changes in the grid, and then even a small performance issue during the solution of the PFEs can have great impact on the overall simulation time. Moreover, also the way the optimization problem described by Equation (3.4) is solved has a huge impact on simulation time. Equation (3.4) describes an unconstrained, mixed-integer, optimization problem that is solved with an heuristic method (details are in Chapter 5) where a population of candidate solutions is generated and transformed through an iterative process. Whenever a member of the population is created or transformed it requires a new EI state estimation: this means that, whenever line 5 of Algorithm 1 is touched, the PFEs have to be solved a number of times proportional to the population size and the number of iterations. For an unconstrained optimization problem, the population size employed in an heuristic method can grow quite quickly: for instance, it grows to 1200 elements for the scenarios presented in Sections 3.6.1 to 3.6.3, and then a huge number of calls to the PFEs solver is triggered, as detailed in Table 3.4.

Table 3.4: Number of CTRL calls, PFEs subroutine calls, and among them how many do not converge, as n increases, for a simulation time of $24h$.

n	# CTRL calls	# PFEs calls	# not converged
33	109	$1.42 \cdot 10^5$	0
66	521	$6.74 \cdot 10^5$	7,594
99	136	$1.79 \cdot 10^5$	$1.3 \cdot 10^5$

In addition, the PFEs are solved using the Newton-Raphson method [68], an iterative method that does not always reach convergence. No convergence implies no estimate of the EI state. Further details are provided in Chapter 5, here it is just important to notice that if this happens inside the local state of a member of the population, when running the heuristic method for the optimization solution, then the member has to be banned from the

population. If there is no convergence during the EI state estimation in other parts of the model, namely at lines 3 and 7 of Algorithm 1, then the EI is considered in blackout or the entire simulation batch is stopped and excluded from the statistical analysis. Notice that if the Newton-Raphson method does not converges it is not possible to estimate the EI state, but this does not means that the real EI is in blackout. The fact that it is not possible to estimate the state of a physical object is dictated by limitations of the selected mathematical formalism, not by behaviors of the object itself. More formally:

- if the method converges and the voltage values are under a predefined threshold then the EI is actually in blackout,
- if the method did not converges the modeler has to chose among the folowing three alternatives:
  - assume the EI is in blackout. This is the most conservative and the one chosen in this thesis,
  - set voltages and powers to a default value, ignoring the outcome of the method,
  - starting from the knowledge of PFEs solutions obtained in previous time instants, elaborate the values to be assigned to each bus.

In this basic model the PFEs were solved through the BBD preconditioned Inexact-Newton-Krylov GMRES method, a variant of the Newton-Raphson method, using the C++ library Sundials/Kinsol [69], but this approach was too unstable to be considered for a general purpose SG model. In particular, Krylov methods need fine tunings of the preconditioner because convergence highly depends on it, thus reducing the ability of the model to tackle a vast range of case studies. Moreover, the Inexact-Newton-Krylov GMRES method is highly sensitive to the equations ordering, as shown in a preliminary work presented in [8]. This additional degree of freedom makes convergence analysis even more complex. Of course, increasing the convergence rate of PFEs is important both to speed up the simulation time and to avoid resulting in too inaccurate estimation of the real blackout condition due to numerical computation problems.

For these reasons, careful investigations, presented in Chapter 5, have been carried out in order to select a convenient variant of the Newton-Method and to develop an appropriate tuning.

3.8 Summary

In this chapter a detailed description of how SGs are represented in the context of this thesis has been provided. Starting from the identification of the main logical components of both the EI and ICT infrastructures, the overall logical architecture has been presented in Section 3.1. The adopted perspective is dictated by a modeling choice: in this thesis the focus is on failures, not on faults. In particular, as explicitly stated in Section 3.2, the selected level of abstraction reflects the aim of measuring the impact of failures that happen within one infrastructure on the other one. Of fundamental importance are then the details discussed in Sections 3.3.1 and 3.3.2, where the EI state representation and estimation are described and the capabilities of MCS are illustrated.

Representing the MCS logic as a combination of an infinite event loop and an optimization problem solution promotes a great flexibility in coupling the two infrastructures' state management, although expensive from the computational viewpoint. In particular, the three high level states defined by Algorithm 1 (*ok*, *critical state* and *critical state not removed*), when the MCS is assumed to work in absence of failures, represent an interface that abstracts the specific modeling choices related to the considered SG at hand or the selected scenario. Specifically, Equations (3.2) and (3.4) can be extended or modified to address different modeling needs or special scenarios.

In Section 3.4 and Section 3.5, technical details about modeling the SG within the Möbius modeling environment have been discussed. The hierarchical structure of both EI and MCS is modeled through the *Join* and *Rep* operators provided by Möbius, while the EI topology is modeled through the *SSRep* approach.

A main contribution of this chapter has been presented in Section 3.6, where three scenarios (comprising the MV only, or both MV and LV levels) are detailed and analyzed. This study concretely demonstrated the usefulness of the developed analyzes framework to gain insights on the impact of failure patterns on indicators representative of the correct operation of the SG, considering the presence of interdependencies. The observed results were not so straightforward, so underlying the usefulness of such analyzes in deeply understanding the relationships between failure phenomena and SG complex operation. However, while fulfilling the goal motivating the modeling effort, the performed analysis drastically pointed out the limitation of the current evaluation framework, which is its ability to scale in order to afford analysis of realistic grid sizes. Then, in Section 3.7, a careful reasoning has been developed to understand the major reasons for the performance weaknesses, thus identifying directions to explore for enhancing the basic SG

model. Such directions are mainly to devise new efficient replication and composition approaches as well as more efficient PFEs solvers. These studies are carried on in the next two chapters.

Chapter 4

Compositional approaches

In this chapter, the focus is on techniques for replication of stochastic models for dependability and performability analysis. General approaches, which are not dependent on any specific tools but implementable on top of a variety of existing stochastic evaluation tools, are pursued.

Moreover, the investigated techniques are first developed without explicit reference to the SG context, but tailored to deal with generic systems sharing characteristics similar to SG in terms of size, heterogeneity of components and interdependencies among components subsets. The motivations are twofold. First, to discuss in isolation the characteristics of the model composition mechanisms, which is the goal of this chapter, the SG scenario presented inChapter 3 would have been too heavy, with also too many phenomena involved in and potentially interfering with the planned basic analysis. Actually, this could have been overcome by adopting a simplified SG scenario, but there is a second motivation, which is that the proposed composition strategies are general ones, not restricted to the SG framework. Thus, this different use case provides a concrete evidence to this claim. However, the embedment of these techniques in the SG context dealt with in Chapter 6.

In order to exploit fast resolution techniques, existing replication mechanisms are mainly directed to anonymous replication. While this fits well many system scenarios, there are a number of other contexts where population of similar components actually require an individual identity to properly analyze the impact of correct/faulty behavior of individual components on the overall system dependability, e.g. because of the position occupied by the component in the system topology. Critical infrastructures, such as in the transportation and electrical sectors, are examples of such systems where an identity-aware replication is necessary to maintain adequate accuracy when modeling the component structure and behavior. Considering that such systems are typically large ones, with dependency relationships

among constituting components, the problem of devising efficient solutions to non-anonymous modeling replication is a challenging issue that is triggering interest in the model-based community. In order to address the issue in general, realistic contexts, such as when targeted system components have non-Markovian behavior (e.g., because of deterministic time delays) and thus analytical approaches are not applicable, this chapter concentrates on modeling strategies well suited to obtain (possibly) efficient simulation-based evaluations. Specifically, this chapter offers the following contributions:

- formalization of the non-anonymous modeling replication concept and of a context to express different approaches in a coherent format;
- refinement and extension of selected solutions taken from the literature. Solutions tailored to specific models, developed within the boundaries of the Möbius Modeling Framework, are here generalized to address a vast variety of system models, also setting the basis for implementing these solutions in other modeling environments;
- refinements are mainly oriented towards speeding up performance;
- comparison of the three major approaches in terms of indicators representative of their efficiency level when employed in dependability system evaluation, at increasing the size and the degree of dependency of the system under analysis. To this purpose, a sophisticated case study is adopted, to illustrate the strengths and limitations of the compared techniques in a variety of realistic system scenarios.

The rest of the chapter is organized as follows. State of the art solutions in non-anonymous modeling replication are overviewed in Section 4.1. Section 4.2 introduces the category of systems under analysis and their characteristics. Section 4.4.1 provides the formal context for expressing replication mechanisms, in terms of formalism and necessary modeling features. This formal context is then further specialized in Sections 4.4.3 to 4.4.5, where the three selected mechanisms *SSRep*, *CSRep* and *DARep* for non-anonymous modeling replication are formalized, respectively.

Going further in the process to make concrete their implementation, Section 4.5 is devoted to present how *SSRep*, *CSRep* and *DARep* have been implemented in the Möbius framework.

A representative case study is described in Section 4.6, followed by a discussion on how to model it using the presented replication approaches. To gain insights on concrete benefits in terms of efficiency indicators, extensive comparisons among *SSRep*, *CSRep*, and *DARep* are carried on in Section 4.6.2, using the adopted case study. This case study is not directly linked

to the SG context, the rationale behind this choice relying on the consideration that a more abstract and, in a certain sense, more general scenario can better address the analysis of the compositional approach selection impact on performance. A SG model could have been too complex to isolate behaviors directly caused by compositional approaches. In addition, the possibility to tackle an abstract case study promotes the adoption of the described approaches in several context. The obtained results show the superiority of the *DARep* approach in all the investigated scenarios.

4.1 Related work

In the literature, modeling of system components replication has been mainly addressed in the form of anonymous replication, well suited to represent a population of identical components, each one interacting with all the others or completely independent, following the approach originally presented in [33]. This configuration favors the application of the strong lumping theorem [70], enhancing the state-space generation based solvers [71], as well as simulation-based solvers [72]. The key point is the possibility to exploit the trivial "all or nothing" symmetry that characterizes the complete directed graph of interactions among components. For identical components connected in arbitrary ways, symmetry exploitation methods [70] have been adopted. In case of the "all or nothing" interconnection pattern, other approaches focus on syncronization [73, 74], in particular when stochastic Petri nets dialects are employed. Alternatively, behavioral aspects [75], such as bisimulation, have been pursued by the stochastic process algebra community.

In case the system logical structure allows a hierarchical decomposition of the model, i.e., components within the same layer are independent and there exists a partial ordering among components in different layers, more general replication strategies have been proposed. In particular, taking advantage of model specification languages, such as those adopted in the SHARPE [76] tool, possibly in conjunction with the use of some kind of scripting language, a template model can be defined and replicated, either anonymously or non-anonymously. The modeler has the ability to specify syntactic and semantic rules, as well as decide how to realize the replication process because it has no impact on the overall system model solution, being performed layer by layer. Unfortunately, if the aim of the analysis is to study interdependencies among system components, the mentioned strategy cannot be considered.

The need to cope with modeling of complex systems, where similar components are arranged according to a topology and have specific peculiarities (such as parameters setting and behavior) that depend on the topological po-

sition, triggered solutions based on non-anonymous replication. As already discussed, given the size, complexity and internal dynamics characterizing the addressed systems, this chapter focuses on modeling solutions tailored to simulation based solvers only.

In [77, 78] a solution, referred in the following as *SSRep* approach, is proposed, where an indexing mechanism is exploited to build non-anonymous replicas of a given template model.

The *SSRep* approach was employed to model the dynamic of the first SG model presented in Chapter 3 and, as pointed out in Section 3.7, can cause inefficiencies. It is a general solution for simulation-based solvers, but its efficiency is limited because it relies on the pessimistic approach of a complete dependency graph among all the replicas. Instead, the vast majority of real-world systems are typically composed by many loosely interconnected components according to regular topologies (tree, mesh, cycle, etc).

This observation triggered studies on more efficient solutions, which exploit the knowledge of the real dependency topology shown by the system under analysis. Promoting higher efficiency is greatly relevant, since it makes possible to enlarge the size of the systems that can be analyzed and to better represent scenarios at the required level of accuracy (especially crucial for dependability critical applications).

The approach introduced in [79], referred in the following as *CSRep* approach, specifically addresses non-anonymous replication of loosely interconnected components. This strategy uses a single channel shared among all the replicas to exchange part of their state during simulation, so reducing the number of state variables that are shared among all the replicas. It has been specifically conceived as an efficient modeling mechanism for system scenarios with low dependency degrees in presence of a large number of replicas, conditions verified for many real systems.

A population of similar components is also considered in [80], where a graph of connections is employed to construct a well shaped matrix in which differences among components correspond to non-zero off-diagonal entries. Similarities help the numerical solver in gaining performance, being the computation based on a linear system solution that is sensitive to the number of non-zero off-diagonal elements. However, in [80], there is no template model and no replication, and the definition and composition of submodels is manually performed.

Another mechanism to non-anonymous replication, named *DARep*, has been more recently proposed in [13]. As for *CSRep*, also *DARep* principle is to exploit again the real dependency graph characterizing the interactions among system entities, but avoiding unneeded additional structures, e.g., a channel, whose management impacts on simulation performance.

In this chapter, the three techniques *SSRep*, *CSRep* and *DARep* are selected as most representative approaches to model non-anonymous replication for simulation-based evaluation, and are compared in terms of efficiency under a variety of parameters setting in Section 4.6.2.

Among popular software tools that explicitly tackle submodels composition, and therefore would be suitable contexts for exploiting the replication techniques addressed in this chapter, there are: GreatSPN [81] and Möbius [5, 57].

4.2 Addressed systems' logical architecture

The system category tackled in this chapter is composed by a large number of components, in general grouped as belonging to different categories in accordance to the nature and function they perform, and organized according to well known structures (topologies, e.g., tree, cycle, mesh, etc..). The connection among two or more components induces a dependency among the state, and so in general the behavior, of the involved components. Therefore, in the following, the terms connection and dependency are used as synonymous, when referred to the relation among components.

Cyber-physical systems in critical sectors, such as transportation, electricity, water and oil, fit well in the addressed system category. For example, an electrical grid encompasses, among its components, a number of collection points called buses. All buses have the same aim, that is collecting and distributing electricity, so they belong to the same component category and their models are similar. The main differences among them are the position occupied in the grid topology and the number and kind of attached electrical equipments for energy production or consumption. Moreover, depending on the topology of the grid under analysis, dependencies are in place among subsets of buses connected through power lines.

Therefore, when abstracting the components (e.g., the buses) of the reference system for analysis purpose, a *generic* component can be assumed as representative of all the *specific* components included in the system belonging to that category. Then, each specific component, characterized by individual peculiarities, can be considered a non-anonymous replica of such generic component, specifically distinguished through an index. Of course, the definition of the logical architecture of the system under analysis, in terms of types and number of components, as well as the abstraction level to be adopted for their modeling, strongly depend on which is the goal of the analysis. As a simple exemplification, if the interest is in determining the overall system failure rate, the system abstraction level is in general different from the case

where the objective is to quantify the impact of failure propagation among specific system components.

In general terms, the logical architecture of the given reference system can be seen as composed by:

- a large number of connected components, in general belonging to different typologies (e.g., buses, or power lines, or substations in electrical power systems). A generic component C^g is associated to each typology, which represents a subset of specific components (i.e., non-anonymous replicas) denoted with C_i, where i is the index of the replica.
- one or more topologies that define the connections among specific components.

Without loosing in generality, but for the sake of simplicity, in the following only one generic component C^g describing n specific connected components $C_0,\ldots, C_{n-1}$ is considered.

The definition of a generic component C^g requires that all the parameters of the specific components C_i (such as those related to the initial state, the state changing, the random choices, the random exponential or non exponential times) can be defined as a function of the index i of the specific component.

4.2.1 System state definition

To define the system states, the generic component C^g includes l *template* SVs:

$$SV^0, SV^1, \ldots, SV^{l-1}, \tag{4.1}$$

where each SV^h represents n different SVs, one for each specific component, in the overall system logical structure:

$$SV_0^h, SV_1^h, \ldots, SV_{n-1}^h. \tag{4.2}$$

Thus, the entire system state is determined by assigning natural or real numbers, i.e., $SV_i^h \in \mathbb{N}$ or $SV_i^h \in \mathbb{R}$, to all the $l \cdot n$ SVs, here listed as follows:

$$\begin{array}{cccc} SV_0^0, & SV_1^0, & \ldots & SV_{n-1}^0, \\ SV_0^1, & SV_1^1, & \ldots & SV_{n-1}^1, \\ & & \vdots & \\ SV_0^{l-1}, & SV_1^{l-1}, & \ldots & SV_{n-1}^{l-1}. \end{array} \tag{4.3}$$

Common to the adopted notation throughout the chapter (both in this section dealing with system formalization and in those dealing with modeling

formalisms) is that superscript are used to indicate the index of a template (e.g., of a SV) whereas a subscript indicates the index of the specific component/model (e.g., SV_i^j represents the state variable SV^j of C_i).

4.2.2 Component interactions definition

The interactions between components C_i and C_j are defined by the $n \times n$ adjacency matrices $\mathcal{T}^h = [\mathcal{T}^h_{i,j}]$ and $\mathcal{T} = [\mathcal{T}_{i,j}]$, representing the *topology of interactions* based respectively on individual SV^h and on all the SVs.

The key idea in modeling interactions between C_i and C_j, $j \neq i$, is through information sharing via one or more State Variables.

Formally, if C_i has read/write access to SV_j^h for $j \neq i$ then $\mathcal{T}^h_{i,j} = 1$, else $\mathcal{T}^h_{i,j} = 0$. In particular, C_i can depend or impact on SV_j^h and, at the same time, C_j can have read/write access to SV_i^h, $i \neq j$. Note that in general, $\mathcal{T}^h_{i,j} = 1$ does not imply $\mathcal{T}^h_{j,i} = 1$. Thus, the SVs SV^h included in C^g can be classified into two categories:

Local state variables If SV_i^h can be accessed by C_i only. $\mathcal{T}^h$ is the identity matrix.

Dependency-aware state variables If SV_j^h can be accessed by C_j and, depending on the topology $\mathcal{T}^h$, by other component(s) C_i, with $i \neq j$. $\mathcal{T}^h$ has no restriction except having all ones on the diagonal.

Therefore, $\mathcal{T}^h$ represents the directed graph of read/write accesses to SV^h, as depicted in Figure 4.1 for a simple example. The matrix $\mathcal{T}$ is then described by:

$$\mathcal{T}_{i,j} = \max_{h=0,\ldots,l-1} \mathcal{T}^h_{i,j}, \tag{4.4}$$

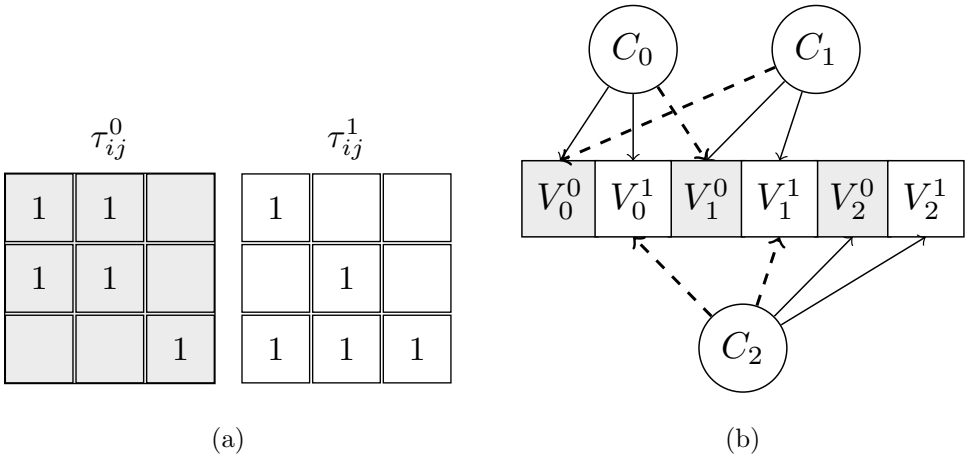
where $\mathcal{T}_{i,j} = 1$ if there exists at least one SV among $SV_j^0, \ldots, SV_j^{l-1}$ that is accessible by C_i, as depicted in the example of Figure 4.2.

The *dependency degree* of component i with respect to SV^h, called $\delta_i^h \in \{0, 1, \ldots, n-1\}$, indicates how many replicas of SV^h among $SV_0^h, \ldots, SV_{n-1}^h$ can be read/written by C_i, formally:

$$\delta_i^h = \sum_j \mathcal{T}^h_{i,j}. \tag{4.5}$$

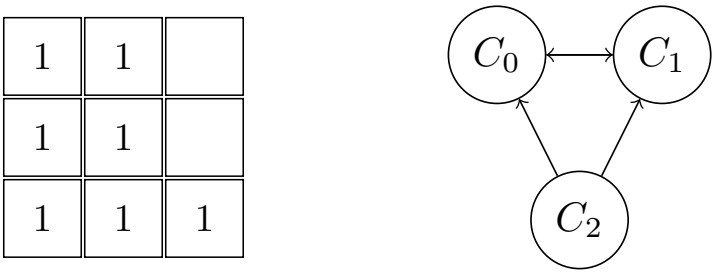
Similarly, fixed SV^h in C^g, the array of those components C_j such that C_i can access SV_j^h is called Δ_i^h and $\Delta_i^h(k)$ is the k-th element of the array, $k = 0, \ldots, \delta_i^h - 1$. Formally:

$$\Delta_i^h = \left(j \,|\, \mathcal{T}^h_{i,j} = 1\right) \in \mathbb{N}^{\delta_i^h}. \tag{4.6}$$



(a) (b)

Figure 4.1: Example of two read/write access topologies $\mathcal{T}^0$ and $\mathcal{T}^1$ (a) for a generic component C^g with two SVs, gray SV^0 and white SV^1, and $n = 3$ (b). Dashed lines represent read/write access of C_i to dependency-aware SVs.



(a) (b)

Figure 4.2: Continuation of the example depicted in Figure 4.1: topology of interactions $\mathcal{T}$ (a) and diagram with interations among specific components (b).

In the example depicted in Figure 4.1 involving three replicas and two state variables, $\Delta_0^0 = (0, 1)$, $\Delta_1^0 = (1, 0)$, $\Delta_2^0 = (2)$, $\Delta_0^1 = (0)$, $\Delta_1^1 = (1)$ and $\Delta_2^1 = (2, 0, 1)$.

An independent component, i.e. a component i such that both the i-th row and column of $\mathcal{T}$ are zero, corresponds to a disconnected node of the graph, while a system where there is full connectivity among its components results in a complete graph.

As detailed in Section 4.4.1, the modeling approaches *CSRep* and *DARep*

strongly differ from *SSRep* in the principle adopted for dealing with interdependencies, namely whether exploiting some properties of $\mathcal{T}$ and of $\mathcal{T}^h$, or assuming a complete graph of interactions.

4.2.3 Component actions definition

The state changes are defined by means of *actions* [59,82]. In particular, the generic component C^g includes m *template* actions:

$$a^0, a^1, \ldots, a^{m-1}, \tag{4.7}$$

where each template action a^k represents n different actions, one for each replica of C^g. Thus, the entire system is governed by the following $m \cdot n$ actions:

$$\begin{array}{cccc} a_0^0, & a_1^0, & \ldots & a_{n-1}^0, \\ a_0^1, & a_1^1, & \ldots & a_{n-1}^1, \\ & & \vdots & \\ a_0^{m-1}, & a_1^{m-1}, & \ldots & a_{n-1}^{m-1}. \end{array} \tag{4.8}$$

The set of all the actions of C^g and C_i is respectively $\mathcal{A}$ and $\mathcal{A}_i$, such that $m = |\mathcal{A}| = |\mathcal{A}_i|$, for all i.

An action can be *enabled* according to rules defined on SVs values. When an enabled action *completes*, it can change the value of all the SVs that are accessed by C_i. SVs values are assumed to evolve in continuous time (time is always a real value) and each action can take a deterministic or stochastic time period between its enabling and completion. The relation between a^k and a_i^k is: whenever in the enabling conditions or in the description of how a^k acts is mentioned SV^h, then in a_i^k will be mentioned SV_i^h. An action in C_i can be enabled and it acts according to rules defined *only* on the SVs that C_i can read/write.

4.2.4 Overall logical component definition

Summing up, a generic component C^g is defined by a set of template state variables SVs and a set of template actions. Formally:

$$C^g = \left(SV^0, \ldots, SV^{l-1}\,;\, a^0, \ldots, a^{m-1}\right). \tag{4.9}$$

The overall system behavior over time is represented by $l \cdot n$ SVs and $m \cdot n$ actions. Starting from the generic component C^g, n specific components are

defined by those SVs that they can access and the set of actions they owns, formally

$$C_i = \Big(\bigcup_h \bigcup_{j \in \Delta_i^h} SV_j^h \, ; \, a_i^0, \ldots, a_i^{m-1} \Big). \tag{4.10}$$

Given the logical structure of the systems of interest in terms of SVs and actions, also the corresponding system model will be defined using formalisms and operators based on SVs and actions. The goal is to generate automatically from a single template model, representing a generic component, the overall system model representing all the specific components and the related interactions. Nevertheless, as will be discussed in Section 4.4.1, the automatic modeling of all the specific components and their interactions is not straightforward, because, to the best of our knowledge, the formalisms and modeling environments currently accessible are not designed to deal with non-anonymous replication through primitives made explicitly available. Thus, the modeling approaches discussed in this chapter are meant to exploit modeling strategies and formalisms offered by existing modeling frameworks to automatically tackle the presented logical architecture.

A final observation is about relaxing the assumption of just one generic component in the system under analysis. In case of multiple generic components, the general concepts and related formalization previously given are all valid, unless for the interdependency aspect. In fact, since interactions can occur not only among replicas of the same generic component, but also among replicas of different generic components, the matrices representing the topology of interactions need to be properly extended. Complete formalization when considering more than one C^g is deferred as future work.

4.3 Example: a load-sharing system

A concrete example of system composed by n components, interconnected through a specified topology is presented in the following. This example is effective in demonstrating features, benefits and limitations of the approaches that will be presented in Section 4.4, since it fully represents the logical architecture of targeted systems described in Section 4.2 and can be considered as a basis to be easily extended and adapted to represent a great variety of real contexts.

4.3.1 System description

Let's consider, in a cloud computing context, two different services, A and B, that can be requested from clients. The infrastructure is composed by n

virtual machines, called here nodes, and each can host one server for A and/or one server for B. In other words, each virtual machine can host: 1. one server for A, or 2. one server for B, or 3. two servers, one for A and one for B. On each node i, the demand (service requests) of service $S \in \{A, B\}$, at any time instant t, is $S^{\text{load}}i \geq 0$. At any time instant t, node i has maximum capacity $C_{S,i}$ to satisfy service S requests, where $S^{\text{load}}i, \leq C_{S,i}$. The connection matrix $\mathcal{T}^S$ represents the topology of connections among servers providing service S on different nodes.

Demand $S^{\text{load}}i$ can increase or decrease due to:

- random interaction with clients: the demand of service S follows a continuous time Markov process so that, after an exponentially distributed period of time, with probability $\mathbb{P}_{\text{incr}}$ demand increases and with probability $1 - \mathbb{P}_{\text{incr}}$ it decreases. The value of $\mathbb{P}_{\text{incr}}$ depends of the demand trend, i.e., if last time demand is changed it is increased, then $\mathbb{P}_{\text{incr}} >> 0.5$, otherwise $\mathbb{P}_{\text{incr}} << 0.5$,

- interaction with neighbors: whenever $S^{\text{load}}i \geq C_{S,i}$ or the server reboots or recovers the server of service S on node i tries to dispatch the exceeding demand respectively $S^{\text{load}}i - C_{S,i}$ or $S^{\text{load}}i$ (for reboot or recovery) to other working servers of the same service on nodes in $\Delta_i^{D_S}$. Thus, $S^{\text{load}}i$ can decrease and $S^{\text{load}}j$ can increase for $j \in \Delta_i^S$.

For the sake of simplicity, in this case study, the following policy to dispatch the exceeding demand has been adopted (although different and more complex policies can be accommodated). The exceeding demand of service S on node i is divided in δ_i^S equal parts $E_{S,i} = (S^{\text{load}}i - C_{S,i})/\delta_i^S$. For each working neighbour $j \in \Delta_i^S$, $E_{S,i}$ is sent to node j, if $E_{S,i} \leq C_{S,j} - S^{\text{load}}j$, decreasing $S^{\text{load}}i$ and incrementing $S^{\text{load}}j$ by $E_{S,i}$. Otherwise, if the server on node j cannot satisfy all the demand $E_{S,i}$, i.e., $E_{S,i} > C_{S,j} - S^{\text{load}}j$, then only a part of $E_{S,i}$ equal to $C_{S,j} - S^{\text{load}}j$ is sent to j, decreasing $S^{\text{load}}i$ by $C_{S,j} - S^{\text{load}}j$ and increasing $S^{\text{load}}j$ to the maximum capacity $C_{S,j}$. Of course, if the node j is down or it does not host a server for service S, then $E_{S,i}$ is not sent to j. Thus, it is not guaranteed that all $S^{\text{load}}i - C_{S,i} \geq 0$ can be redispatched.

To make the case study more sophisticated, so to resemble more realistic application scenarios, if the total time $T_{S,i}^{overc}(t)$ in which $S^{\text{load}}i \geq C_{S,i}$ overcomes a pre-set threshold $T_{S,i}^{threshold}$, for at least one service S, then the state of node i is switched from working to down and a reboot is performed. The reboot takes a deterministic amount of time and has success with probability c. In case it fails, or after a given number of reboots n^{rcv}, a recovery process is launched on node i. The recovery operation can take considerably longer

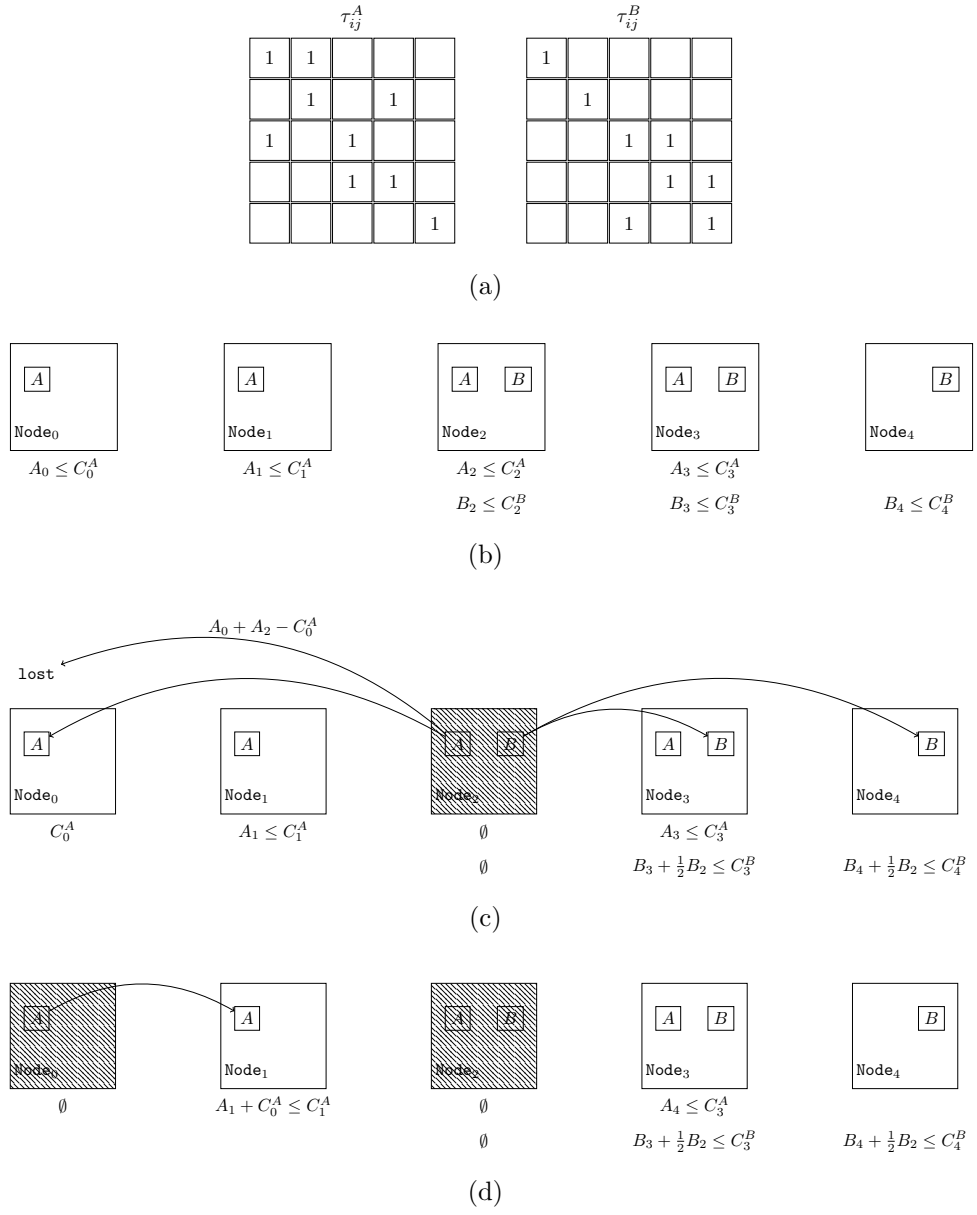


Figure 4.3: Example described in Section 4.3. The topology of interactions among components is represented in (a), the starting configuration of the system is depicted in (b), the load redispatching after a failure of $\texttt{Node}_2$ is represented in (c), the failure, and consequent reboot, of $\texttt{Node}_0$ after $T_{A,0}^{\text{threshold}}$ time units is depicted in (d).

time with respect to reboot, but guarantees a complete reset of node i. During reboot and recovery, all demand $S^{\text{load}}i$, for each service S, is not satisfied.

In addition to overload conditions, after an exponentially distributed period of time, with rate depending on the value of the current demand, each node also experiences a failure, which requires a recovery action.

4.3.2 Logical architecture of the example

The first step to tackle a system as the one described in Section 4.3.1 is to describe its logical architecture. At this stage, it is necessary to identify the generic component C^g, list the *state variables* that characterize its state, the subset of *Dependency-aware state variables*, and the *actions* that characterize state changes. Notice that several different choices are ofter available, and the modeler has to select one of them according to the level of details required by the analysis and the candidates impact on performance. For instance, in representing the example of Section 4.3.1, a generic service can be considered as C^g, abstracting the concept of node and recasting reboot and recovery to behaviors of the service. Another choice could be to consider a generic demand as C^g, abstracting both services and nodes, and interpreting load addressing, sharing or loosing as components' states.

Here, a generic node is selected as C^g, defining the presence/absence of service $S \in \{A, B\}$ as a feature of specific components. The example logical architecture can be depicted as in Figure 4.3b where, for $n = 5$, the specific components are represented as squares with/without services A or B, and with a specified load S_i and capacity C_i^S. Topologies of interdependencies are represented in Figure 4.3a. To illustrate the system behavior, Figure 4.3c depicts how the failure of $\texttt{Node}_2$ triggers the load-sharing for both service A and B: $\texttt{Node}_3$ and $\texttt{Node}_4$ can accept the received load from $\texttt{Node}_2$ without problems, whereas $\texttt{Node}_0$ reaches its capacity limit for service A and part of the load is lost. As depicted in Figure 4.3d, after $T_{A,0}^{\text{threshold}}$ time units, $\texttt{Node}_0$ fails and sends its load to $\texttt{Node}_1$ just before rebooting.

A detailed description of SVs and actions is posponed to Section 4.6.1, where the SAN formalism is adopted, but it is interesting to note here that the SV capturing node up/down state can be local to each specific component. Instead, in order to describe information exchange among components, the template SVs describing the demand for service S on $\texttt{Node}_i$ and the up/down state of service S, called in the following *Demand_S* and *Up_S* , respectively, have to be dependency-aware.

4.4 Theoretical discussion

In this section, a detailed analysis about the template replication process, as defined by the *SSRep*, *CSRep* and *DARep* approaches, is provided. The adopted notation has been chosen focusing on the possibility to compare the approaches. In particular, the core logic relies on indexing mechanisms, one for each approach, in the following indicated with f_{SS}, f_{CS} and $\mathcal{D}$, respectively.

4.4.1 Formalism and modeling features

Following the formal definition of the category of addressed systems, the aim of this section is to present a formalism for:

1. defining a template model M^t to realize the generic component C^g,
2. automatically producing from M^t the *specific* models $M_0,\ldots, M_{n-1}$ to realize the specific components $C_0,\ldots, C_{n-1}$, respectively, and
3. composing the *specific* models $M_0,\ldots, M_{n-1}$ to form the overall system model M^{sys}.

Similarly to the notation used for components SVs and actions, define a template model and a specific model as follows:

$$M^t = \left(W^0,\ldots,W^{l_t-1}; b^0,\ldots,b^{m_t-1}\right), \tag{4.11}$$

$$M_i = \left(W_i^0,\ldots,W_i^{l_t-1}; b_i^0,\ldots,b_i^{m_t-1}\right), \tag{4.12}$$

where W and b represent, respectively, the SVs and the actions of the models.

Let $\mathcal{B}$ and $\mathcal{B}_i$ be the set of actions of the template model and of the specific model, respectively, such that $m_t = |\mathcal{B}| = |\mathcal{B}_i|$, for all i. Depending on the characteristics of the approach used to model non-anonymous replication, the numbers l_t and m_t of SVs and actions for the template model can be different from the corresponding ones (l and m) of a generic component.

There exists a great variety of formalisms to define the model M^t based on SVs and actions, each one with its specificities. To keep the presentation of the needed concepts and facilities for modeling non-anonymous replication as general as possible, none of such existing formalisms is fully described. However, to facilitate the reading of this section, a few representatives of widely adopted families of formalisms are briefly recalled in the following, also discussing at high level the correspondence with the basic concepts of the general formalism used in this chapter.

- Stochastic Petri Nets [82–84]. Each *place* corresponds to a SV, each *transition* corresponds to an action. To the best of our knowledge, the explicit associations transition-action has been proposed in [82] and SPNs have been the first formalism to benefit from this abstract interpretation. Dialects introduce different kinds of enabling conditions, different probability distributions for the time period between activities enabling and completion, and different *reactivation* policies, such as aging or enabling memory and preselection [82].

- Performance Evaluation Process Algebra [50, 51]. This formalism explicitly tackles actions, while SVs are implicitly defined by composition rules among actions. Traditionally, the stochastic process that has as state space the process algebra derivation graph is considered in the analysis. Nevertheless, there exists an interesting dialect, called PEPAk [85], that allows a direct definition of SVs and can be easily employed to implement the approaches described in this chapter. Although the PEPA family of formalisms was originally introduced by the performance community, it is adopted also for dependability modeling, especially for availability evaluation.

- Stochastic Fault Trees [52]. Each *node* corresponds to a SV, each *failure event* corresponds to an action and *gates* are defined to compose failure events. The SFTs family of formalisms is mainly adopted in reliability modeling, and is rather popular in industrial environments, especially the static and stateless ancestor, Fault Trees [43, 86].

As shown, the building blocks SVs and actions are general concepts actually at the basis of a vast set of formalisms and dialects, suitable to define Markovian, semi-Markov and non continuous time Markov process stochastic models. All of them are then appropriate to consider the techniques for modeling non-anonymous replication addressed in this chapter.

In order to obtain a system model based on a replicated template model, different compositional formalisms and operators have been introduced and are commonly used [87, 88], as already pointed out in Section 4.1. Some approaches to submodels composition are based on syncronizations [50], where actions are shared among submodels. Others employ specific operators for models replication and joining, namely *Rep* and *Join* [33], which are based on sharing of SVs among submodels. In this chapter, the state-sharing based operators are adopted to generate or compose the submodels M_i. Notice that the Rep operator is a state of the art submodels compositional operators that has proved its power when employed where anonymous replication is enough, being adopted in a vast range of system models.

The SVs W_i^p and W_j^q, defined, respectively, in the models M_i and M_j, are *shared* if $W_i^p = W_j^q$ always holds. In this way, the actions of M_i can modify W_i^p and at the same time W_j^q of M_j, thus representing the impact of C_i on C_j. A SV is *local* if it is not shared among any of the SVs of other models.

The *Join* operator composing two or more models has as a result a new model that contains all the actions and the local SVs of the composing models, and all the SVs shared among the composing models. Notation adopted for the *Join* operator and the resulting model are shown in the following example, where two models $M_x = \left(W_x^0, W_x^1, W_x^2\,;\, b_x^0, b_x^1\right)$ and $M_y = \left(W_y^0, W_y^1, W_y^2\,;\, b_y^0, b_y^1\right)$ are joined sharing the variables $W_x^1 = W_y^0$ and $W_x^2 = W_y^1$:

$$\begin{aligned}\mathcal{J}\Big(M_x, M_y \,\Big|\, W_x^1 = W_y^0, W_x^2 = W_y^1\Big) = \\ = \Big(W_x^0, W_y^2, W_{x,y}^{1,0}, W_{x,y}^{2,1}\,;\, b_x^0, b_x^1, b_y^0, b_y^1\Big),\end{aligned} \tag{4.13}$$

where the compact notation $W_{i,j}^{p,q}$ is used in the final joined model to represent the SV obtained sharing the variables W_i^p and W_j^q (instead, when $p = q$ or $i = j$ only one index is used). For the sake of brevity, the name of the models is omitted in the notation for shared SV.

The *Rep* operator applied to a template model has as result a new model obtained joining n anonymous replicas (identical copies) of the template model, where the shared SVs are shared among all the replicas. Thus, a SV can be local to each replica or shared among all replicas (all-or-none sharing strategy). The replicas are anonymous in the sense that the Rep operator does not assign to them any index that could be used in the template model to identify a specific replica. This is actually what is currently available in most popular model-based analysis tools for dependability and performance assessment.

Composing operator *Rep* can be used to automatically generate n replicas $M_0,\ldots,M_{n-1}$ of the template model M^t, where the index i of each replica is the different value of a SV defined as local to each replica. Notation adopted for the *Rep* operator and the resulting model are shown in the following example, where a template model $M^t = \left(W^0, W^1\,;\, b^0\right)$ is replicated to obtain the replicas $M_0 = \left(W_0^0, W_0^1\,;\, b_0^0\right)$, …, $M_{n-1} = \left(W_{n-1}^0, W_{n-1}^1\,;\, b_{n-1}^0\right)$ and the final composed model is obtained by sharing W^0:

$$\begin{aligned}\mathcal{R}_n\Big(M^t \,|\, W^0\Big) = \mathcal{J}\Big(M_0, \ldots, M_{n-1} \,\Big|\, W_0^0 = \cdots = W_{n-1}^0\Big) \\ = \Big(W_{0,\cdots,n-1}^0, W_0^1, W_1^1, \ldots, W_{n-1}^1\,;\, b_0^0, \ldots, b_{n-1}^0\Big).\end{aligned} \tag{4.14}$$

Observe that:

1. Join takes as input a set of already defined models, thus cannot be used to *automatically generate* system component models.

2. Join can directly address $\mathcal{T}^h$ for $h = 0, \ldots, l-1$, because the modeler can manually define all M_i following (4.10) and share only the SVs, as defined in $\mathcal{T}^h$. This process is error prone and does not scale at increasing of the number n of system components.

3. Rep takes as input a template model and *automatically generates* n models, but they are identical (anonymous), i.e., Rep operator does not provide an index to use in the template to refer specific replicas. Although possible, it is not straightforward to tackle a system whose components are different (non-anonymous).

4. In Rep, a SV can be shared among all the replicas or none of them, thus this operator cannot directly address neither $\mathcal{T}$ nor $\mathcal{T}^h$, when W^h is a dependency-aware state variable.

Thus, using only the Join operator as it is, no template model M^t can be defined to represent C^g and then produce individual components models. On the other hand, it is possible to work around the fact that Rep is natively an anonymous replication operator and adopt it to produce non-anonymous replicas, by defining clever maps that relate the SVs of the specific components to the SVs of the template model, and similarly for the actions. The specific models will inherit the right set of SVs, with the right indices, as detailed in Sections 4.4.3 and 4.4.4.

Before describing in details the modeling approaches, general considerations can be done about model-based simulation. In order to evaluate the relevant measures defined in the model, the simulator needs to keep track of SVs values. A common strategy is to define for each action an *event* that encapsulates information about how the SVs are modified and the timing. Traditionally the events are stored in a list, the *event list*, although other choices are possible [89]. After an initialization phase in which all the SVs are declared and instantiated to their initial value, the simulation consists in a vast number of *batches*, each following a path of events and accumulating measures. Both the number of SVs and events have impact on simulator performance, as shown at the end of Sections 4.4.3, 4.4.4 and 4.4.5, where theoretical predictions about performance are reported.

4.4.2 Non-anonymous replication

Exploiting the notation introduced in Section 4.4.1, a formal definition of three different approaches, *SSRep*, *CSRep*, and *DARep* as described in Sec-

tion 4.2, is presented. Notice that the original idea of non-anonymous replication was stated only for specific models, not studying it as a general approach. In particular, to the best of this thesis author's knowledge:

- *SSRep* first appeared in [77, 78], in the electrical context,
- an approach similar to *DARep* first appeared in [**?**], where – in the context of Wireless Sensor Networks – a replication process based on template has been considered.

Both *CSRep*, presented in [79], and *DARep*, presented in [13], have been developed as part of the candidate's PhD work. In this thesis a new formalization of the three approaches is discussed. The general treatment promotes not only the definition of a "framework within a framework", i.e., the design of an abstract procedure to model systems through non-anonymous replication within the Möbius Modeling Framework, but also the possibility to study the complexity of the approaches and the definition of rigorous specifications that can guide the implementation of the approaches in other modeling environments.

To exploit non-anonymous replication of a given template model, the following features are required:

i) data structures to manage the parameters of the template model M^t, as a function of a generic replica index of the template, in particular the topology associated to each dependency-aware SV, that is the array Δ^h;

ii) an indexing mechanism for M_i, used to access the parameters associated to each replica of the template and to define the actions as a function of the index;

iii) a sophisticated one-to-one mapping f to relate the SV $W^{f(h,j)}$ of the template model with the SV SV^h_j of the specific components;

iv) a one-to-one mapping $g : \mathcal{A} \to \mathcal{B}$ from the set of generic component actions and corresponding template model actions;

v) for each replica M_i of the template, constant-time read/write access, using a relative index, to the replicas of each dependency-aware SV SV^h shared among the other replicas of the template, as defined in Δ^h_i;

vi) automatic generation of M_i from the template M^t, where interactions among M_i occur through dependency-aware SVs, differently modeled in each of the three approaches;

vii) automatic generation of the final composed model joining the specific models M_i and addressing the interactions among M_i through dependency-aware SVs.

SSRep and *CSRep* use the same indexing mechanism, that relies on a SV 'Index" local to each replica M_i, i.e., the index of each replica is part of the state of the replica. Moreover, *SSRep* and *CSRep* are based directly on the *Rep* operator, that supports above features vi) and vii).

DARep is based on a new operator that explicitely supports ii) and vi) and on the *Join* operator that supports vii).

For all the approaches, feature v) is obtained defining the replicas of the dependency-aware SVs with arrays (for *SSRep* and *CSRep* at definition of the template, while for *DARep* automatically when the replicas M_i are generated).

In the following, to avoid too heavy notation, the topology-related parameters are not explicitly included in the formal definition of M^t, although they are used in the models.

4.4.3 *State-Sharing Replication*

The *State-Sharing Replication* (*SSRep*) approach relies on n SVs, one for each replica of the template, for each dependency-aware SV^h of the modeled component C^g. *State-Sharing Replication* hasn't been conceived to exploit the dependency topology when defining the variables shared among the replicas of the template. Thus all the dependency-aware SVs are shared among all the replicas of the template model, independently on the topology. Therefore, the efficiency of *State-Sharing Replication* is limited by the fact that it assumes a complete graph of interactions among the replicated components.

The *SSRep* approach consists of a simple architecture: a single template model, called M^{ss}, is replicated n times by the *Rep* operator to obtain the system model M^{sys}, where an index is assigned to each replica.

The model M^{ss} is structured in two parts: one defining the generic component M^{g} as a function of the index of the replicas, represented by the SV $Index \in \mathbb{N}$, including W_i^h and b^k, and one that initializes the indices for each replica of the template, including the SVs $Count \in \mathbb{N}$ and $Start \in \mathbb{N}$, and the instantaneous action *tInit*. Formally,

$$M^{ss} = \Big(W^0, \ldots, W^{n(l-1)+n-1}, Index, Count = n, Start = 1\,; \\ b^0, b^1, \ldots, b^{m-1}, tInit\Big), \tag{4.15}$$

where the matching between $\{W^{f_{\rm SS}}\}$ and $\{SV_j^h\}$ is based on the function $f_{\rm SS}(h,j) = n \cdot h + j$, such that

$$W^{n\cdot h+j} = W^{f_{\rm SS}(h,j)} = SV_j^h, \text{ for all } h \text{ and } j, \tag{4.16}$$

$$b^k = g_{\rm SS}(a^k), \text{ for all } k, \tag{4.17}$$

where $g_{\rm SS}$ simply adds "$Count$==0" to the enabling condition of a^k. Thus, the replicas of SV^h are modeled by n SVs:

$$W^{n\cdot h}, W^{n\cdot h+1}, \ldots, W^{n\cdot h+n-1}. \tag{4.18}$$

The *Rep* operator generates n replicas M_i of M^{ss}:

$$M_i = \Big(W_i^0, \ldots, W_i^{n(l-1)+n-1}, \mathit{Index}_i, \mathit{Count}_i, \mathit{Start}_i\,; \\ b_i^0, b_i^1, \ldots, b_i^{m-1}, \mathit{tInit}_i\Big), \tag{4.19}$$

that are composed to generate the overall system model:

$$M^{\rm sys} = \mathcal{R}_n\Big(M^{ss} \,\Big|\, W^0, \ldots, W^{n(l-1)+n-1}, \text{Count}\Big) \tag{4.20}$$

$$= \Big(W^0_{0,\cdots,n-1}, \ldots, W^{n(l-1)+n-1}_{0,\cdots,n-1}, \text{Index}_0, \ldots, \text{Index}_{n-1}, \\ \text{Count}_{0,\cdots,n-1}, \text{Start}_0, \ldots, \text{Start}_{n-1};\, b_0^0, \ldots, b_{n-1}^0, \\ \ldots, b_0^{m-1}, \ldots, b_{n-1}^{m-1}, \text{tInit}_0, \ldots, \text{tInit}_{n-1}\Big),$$

where each $W^{n\cdot h+j}$ is shared among the n replicas of the template, as defined by $W^{n\cdot h+j}_{0,\ldots,n-1}$.

The SV *Count* is shared among all the replicas M_i and is used to initialize the index of each replica Index_i at completion of tInit_i, through the steps in the order specified as follows:

$$\begin{aligned} &\text{Count}_{0,\cdots,n-1} = \text{Count}_{0,\cdots,n-1} - 1, \\ &\text{Index}_i = \text{Count}_{0,\cdots,n-1}, \\ &\text{Start}_i = 0, \end{aligned} \tag{4.21}$$

where Index_i and Start_i are local to each replica, and tInit_i is enabled if $\mathit{Start}_i = 1$ and $\mathit{Count}_{0,\cdots,n-1} > 0$. All the actions b_i^k are enabled after the initialization of all the indices, i.e., when $\mathit{Count}_{0,\cdots,n-1} = 0$.

Notice that M_i has $n \cdot l + 3$ SVs and $m + 1$ actions, whereas $M^{\rm sys}$ has $(2 + l)n + 1$ SVs and $n(m + 1)$ actions. During the initialization phase, the

simulator has to assign $n \cdot l + 3$ SVs to each replica and then shares the corresponding SVs.

Moreover, each $W^{n \cdot h+j}$, i.e., each replica of SV^h, can be read/write accessed by each M_i, independently of Δ_i^h. In fact, the function f_{SS} does not depend of the actual read/write access topology, but only on n. Thus, *SSRep* assumes a complete graph of interactions among components, even if the actual number of ones in $\mathcal{T}$ is much less than n^2. Obviously, the definition of each action b^k is based on the actual interactions among components, as defined in $\mathcal{T}$, using Δ^h. For example, if the enabling condition of the action b^k is given in the template model by $W^{n \cdot h+j} = 1 | j \in \Delta^h$, then for each replica i the enabling condition of b_i^k is $W_i^{n \cdot h+j} = 1 | j \in \Delta_i^h$.

During the system model simulation, the generation of new events in the event list, which requires evaluating the enabling conditions of b_i^k, can involve the unnecessary checking of all the values of $W_i^{f_{\text{SS}}}$, thus leading to degraded efficiency of the method.

Finally, calling k_i^h the number of possible values that SV_i^h can assume (maybe infinite), and considering that all the $W_i^{f_{\text{SS}}}$ are shared among all the replicas, the number of states in the system model is at most

$$\prod_{h=0}^{l-1} \prod_{i=0}^{n-1} k_i^h, \tag{4.22}$$

where the SVs related to the indexing mechanism do not count because the mechanism itself is implemented using instantaneous actions. Also, it is interesting to notice that the number of states of each replica M_i can be different from that of the other replicas, depending on the number of shared SVs.

4.4.4 *Channel-Sharing Replication*

The *Channel-Sharing Replication* (*CSRep*) approach has been introduced to mitigate the inefficiency of *State-Sharing Replication* at inizialization time. The approach relies on:

1. defining as local to each replica all its dependency-aware SVs following the actual dependency topology,

2. using a small-sized channel, i.e., a channel composed by a small number of SVs, shared among all the replicas, to exchange the values of dependency-aware SVs only among the interested replicas, each time they are updated.

In particular, for each dependency-aware SV^h of the modeled component C^g, δ^h local state variables W^p are defined in the template to represent the set Δ_i^h of replicas of SV^h.

Differently from *State-Sharing Replication*, for which all the dependency-aware SVs are shared among all the replicas, with *CSRep* all the dependency-aware SVs are considered local to the replicas and only the channel requires a small set of SVs to be shared among all the replicas. Therefore, a reduction in the simulation time overhead due to the sharing of a high number of SVs among replicas is expected.

As for *SSRep*, also the *CSRep* approach defines a single template model, called M^{cs}, that is replicated n times by the *Rep* operator. The template M^{cs} is structured in three parts: 1) the generic component M^{g} as a function of the index of the replicas, represented by the SV $Index \in \mathbb{N}$, including W_i^h and b^k, 2) the initialization of the indices (following the description already provided for *SSRep*), 3) the channel and the read/write channel operations, including $Channel \in \mathbb{N} \times \mathbb{N} \times DataType$, $ToChannel \in \mathbb{N}$, $ReadyChannel \in \mathbb{N}$ and the instantaneous actions *read* and *write*. Formally, the architecture of *CSRep* is:

$$\begin{aligned} M^{cs} = \Big(& W^0, \ldots, W^{f_{\text{CS}}(l-1,n-1)}, Index, Count = n, Start = 1, \\ & Channel, ReadyChannel = 1, ToChannel = 0\,; \\ & b^0, b^1, \ldots, b^{m-1}, tInit, read, write \Big). \end{aligned} \tag{4.23}$$

The SV *ToChannel*, initialized to 0, is local to each replica and it is used to trigger (*ToChannel*== 1) the writing on the channel each time a dependency-aware SV is updated.

The SV *ReadyChannel*, initialized to 1, is shared among all replicas and is used to lock the channel (*ReadyChannel*== 0) from writing until the data are read by all the destination replicas.

The channel is shared among all replicas and is defined by 3 fields: the SV index $\in \mathbb{N}$, the SV ndest $\in \mathbb{N}$ and the set of SVs data $\in$ DataType, used to represent, respectively, the index of the sender replica, the current number of destination replicas that have not yet read the data and the value of (or in general the data record of type *DataType* associated to) the dependency-aware SV to send to the replicas. The field *index* is used to identify whether a replica i is a destination depending on the topology of the dependencies, i.e., to get the entry of a constant n-sized array $\hat{\Delta}_i^h$ associated to each replica i, defined as:

$$\hat{\Delta}_i^h[j] = \begin{cases} 1 & \text{if } j \in \Delta_i^h, \\ 0 & \text{otherwise.} \end{cases} \tag{4.24}$$

The replica i is a destination of the data of the channel if $\hat{\Delta}_i^h$[Channel.index] == 1, i.e., if Channel.index $\in \Delta_i^h$. The field *ndest* is used to unlock the channel when all the destination replicas have read all the data of the channel (Channel.ndest == 0).

The *Rep* operator generates n replicas M_i of M^{cs}:

$$\begin{aligned} M_i = \Big(& W_i^0, \ldots, W_i^{f_{\text{CS}}(l-1,n-1)}, \mathit{Index}_i, \mathit{Count}_i, \mathit{Start}_i, \\ & \mathit{Channel}_i, \mathit{ReadyChannel}_i, \mathit{ToChannel}_i\,; \\ & b_i^0, b_i^1, \ldots, b_i^{m-1}, \mathit{tInit}_i, \mathit{read}_i, \mathit{write}_i \Big), \end{aligned} \tag{4.25}$$

that are composed to generate the overall system model, where *none* of $W^{f_{\text{CS}}}$ is shared among replicas:

$$\begin{aligned} M^{\text{sys}} &= \mathcal{R}_n\Big(M^{ss} \;\Big|\; \mathit{Count}, \mathit{Channel}, \mathit{ReadyChannel} \Big) \\ &= \Big(W_0^0, \ldots, W_{n-1}^{f_{\text{CS}}(l-1,n-1)}, \mathit{Index}_0, \ldots, \mathit{Index}_{n-1}, \\ &\quad \mathit{Count}_{0,\ldots,n-1}, \mathit{Start}_0, \ldots, \mathit{Start}_{n-1}, \mathit{Channel}_{0,\ldots,n-1}, \\ &\quad \mathit{ToChannel}_0, \ldots, \mathit{ToChannel}_{n-1}, \mathit{ReadyChannel}_{0,\ldots,n-1}; \\ &\quad b_0^0, \ldots, b_{n-1}^0, \ldots, b_0^{m-1}, \ldots, b_{n-1}^{m-1}, \mathit{tInit}_0, \ldots, \mathit{tInit}_{n-1}, \\ &\quad \mathit{read}_0, \ldots, \mathit{read}_{n-1}, \ldots, \mathit{write}_0, \ldots, \mathit{write}_{n-1} \Big). \end{aligned} \tag{4.26}$$

The mapping between $\{W_i^{f_{\text{CS}}}\}$ and $\{SV_i^h\}$ is based on the function $f_{\text{CS}}(h,r) = \sum_{k=0}^{h} \delta^k + r$, such that:

$$W_i^{\sum_{k=0}^{h} \delta^k + r} = W_i^{f_{\text{CS}}(h,r)} = \begin{cases} SV_j^h & \text{for } j = \Delta_i^h(r), r < \delta_i^h, \\ 0 & \text{for } r \geq \delta_i^h, \end{cases} \tag{4.27}$$

where δ^h is the number of local state variables W^p included in the template M^{cs} for each SV^h, defined as

$$\delta^h = \max_i \delta_i^h = \max_i \sum_j \mathcal{T}_{i,j}^h. \tag{4.28}$$

In fact, being generic, M^{cs} must contain, for each SV^h, δ^h local state variables W^p necessary to represent the set Δ_i^h of δ_i^h replicas of SV^h that M_i can access in read/write mode. Local state variables W^p are listed in the following,

where each row models the replicas of a different state variable SV^h:

$$\begin{array}{ccc} W^0 & \dots & W^{\delta^0-1}, \\ W^{\delta^0} & \dots & W^{\delta^0+\delta^1-1}, \\ & \vdots & \\ W^{\sum_{k=0}^{h}\delta^k} & \dots & W^{\sum_{k=0}^{h+1}\delta^k-1}, \\ & \vdots & \\ W^{\sum_{k=0}^{l-2}\delta^k} & \dots & W^{\sum_{k=0}^{l-1}\delta^k-1}. \end{array} \tag{4.29}$$

The mapping between component and model actions relies on the function g_{CS}, as follows:

$$b^k = g_{\mathrm{CS}}(a^k), \text{ for all } k, \tag{4.30}$$

where g_{CS} extends b^k with the assignment ToChannel = 1, each time a local state variable W^p is updated.

The actions *write* and *read* are enabled respectively by the sender replica M_j to write the data into the channel, and by all the destination replicas M_i to read the data from the channel.

In particular, *write* is enabled if:

$$\text{ToChannel} == 1 \text{ and } \text{ReadyChannel} == 1, \tag{4.31}$$

i.e., when, respectively, there are new data to send and when the channel is ready to accept the data. At completion of *write* the channel is updated and locked, performing the following assignments:

$$\begin{aligned} &\text{Channel.index} = \text{Index}, \\ &\text{Channel.ndest} = \delta^h_{\text{Index}}, \\ &\text{Channel.data} = W^{f_{\mathrm{CS}}(h,r)}, \\ &\text{ToChannel} = 0, \\ &\text{ReadyChannel} = 0, \end{aligned} \tag{4.32}$$

where $W^{f_{\mathrm{CS}}(h,r)}$ is the value that has to be transmitted through the channel.

The action *read* is only enabled when the channel has been updated with new data to send and the current replica is the destination of the data of the channel, i.e.:

$$\text{ReadyChannel} == 0 \text{ and } \hat{\Delta}^h[\text{Channel.index}] == 1. \tag{4.33}$$

At completion of the action *read* the following assignments are performed in order:

$$\begin{aligned} &W^{f_{\mathrm{CS}}(h,r)} = \text{Channel.data}, \\ &\text{Channel.ndest} = \text{Channel.ndest} - 1, \\ &\text{if Channel.ndest} == 0 \text{ then ReadyChannel} = 1, \end{aligned} \tag{4.34}$$

i.e., the local state variable $W^{f_{\mathrm{CS}}(h,r)}$, corresponding to the current replica, and the state of the channel are updated. Then, if the replica is the last destination of the data, the channel is unlocked, so to be ready to accept new data to send.

Thus, whenever an action b_j^k of M_j changes the value of $W_j^{f_{\mathrm{CS}}(h,r)}$, also the value of $ToChannel_j$ switches from 0 to 1. Then the action $write_j$ is enabled and, being instantaneous, immediately completes performing the assignments in (4.32). In particular, it assigns $\text{Channel}_{0,\ldots,n-1}.\text{index} = j$ and $\text{Channel}_{0,\ldots,n-1}.\text{data} = W_j^{f_{\mathrm{CS}}(h,r)}$, locks the channel assigning the value $ReadyChannel_{0,\ldots,n-1} = 0$ and then all the $read_i$ for which $j \in \Delta_i^h$ are enabled. Instantaneously, after each $read_i$ for which $\hat{\Delta}_i^h[\text{Channel}_{0,\ldots,n-1}.\text{index}] == 1$ read the channel and assigned $W_i^{f_{\mathrm{CS}}(h,s)} = Channel_{0,\ldots,n-1}.data$, *ReadyChannel* can switch from 0 to 1, freeing the channel for further communications.

When a replica of more than one dependency-aware SV or more than one replica of the same dependency-aware SV are updated at the same time, a sequence of consecutive transmissions using the same channel can be easily modeled, or alternatively the channel can be easily extended by adding new fields.

The following considerations apply to the *CSRep* approach:

- it explicitly tackles the interaction among replicas following the topology $\mathcal{T}^h$ and uses only a constant small number of SVs shared among replicas. Then, depending on how sparse is the matrix $\mathcal{T}^h$, a (potentially great) performance improvement can be achieved with respect to *SSRep* that always relies on a complete dependency graph;

- during the system model simulation, each change of W_j^q produces 1 new event for $write_j$ and 1 new event for each $read_i$, for which $j \in \Delta_i^h$. This implies a time overhead due to the synchronization of the dependency-aware SVs each time they are updated;

- the state space of M^{sys} obtained with *CSRep* is equal to the state space obtained with *SSRep* [43].

4.4.5 *Dependency-Aware Replication*

The *Dependency-Aware Replication* (*DARep*) approach takes advantage of the topology of dependencies among system components, by sharing subsets of the replicas of each dependency-aware SV among only those replicas that need to access them. The goal is to improve the performance of the simulation solvers with respect to the approaches *SSRep* and *CSRep*. In *SSRep*, all the replicas of each dependency-aware SV are shared among all the replicas of the template. In *CSRep*, the replicas of each dependency-aware SV are local to the replicas of the template, but the adopted channel structure needs do be managed. Thus, with *SSRep* and *CSRep*, extra computation time and possibly intricate data structures are necessary to properly manage the correct dependency relations at model definition and solution.

To overcome this drawback, the *DARep* approach extends the *Rep* composition operator in order to: 1) generate automatically the replicas of the dependency-aware SV associated to each replica of the template model, 2) share different subsets of the replicas of the dependency-aware SV among different subsets of the replicas of the template model. Thus, it merges the advantages of the *Join* and *Rep* compositional operators.

The *DARep* approach is based on:

1. extending the template model with two functions *Index()* and *Deps()* that represent, respectively, the index of the replicas of the template and the replicas of the dependency-aware SVs included in each replica of the template, following the actual system topology,

2. defining the operator $\mathcal{D}$ that, automatically, i) generates the indexed replicas of the template model supporting the functions *Index()* and *Deps()*, and ii) defines the overall system model, composing through the *join* operator the indexed replicas of the template.

In particular, for each dependency-aware SV^h of the modeled component C^g, δ_i^h shared state variables W_q^p are automatically generated in M_i to represent the set Δ_i^h of replicas of SV^h. Differently from *State-Sharing Replication*, for which all the replicas of the dependency-aware SVs are shared among all the replicas, and from *CSRep*, for which local replicas of the dependency-aware SVs are considered, in *DARep* the replicas of the dependency-aware SVs are shared among M_i following the actual system topology. Thus, both unnecessary interactions among replicas (a major limitation of *SSRep*) and synchronization of replicas values (a major limitation of *CSRep*) are avoided. This improvement is expected to be reflected in a reduced simulation time for *DARep* with respect to *SSRep* and *CSRep*.

The *DARep* approach defines a single template model, called M^{darep}, that is replicated n times by the newly defined $\mathcal{D}$ operator.

The template M^{darep} is defined as follows:

$$M^{darep} = \big(W^0, \ldots, W^{l-1}; \\ b^0, \ldots, b^{m-1}; \text{Index}(), \text{Deps}()\big). \tag{4.35}$$

Index() and *Deps()* are two functions used in the definition of the actions of the template and supported by the $\mathcal{D}$ operator.

The $\mathcal{D}$ operator generates n replicas M_i of M^{darep} and the list $\mathcal{J}^{shared}$ of the replicas of the dependency-aware SVs shared among M_i, following the topology $\mathcal{T}^h$ associated to each dependency-aware SV:

$$\mathcal{D}\big(\mathcal{T}^0, \ldots, \mathcal{T}^{l-1}, M^{darep}\big) = \big(M_0, \ldots, M_{n-1}, \mathcal{J}^{shared}\big). \tag{4.36}$$

Each M_i is defined as:

$$M_i = \big(\{W_j^0 \, \forall j \in \Delta_i^0\}, \ldots, \{W_j^{l-1} \, \forall j \in \Delta_i^{l-1}\}, \\ b_i^0, b_i^1, \ldots, b_i^{m-1}, \mathit{Index}_i(), \mathit{Deps}_i()\big), \tag{4.37}$$

where

$$\text{Index}_i() = i, \text{ for all } i, \\ \text{Deps}_i(h, s) = W_j^h | j = \Delta_i^h(s), \text{ for all } i, h \text{ and } s, \tag{4.38}$$

and the matching between $\{W^{f_{\text{darep}}}\}$ and $\{SV^h\}$ and between $\{b^k\}$ and $\{a^k\}$ is based on the identity functions $f_{\text{darep}}(h) = h$ and $g_{\text{darep}}(a) = a$, respectively, such that

$$W_j^h = W_j^{f_{\text{darep}}(h)} = SV_j^h, \text{ for all } h \text{ and } j, \tag{4.39}$$

$$b_i^k = g_{\text{darep}}(a_i^k) = a_i^k, \text{ for all } k \text{ and } i. \tag{4.40}$$

The list $\mathcal{J}^{shared}$ is generated by the $\mathcal{D}$ operator considering as shared all the SVs of different replicas M_i having the same name, i.e., formally:

$$\mathcal{J}^{shared} = \cup_{i,h,k}\{M_i.W_j^h = M_j.W_j^h\}, \tag{4.41}$$

where the low dot operator $M.W$ is used to refer to the SV W of the model M.

The models M_i are composed to generate the overall system model using the *Join* operator:

$$\begin{aligned} M^{\text{sys}} &= \mathcal{J}\big(\mathcal{D}\big(\mathcal{T}^0, \ldots, \mathcal{T}^{l-1}, M^{darep}\big)\big) \\ &= \mathcal{J}\big(M_0, \ldots, M_{n-1} | \mathcal{J}^{shared}\big) \\ &= \big(W^0_{\mathcal{J}_0^0}, \ldots, W^0_{\mathcal{J}_{n-1}^0}, \ldots, W^{l-1}_{\mathcal{J}_0^{l-1}}, \ldots, W^{l-1}_{\mathcal{J}_{n-1}^{l-1}}; \\ &\quad b_0^0, \ldots, b_{n-1}^0, \ldots, b_0^{m-1}, \ldots, b_{n-1}^{m-1}\big), \end{aligned} \tag{4.42}$$

where $\mathcal{J}_j^h = \{i | j \in \Delta_i^h\}$ is the list of replicas of the templates that access W_j^h. Thus, $W^h_{\mathcal{J}_j^h}$ represents the replica W_j^h shared among all the replicas of the templates listed in $\mathcal{J}_j^h$.

4.5 Implementation in Möbius

To implement and evaluate the approaches presented in this chapter, the Möbius modeling framework [5] and its supporting tool Möbius [57] have been used. Among the formalisms available in Möbius, our models are defined using the SAN formalism [53]. In this section the goal is not to provide a complete and detailed model representing a concrete instance of a targeted system, but to describe how the basic characteristics of each approach can be actually obtained. When appropriate, the case study presented in Section 4.3 is used to make concrete examples. Then, in Section 4.6 the SAN model for the load-sharing case study is discussed and evaluated to compare the performance of the three replication approaches, assuming $n = 100$ and δ varing between 1 and 148.

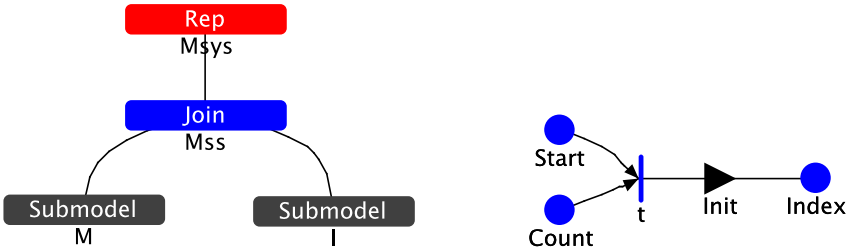
The indexing mechanism implemented for *SSRep* and *CSRep* is the same, and is identical to the one detailed in Section 3.5. The implementation of *SSRep* is employed in the first SG model, as discussed in Section 3.7.

The *DARep* does not require an indexing mechanism implemented using the SAN formalism but instead a way to define the auxiliary functions *Index()* and *Deps()*. The Möbius GUI produces an XML file for each submodel, composed model, etc. The choice fully discussed in this thesis, in particular in Section 4.5.3 and in the second SG model described in Chapter 6, is to implemenet the auxiliary functions *Index()* and *Deps()*, and the operator $\mathcal{D}$, through a Perl [60] program that manipulates those XML files.

4.5.1 *State-Sharing Replication*

The *SSRep* approach can be implemented in the Möbius framework by defining the composed model M^{sys} as depicted in Figure 4.4a, where:

- the operator *Rep* is used to automatically construct the overall system model M^{sys}, composed by the indexed replicas M_i of the template M^{ss}, as in (4.20);
- the template M^{ss} is defined by the operator *Join*, that composes the atomic SAN model M, representing the generic component M^{g}, and the atomic SAN model I, that initializes the index of the replicas, as in (4.15).



(a) (b)

Figure 4.4: *SSRep* approach: composed *Rep* model M^{sys} (a) and template SAN model I initializing the index of the replicas (b).

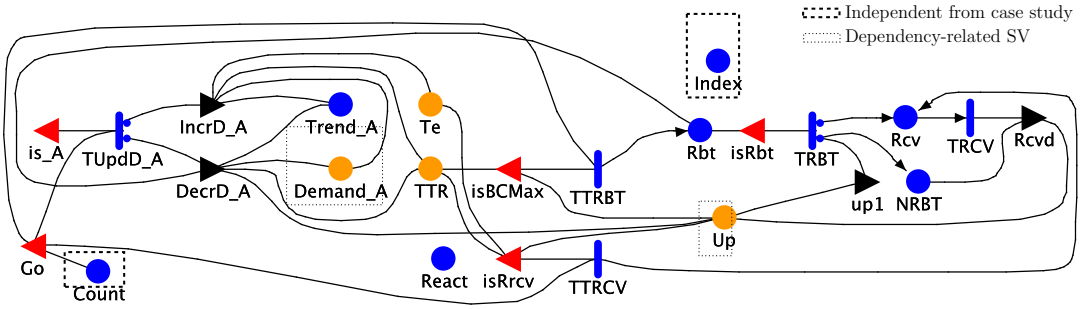


Figure 4.5: *SSRep* approach: template SAN model M defining the generic component for the case study described in Section 4.3.

The index of the replicas is defined by the place *Index* that is shared among the models M and I, but is local to M^{ss}. The action *tInit* of (4.15) and the steps in (4.21) are implemented by the activity t and the output gate *Init*, as shown in Figure 4.4b. When t completes, one token is removed from the places *Start* (local to M^{ss}) and *Count* (shared among replicas), and the following C++ code of *Init* is executed to set the index of the replica: $Index{->}Mark() = Count{->}Mark()$. This indexing mechanism was adopted in the first SG model, described in Section 3.5.

The n SVs listed in (4.18), representing the replicas of a dependency-aware SV^h, are automatically implemented by an n-sized array-type extended place W defined in the model M and shared among all M_i, such that the i-th entry of W, obtained with $W{->}Index(i){->}Mark()$, corresponds to $W^{n \cdot h+i}$. For example, in the atomic model M shown in Figure 4.5, there are two n-sized array-type extended places *Demand_A* and *Up*, representing the dependency-aware SVs.

Each action b^k in (4.15) is implemented by: an activity, the input and

output gates linked to the activity and the direct arcs linking the activity and places. The place *Index* can be used to define gates and parameters of the activity. For example, one of the 5 actions defined in the atomic model M shown in Figure 4.5 is implemented by the activity *TUpD_A*, the input gates *is_A* and *Go*, and the output gates *IncrD_A* and *DecrD_A*. Moreover, the constant parameter rate of the activity *TUpD_A*, depending on the indexed replica, is defined as a function of the place *Index*, as $LambdaD_A(Index{->}Mark())$.

The topology $\mathcal{T}$ is modeled by C++ constant data structures defined at compilation time, e.g., Δ^h is a C++ array of n different-sized arrays of short, one for each replica of the template.

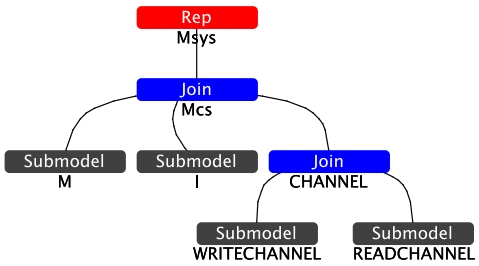
Thus, for example, in the atomic model M shown in Figure 4.5, the entry of the place *Up* corresponding to the generic replica of the template M^{ss} is given by $Up{->}Index(Index{->}Mark()){->}Mark()$. The entries of *Up* that are accessed by the generic replica of the template are given by the indices obtained scanning all the entries of $\Delta^h[Index{->}Mark()]$, i.e., by $Up{->}Index(\Delta^h[Index{->}Mark()][k]){->}Mark()$ for $k = 0, \ldots, \delta_i$, with $i = Index{->}Mark()$. Obviously, being the place *Up* shared among all M_i, the template M^{ss}, and so each M_i, can access all the entries of *Up*, although the entries out of the topology are not relevant.

From the inspection of the C++ code generated by Möbius, in particular the composed model *Comp*, it follows that during the initialization phase of the simulator each replica is created declaring *all* the $n \cdot l + 3$ SVs as local and then *all* the SVs are re-scanned and shared according to the description of Section 4.4.3. Considering the creation of a single SV as an atomic operation with constant complexity, the *SSRep* approach produces a time complexity of $\mathcal{O}(n^4)$ during the initialization phase, as detailed in Section 4.7.1.

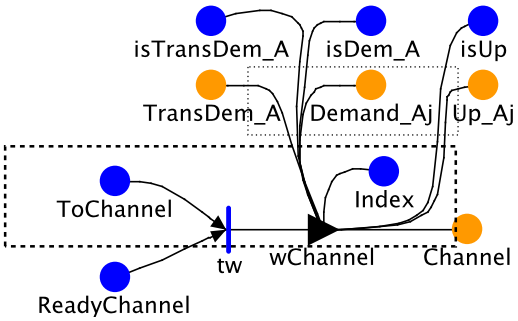
4.5.2 *Channel-Sharing Replication*

The *CSRep* approach can be implemented in the Möbius framework by defining the composed model M^{sys} as depicted in Figure 4.6a, where:

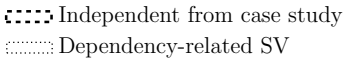
- like for the *SSRep* approach, the operator *Rep* is used to automatically generate the overall model M^{sys}, composed by the indexed replicas M_i of the template M^{cs}, as in (4.26);
- the template M^{cs} is the model defined by the operator *Join*, that composes three models: M, representing the generic component M^{g}, I, that initializes the index of the replicas, and the composed model *CHANNEL*, that models the channel and the related read and write operations, as in (4.23);

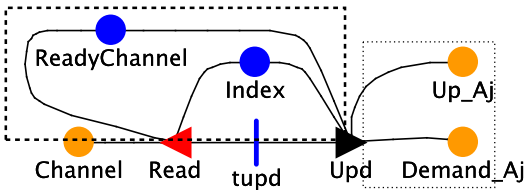


(a)



(b)





(c)

Figure 4.6: *CSRep* approach: composed *Rep* model (a) and SAN models *WRITECHANNEL* (b) and *READCHANNEL* (c) used respectively to write and to read the shared data into the channel.

- the model *CHANNEL* is defined by the operator *Join*, that composes the two atomic SAN models *WRITECHANNEL* and *READCHANNEL*

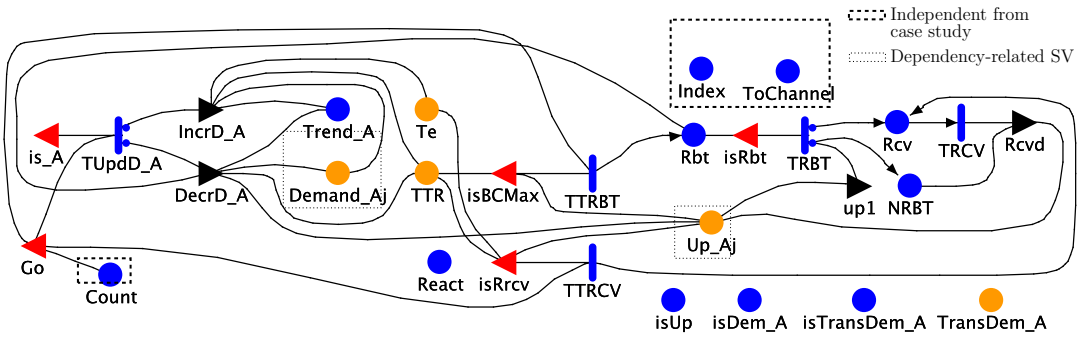


Figure 4.7: *CSRep* approach: template SAN model M defining the generic component for the case study described in Section 4.3.

that are used, respectively, by the sender replica to write the data into the channel and by all the destination replicas to read the data from the channel.

The δ^h local state variables necessary to represent the set Δ_i^h of δ_i^h replicas of SV^h, as listed in (4.29), are automatically implemented by an δ^h-sized array-type extended place W defined in the model M and local to each M_i. Therefore, the r-th entry of W on the replica M_i, obtained with $W{->}Index(r){->}Mark()$, corresponds to $W_i^{\sum_{k=0}^{h}\delta^k+r}$, as defined in (4.27).

For example, in the atomic model M shown in Figure 4.7, there are two local δ^h-sized array-type extended places *Demand_Aj* and *Upj*, representing the dependency-aware SVs. Moreover M also includes, as defined in (4.23), the place *Count*, shared among all the replicas of M^{cs}, the place *Index*, that is local to M^{cs}, and the place *ToChannel*, that is local to the template M^{cs}, but it is shared with the model *CHANNEL* and it is used to trigger the activation of the model *WRITECHANNEL* each time a dependency-aware SV is updated. For example, in Figure 4.7, at completion of the activity *TUpd_A* when the output gate *DecrD_A* updates one of the entries of *Demand_Aj*, then it also sets $ToChannel{->}Mark() = 1$.

The composed model *CHANNEL* is used to send the values of the dependency-aware SVs of a replica to other replicas, each time the dependency-aware SVs are updated by the model M. Each time one or more dependency-aware SVs of a replica are updated, the following steps are performed in the listed order:

1. the model *WRITECHANNEL* writes the new values (the data) in the channel and locks the channel (the channel is busy),

2. the model *READCHANNEL* of each destination replica updates the SVs of the replica with the data received from the channel,

3. when all destination replicas have received the data, the channel is unlocked, and can be used to transmit new data.

Figure 4.6b and 4.6c depict respectively the models *WRITECHANNEL* and *READCHANNEL*, when two dependency-aware SVs *Demand_Aj* and *Upj* are considered for the case study described in Section 4.3. The channel is the record-type extended place *Channel* that is shared among *WRITECHANNEL*, *READCHANNEL* and all replicas. In this example, the two dependency-aware SVs can be updated at the same time, thus an extended version of the definition of the channel given in 4.4.4 has been implemented, by adding new fields to the record place *Channel*.

The activity *tw*, the two input arcs and the output gate *wChannel* implement the action *write* in (4.23), (in particular, the enabling condition defined in (4.31) and the marking change in (4.32), that updates and locks the channel). The activity *tupd*, with the linked input and ouptut gates *Read* and *Upd*, implements the action *read* in (4.23), in particular, in *Read* the enabling condition defined in (4.33) and in *Upd* the marking change defined in (4.34).

4.5.3 *Dependency-Aware Replication*

The *DARep* approach can be implemented in the Möbius framework through the following steps, listed in order of execution:

1. Manual or automatic definition of the dependency topology associated to each dependency-aware SV, as in (4.6), and of the parameters of the system.

2. Automatic generation, based on the dependency topology and on the parameters of the system, of the C++ user defined library supported by the Möbius tool.

3. Manual definition of the template atomic SAN model M^{darep} that represents the generic component, as in (4.35).

4. Automatic generation of the atomic models M_i, $i = 1, \ldots, n-1$, as in (4.36) and (4.37).

5. Automatic generation of the list $\mathcal{J}^{shared}$ of the replicas of the dependency-aware SVs shared among M_i, as in (4.41).

6. Automatic definition, using the outputs of the steps 4 and 5, of the composed model obtained through the operator *Join*, as in (4.42).

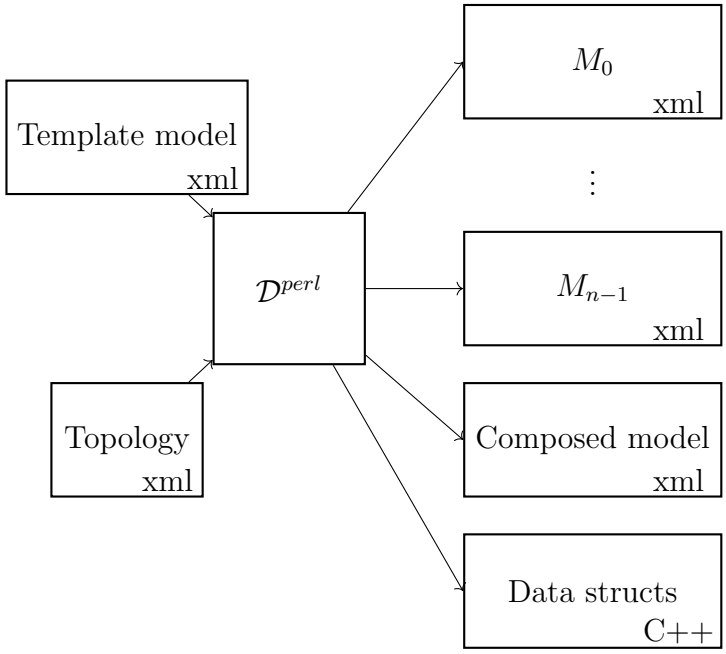


Figure 4.8: Architecture of $\mathcal{D}^{perl}$.

In this chapter, a *Perl* script, denoted by $\mathcal{D}^{perl}$, implements all the automatic steps 2, 4, 5 and 6, as depicted in Figure 4.8. In particular, in 1 the modeler defines an XML file representing the topology of interdependencies. An example is depicted in Figure 4.9 for the case study model where $n = 2$. The topology file can in turn be the result of some automatic procedure that consumes an higher level representation of the system. This is the case for the enhanced SG model that will be presented in Chapter 6. When Möbius saves an atomic model three files are generated: a couple .h and .cpp, that are compiled to obtain object file, and .xml file. Thus, when the modeler saves the template atomic model a template XML file is generate. This is step 3. The topology and template atomic model XML files are the input of $\mathcal{D}^{perl}$, as shown in Figure 4.8. Step 2, using the results of step 1, implements Δ_i^h, for each i, h, with a different C++ constant array of short, and the other system parameters with C++ data structures, like arrays, records and plain types. These C++ data structures, statically defined at compilation time, are automatically generated and included in the user defined library supported by the Möbius tool, and then they can be used in each model defined in the tool.

Steps 4, 5 and 6 implement the function $\mathcal{D}$, as in (4.36). Each model generated at steps 4 and 6 is defined in an *XML* file, automatically generated

```
<SANDAREP>
  <name>SAN</name>
  <darepcomposedname>Comp</darepcomposedname>
  <darepnodename>SANSANDARep</darepnodename>
  <replicasNumber>2</replicasNumber>
  <DRSVsNumber>4</DRSVsNumber>
  <topology>
    <drsvs>
      <drsv>
        <name>Demand_A</name>
        <type>Place</type>
      </drsv>
      <drsv>
        <name>Demand_B</name>
        <type>Place</type>
      </drsv>
      <drsv>
        <name>Up_A</name>
        <type>Place</type>
      </drsv>
      <drsv>
        <name>Up_B</name>
        <type>Place</type>
      </drsv>
    </drsvs>
    <deps>
      <ind>0</ind>
      <dep>
        <pos>0</pos>
        <ind>0</ind>
      </dep>
      <dep>
        <pos>1</pos>
        <ind>1</ind>
      </dep>
      <ind>1</ind>
      <dep>
        <pos>0</pos>
        <ind>1</ind>
      </dep>
      <dep>
        <pos>1</pos>
        <ind>0</ind>
      </dep>
    </deps>
  </topology>
<SANDAREP>
```

Figure 4.9: Example of Topology XML file.

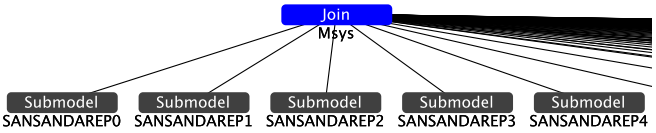


Figure 4.10: *DARep* approach: composed *Join* model generated from the template M^{darep} for the case study described in Section 4.3. Notice that the figure represents only a portion of the composed model and has being generated by the *DARep* script.

with *XML::LibXML*, a Perl Binding for libxml2, using the dependency topology described with an *XML* input file defined at step 1. In particular, the XML file of M_i is almost identical to the template XML file, the difference relies on the fact that *Index* and *Deps* are replaced by the corresponding (set of) indices.

Each time the template SAN model undergoes updates at steps 1 or 3, implying changes either in the dependency topology or in the number of replicas n, steps 4, 5 and 6 must be repeated, to update the resulting *XML* files and C++ files. Consequently the overall model must be compiled again.

Figure 4.10 depicts the left part (the overall picture is omitted for the sake of space) of the composed model M^{sys} automatically generated at step 6, as in (4.42), for the case study described in Section 4.6, for $n = 100$ and $\delta = 74$. The SAN models $SANSANDARep0, \ldots, SANSANDARep99$ (their names are obtained merging the name *SAN* of the template, the string *SANDARep* and the index of the replica) are the models automatically generated at step 4, corresponding to the replicas M_i, $i = 1, \ldots, n-1$, as in (4.36) and (4.37).

At step 4, a place is automatically generated in each model M_i for each entry of the array Δ_i^h associated to the dependency-aware SV W^h. The name of each place $W^h DRSVDARep_j$ is obtained merging the name W^h, the string *DRSVDARep* and the index j of the replica W_j^h that the place represents. Thus, these places model only the replicas of each dependency-aware SV that are accessed by each replica of the template model, as defined in Δ^h. Moreover, these places are shared among the replicas M_i by the operator *Join*, as defined in the list $\mathcal{J}^{shared}$ automatically generated at step 5.

Figure 4.11 shows the SAN model *SANSANDARep0* generated for the 0-th replica M_0. The bottom of the figure is omitted for the sake of space. The places *Demand_ADRSVDARep0, ..., Demand_ADRSVDARep74* and *Up_ADRSVDARep0, ..., Up_ADRSVDARep74* represent respectively the 75 replicas of the SVs *Demand_A* and *Up_A* that are accessed by the model

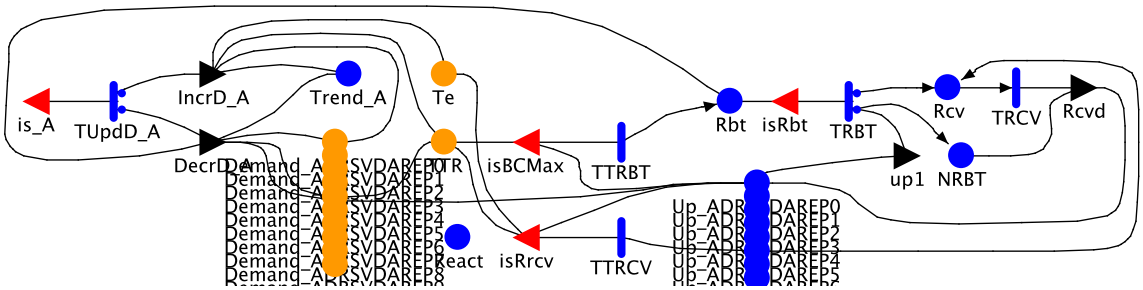


Figure 4.11: *DARep* approach: SAN model *SANSANDAREP0* generated from the template M^{darep} defining the generic component for the case study described in Section 4.3 with $n = 100$ and $\delta = 74$. Notice that the figure represents only a portion of the composed model and has being generated by the *DARep* script.

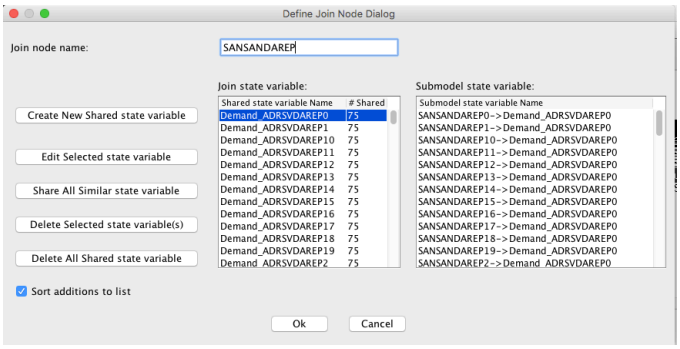


Figure 4.12: *DARep* approach: a snapshot of the "Define Node Join Dialog" of the Möbius tool for the composed model *Msys* generated from the template M^{darep} for the case study described in Section 4.3 with $n = 100$ and $\delta = 74$. Notice that the figure has been generated by the *DARep* script.

SANSANDARep0.

Figure 4.12 is a snapshot of the "Define Node Join Dialog" of the Möbius tool for the composed model *Msys*, depicted in Figure 4.10. It shows part of the list $\mathcal{J}^{shared}$ automatically generated for the SV *Demand_A* at step 5, as in (4.41), where the names of each replica are those defined at step 4.

The template model M^{darep} defined at step 3 is an atomic SAN model, where either plain places or extended places are used to represent dependency-aware SVs. In particular, struct-type extended place or array-type extended place can be both used to model different dependency-aware SVs.

The functions *Index()* and *Deps()* are implemented by two C++ func-

tions, that can be only used in the SAN template model, as follows:

$$SANSANDARep::M^{darep}::Index(),$$
$$SANSANDARep::M^{darep}::W^h->Deps(j). \quad (4.43)$$

where the names *SANDARep* and M^{darep} are C++ namespaces introduced to avoid names conflicting with Möbius C++ code. In particular, a different namespace M^{darep} is defined in the namespace *SANDARep* for each template SAN M^{darep}.

The C++ statement $SANSANDARep::M^{darep}::Index()$ is used in the template model M^{darep} to get the index of the replica. It is replaced, in each atomic SAN model generated at step 4, by the actual index of the modeled replica.

The C++ statement $SANSANDARep::M^{darep}::W^h->Deps(s)$ refers to the place Δ^h(s), as defined in (4.38). Without the index s, the statement can be used to pass to each user defined C++ function the list of references to all the places of Δ^h for the current replica of M^{darep}, such that the values of these places can be accessed. In each atomic SAN model M_i generated at step 4, the statement $SANSANDARep::M^{darep}::W^h->Deps(s)$ is replaced by

$$SANDARep::M^{darep}::rep(s).W^h(), \quad (4.44)$$

where a C++ object *rep* calls the actual method that returns the reference to the place $W^h DRSVDARep_j | j = \Delta_i^h(s)$, automatically defined in the generated SAN M_i, that models Δ_i^h(s). Moreover, in each SAN primitive where this statement is used (e.g., in the enabling condition of an input gate), all the actual names of the automatically generated places modeling all the SV of the array Δ_i^h are included in the primitive through a call to a dummy empty function having all these names as arguments. This is needed because the dependencies among SAN primitives (like e.g, the activities) and places in the Möbius tool are statically defined when the C++ code describing the model is generated, based on the names of the places. Thus, for example, an enabling condition defined by a statement that accesses a place by reference, like the above statement generated by the *DARep* approach, is checked at each update of the place only if the enabling condition includes also the name of the place.

The C++ code (definition of classes and initialization of objects and constants) used to implement the method called by *rep* in (4.44) and that relies on the dependency topology, is automatically generated and included in each submodel at step 4. In particular, the code to initialize the object *rep* with the array of the pointers to places modeling all the SV of the array

Δ_i^h, is included in the field "Custom Initialization" of each *SAN*; thus, the C++ data structures are set before the model evaluation starts.

Compared to *SSRep*, a reduction in the time overhead is expected for *DARep*, both during the initialization of the simulation solver and during the execution of the simulation batches. Compared to *CSRep*, a reduction in the time overhead is expected for *DARep*, mainly during the execution of the simulation batches.

However, *DARep* introduces a time overhead at generation time of the atomic and composed models (steps 4, 5 and 6) and at compilation time, due to the number n of the atomic models and to the size of the composed model. In particular, for very large values of n and δ_i, the time required for the generation and compilation of the composed model could have a relevant impact on the efficiency of the model evaluation.

4.6 Model a load-sharing systems

In this section the example presented in Section 4.3 is modeled in Möbius exploiting all the three approaches described so far.

The process representing the system proposed in this case study is a non-Markovian stochastic process, because of actions with deterministic time.

To perform comparison among the considered replication approaches, the system described above has been evaluated in terms of the following dependability measures:

- steady-state expected unsatisfied demand on node i for the server providing service S, i.e.,

$$\mathrm{UD}_{S,i} = \mathbb{E}[\max\{D_{S,i}(t \to \infty) - C_{S,i}, 0\}], \tag{4.45}$$

- steady-state expected unsatisfied demand on node i, i.e.,

$$\mathrm{UD}_i = \mathrm{UD}_{A,i} + \mathrm{UD}_{B,i}, \tag{4.46}$$

- steady-state expected global unsatisfied demand, i.e.

$$\mathrm{UD} = \sum_i \mathrm{UD}_i. \tag{4.47}$$

Since the goal is to compare the performance of the three replication approaches, the obtained dependability results for these measures are out of the scope of this thesis, and are not shown. However, as analysis target for the system under evaluation, the definition of the models described in the following is based on these dependability measures.

4.6.1 SAN models

For each approach *SSRep*, *CSRep* and *DARep*, the SAN model representing the generic component is composed by two parts, one for the service A and one for B. For the sake of brevity, only the part modeling the service A is described. The part modeling the service B is obtained duplicating all the primitives ending with "_A" (and the related arcs) and replacing the occurrence of "_A" with "_B" at the end of each string.

As shown in Figure 4.5, 4.7 and 4.11, the template SAN models (excluding the SAN models used for the channel in the *CSRep* approach) defined for all the considered replication approaches have a similar structure. They mainly differ in the definition of the index, in the definition of the dependency-aware SVs and in the channel used for the *CSRep* approach. In particular, the demand $S^{\text{load}}i$ and the state (working or down) associated to the generic node i are modeled in the template SAN by the following dependency-aware SVs:

- the extended places *Demand_A* (n-sized arrays of double) and *Up* (n-sized arrays of short), that are shared among all the replicas, for the *SSRep* approach,
- the extended places *Demand_Aj* (δ^h-sized arrays of double) and *Up_Aj* (δ^h-sized arrays of short), that are local to all the replicas, for the *CSRep* approach,
- the double-type extended place *Demand_A* and the place *Up_A*, that replace W^h in (4.43), for the *DARep* approach.

The double-type local extended place *TTR* models the value $T_{S,i}^{threshold} - T_{S,i}^{overc}(t)$, used to set the deterministic time to reboot, represented by the activity *TTRBT*, when the demand overcomes a pre-set threshold. The double-type local extended place *Te* models the instant of time when the demand overcomes a pre-set threshold, used to evaluate the new value for *TTR* when the demand returns below the threshold.

The activity *TUpdD_A* and the associated cases model respectively the random time to update the demand $S^{\text{load}}i$ and the probability $\mathbb{P}_{\text{incr}}$ (upper case 0), depending on the value of the local place *Trend_A* (if it is 1 the demand trend is increasing). The output gate *DecrD_A* updates the demand each time it decreases and also updates the value of *TTR* to *TTR–>Mark()-BaseModelClass::LastActionTime+Te–>Mark()* (the current value of *TTR* minus the current time plus the time when *TTRBT* has been enabled), if the demand returned below the threshold for both services. The activity *TTRCV*, enabled when the service A on the node is working, models the

random time to failure of the node. The rate of this activity is marking depending, being a function of the demand $S^{\text{load}}i$. At completion of $TTRCV$, when a failure occurs, 1 token is added to the place *Rcv* to enable the activity $TRCV$ that models the duration of the recovery. The activity $TRBT$ models the duration of the reboot. At completion of $TRBT$ the reboot ends successfully with probability c associated to the lower case 2, when one token is added to the short-type place $NRBT$ counting the number of successfully reboots, otherwise the reboot is considered failed, and 1 token is added to the place *Rcv* to enable the recovery action. The recovery is also enabled, selecting case 1, when at completion of $TRBT$ the value of $NRBT$ is greater than n^{rcv}. The place *React* is used to reactivate the activity $TTRCV$ each time the demand $S^{\text{load}}i$ is updated. Finally, with the *CSRep* approach, the places *isTransDem_A*, *isDem_A*, *isUp_A* and *TransDem_A*, shared among the SAN models shown in Figure 4.6b and 4.7, are used to set the information that have to be transferred with the channel (for example, *isUp_A* is equal to 1, if the place *Up_A* has been updated).

4.6.2 Simulation-based model

To understand strengths and weaknesses of the considered approaches, a comparison of the performance results of the Möbius implementations of *SSRep*, *CSRep* and *DARep* has been conducted, considering the case study described in Section 4.3. This analysis allows to derive useful observations on which is the best solution to employ, depending on the characteristics of the system under analysis in terms of size and dependency degree, and on accuracy requested to the analysis output. To accomplish such comparison, the terminating Möbius simulator [5] has been used to evaluate the dependability related measures defined by (4.45), (4.46) and (4.47).

The validation of the models has been carried out as follows:

- for some sample (small) models, the Möbius Transformer has been employed to verify that the correct number of states, and the correct set of transitions among them, was actually produced. It is in fact possible, for small models, to predict theoretically these information,

- for some sample (small) models, the measures have been evaluate with both simulation and Möbius analytical solvers, checking that in all the considered cases the values where the same,

- having the possibility to control the pseudo-random number generation (managing the seed), it has been verified that, for all the cases

irrespectively to the dimension of the model, simulations of the models produced by the three approaches result in:

- the same values of the considered measures,
- the same events, checked analyzing simulation traces.

Different reward structures [73, 90] over different set of markings have a different impact on simulation times. Thus, to improve accuracy and fairness of the comparison, a high number of reward variables (around 40) has been considered in the study. However, since here the analysis focuses on the comparison of the performance of the approaches, details on these measures and the obtained results are out of the scope of this thesis.

Each execution of the terminating Möbius simulator [5] is defined for a specific setting of all the parameters of the considered models (corresponding to an experiment in the Möbius terminology). The terminating simulator has been selected because the measures of interest are defined or at instant of time or on an interval of time. Here the interest is not on measures defined at the steady-state. Each execution of the terminating simulator starts initializing the data structures, then runs k batches (replications in Möbius terminology) with $k \geq 1$.

For the comparison, the following performance measures have been considered, relative to one execution of the Möbius simulator that runs k batches, with $k \geq 1$:

- $\tau(k)$: Total amount of *CPU* time, in seconds, used by the Möbius simulator. It includes both the initialization time and the time necessary to run k batches;

- τ_{init} or $\tau(0)$: The amount of *CPU* time, in seconds, used by the Möbius simulator to initialize its data structures. This is the *CPU* time used by the simulator to output the string "SIMULATOR::Preparing to run()". The definition of τ_{init} as a function of $\tau(k)$ is: $\tau_{init} = \tau(1) - (\tau(2) - \tau(1)) = 2\tau(1) - \tau(2)$, where $\tau(2) - \tau(1)$ is the total amount of *CPU*, in seconds, used by the Möbius simulator to run a batch;

- $\Delta\tau(k)$: Difference between the total CPU time to run k batches and the initialization time:

$$\Delta\tau(k) = \tau(k) - \tau_{init}.$$

- ${}^{D}_{S}\mathcal{C}\tau(k)$: compared simulation performance (pure number) between

DARep and *SSRep*, defined as follows:

$$ {}^{D}_{S}\mathcal{C}\tau(k) = \frac{\tau^{DARep}(k)}{\tau^{SSRep}(k)}, $$

where $k > 0$ and the superscripts "*SSRep*" and "*DARep*" refer to *State-Sharing Replication* and *Dependency-Aware Replication* approaches, respectively. This is a useful indicator to immediately perceive the ratio between the performance of the two approaches;

- ${}^{C}_{S}\mathcal{C}\tau(k)$: Similarly to the previous measure, this is the compared simulation performance (pure number) between *CSRep* and *SSRep*, defined as follows:

$$ {}^{C}_{S}\mathcal{C}\tau(k) = \frac{\tau^{CSRep}(k)}{\tau^{SSRep}(k)}, $$

where $k > 0$ and the superscripts "*SSRep*" and "*CSRep*" refer to *State-Sharing Replication* and *Channel-Sharing Replication* approaches, respectively;

- ${}^{D}_{C}\mathcal{C}\tau(k)$: Similarly to the previous two measures, this is the compared simulation performance (pure number) between *DARep* and *CSRep*, defined as follows:

$$ {}^{D}_{C}\mathcal{C}\tau(k) = \frac{\tau^{DARep}(k)}{\tau^{CSRep}(k)}, $$

where $k > 0$ and the superscripts "*DARep*" and "*CSRep*" refer to *Dependency-Aware Replication* and *Channel-Sharing Replication* approaches, respectively.

The considered *CPU* time includes both user and system *CPU* times.

The following topology of read/write access is chosen for the case study. The first three quarts of the virtual machines host a server for service A, i.e., $i = 0, \ldots, \lceil 0.75 \cdot n \rceil - 1$, the last three quarts host a server for service B, i.e., $i = \lceil 0.25 \cdot n \rceil - 1, \ldots, n - 1$ (note that virtual machines in the intersection host servers for both service A and service B). The degree of interaction is the same for all i, i.e., $\delta_i^A = \delta_i^B = \delta$. The demand re-dispatching follows a cyclic graph, i.e., $\Delta_i^S = \{i, i + 1 \bmod 0.75 \cdot n, \ldots, i + \delta \bmod 0.75 \cdot n\}$. Notice that $\mathcal{T}^A$ and $\mathcal{T}^B$ have the same structure but involve different indices.

To exercise the approaches in a variety of relevant contexts, scenarios generated as combinations of the following values for n, δ and k, have been considered:

- number n of replicas ranging from 10 to 1000,

- dependency degree δ varying from 1 (minimum connectivity) to 148,
- number of batches k varying from 1 to 1000. The value of k impacts on the precision of the obtained results and 1000 has been selected to assure convergence satisfying the chosen requirement (relative confidence interval of width less than 10^{-5}).

Simulations were sequentially performed on Intel(R) Core(TM) i7-5960X with fixed 3.50 GHz CPU, 20M cache and 32GB RAM, and an up to date GNU/Linux Operating System. Each $\tau(k)$ has been evaluated 10 times and the arithmetic mean is taken as final result. It is important to notice that, for the *SSRep* and *CSRep* approaches, models compilation times are negligible, being constituted by a few single atomic SANs and one composed model, while for the *DARep* approach component models compilation time can be relevant. In particular, if $n \leq 100$ then atomic SANs compilation times and composed model compilation time can take few minutes, while, for $n = 1000$ and $\delta = 148$, composed model compilation time can grow up to about 10 hours. However, it is important to notice that recompilation is required *only* if the structure of the template model or the topology of interdependencies $\mathcal{T}$ are changed. Thus, changing all the other model parameters, such as failure rates or service request rates, as well as changing the measures under evaluation, do not impact on compilation time. In addition, investigations are currently in progress to understand how to promote parallel models compilation to improve on compilation time.

Another relevant issue for *DARep* is RAM usage. Depending on the number of SANs in the model, the amount of memory occupation can vary from few Mbs to about 10 Gbs for $n = 1000$.

Table 4.1: τ_{init} for the *SSRep* approach.

		δ			
		1	7	74	148
n	10^1	0.019	0.013		
	10^2	3.56	3.81	3.62	
	10^3	29600	29700	29600	29700

First, the results relative to the τ_{init} indicator for the three approaches are presented in Table 4.1, Tables 4.2 and 4.3, and depicted in Figures 4.14 to 4.16, respectively. From the inspection of Table 4.1 two important conclusions can be drawn: for *SSRep*, the initialization phase time is almost independent from the dependency degree δ, but strongly correlated with the system size n. This confirms the already predicted behavior, as discussed

Table 4.2: τ_{init} for the *CSRep* approach.

		δ			
		1	7	74	148
n	10^1	0.015	0.018		
	10^2	0.118	0.142	0.438	
	10^3	4.789	5.592	12.251	22.452

Table 4.3: τ_{init} for the *DARep* approach.

		δ			
		1	7	74	148
n	10^1	0.015	0.012		
	10^2	0.021	0.033	1.746	
	10^3	0.309	0.746	23.414	137.610

when presenting this approach in Section 4.4.3. Instead, Tables 4.2 and 4.3 show that both *CSRep* and *DARep* initialization times increase of about one order of magnitude when n increases of one order of magnitude, and differently correlated with δ, with τ_{init} of *DARep* growing faster than *CSRep* at increasing δ. Again, these results are in line with what already predicted in Sections 4.4.4 and 4.4.5, by observations regarding the behavior of these approaches. It is important to notice that the presence of two read/write access topologies, $\mathcal{T}^A$ and $\mathcal{T}^B$, drastically impacts on the *SSRep* initialization phase performance. Also, there are insights that the numbers presented in Table 4.1 suffer from implementation issues concerning the model construction operated by Möbius (see Section 4.7). The initialization time for *CSRep* for $n = 1000$ and $\delta = 148$ has not been reported because the total execution time overcome an upper limit. A graphical comparison is depicted in Figure 4.13. The results obtained for the next performance indicator under

Table 4.4: $\Delta\tau(1000)$ for the *SSRep* approach.

		δ			
		1	7	74	148
n	10^1	0.412	0.437		
	10^2	48.873	48.476	50.306	
	10^3	5382.400	5216.000	5373.300	5455.600

analysis, $\Delta\tau(k)$, are presented for each approach in Table 4.4, Table 4.5 and Table 4.6, respectively. In detail, Table 4.4 shows $\Delta\tau(1000)$ for *SSRep*. The evaluation of batches during the simulation, as predicted in Section 4.4.3,

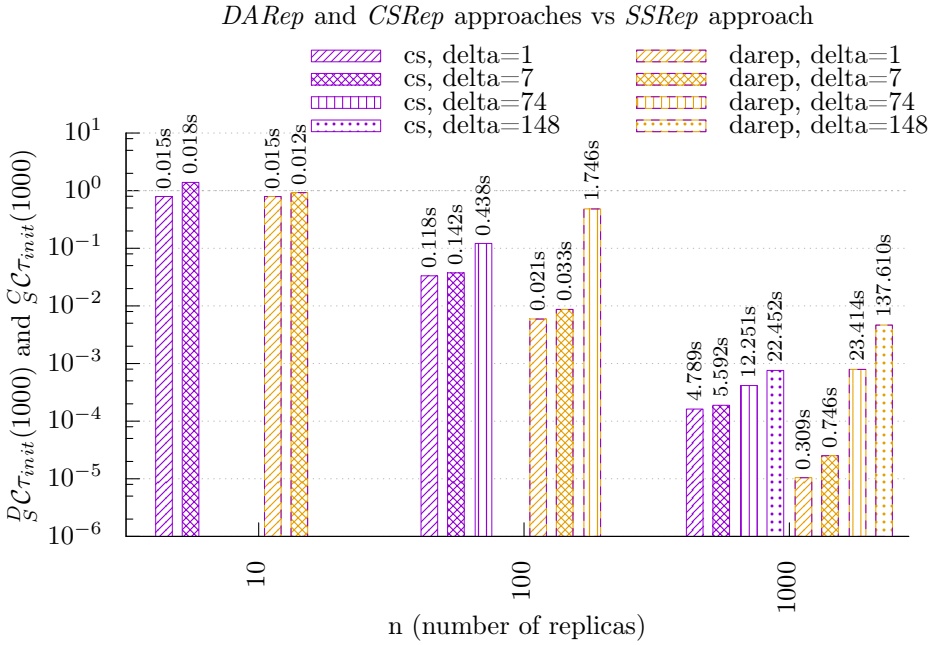


Figure 4.13: CPU time for the initialization phase of *DARep* and *CSRep* relative to *SSRep* where δ varies. Bars on the left of the n value refer to *CSRep*, on the right to *DARep*.

Table 4.5: $\Delta\tau(1000)$ for the *CSRep* approach.

		δ			
		1	7	74	148
n	10^1	0.838	2.580		
	10^2	74.272	272.091	2455.352	
	10^3	9273.481	24185.708	216642.749	*NaN*

Table 4.6: $\Delta\tau(1000)$ for the *DARep* approach.

		δ			
		1	7	74	148
n	10^1	0.164	0.204		
	10^2	6.786	8.067	17.513	
	10^3	1185.831	1292.934	1536.186	2482.280

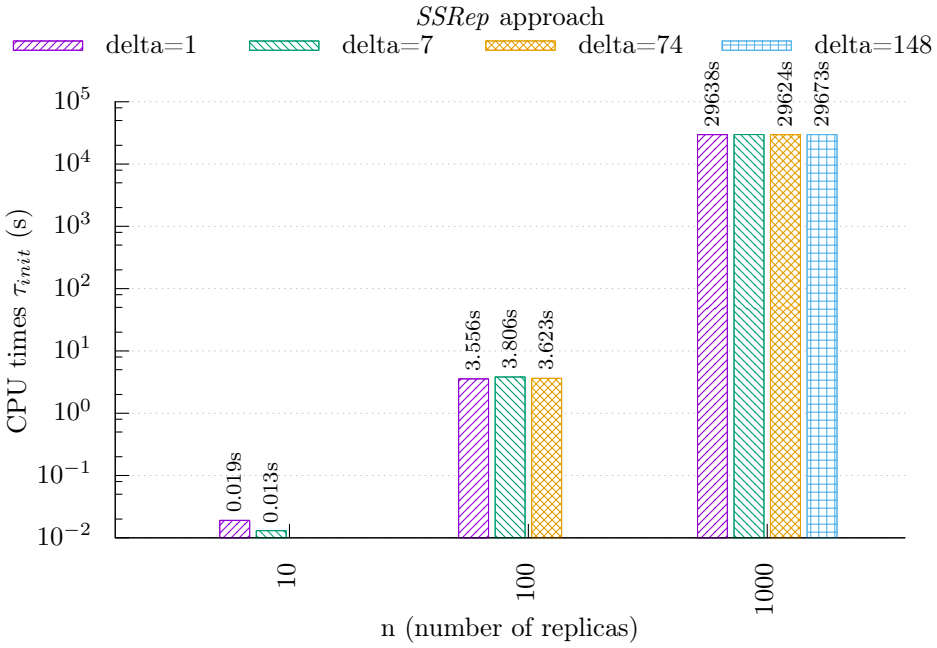


Figure 4.14: initialization CPU time for *SSRep* at increasing of δ.

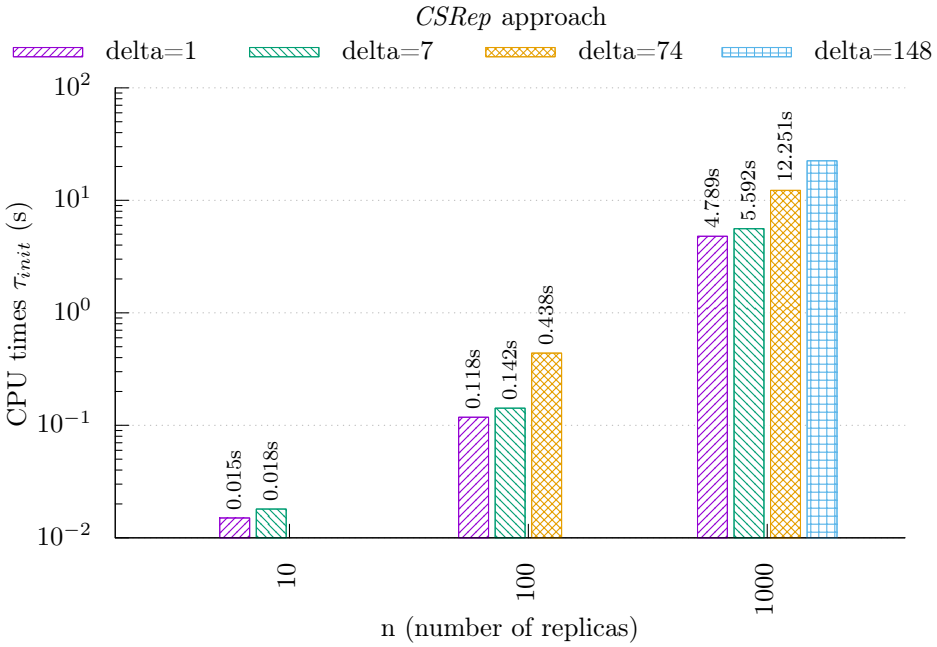


Figure 4.15: initialization CPU time for *CSRep* at increasing of δ.

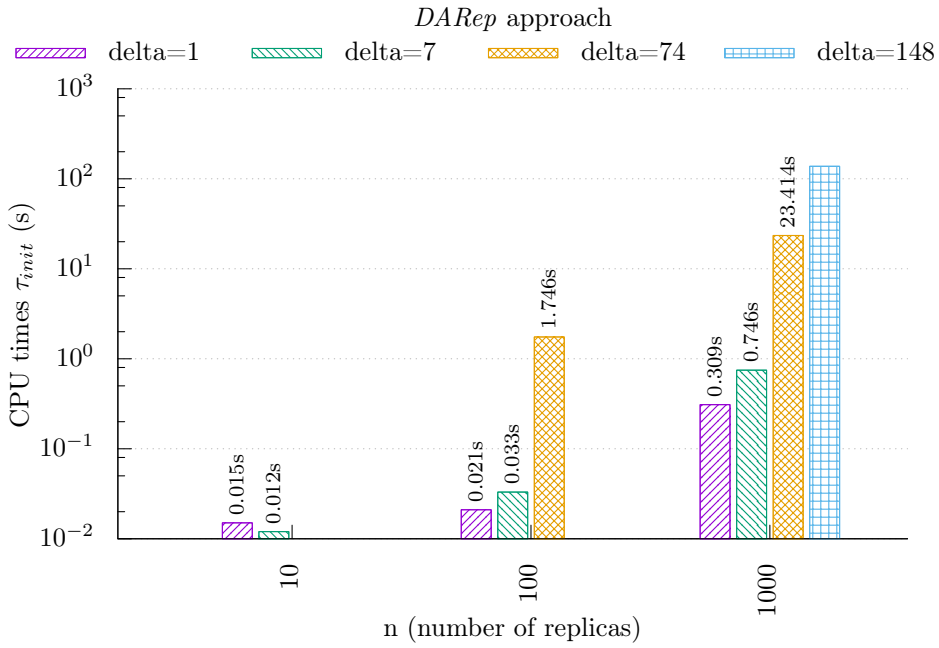


Figure 4.16: initialization CPU time for *DARep* at increasing of δ.

has a time complexity of $\mathcal{O}(n^2)$ and it is almost independent from δ.

For the two approaches *CSRep* and *DARep*, the results are shown in Table 4.5 and Table 4.6, respectively. Not surprisingly, since they have been defined with the purpose to exploit the real dependency topology among system components, their batches execution times depend on both n and δ. In particular, it can be observed how *CSRep* batches time explodes at increasing of δ. In fact, $\Delta\tau(1000)$ for $n = 1000$ and $\delta = 148$ has not been reported because the total execution time exceeds a fixed upper limit.

Comparing *SSRep* and *CSRep* values, an interesting phenomenon can be highlighted, in particular considering the trends visible in Figures 4.17 to 4.19. Summing up τ_{init} and $\Delta\tau(1000)$ for the case $n = 1000$, if $\delta \leq 7$ then $\tau_{init} + \Delta\tau(1000)$ for *SSRep* is greater than $\tau_{init} + \Delta\tau(1000)$ for *CSRep*, whereas for higher values of δ it is clear that $\tau_{init} + \Delta\tau(1000)$ for *CSRep* is greater than $\tau_{init} + \Delta\tau(1000)$ for *SSRep*. Thus, from the point of view of the modeler, if the target is to optimize the total simulation time, it is the value of δ that determines which approach is more convenient to adopt between *SSRep* and *CSRep*. If system components are loosely interconnected, i.e., δ is small, then *CSRep* performs better, otherwise *SSRep* overcomes *CSRep*.

Of course, also *DARep* fully exploits the topologies of read/write accesses, thus outperforming *SSRep*, and, having no overhead due to the special SVs

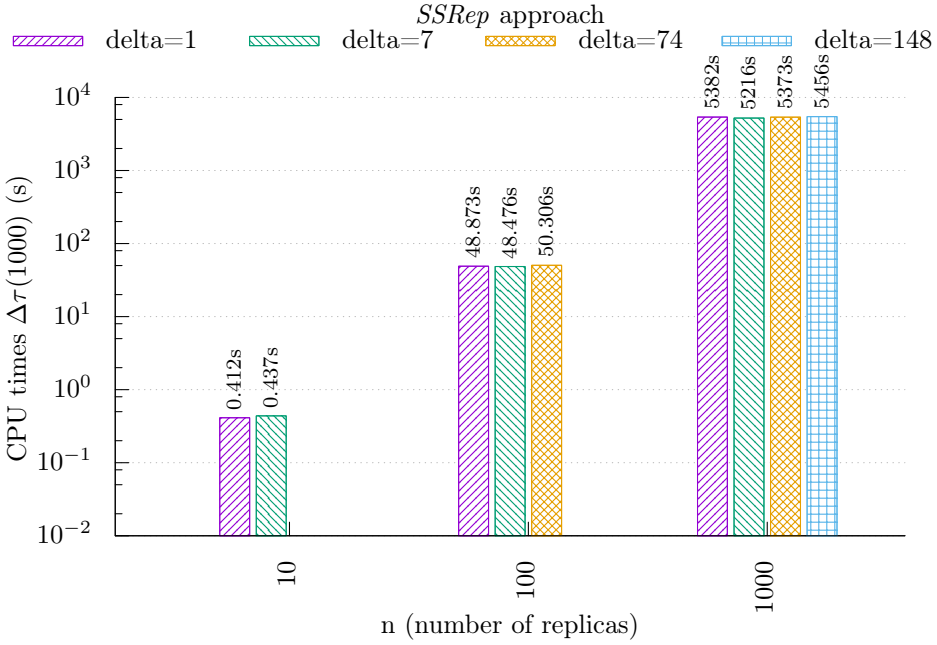


Figure 4.17: batch time for *SSRep* at increasing of δ.

management (the channel), is always better than *CSRep*. Thus, without considering compilation times, *DARep* is the best choice for complex systems whatever the value of δ is.

Table 4.7: ${}^{D}_{S}\mathcal{C}\tau(k)$ between the *DARep* and *SSRep* when $\delta = 10$.

		k				
		1	10	100	1000	10000
n	10^2	0.0168	0.0378	0.104	0.156	0.195
	10^3	0.0001	0.0005	0.004	0.042	0.223

The final part of the comparison is carried out in terms of the $\mathcal{C}\tau(k)$ indicator, for different values of k. Varying the number of batches k has an impact on the accuracy of the simulation results, so it is a parameter to be cautiously selected, in accordance with the criticality of the system under analysis and of the purpose of the analysis itself. Tables 4.7, 4.8 and 4.9 show the results of ${}^{D}_{S}\mathcal{C}\tau(k)$, ${}^{C}_{S}\mathcal{C}\tau(k)$ and ${}^{D}_{C}\mathcal{C}\tau(k)$, respectively, keeping δ fixed at 10. From Table 4.7, it can be immediately concluded that the *DARep* approach is always better than *SSRep*. From Table 4.8, a similar trend already observed between *CSRep* and *SSRep* at varying δ is shown also at varying k. In fact, at increasing k, *CSRep* initially outperforms *SSRep* (also depending on the

Table 4.8: ${}^{C}_{S}\mathcal{C}\tau(k)$ between the *CSRep* and *SSRep* when $\delta = 10$.

		k				
		1	10	100	1000	10000
n	10^2	0.139	0.975	4.141	7.195	7.905
	10^3	0.002	0.016	0.118	1.052	6.696

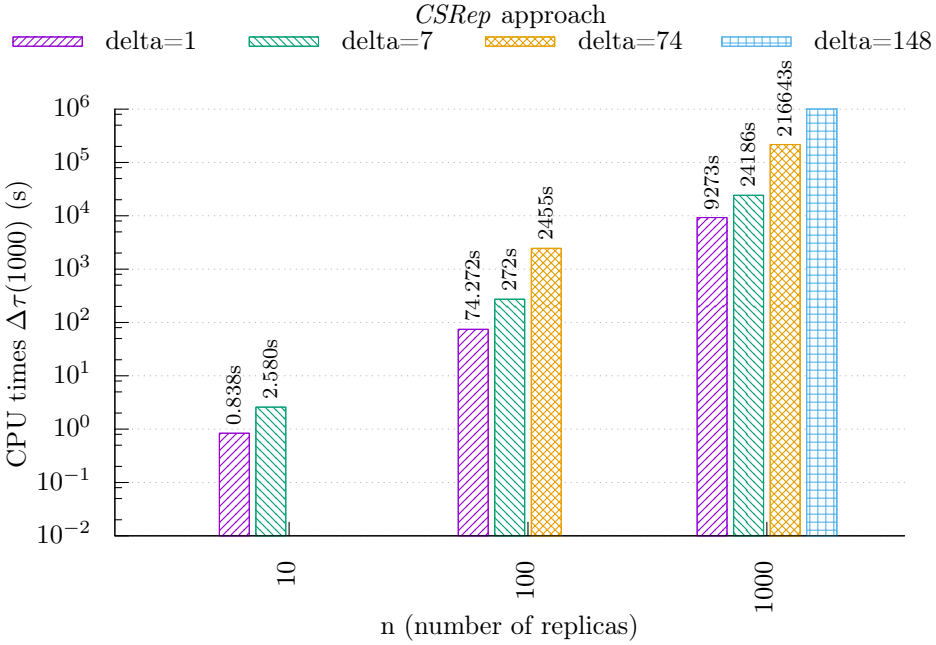


Figure 4.18: batch time for *CSRep* at increasing of δ.

Table 4.9: ${}^{D}_{C}\mathcal{C}\tau(k)$ between the *DARep* and *CSRep* when $\delta = 10$.

		k				
		1	10	100	1000	10000
n	10^2	0.121	0.039	0.025	0.022	0.025
	10^3	0.043	0.030	0.038	0.040	0.033

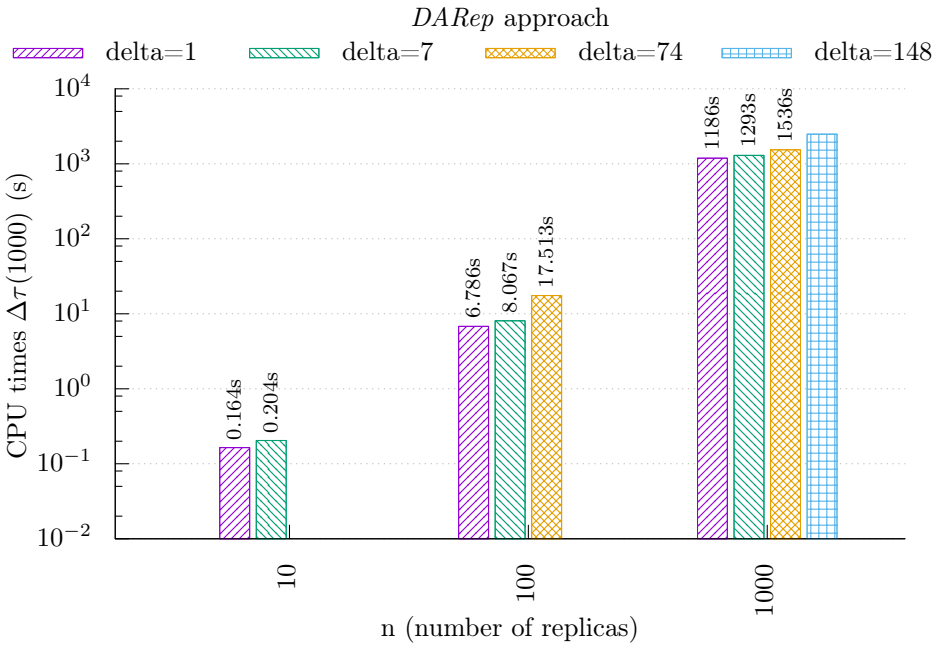


Figure 4.19: batch time for *DARep* at increasing of δ.

value of n), but then it is the reverse. Actually, the assumed value for the dependency degree ($\delta = 10$) in this evaluation is already rather high for having *CSRep* really competitive with respect to *SSRep*. Last, Table 4.9 shows values of *DARep* always much better than those of *CSRep* (between 30 and 40 times better for the highest values of k and n).

The performance of all the three approaches is affected by the size of the system and by the number of simulation batches: not surprisingly, the obtained values degrade at increasing of both n and k. With respect to the other considered parameter, namely the topology of interaction that is a key aspect of complex systems, the behavior among the three replication solutions is diversified. The *SSRep* approach is insensitive to δ, which results in high values of the initialization time τ_{init}: it grows higher than 8 hours when n is 1000 and k is 1000. *CSRep* exploits not only the topology of interactions but also the fine grained topology of read/write accesses with the channel mechanism. The benefits on the initialization time τ_{init} are great (just 22 seconds for the highest considered values for δ and n), but correspondingly the overhead required to manage the channel grows significantly (up to around 60 hours when δ is 74 and n is 1000). Therefore, *CSRep* overcomes *SSRep* when the dependency degree is low (which is however not uncommon in relevant application sectors, such as in power grid systems, as already discussed

in [79]). Applying a totally different idea as done by the *DARep* approach, which relies on an external *Perl* program in its implementation in Möbius, the performance is greatly improved: *DARep* is always the best in all the Tables showing the evaluation results, whichever be the values of δ, n and k. However, it might suffer from long compilation times, although it has to be reminded that a recompilation is needed only when there are changes in the structure of the template model and/or in the interdependency topology.

4.7 New proposals for Möbius

Considering the outcome of the comparison effort, *DARep* is the approach that has been selected for implementing the enhanced SG model described in Chapter 6. In parallel to the SG model design, additional investigations have been carried out in order to understand why the implementation of the *SSRep* approach has a time complexity of $\mathcal{O}(n^4)$ during the initialization phase, and how to reduce it to $\mathcal{O}(n^2)$ value. For the *SSRep* approach, $\mathcal{O}(n^2)$ is the best value reachable, assuming a complete graph of interdependencies. Thus, Section 4.7.1 describes in more details the current implementation of the anonymous *Rep* in Möbius and Section 4.7.2 presents a proposal for a different implementation, that has been already accepted by the PERFORM group [91] and will appear in a forthcoming release of Möbius. Numerical experiments, briefly presented in Section 4.7.3 confirm that the theoretical limit of $\mathcal{O}(n^2)$ can be reached. Results discussed in these sections have been published in [11].

4.7.1 Anonymous Rep in Möbius

Möbius [5] embraces the modularity principle adopting the object oriented programming paradigm to represent atomic models, composed models, reward structures (performance measures) and model solvers as classes. The class hierarchy describes behavioral aspects. For instance, different modeling formalisms are obtained specializing the base model class where SVs and actions are defined [92]. The architecture of the system model is reflected by the methods call graph. Composed model are designed defining a *compositional tree* [93] where nodes are either atomic of composed submodels. Atomic submodels can be only used to define the leves of the tree, *Join* and *Rep* composed submodel can be used to define all the other nodes of the tree, root included. Designing a model means defining a hierarchy of classes that describe atomic and composed models; constructing a model means instantiating an object whose class is the class of the root in the compositional tree.

This object is then passed to the selected solver.

In Möbius 2.5 (the current version), model classes are not aware of the compositional tree between the root and them. In particular, each node constructor does not receive information from its parent about already created SVs, thus multiple copies of the same shared SV are created.Calling $A.B$ the object B of class $\mathbb{B}$ inside the object A of class $\mathbb{A}$, the constructor method of *Join* performs the following steps:

$\mathcal{J}_{\text{Current}}$ 1) creates an object for each shared SV,

$\mathcal{J}_{\text{Current}}$ 2) creates an object for each of its children,

$\mathcal{J}_{\text{Current}}$ 3) shares its shared SVs with the corresponding SVs of its children objects,.

In this way the compositional tree is traversed recursively from root to leaves. The application of $\mathcal{J}_{\text{Current}}$ to the particular case of the *Rep* operator produces:

$\mathcal{R}_{\text{Current}}$ 1) creates an object for each shared SV,

$\mathcal{R}_{\text{Current}}$ 2) creates n objects (replicas) of the class specified by its children,

$\mathcal{R}_{\text{Current}}$ 3) shares its shared SVs with the corresponding SVs of its children objects.

For both *Join* and *Rep*, multiple copies of the same shared SV are created in 1) and 2). Then, in 3), all these copies have to be scanned and reduced to a unique object. A detailed description of the sharing mechanism is out of this thesis' scope. The complete characterization is in [93]; here it is sufficient to notice that it involves not only the shared SVs, but also the actions that are affected by, or can have impact on, the shared SVs. For the *Join* operator each sharing has a time complexity of $\mathcal{O}(n^2)$, where n is the number of replicas.

Models with a complex compositional tree, i.e., one or more *Rep* operators with many replicas, do not pose performance issues as long as only a few shared SVs are involved. However, when system components have complex interactions, as it increasingly occurs in modern systems, a large number of shared SVs is needed. The performance of the solver initialization phase quickly degrades if the number of shared SVs in the overall model is $\mathcal{O}(n)$ or higher, as discussed in Section 4.6.2.

4.7.2 Improving the anonymous Rep

The modeler follows an ordering when designing the model: first, the atomic models are defined independently from the others, then atomic models are composed to form the overall system model. So the design ordering is from leaves to root. Unfortunately, the objects construction in the current version of Möbius follows the same ordering and this can lead to long startup times.

The key idea is to revert the design ordering during construction, i.e., calling constructor methods from root to leaves: 1) introduce a data structure that keeps track of the shared SVs already created in the compositional tree; 2) redefine all the constructors, for both atomic and composed submodels, to search in the data structure for the presence of a SV before creating it. The data structure of choice is an unordered map $\mathcal{M}$ whose keys are SV names and values are SV object references.

In the proposal, labeled *New*, the empty map $\mathcal{M} = \{\}$ is created by the solver and passed to the constructor of the compositional tree root, that can be a *Join* or a *Rep*. Notice that *Rep* is a special case of *Join*, thus the new algorithm equally impacts on both *Join* and *Rep*.

Each node constructor, either atomic or composed model, performs the following steps:

New 1) receives $\mathcal{M}$,

New 2) checks if shared SVs exist within $\mathcal{M}$:

- if yes, then it copies the object references instead of creating new ones,
- else, it creates the new objects and inserts the object references into $\mathcal{M}$,

New 3) calls the constructor of children nodes,

New 4) if the node is a composed model, then removes distinguished SV references from $\mathcal{M}$.

Notice that *New* 4) is needed because two leaves of the compositional tree can have SVs with the same name that are local to different models. The sharing in our new algorithm is no more performed after node objects creation, but is part of the objects creation itself.

The time complexity of each sharing is bounded by the worst complexity among searching, inserting and removing couples from the map $\mathcal{M}$. In particular, the map $\mathcal{M}$ has been implemented using the unordered map of the C++ standard library [94], where searching, inserting and deleting are

preformed in $\mathcal{O}(1)$. Thus, compared with the $\mathcal{O}(n^2)$ shown by the *Current* algorithm, the *New* algorithm performs significantly better. Experiments in Section 4.7.3 quantitatively assess the improvements on a simple case study.

4.7.3 Comparison between anonymous Rep variants

Similarly to the logical structure of the system presented in Section 4.6, consider n working stations, called $worker_1$, ..., $worker_n$, dedicated to performing the same task in parallel. At every time instant, each station can be either working or failed and its failure rate is a function of the workload assigned to it. The failure of a station implies a reconfiguration of the workload assigned to the other stations to continue accomplishing the tasks of the failed station. Just before failing, a station redirects its tasks to one or more of the other stations it is connected with. Connections, and then neighbouring relations, follow a predefined oriented graph. The system model is obtained by non-anonymous replication, that means:

- a generic worker is modeled and replicated n times,
- when the system model is constructed, replicas are indistinguishable,
- a sophisticated mechanism promotes each replica to become self-aware of its index and then interact with other replicas according to the connections graph.

Detailed logical structure and the complete model, implemented using the SAN formalism, where SVs correspond to places in Petri Net dialects, hve been published in [79]. Here, in order to simplify the notation, the model compositional tree is described as

$$system = \mathcal{R}_n\big(worker, \mathbb{S}_1, \ldots, \mathbb{S}_n\big),$$

where $\mathbb{S}_j$ are shared SVs that can be read and written by $worker_i$ if the topology of interactions contains the edge (i, j). The model is designed to tackle each possible connection graph, thus all the $\mathbb{S}_j$ are shared among all the *worker* replicas. The compositional tree has 2 nodes: one atomic model class $\mathbb{N}_2$ for *worker*, and one composed model class $\mathbb{N}_1$ for the root. In the current version of Möbius the following steps are performed:

Current 1) for all $j = 1, \ldots, n$ creates $N_1.S_j$,

Current 2) for all $i = 1, \ldots, n$ creates $N_1.N_2^i$, that in turn, for all $j = 1, \ldots, n$, creates $N_1.N_2^i.S_j$,

Current 3) for all $i, j = 1, \ldots, n$ shares $N_1.S_j$ with $N_1.N_2^i.S_j$.

The time complexity of Current 1) and Current 2) is $\mathcal{O}(n^2)$, but the complexity of Current 3) is $\mathcal{O}(n^4)$ because there are n distinguished SVs, shared among all the n replicas (therefore, n^2 shares, each of complexity $\mathcal{O}(n^2)$).

Instead, the new algorithm performs the following steps:

New 1) receives $\mathcal{M} = \{\}$,

New 2) for all $j = 1, \ldots, n$ creates $N_1.S_j$ and inserts $N_1.S_j$ inside $\mathcal{M}$, i.e., $\mathcal{M} = \mathcal{M} \cup \{(S_j, N_1.S_j)\}$,

New 3) for all $i = 1, \ldots, n$ creates $N_1.N_2^i$, that in turn, for all $j = 1, \ldots, n$, searches S_j in $\mathcal{M}$ and copies $N_1.S_j$ into $N_1.N_2^i.S_j$,

New 4) removes nothing.

The time complexity of *New* 3) is $\mathcal{O}(n^2)$.

A simplified version of the load-sharing case study has been modeled to assess the Mean Time to Failure of $worker_i$ for $i = 1, \ldots, n$, through the Möbius simulator [95] (running 1000 batches). To compare the two algorithms, the CPU time they need for the initialization phase is computed, as reported in Table 4.10. To compare performance, the CPU time of initialization are shown in Table 4.10. All the experiments have been performed on a computer with an Intel i7-4710MQ CPU running at 2.50 GHz and with 16 GB of RAM clocked at 1333 MHz. Within the Möbius modeling environment, models are designed as a hierarchy of C++ classes. Thus the theoretical prediction of the complexity for the simulator initialization phase

Table 4.10: CPU times for initialization as n increases.

n	$t_{\text{init\_Current}}$ (s)	$t_{\text{init\_New}}$ (s)
100	0.22	$4.3 \cdot 10^{-2}$
200	1.78	$9.2 \cdot 10^{-2}$
300	6.99	0.20
400	18.90	0.36
500	41.69	0.59
600	82.56	0.91
700	151.07	1.38
800	254.02	1.99
900	414.83	2.69
1000	631.35	3.60

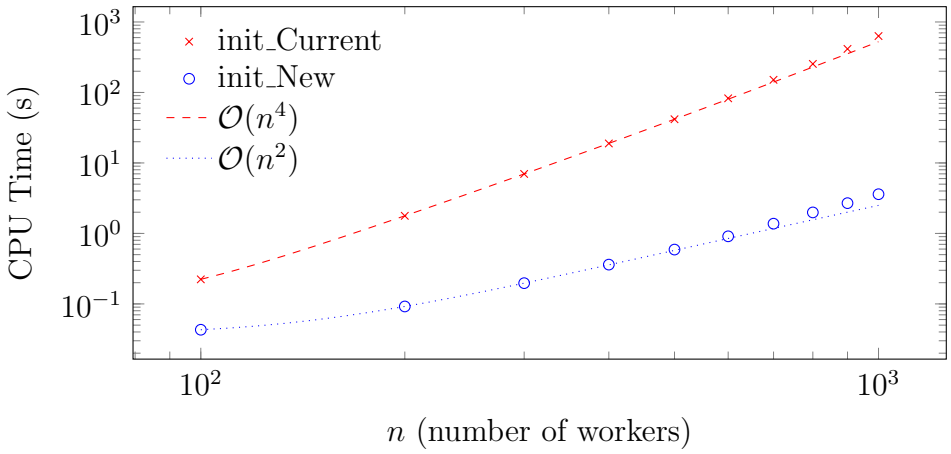


Figure 4.20: Comparison of time complexity for the solver initialization phase in the current version of Möbius and for our new algorithm as the number n of workers increases.

is $\mathcal{O}(n^4)$ for the current version of Möbius and $\mathcal{O}(n^2)$ for the new algorithm. The predicted time complexities are confirmed by experimental results, as reported in Table 4.10 and clearly visible in Figure 4.20. In addition, notice that the sharing in the current version of Möbius produces a complex set of data structures that have an impact on batches performance, whereas in the new algorithm no additional data structure is needed. Thus, also batches performance is improved, as depicted in Table 4.11.

Table 4.11: CPU times for 1000 batches as n increases.

n	$t_{\text{batches\_Current}}$ (s)	$t_{\text{batches\_New}}$ (s)
100	9.99	8.78
200	43.47	35.83
300	121.34	84.15
400	246.35	155.98
500	505.52	277.18
600	783.10	436.57
700	1142.66	735.40
800	1561.30	959.69
900	2183.85	1292.19
1000	2796.18	1676.77

4.8 Summary

In this chapter a detailed description of existing and new proposed non-anonymous replication mechanisms has been carried out, as well as a comparison among them in terms of efficiency related indicators. The contribution develops along two directions: 1) formal definition of new model composition approaches able to address the modeling and analysis of large and complex systems; 2) provide evidence of the capabilities of existing and new proposed model composition solutions, through a case study (see Section 4.3). Such case study allows on one side to illustrate the ideas and the dynamics of the proposals, and on the other side to select the most appropriate solution to employ in our enhanced SG framework, based on a comparative assessment of their performances.

In particular, in Section 4.2 the logical architecture of a class of systems general enough to represent a variety of large, interconnected systems has been presented. Since the analysis of *SSRep*, the state-of-the-art approach adopted in the development of the basic model (see Chapter 3) revealed several weaknesses when large models are addressed, two new approaches, *CSRep* and *DARep*, have been proposed, having in mind both the general applicability and the concrete replacement of *SSRep* with one of them in the context of the basic SG model. From the comparison analysis, *DARep* ranked as the most efficient approach and the less impacted by increasing level of dependencies among system components. Thus, for the enhanced SG model, described in the next chapter, the *DARep* approach has been selected.

Also, a new implementation of the Möbius *Rep* operator, still based on anonymous replication, has been developed and compared with the current one, resulting in improved performance during the initialization time (Section 4.7.3).

Chapter 5

State estimation

As discussed in Chapter 3, the state of a SG can be defined as the Cartesian product of the EI state, the working/failure state of its components and the state of the MCS. As already seen in Chapter 3, the ESTATE_SAN model is responsible for all the computations related to the EI state estimation, thus the results discussed in this chapter are at the basis of the enhanced version of ESTATE_SAN detailed in Section 6.2.1.

The EI is a complex physical reality, so the first goal of this chapter is to describe the assumptions considered to model its state and the strategy adopted to estimate it. In Section 5.2 a detailed description of the state of art is presented, pointing out assumptions and limitations of existing strategies. The contribution of this thesis in state estimation is twofold: first, a new formulation of an existing strategy is introduced; second, comparisons among a considerable number of strategies are presented. The final goal is to be able to decide which strategy, or combination of strategies, can be adopted to replace the one implemented within the basic model presented in Chapter 3. In particular, after presenting a discussion on the role the PFEs have in both the basic and the enhanced model, stressing the fact that a relation between powers and voltages can be established without explicit dependency on time, the detailed analysis of old and new ideas concerning PFEs solution methods is the main topic of this chapter. Both theory and implementation are considered, connecting questions arising from the EI state estimation to well-known issues addressed by numerical analysis and complex calculus. Here, the main problem is that, on one side, the PFEs are formulated in terms of complex numbers and have special analytical properties but, on the other side, computers offer at the level of hardware computation units the possibility to manipulate floating points numbers. Thus, the PFEs have to be recasted to a computer-friendly form and then solved. The wideness of the topic is based mainly on the fact that there are quite a number of ways

to do the recasting, each with pros and cons in terms of the corresponding implementation in software.

5.1 Power Flow Equations

For an AC electrical grid, powers, voltages, currents, etc, are time-varing quantities: they oscillate with a base frequency of 50 or 60 Hz and can be affected by several sources of disturbance. As discussed in Section 3.3.1, at every time instant t, the impact of the components directly linked to the i-th bus on its power, voltage, nodal current, etc, is represented by Equation (3.2); so, whenever one of those components produces a disturbance, the power, voltage, nodal current, etc, of the buses are affected too. This means that the EI state estimation at time t is performed as follows:

1. Assign, according to a predefined formula or reading from a data set, the values to all the variables on the right hand side of Equation (3.2),

2. Evaluate the left hand side of Equation (3.2) in order to obtain the independent variables that appear within the PFEs,

3. Solve the PFEs to obtain the dependent variables.

If the interest is in capturing the fine grained behavior of the EI, then the frequency at which the state estimation has to occur must be greater than 50Hz. The impact of disturbances on values that appear on the right hand side of Equation (3.2) to those of the dependent variables is addressed.

On the contrary, in the simulation context of this thesis the interest is on events that can happen at different time scales, e.g., 10^{-2} and 10^{2} seconds, and then the frequency at which the state estimation takes place cannot be as high as 50Hz, otherwise it would be unfeasible to perform the evaluation over an entire day. In particular, within the ESTATE_SAN (detailed in Section 3.3.1 for the basic model and in Section 6.2.1 for the enhanced model) the action *UpdateES* can fire due to several conditions related to what happened in the other SANs models, each focused on the management of a single variable among those that appear on the right hand side of Equation (3.2). The actions inside each SAN model can represent different phenomena, such as:

- a change in active power, e.g., a change of a DG P^{inj},

- a failure of some hardware components, such as the *AccidentalFailure* action in the MV_PL_SAN depicted in Figure 3.14,

- the application of control policies,

and many others. The key point here is that all the listed phenomena can happen at a different time scale. Being interested in modeling the interaction between ICT and EI, that takes place within seconds, the root mean square of powers, currents and voltages are then considered, thus abstracting away events happening with a frequency higher than 1 Hz.

Of central importance becomes then a special property of the PFEs: they express a relation between powers and voltages that, without explicitly referring to time instants, can enforce a phase shift among them.

The PFEs derive formally from the Ohm's law, the Kirchhoff's current law and the formula for the complex power (S) [96]. The common complex formulation of the resulting equation associated to a node i is given by:

$$S_i = E_i \overline{I_i^{\text{bus}}} = E_i \sum_{k \simeq i} \overline{Y_{ik}^{\text{bus}}} \overline{E}_k \tag{5.1}$$

where E is the complex voltage, I^{bus} is the bus current, Y^{bus} is the bus admittance and $\simeq$ means *connected or equal to*. The phase shift between S_i and E_j is represented exploiting the complex conjugation instead of an explicit reference to time.

For convenience, equation (5.1) can be seen as a row of the matrix equation:

$$\mathbf{S} = [\mathbf{E}] \overline{\mathbf{Y}^{\text{bus}}} \overline{\mathbf{E}} \tag{5.2}$$

where $\overline{\mathbf{E}}$ and $\mathbf{S}$ are vectors, $\mathbf{Y}^{\text{bus}}$ is a sparse matrix and $[\mathbf{E}]$ is the square matrix with the entries of the vector $\mathbf{E}$ on its diagonal and all zeros elsewhere.

In order to classify the buses nature, the complex power S and the complex voltage E are written using the *cartesian* and the *polar* coordinates respectively:

$$\begin{aligned} S &= \Re(S) + \jmath \Im(S) = P + \jmath \mathrm{Q} \\ E &= |E| e^{\jmath \varphi(E)} = V e^{\jmath \delta} = V(\cos(\delta) + \jmath \sin(\delta)) \end{aligned} \tag{5.3}$$

where P and Q are real quantities called *active* and *reactive* power, while *magnitude* and *phase* of E are indicated with V and δ respectively. Given these coordinates, each bus of an electrical grid can be classified depending on which of the four real quantities just introduced are constant:

- there is always a single $V\delta$-bus that has V and δ constant, called *slack* bus or *slack* bus;
- PQ-buses have P and Q constant;

- PV-buses have P and V constant.

Then, the number of buses in a grid is $1 + n_{PQ} + n_{PV} = n$.

In the following matrix equations, the subscripts PQ, PV and $V\delta$ indicate the selection of rows corresponding to PQ-buses, PV-buses and $V\delta$-bus respectively.

The unknown values in the PFEs depend on the bus type associated to the index i: if i refers to the $V\delta$-bus, its complex power S has to be determined; if i is the index of a PQ-bus, then its voltage E is a complex unknown; if i is a PV-bus, the unknowns are Q and δ. Hence, the system (5.2) contains n complex equations in $1 + n_{PQ}$ complex and $2n_{PV}$ real unknowns. Since $\mathbb{C}^{1+n_{PQ}} \times \mathbb{R}^{2n_{PV}}$ and $\mathbb{C}^n$ have the same dimension as vector spaces over $\mathbb{R}$, the number of equations in (5.2) is equal to its degree of freedom and, since the equations are linearly independent, the system is *consistent* (i.e., admits at least one solution). It is important to observe that the unknowns related to PV-buses can only be described with their real form, so that complex formulations of the equations are disadvantaged.

In this thesis buses and branches can be either working or failed, and this is reflected in how $\mathbf{S}$, $\mathbf{E}$ and $\mathbf{Y}^{\text{bus}}$ are treated. More details are in [64], here it is important just to remark that if, after a failure, the electrical grid presents islands then our formulation does not allow to solve the PFEs.

Solutions of a non-linear system of complex or real equations with many unknowns are often not computable with a closed formula. Hence, the most straightforward way to solve systems like (5.2) is using an iterative method [97]. One of the most used is the Newton-Raphson method (NR) [98]: starting from an initial approximation of the solution $\mathbf{X}^0$ of a system of equations $\mathbf{F}(\mathbf{X}) = \mathbf{0}$, the corrections $\Delta\mathbf{X}^k = \mathbf{X}^{k+1} - \mathbf{X}^k$ are obtained by solving the linear equation system

$$\mathbf{J}(\mathbf{X}^k)\Delta\mathbf{X}^k = -\mathbf{F}(\mathbf{X}^k) \tag{5.4}$$

where $\mathbf{J} = (\partial F_i / \partial X_j)$ is the Jacobian matrix of $\mathbf{F}$. The iterative process is stopped when $|\mathbf{F}(\mathbf{X}^k)|_\infty < tol$. So, as shown in Algorithm 2, the non-linear equations system is solved through a sequence of linear system solutions. In turn, there are many ways to solve the linear system of Equation (5.4) and line 4 of Algorithm 2. In the basic model of Chapter 3 an iterative method [99], the Krylov method, has been considered also for the linear system solution. One of the advantages is that a basic implementation of the Krylov method requires only two case specific subroutines: one to evaluate $\mathbf{F}(\mathbf{u})$, where $\mathbf{u}$ is a generic vector, and the other to evaluate the Jacobian-vector product $\mathbf{J}(\mathbf{X}^k) \cdot \mathbf{u}$. An additional enhancement can be considered: the exact Jacobian-vector product can be approximated, fixing a small σ,

Algorithm 2: Newton-Raphson method.

1: $\mathbf{X}^0 \leftarrow$ initial guess, $k \leftarrow 0$
2: **repeat**
3: $f \leftarrow \mathbf{F}(\mathbf{X})$, $J \leftarrow \mathbf{J}(\mathbf{X})$
4: $\Delta\mathbf{X}^k \leftarrow$ solve linear system $J\Delta\mathbf{X}^k = -f$
5: $\mathbf{X}^{k+1} \leftarrow \mathbf{X}^k + \Delta\mathbf{X}^k$
6: $k \leftarrow k + 1$
7: **until** $|\mathbf{F}(\mathbf{X}^k)|_\infty < tol$

with the finite difference $[\mathbf{F}(\mathbf{X} + \sigma\mathbf{u}) - \mathbf{F}(\mathbf{X})]/\sigma$ that only requires the $\mathbf{F}(\mathbf{u})$ evaluation subroutine [100]. The resulting method is called Inexact-Newton-Krylov. Unfortunately, two relevant drawbacks emerged:

- the Inexact-Newton-Krylov method applied directly to the PFEs rarely converges to a solution. A common strategy to overcome this issue is to use a preconditioner [97, 101, 102]. A fine tuning, highly dependent on the specific case study, is needed to make the method convergent.

- Krylov iterative methods are highly influenced by the equations ordering. In the basic model discussed in Chapter 3, the equations ordering depends on the presence/absence of electrical components attached to the grid buses [8].

Thus, a different design for the PFEs solver has been considered to free the enhanced model of Chapter 6 from an highly case-specific fine tuning. In order to do that, a direct method [103] has been employed to solve the linear system of Equation (5.4).

More details will be discussed in the following sections, but one key aspect of the PFEs guided the investigations: their analytical properties. The PFEs are formulated imposing relations among complex numbers, but the presence of the complex conjugate, needed to impose the phase relation between powers and voltages, make them *non-holomorphic*, i.e., non complex analytic. As a consequence of this fact, it is not possible to form the complex Jacobian of Equation (5.4). In this chapter then a review of existing strategies to overcome this issue will be presented. Remaining within the context of the Newton-Raphson method, three main ideas have been considered:

- translating the PFEs in real-valued non-linear equations. This strategy, the first in chronological order [104], is considered the "standard approach" and can produce a vast set of real non-linear equations, depending on how each complex parameter or unknown is interpreted

as a couple of real values. The two most common choices are: using the Cartesian representation for every parameter and the polar representation for every unknown, or using the Cartesian representation for both parameters and unknowns. The former, described in Equation (3.1), was employed in the basic SG model of Chapter 3 and is at the basis of the bus type definition. Additional details are provided in Section 5.2.1. The latter will be discussed in Section 5.2.1. In any case, the equations are real analytic and the real Jacobian is used within the Newton-Raphson method;

- doubling complex equations and unknowns. Considering Equation (5.2) and its complex conjugate, and introducing a new set of independent variable W that substitute the complex conjugate of E, a new complex system of equations is obtained. A vector solves this new system if and only if it solves the PFEs, and in addition the new equations are complex holomorphic so the complex Jacobian is well defined. Additional details are in Section 5.2.2;

- exploit the Wirtinger calculus. The Wirtinger calculus is a real-based calculus but exploits the algebraic structure of complex PFEs, so is someway in between the previous approaches. More details are in Section 5.3.

Detailed investigations have been carried out in order to compare all the mentioned strategies from the computational point of view. Theoretical predictions and experiments will be presented in Section 5.4.1.

Another direction of research has been the study of the strategies convergence properties. Notice that it is not always possible to solve the PFEs, and then obtain an estimate of the EI state. The NR method is an iterative process and there are circumstances under which it does not converges to a solution. In both the basic and enhanced SG models of Chapters 3 and 6 when NR does not converges the EI is considered in blackout and the simulation stopped, so those strategies that manifest a better convergence profile are considered preferable with respect to those that often does not converge. In Section 5.4.2 all the considered strategies have been compared in different scenarios in order to gain information about their convergence properties.

5.2 Available solution methods

In order to make NR exploitable in the PFEs context, most of the strategies formulate the complex system (5.2) with real variables and then construct $\mathbf{J}$ using the real derivatives (Section 5.2.1). Other strategies, instead,

maintain a complex formulation but calculate $\mathbf{J}$ with different derivatives (Sections 5.2.2 and 5.3).

In both cases, NR is particularly powerful in solving the PFEs because $\mathbf{J}$ maintains the sparsity of $\mathbf{Y}^{\text{bus}}$.

5.2.1 Real-based strategies

Depending on the real coordinates used for the voltages, there are two main real-based strategies.

Polar Coordinates [64]

when writing voltages with their polar coordinates, equation (5.1) becomes:

$$\Delta S_i = E_i e^{\jmath\delta_i} \sum_{k \simeq i} \overline{Y_{ik}^{\text{bus}}} V_k e^{-\jmath\delta_k} - S_i = 0 \tag{5.5}$$

and a real equation system can be obtained by considering $\Re(\Delta\mathbf{S}) = \Delta\mathbf{P}$ and $\Im(\Delta\mathbf{S}) = \Delta\mathbf{Q}$. In this case, the subsystem

$$\begin{cases} \Delta\mathbf{P}_{PQ} = \Re\Big([\mathbf{E}_{PQ}]\overline{\mathbf{Y}_{PQ}^{\text{bus}}\mathbf{E}}\Big) - \mathbf{P}_{PQ} = \mathbf{0} \\ \Delta\mathbf{P}_{PV} = \Re\Big([\mathbf{E}_{PV}]\overline{\mathbf{Y}_{PV}^{\text{bus}}\mathbf{E}}\Big) - \mathbf{P}_{PV} = \mathbf{0} \\ \Delta\mathbf{Q}_{PQ} = \Im\Big([\mathbf{E}_{PQ}]\overline{\mathbf{Y}_{PQ}^{\text{bus}}\mathbf{E}}\Big) - \mathbf{Q}_{PQ} = \mathbf{0} \end{cases} \tag{5.6}$$

contains $2n_{PQ} + n_{PV}$ real equations in $2n_{PQ} + n_{PV}$ real unknowns and determines $\mathbf{F}(\boldsymbol{\delta}_{PQ,\,PV}, \mathbf{V}_{PQ}) = \mathbf{0}$ to be solved with NR. Its solutions, once found, can be replaced in the other subsystem containing the equations $\Delta\mathbf{P}_{V\delta} = \mathbf{0}$ and $\Delta\mathbf{Q}_{PV,V\delta} = \mathbf{0}$ to easily calculate all the remaining real unknowns $(\mathbf{P}_{V\delta}, \mathbf{Q}_{PV,\,V\delta})$. Since the derivatives of (5.6) can be calculated as the real and imaginary parts of:

$$\begin{aligned} \frac{\partial\Delta\mathbf{S}}{\partial\mathbf{V}} &= [\mathbf{E}]\Big([\overline{\mathbf{Y}^{\text{bus}}\mathbf{E}}] + \overline{\mathbf{Y}^{\text{bus}}}[\overline{\mathbf{E}}]\Big)[\mathbf{V}^{-1}] \\ \frac{\partial\Delta\mathbf{S}}{\partial\boldsymbol{\delta}} &= \jmath[\mathbf{E}]\Big([\overline{\mathbf{Y}^{\text{bus}}\mathbf{E}}] - \overline{\mathbf{Y}^{\text{bus}}}[\overline{\mathbf{E}}]\Big) \end{aligned} \tag{5.7}$$

the real Jacobian of $\mathbf{F}$ for (5.4) is given by:

$$\mathbf{J} = \left(\begin{array}{c|c} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \hline \mathbf{J}_{21} & \mathbf{J}_{22} \end{array}\right) \in \mathbb{R}^{2n_{PQ}+n_{PV},\,2n_{PQ}+n_{PV}} \tag{5.8}$$

where:

$$\begin{aligned} \mathbf{J}_{11} &= \Re\left(\frac{\partial \Delta \mathbf{S}_{PQ,PV}}{\partial \boldsymbol{\delta}_{PQ,PV}}\right) \qquad \mathbf{J}_{12} = \Re\left(\frac{\partial \Delta \mathbf{S}_{PQ,PV}}{\partial \mathbf{V}_{PQ}}\right) \\ \mathbf{J}_{21} &= \Im\left(\frac{\partial \Delta \mathbf{S}_{PQ}}{\partial \boldsymbol{\delta}_{PQ,PV}}\right) \qquad \mathbf{J}_{22} = \Im\left(\frac{\partial \Delta \mathbf{S}_{PQ}}{\partial \mathbf{V}_{PQ}}\right) \end{aligned} \tag{5.9}$$

The strategy, consisting in solving the formulation (5.6) with NR through the Jacobian (5.8), is here indicated with NR^{PC}.

Rectangular Coordinates [98]

if the voltages are written with their rectangular coordinates, then equation (5.1) becomes:

$$\Delta S_i = (E_i^x + \jmath E_i^y) \sum_{k \simeq i} \overline{Y_{ik}^{\text{bus}}} (E_j^x - \jmath E_j^y) - S_i = 0 \tag{5.10}$$

As in the previous formulation, the subsystem containing the equations $\Delta \mathbf{P}_{V\delta} = \mathbf{0}$ and $\Delta \mathbf{Q}_{PV,V\delta} = \mathbf{0}$ can be easily solved after obtaining the voltages of PQ-buses and PV-buses. However, the system (5.6) contains only $2n_{PQ} + n_{PV}$ equations while the real unknowns $(\mathbf{E}_{PQ,PV}^x, \mathbf{E}_{PQ,PV}^y)$ are $2(n_{PQ} + n_{PV})$. Consequently, $\mathbf{F}(\mathbf{E}_{PQ,PV}^x, \mathbf{E}_{PQ,PV}^y) = \mathbf{0}$ has to be:

$$\begin{cases} \Delta \mathbf{P}_{PQ} = \Re\left([\mathbf{E}_{PQ}]\overline{\mathbf{Y}_{PQ}^{\text{bus}}}\overline{\mathbf{E}}\right) - \mathbf{P}_{PQ} = \mathbf{0} \\ \Delta \mathbf{P}_{PV} = \Re\left([\mathbf{E}_{PV}]\overline{\mathbf{Y}_{PV}^{\text{bus}}}\overline{\mathbf{E}}\right) - \mathbf{P}_{PV} = \mathbf{0} \\ \Delta \mathbf{Q}_{PQ} = \Im\left([\mathbf{E}_{PQ}]\overline{\mathbf{Y}_{PQ}^{\text{bus}}}\overline{\mathbf{E}}\right) - \mathbf{Q}_{PQ} = \mathbf{0} \\ \Delta \mathbf{V}_{PV}^2 = |\mathbf{E}_{PV}|^2 - \mathbf{V}_{PV}^2 = \mathbf{0} \end{cases} \tag{5.11}$$

so that there are $2(n_{PQ} + n_{PV}) = 2n - 2$ equations and the PV-buses voltages respect the magnitude constraint.

Derivatives of (5.11) are given by:

$$\begin{aligned} \frac{\partial \Delta \mathbf{P}}{\partial \mathbf{E}^x} &= [\mathbf{G}^{\text{bus}}\mathbf{E}^x - \mathbf{B}^{\text{bus}}\mathbf{E}^y] + [\mathbf{E}^x]\mathbf{G}^{\text{bus}} + [\mathbf{E}^y]\mathbf{B}^{\text{bus}} \\ \frac{\partial \Delta \mathbf{P}}{\partial \mathbf{E}^y} &= [\mathbf{G}^{\text{bus}}\mathbf{E}^y + \mathbf{B}^{\text{bus}}\mathbf{E}^x] + [\mathbf{E}^y]\mathbf{G}^{\text{bus}} - [\mathbf{E}^x]\mathbf{B}^{\text{bus}} \\ \frac{\partial \Delta \mathbf{Q}}{\partial \mathbf{E}^x} &= -[\mathbf{G}^{\text{bus}}\mathbf{E}^y + \mathbf{B}^{\text{bus}}\mathbf{E}^x] + [\mathbf{E}^y]\mathbf{G}^{\text{bus}} - [\mathbf{E}^x]\mathbf{B}^{\text{bus}} \\ \frac{\partial \Delta \mathbf{Q}}{\partial \mathbf{E}^y} &= [\mathbf{G}^{\text{bus}}\mathbf{E}^x - \mathbf{B}^{\text{bus}}\mathbf{E}^y] - [\mathbf{E}^x]\mathbf{G}^{\text{bus}} - [\mathbf{E}^y]\mathbf{B}^{\text{bus}} \\ \frac{\partial \Delta \mathbf{V}^2}{\partial \mathbf{E}^x} &= [2\mathbf{E}^x] \qquad \frac{\partial \Delta \mathbf{V}^2}{\partial \mathbf{E}^y} = [2\mathbf{E}^y] \end{aligned} \tag{5.12}$$

where $\mathbf{G}^{\text{bus}} = \Re\big(\mathbf{Y}^{\text{bus}}\big)$ is the bus conductance matrix and $B^{\text{bus}} = \Im\big(\mathbf{Y}^{\text{bus}}\big)$ is the bus susceptance matrix.

Then, in (5.4) the Jacobian of $\mathbf{F}$ is:

$$\mathbf{J} = \left(\begin{array}{c|c} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \hline \mathbf{J}_{21} & \mathbf{J}_{22} \\ \hline \mathbf{J}_{31} & \mathbf{J}_{32} \end{array}\right) \in \mathbb{R}^{2n-2,\,2n-2} \tag{5.13}$$

where:

$$\begin{aligned} \mathbf{J}_{11} &= \frac{\partial \Delta \mathbf{P}_{PQ,PV}}{\partial \mathbf{E}^{x}_{PQ,PV}} & \mathbf{J}_{12} &= \frac{\partial \Delta \mathbf{P}_{PQ,PV}}{\partial \mathbf{E}^{y}_{PQ,PV}} \\ \mathbf{J}_{21} &= \frac{\partial \Delta \mathbf{Q}_{PQ}}{\partial \mathbf{E}^{x}_{PQ,PV}} & \mathbf{J}_{22} &= \frac{\partial \Delta \mathbf{Q}_{PQ}}{\partial \mathbf{E}^{y}_{PQ,PV}} \\ \mathbf{J}_{31} &= \frac{\partial \Delta \mathbf{V}^{2}_{PV}}{\partial \mathbf{E}^{x}_{PQ,PV}} & \mathbf{J}_{32} &= \frac{\partial \Delta \mathbf{V}^{2}_{PV}}{\partial \mathbf{E}^{y}_{PQ,PV}} \end{aligned} \tag{5.14}$$

Solving (5.11) with NR and (5.13) as Jacobian determines a strategy here called NR^{RC}.

5.2.2 Complex-based strategies

Despite the fact that the complex conjugate makes equations (5.1) non-holomorphic with respect to the voltages, and hence impossible to differentiate using the classic complex derivatives, there are particular processes that make complex PFEs solvable with NR without converting them to their real version. An example, concerning an approximation of the complex Jacobian with respect to $\overline{E}$, can be seen in [105].

Another possibility consists in considering E and $\overline{E}$ as independent variables and use complex derivatives [106]. To emphasize this independence, $\overline{E}$ will be indicated with W. In this case, equation (5.1) becomes:

$$\Delta S_i = E_i \sum_{k \simeq i} \overline{Y^{\text{bus}}_{ik}} W_k - S_i = 0 \tag{5.15}$$

and it is holomorphic with respect to E and W.

As in Section 5.2.1, $\mathbf{P}_{V\delta}$ and $\mathbf{Q}_{PV,\,V\delta}$ can be determined from voltages, so that the unknowns of the system are $\mathbf{E}_{PQ,\,PV}$ and $\mathbf{W}_{PQ,\,PV}$. Since $2(n_{PQ} + n_{PV})$ complex unknowns are involved, in order to have $2(n_{PQ} + n_{PV})$ linearly

independent equations, [106] considers the complex formulation:

$$\begin{cases} \Delta\mathbf{S}_{PQ} = [\mathbf{E}_{PQ}]\overline{\mathbf{Y}_{PQ}^{\text{bus}}}\mathbf{W} - \mathbf{S}_{PQ} = \mathbf{0} \\ \Delta 2\mathbf{P}_{PV} = [\mathbf{E}_{PV}]\overline{\mathbf{Y}_{PV}^{\text{bus}}}\mathbf{W} + [\mathbf{W}_{PV}]\mathbf{Y}_{PV}^{\text{bus}}\mathbf{E} - 2\mathbf{P}_{PV} = \mathbf{0} \\ \Delta\overline{\mathbf{S}}_{PQ} = [\mathbf{W}_{PQ}]\mathbf{Y}_{PQ}^{\text{bus}}\mathbf{E} - \overline{\mathbf{S}}_{PQ} = \mathbf{0} \\ \Delta\mathbf{V}_{PV}^{2} = [\mathbf{E}_{PV}]\mathbf{W}_{PV} - \mathbf{V}_{PV}^{2} = \mathbf{0} \end{cases} \tag{5.16}$$

In this case, complex derivatives are calculated as:

$$\begin{aligned} &\frac{\partial\Delta\mathbf{S}}{\partial\mathbf{E}} = [\overline{\mathbf{Y}^{\text{bus}}}\mathbf{W}] \qquad \frac{\partial\Delta\mathbf{S}}{\partial\mathbf{W}} = [\mathbf{E}]\overline{\mathbf{Y}^{\text{bus}}} \\ &\frac{\partial\Delta 2\mathbf{P}}{\partial\mathbf{E}} = [\mathbf{W}]\mathbf{Y}^{\text{bus}} + [\overline{\mathbf{Y}^{\text{bus}}}\mathbf{W}] \\ &\frac{\partial\Delta 2\mathbf{P}}{\partial\mathbf{W}} = [\mathbf{E}]\overline{\mathbf{Y}^{\text{bus}}} + [\mathbf{Y}^{\text{bus}}\mathbf{E}] \\ &\frac{\partial\Delta\overline{\mathbf{S}}}{\partial\mathbf{E}} = [\mathbf{W}]\mathbf{Y}^{\text{bus}} \qquad \frac{\partial\Delta\overline{\mathbf{S}}}{\partial\mathbf{W}} = [\mathbf{Y}^{\text{bus}}\mathbf{E}] \\ &\frac{\partial\Delta\mathbf{V}^{2}}{\partial\mathbf{E}} = [\mathbf{W}] \qquad \frac{\partial\Delta\mathbf{V}^{2}}{\partial\mathbf{W}} = [\mathbf{E}] \end{aligned} \tag{5.17}$$

Thus, in [106], (5.4) has $\mathbf{J}$ structured as follows:

$$\mathbf{J} = \left(\begin{array}{c|c} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \hline \mathbf{J}_{21} & \mathbf{J}_{22} \\ \hline \mathbf{J}_{31} & \mathbf{J}_{32} \\ \hline \mathbf{J}_{41} & \mathbf{J}_{42} \end{array}\right) \in \mathbb{C}^{2n-2,\,2n-2} \tag{5.18}$$

where:

$$\begin{aligned} &\mathbf{J}_{11} = \frac{\partial\Delta\mathbf{S}_{PQ}}{\partial\mathbf{E}_{PQ,PV}} \qquad \mathbf{J}_{12} = \frac{\partial\Delta\mathbf{S}_{PQ}}{\partial\mathbf{W}_{PQ,PV}} \\ &\mathbf{J}_{21} = \frac{\partial\Delta 2\mathbf{P}_{PV}}{\partial\mathbf{E}_{PQ,PV}} \qquad \mathbf{J}_{22} = \frac{\partial\Delta 2\mathbf{P}_{PV}}{\partial\mathbf{W}_{PQ,PV}} \\ &\mathbf{J}_{31} = \frac{\partial\Delta\overline{\mathbf{S}}_{PQ}}{\partial\mathbf{E}_{PQ,PV}} \qquad \mathbf{J}_{32} = \frac{\partial\Delta\overline{\mathbf{S}}_{PQ}}{\partial\mathbf{W}_{PQ,PV}} \\ &\mathbf{J}_{41} = \frac{\partial\Delta\mathbf{V}_{PV}^{2}}{\partial\mathbf{E}_{PQ,PV}} \qquad \mathbf{J}_{42} = \frac{\partial\Delta\mathbf{V}_{PV}^{2}}{\partial\mathbf{W}_{PQ,PV}} \end{aligned} \tag{5.19}$$

The strategy introduced in [106], that aims to solve (5.16) by using (5.18) in a complex version of NR, here is called NR^{NV}.

5.2.3 Discussion

Some considerations on the above reviewed strategies are drawn in the following.

In $\mathrm{NR}^{\mathrm{PC}}$, trigonometric functions are required and their evaluations imply numerical approximations that impact on the numerical precision of the results [68]. For $\mathrm{NR}^{\mathrm{RC}}$, the equations in (5.11) are polynomials so that trigonometric functions are avoided. A comparison between $\mathrm{NR}^{\mathrm{PC}}$ and $\mathrm{NR}^{\mathrm{RC}}$ with and without optimal multipliers is presented in [107].

Concerning the complex-based strategy $\mathrm{NR}^{\mathrm{NV}}$, it allows to model particular devices, as reported in [106], but all computational costs are increased because of the complex calculus. In addition, since E and W are treated as independent variables, $\overline{E}$ and W can differ during $\mathrm{NR}^{\mathrm{NV}}$ steps, resulting in less accurate outcomes.

Moving from the above considerations and aiming at improving on the reported weaknesses, new formulations are investigated, resorting to the Wirtinger calculus because it is directly applicable to the complex formulation of the PFEs.

5.3 Exploring Wirtinger calculus

The *Wirtinger derivatives* [108] exploit the fact that each complex function $\mathbf{F} : \mathbb{C}^m \to \mathbb{C}^m$ can be seen as $\tilde{\mathbf{F}} : \mathbb{R}^{2m} \to \mathbb{C}^m$ by writing its complex variables as $Z_i = X_i + \jmath Y_i$.

Hence, if $\tilde{\mathbf{F}}$ is differentiable in the real variables X_i and Y_i, the Wirtinger derivatives of $\mathbf{F}$ are defined as the following partial differential operators:

$$\frac{\partial^W \mathbf{F}}{\partial Z_i} = \frac{1}{2}\left(\frac{\partial \tilde{\mathbf{F}}}{\partial X_i} - \jmath \frac{\partial \tilde{\mathbf{F}}}{\partial Y_i}\right) \quad \frac{\partial^W \mathbf{F}}{\partial \overline{Z}_i} = \frac{1}{2}\left(\frac{\partial \tilde{\mathbf{F}}}{\partial X_i} + \jmath \frac{\partial \tilde{\mathbf{F}}}{\partial Y_i}\right) \tag{5.20}$$

It is important to observe that Wirtinger derivatives can be calculated without introducing $\tilde{\mathbf{F}}$. In this case, $\partial^W \mathbf{F}/\partial Z_i$ works as the classic complex derivative $\partial \mathbf{F}/\partial Z_i$ in which $\overline{Z}_i$ is considered as a constant, and the behavior of $\partial^W \mathbf{F}/\partial \overline{Z}_i$ is the same of $\partial \mathbf{F}/\partial \overline{Z}_i$ with Z_i constant.

This property makes Wirtinger derivatives defined also for complex functions that are non-holomorphic because of the complex conjugate.

As for the classic complex derivatives, also for Wirtinger derivatives a Jacobian can be considered. However, since the matrix containing all Wirtinger

derivatives of $\mathbf{F}$ is not square, the *Wirtinger Jacobian* is defined as:

$$\mathbf{J}^{W} = \left(\begin{array}{c|c} \frac{\partial^{W}\mathbf{F}}{\partial\mathbf{Z}} & \frac{\partial^{W}\mathbf{F}}{\partial\overline{\mathbf{Z}}} \\ \hline \frac{\partial^{W}\overline{\mathbf{F}}}{\partial\mathbf{Z}} & \frac{\partial^{W}\overline{\mathbf{F}}}{\partial\overline{\mathbf{Z}}} \end{array}\right) = \left(\begin{array}{c|c} \frac{\partial^{W}\mathbf{F}}{\partial\mathbf{Z}} & \frac{\partial^{W}\mathbf{F}}{\partial\overline{\mathbf{Z}}} \\ \hline \overline{\frac{\partial^{W}\mathbf{F}}{\partial\overline{\mathbf{Z}}}} & \overline{\frac{\partial^{W}\mathbf{F}}{\partial\mathbf{Z}}} \end{array}\right) \tag{5.21}$$

If the complex system $\mathbf{F}(\mathbf{Z}) = \mathbf{0}$ has to be solved, the Wirtinger Jacobian of $\mathbf{F}$ can be used in NR. Thus, analogously to (5.4), the correction $\Delta\mathbf{Z}$ is obtained by solving:

$$\mathbf{J}^{W}\left(\begin{array}{c}\Delta\mathbf{Z}\\ \hline \overline{\Delta\mathbf{Z}}\end{array}\right) = -\left(\begin{array}{c}\mathbf{F}(\mathbf{Z})\\ \hline \overline{\mathbf{F}(\mathbf{Z})}\end{array}\right) \tag{5.22}$$

Since the PFEs contain complex conjugates that make them non-holomorphic with respect to the complex voltages, Wirtinger calculus is very appropriate to differentiate them.

An approach similar to the proposed one was introduced in [109] to differentiate the PFEs with respect to the electric currents, but for the different purpose of finding a necessary condition for power-flow insolvability in systems with distributed generators. Another interesting similar approach was introduced in [110] where, after considering Wirtinger derivatives, a solving method using complex tableaux is presented.

In order to use Wirtinger calculus, the PFEs have to be formulated with an appropriate system of complex equations. The number of these equations must be $n_{PQ} + n_{PV}$, which is the number of the unknown complex voltages $\mathbf{E}_{PQ,\,PV}$ (as in Section 5.2, $\mathbf{P}_{V\delta}$ and $\mathbf{Q}_{PV,\,V\delta}$ can be easily determined later).

The PFEs and the constraints given by the bus types must be respected. Thus the first n_{PQ} equations can be simply (5.1) while, for PV-buses, the following n_{PV} complex equations, introduced in [111], are considered:

$$\Delta F_i = 2\Re\left(E_i\sum_{k\simeq i}\overline{Y_{ik}^{\text{bus}}}\,\overline{E}_k\right) - 2P_i + \jmath(|E_i|^2 - V_i^2) = 0 \tag{5.23}$$

The complex system $\mathbf{F}(\mathbf{E}_{PQ,PV}) = \mathbf{0}$ is consequently given by:

$$\begin{cases} \Delta\mathbf{S}_{PQ} = [\mathbf{E}_{PQ}]\,\overline{\mathbf{Y}_{PQ}^{\text{bus}}}\,\overline{\mathbf{E}} - \mathbf{S}_{PQ} = \mathbf{0} \\ \Delta\mathbf{F}_{PV} = 2\Re\left([\mathbf{E}_{PV}]\,\overline{\mathbf{Y}_{PV}^{\text{bus}}}\,\overline{\mathbf{E}}\right) - 2\mathbf{P}_{PV} + \jmath(|\mathbf{E}_{PV}|^2 - \mathbf{V}_{PV}^2) = \mathbf{0} \end{cases} \tag{5.24}$$

Equations (5.23) imply that the active power and the voltage magnitude of each PV-bus respect the constraints and, in addition, $\mathbf{F}$ has a non-singular

Wirtinger Jacobian. Other equations for PV-buses can be found, paying attention to obtain a non-singular Wirtinger Jacobian matrix.

The Wirtinger derivatives of $\mathbf{F}$ are:

$$\begin{aligned}
\frac{\partial^W \Delta \mathbf{S}}{\partial \mathbf{E}} &= [\overline{\mathbf{Y}^{\text{bus}}}\overline{\mathbf{E}}] \qquad \frac{\partial^W \Delta \mathbf{S}}{\partial \overline{\mathbf{E}}} = [\mathbf{E}]\overline{\mathbf{Y}^{\text{bus}}} \\
\frac{\partial^W \Delta \mathbf{F}}{\partial \mathbf{E}} &= [\overline{\mathbf{E}}]\mathbf{Y}^{\text{bus}} + [\overline{\mathbf{Y}^{\text{bus}}}\overline{\mathbf{E}} + \jmath\overline{\mathbf{E}}] \\
\frac{\partial^W \Delta \mathbf{F}}{\partial \overline{\mathbf{E}}} &= [\mathbf{E}]\overline{\mathbf{Y}^{\text{bus}}} + [\mathbf{Y}^{\text{bus}}\mathbf{E} + \jmath\mathbf{E}]
\end{aligned} \tag{5.25}$$

It is noteworthy that, although the derivatives in (5.25) appear similar to the ones of NR^{NV} in (5.17), their number is halved thanks to the definition given in (5.21).

The Wirtinger Jacobian of $\mathbf{F}$ can be structured with only four blocks, as observed in (5.21):

$$\mathbf{J}^W = \left(\begin{array}{c|c} \mathbf{J}^W_{11} & \mathbf{J}^W_{12} \\ \hline \mathbf{J}^W_{21} & \mathbf{J}^W_{22} \\ \hline \overline{\mathbf{J}^W_{12}} & \overline{\mathbf{J}^W_{11}} \\ \hline \overline{\mathbf{J}^W_{22}} & \overline{\mathbf{J}^W_{21}} \end{array}\right) = \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \overline{\mathbf{B}} & \overline{\mathbf{A}} \end{array}\right) \tag{5.26}$$

where:

$$\begin{aligned}
\mathbf{J}^W_{11} &= \frac{\partial^W \Delta \mathbf{S}_{PQ}}{\partial \mathbf{E}_{PQ,PV}} \qquad \mathbf{J}^W_{12} = \frac{\partial^W \Delta \mathbf{S}_{PQ}}{\partial \overline{\mathbf{E}}_{PQ,PV}} \\
\mathbf{J}^W_{21} &= \frac{\partial^W \Delta \mathbf{F}_{PV}}{\partial \mathbf{E}_{PQ,PV}} \qquad \mathbf{J}^W_{22} = \frac{\partial^W \Delta \mathbf{F}_{PV}}{\partial \overline{\mathbf{E}}_{PQ,PV}}
\end{aligned} \tag{5.27}$$

Despite starting from a different formulation, the Wirtinger derivatives exploited to obtain the matrix (5.26) actually returns the same Jacobian obtained in [111] through differential forms.

The strategy that uses the Wirtinger Jacobian (5.26) in NR to solve (5.24), here is called $\text{NR}^{\text{W}\mathbb{C}}$.

An improvement of $\text{NR}^{\text{W}\mathbb{C}}$ can be obtained through a real conversion of $\mathbf{J}^W$, exploiting its particular structure as follows.

The new-found conversion starts with inverting the order of the last $n = n_{PQ} + n_{PV}$ rows and columns of $\mathbf{J}^W$, in order to obtain a *centrohermitian* matrix [112], defined as:

$$\mathbf{M} \in \mathbb{C}^{m,m} | M_{ij} = \overline{M}_{m+1-i,m+1-j}, \text{ for } 1 \leq i,j \leq m \tag{5.28}$$

The permutation of indices that allows the conversion of $\mathbf{J}^W$ into a centrohermitian matrix is $\pi : i \mapsto 3n + 1 - i$ for $n < i \leq 2n$. The matrix form $\mathbf{P}_\pi$

of π contains the matrices:

$$\mathbb{I}_n = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}, \quad \mathbb{J}_n = \begin{pmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{pmatrix} \in \mathbb{R}^{n,n} \tag{5.29}$$

known as *identity* and *exchange* matrix respectively, so that:

$$\mathbf{P}_\pi = \left(\begin{array}{c|c} \mathbb{I}_n & \mathbf{0}_n \\ \hline \mathbf{0}_n & \mathbb{J}_n \end{array}\right) \tag{5.30}$$

The action of π on $\mathbf{J}^W$, using $\mathbf{P}_\pi$, determines:

$$\mathbf{P}_\pi \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \overline{\mathbf{B}} & \overline{\mathbf{A}} \end{array}\right) \mathbf{P}_\pi = \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B}\mathbb{J}_n \\ \hline \mathbb{J}_n\mathbf{B} & \mathbb{J}_n\mathbf{A}\mathbb{J}_n \end{array}\right) \tag{5.31}$$

that satisfies (5.28) and thus is a centrohermitian matrix.

In particular, using π as in (5.31), every Wirtinger Jacobian is similar to a centrohermitian matrix. Following the classical definition, two square matrices A and B are similar if there exists an invertible matrix V such that $B = V^{-1}AV$, i.e., if they represent the same linear operator under different basis. In particular, here the similarity is obtained through the permutation matrix $\mathbf{P}_\pi$. Similarity preserves eigenvalues and then convergence behavior.

As demonstrated in [112], every centrohermitian matrix is in turn similar to a real matrix. In fact, given the *unitary* (i.e., $\mathbf{U}^{-1} = \mathbf{U}^* = \overline{\mathbf{U}^t}$) matrix:

$$\mathbf{U} = \tfrac{1}{\sqrt{2}} \left(\begin{array}{c|c} \mathbb{I}_n & \jmath\mathbb{I}_n \\ \hline \mathbb{J}_n & -\jmath\mathbb{J}_n \end{array}\right) \in \mathbb{C}^{2n,2n} \tag{5.32}$$

the centrohermitian matrix determined in (5.31) is similar to the real matrix:

$$\begin{aligned} &\mathbf{U}^* \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B}\mathbb{J}_n \\ \hline \mathbb{J}_n\mathbf{B} & \mathbb{J}_n\mathbf{A}\mathbb{J}_n \end{array}\right) \mathbf{U} \\ &= \left(\begin{array}{c|c} \Re(\mathbf{A}+\mathbf{B}) & \Im(\mathbf{B}-\mathbf{A}) \\ \hline \Im(\mathbf{A}+\mathbf{B}) & \Re(\mathbf{A}-\mathbf{B}) \end{array}\right) = \mathbf{J}^{W\mathbb{R}} \end{aligned} \tag{5.33}$$

The matrix transformations described in (5.31) and (5.33) influence the system (5.22) as follows. Starting from:

$$\mathbf{J}^W \begin{pmatrix} \Delta\mathbf{E} \\ \hline \Delta\overline{\mathbf{E}} \end{pmatrix} = \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \overline{\mathbf{B}} & \overline{\mathbf{A}} \end{array}\right) \begin{pmatrix} \Delta\mathbf{E} \\ \hline \Delta\overline{\mathbf{E}} \end{pmatrix} = - \begin{pmatrix} \mathbf{F} \\ \hline \overline{\mathbf{F}} \end{pmatrix} \tag{5.34}$$

where $\Delta\mathbf{E} = \Delta\mathbf{E}_{PQ,PV}$ and $\mathbf{F} = \mathbf{F}(\mathbf{E}_{PQ,PV})$ is the left-hand side of the system in (5.24), the result of the permutation introduced in (5.31) is:

$$\begin{aligned}
\mathbf{P}_\pi \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \overline{\mathbf{B}} & \overline{\mathbf{A}} \end{array}\right) \mathbf{P}_\pi \mathbf{P}_\pi \left(\begin{array}{c} \Delta\mathbf{E} \\ \hline \Delta\overline{\mathbf{E}} \end{array}\right) &= -\mathbf{P}_\pi \left(\begin{array}{c} \mathbf{F} \\ \hline \overline{\mathbf{F}} \end{array}\right) \\
\left(\begin{array}{c|c} \mathbf{A} & \mathbf{B}\mathbb{J}_n \\ \hline \mathbb{J}_n\mathbf{B} & \mathbb{J}_n\mathbf{A}\mathbb{J}_n \end{array}\right) \left(\begin{array}{c} \Delta\mathbf{E} \\ \hline \mathbb{J}_n\Delta\overline{\mathbf{E}} \end{array}\right) &= -\left(\begin{array}{c} \mathbf{F} \\ \hline \mathbb{J}_n\overline{\mathbf{F}} \end{array}\right)
\end{aligned} \tag{5.35}$$

and, considering the similitude in (5.33), (5.35) becomes:

$$\begin{aligned}
\mathbf{U}^* \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B}\mathbb{J}_n \\ \hline \mathbb{J}_n\mathbf{B} & \mathbb{J}_n\mathbf{A}\mathbb{J}_n \end{array}\right) \mathbf{U}\mathbf{U}^* \left(\begin{array}{c} \Delta\mathbf{E} \\ \hline \mathbb{J}_n\Delta\overline{\mathbf{E}} \end{array}\right) &= -\mathbf{U}^* \left(\begin{array}{c} \mathbf{F} \\ \hline \mathbb{J}_n\overline{\mathbf{F}} \end{array}\right) \\
\mathbf{J}^{W\mathbb{R}} \tfrac{1}{\sqrt{2}} \left(\begin{array}{c} \Delta\mathbf{E} + \Delta\overline{\mathbf{E}} \\ \hline \Delta\mathbf{E} - \Delta\overline{\mathbf{E}} \end{array}\right) &= -\tfrac{1}{\sqrt{2}} \left(\begin{array}{c} \mathbf{F} + \overline{\mathbf{F}} \\ \hline \mathbf{F} - \overline{\mathbf{F}} \end{array}\right) \\
\mathbf{J}^{W\mathbb{R}} \left(\begin{array}{c} \Re(\Delta\mathbf{E}) \\ \hline \Im(\Delta\mathbf{E}) \end{array}\right) &= -\left(\begin{array}{c} \Re(\mathbf{F}) \\ \hline \Im(\mathbf{F}) \end{array}\right)
\end{aligned} \tag{5.36}$$

so that the resulting system has size $2(n_{PQ} + n_{PV})$ and contains only real values. The obtained real implementation of $\mathrm{NR}^{\mathrm{W}\mathbb{C}}$ is called $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$.

The determined linear equation system of $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$, although obtained through novel developments, is the same as the one of $\mathrm{NR}^{\mathrm{RC}}$. However, despite the same outcome from both strategies, the comparison studies described in Section 5.4 show that the differences in the construction of the Jacobian have an impact on the overall performance of the strategies, with $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$ performing better than the classical $\mathrm{NR}^{\mathrm{RC}}$.

The structure of $\mathbf{J}^{W\mathbb{R}}$ is exploitable to further reduce the number of equations in (5.36).

In particular, in (5.33) the last n_{PV} rows of $\mathbf{J}^{W\mathbb{R}}$ are:

$$\left(\begin{array}{c|c} \cdots & \cdots \\ \hline \Im(\mathbf{J}^W_{21} + \mathbf{J}^W_{22}) & \Re(\mathbf{J}^W_{21} - \mathbf{J}^W_{22}) \end{array}\right) \tag{5.37}$$

where, from (5.25):

$$\begin{aligned}
\frac{\partial^W \Delta\mathbf{F}}{\partial\mathbf{E}} + \frac{\partial^W \Delta\mathbf{F}}{\partial\overline{\mathbf{E}}} &= 2\Re\Big([\overline{\mathbf{E}}]\mathbf{Y}^{\mathrm{bus}} + [\overline{\mathbf{Y}^{\mathrm{bus}}\overline{\mathbf{E}}}]\Big) \\
&\quad + \jmath 2\Re([\![\mathbf{E}]\!]) \\
\frac{\partial^W \Delta\mathbf{F}}{\partial\mathbf{E}} - \frac{\partial^W \Delta\mathbf{F}}{\partial\overline{\mathbf{E}}} &= 2\Im([\![\mathbf{E}]\!]) \\
&\quad + \jmath 2\Im\Big([\overline{\mathbf{E}}]\mathbf{Y}^{\mathrm{bus}} + [\overline{\mathbf{Y}^{\mathrm{bus}}\overline{\mathbf{E}}}]\Big)
\end{aligned} \tag{5.38}$$

so that, from (5.27):

$$\begin{aligned}\Im(\mathbf{J}_{21}^{W}+\mathbf{J}_{22}^{W}) &= \Big(\mathbf{0} \;\Big|\; [\,2\Re(\mathbf{E}_{PV})\,]\Big) \in \mathbb{R}^{n_{PQ}+n_{PV},\,n_{PV}} \\ \Re(\mathbf{J}_{21}^{W}-\mathbf{J}_{22}^{W}) &= \Big(\mathbf{0} \;\Big|\; [\,2\Im(\mathbf{E}_{PV})\,]\Big) \in \mathbb{R}^{n_{PQ}+n_{PV},\,n_{PV}}\end{aligned} \tag{5.39}$$

Hence, the system (5.36) can be written as:

$$\left(\begin{array}{c|c|c|c} & & & \\ \mathbf{J}_1 & \mathbf{J}_2 & \mathbf{J}_3 & \mathbf{J}_4 \\ & & & \\ \hline \mathbf{0} & \mathbf{D}_1 & \mathbf{0} & \mathbf{D}_2 \end{array}\right) \left(\begin{array}{c} \Re(\Delta\mathbf{E}_{PQ}) \\ \Re(\Delta\mathbf{E}_{PV}) \\ \Im(\Delta\mathbf{E}_{PQ}) \\ \Im(\Delta\mathbf{E}_{PV}) \end{array}\right) = -\left(\begin{array}{c} \Re(\mathbf{F}) \\ \\ \Im(\mathbf{F}_{PQ}) \\ \Im(\mathbf{F}_{PV}) \end{array}\right) \tag{5.40}$$

where $\mathbf{D}_1 = [\,2\Re(\mathbf{E}_{PV})\,]$ and $\mathbf{D}_2 = [\,2\Im(\mathbf{E}_{PV})\,]$ are diagonal matrices. Calling $\hat{\mathbf{F}}$ the vector containing $\Re(\mathbf{F})$ and $\Im(\mathbf{F}_{PQ})$, the submatrices equations related to (5.40) are:

$$\begin{cases} \mathbf{J}_1\Re(\Delta\mathbf{E}_{PQ}) + \mathbf{J}_2\Re(\Delta\mathbf{E}_{PV}) \\ \qquad\qquad + \mathbf{J}_3\Im(\Delta\mathbf{E}_{PQ}) + \mathbf{J}_4\Im(\Delta\mathbf{E}_{PV}) = -\hat{\mathbf{F}} \\ \mathbf{D}_1\Re(\Delta\mathbf{E}_{PV}) + \mathbf{D}_2\Im(\Delta\mathbf{E}_{PV}) = -\Im(\mathbf{F}_{PV}) \end{cases} \tag{5.41}$$

In the second equation of (5.41), the real diagonal matrix $\mathbf{D}_1$ is always invertible, so that $\Re(\Delta\mathbf{E}_{PV})$ can be isolated:

$$\Re(\Delta\mathbf{E}_{PV}) = -\mathbf{D}_1^{-1}\big(\mathbf{D}_2\Im(\Delta\mathbf{E}_{PV}) + \Im(\mathbf{F}_{PV})\big) \tag{5.42}$$

and then substituted in the first equation of (5.41):

$$\begin{aligned}\mathbf{J}_1\Re(\Delta\mathbf{E}_{PQ}) - \mathbf{J}_2\mathbf{D}_1^{-1}\big(\mathbf{D}_2\Im(\Delta\mathbf{E}_{PV}) + \Im(\mathbf{F}_{PV})\big) \\ + \mathbf{J}_3\Im(\Delta\mathbf{E}_{PQ}) + \mathbf{J}_4\Re(\Delta\mathbf{E}_{PV}) = -\hat{\mathbf{F}}\end{aligned} \tag{5.43}$$

In conclusion, only the following submatrices equation has to be solved with an iterative method:

$$\begin{aligned}&\mathbf{J}_1\Re(\Delta\mathbf{E}_{PQ}) + \mathbf{J}_3\Im(\Delta\mathbf{E}_{PQ}) \\ &+ \big(\mathbf{J}_4 - \mathbf{J}_2\mathbf{D}_1^{-1}\mathbf{D}_2\big)\Im(\Delta\mathbf{E}_{PV}) = -\hat{\mathbf{F}} + \mathbf{J}_2\mathbf{D}_1^{-1}\Im(\mathbf{F}_{PV})\end{aligned} \tag{5.44}$$

that is a linear system of equations of size $2n_{PQ} + n_{PV}$. The square matrix associated to (5.44) is given by:

$$\mathbf{J}_{\text{red}}^{W\mathbb{R}} = \left(\begin{array}{c|c} & \\ \mathbf{J}_1 & \mathbf{J}_3 \\ & \\ \hline & \overline{\mathbf{J}_4 - \mathbf{J}_2\mathbf{D}_1^{-1}\mathbf{D}_2} \end{array}\right) \tag{5.45}$$

and determines a real and reduced version of $\text{NR}^{\text{W}\mathbb{C}}$ that here is called $\text{NR}_{red}^{\text{W}\mathbb{R}}$.

5.4 Comparison results

This section is devoted to compare the six strategies $\mathrm{NR}^{\mathrm{PC}}$, $\mathrm{NR}^{\mathrm{RC}}$, $\mathrm{NR}^{\mathrm{NV}}$, $\mathrm{NR}^{\mathrm{W}\mathbb{C}}$, $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$ and $\mathrm{NR}^{\mathrm{W}\mathbb{R}}_{red}$, in terms of two major indicators representative of the quality of a PFEs solution strategy: computational effort and convergence to a solution. Therefore, theoretical analysis and experimental results about the computational costs are presented in Section 5.4.1, while convergence rates are discussed in Section 5.4.2.

All the experimental results are obtained through MATLAB [113] implementations of the strategies. $\mathrm{NR}^{\mathrm{PC}}$ was already part of MATPOWER [64], a MATLAB package for solving power-flow and optimal power-flow problems, while the other strategies have been implemented by the authors. The choice of MATLAB presents two major advantages: first, it is based on matrices manipulations so that the implementations directly reflect the mathematical descriptions presented in this article; second, MATPOWER functionalities can be fully exploited.

Eleven case studies, included in the MATPOWER package [3] and inspired by real electrical networks, are considered. In particular, some of the larger cases are selected, precisely case1354pegase, case1888rte, case2000tex, case2383wp, case2746wp, case2868rte, case2869pegase, case3375wp, case6470rte, case9241pegase and case13659pegase (the number in the name is the number of buses involved).

5.4.1 Computational analysis

The focus is on a single step of NR in each strategy. The analysis distinguishes between the two major parts composing a step: the matrices construction and the linear system solution.

In order to cover in more general terms the comparison study and provide a comprehensive analysis, the strategies are compared with: 1) a theoretical analysis of the computational burden for the matrices construction; 2) a count of the operations required in the linear system solver; 3) an assessment of the elapsed times in MATLAB.

Matrices Construction

The first theoretical comparison consists in counting the number of loads, stores, real additions and real multiplications required to calculate $\mathbf{F}$ and $\mathbf{J}$ for (5.4) in each strategy step. This is a technology independent analysis, since it does not make any assumption on the specific computer architecture adopted to execute the strategy.

Table 5.1: Theoretical estimations of the loads required for **F** and **J** at each step of each strategy

case	load	NR^{PC}	NR^{RC}	NR^{NV}	$NR^{W\mathbb{C}}$	$NR^{W\mathbb{R}}$	$NR^{W\mathbb{R}}_{red}$
1354peg.	**F**	53.3k	52.1k	103k	54.8k	52.1k	54.6k
	J	101k	136k	143k	88.9k	120k	124k
1888rte	**F**	70.4k	68.3k	136k	72.1k	68.3k	68.1k
	J	136k	180k	186k	114k	157k	159k
2000tex	**F**	80.5k	78.2k	156k	82.2k	78.2k	77.8k
	J	156k	206k	216k	130k	178k	181k
2383wp	**F**	84.1k	81.3k	162k	86.1k	81.3k	84.1k
	J	160k	214k	221k	138k	192k	198k
2746wp	**F**	96.0k	92.7k	184k	98.2k	92.7k	95.3k
	J	183k	244k	251k	157k	218k	224k
2868rte	**F**	106k	103k	205k	109k	103k	103k
	J	205k	271k	281k	172k	236k	241k
2869peg.	**F**	119k	116k	231k	122k	116k	121k
	J	229k	305k	323k	199k	270k	277k
3375wp	**F**	120k	116k	230k	123k	116k	118k
	J	230k	306k	315k	195k	271k	277k
6470rte	**F**	247k	237k	472k	250k	237k	231k
	J	480k	631k	654k	392k	541k	546k
9241peg.	**F**	403k	393k	782k	412k	393k	408k
	J	786k	1.05M	1.11M	682k	927k	951k
13659peg.	**F**	542k	539k	1.07M	566k	539k	565k
	J	1.05M	1.40M	1.48M	925k	1.26M	1.30M

The methodology selected is inspired by the work in [114]. Since loads are the most numerous and onerous operations [114], their actual amounts have been evaluated for the considered case studies and collected in Table 5.1. More details are provided in **??**

It is important to observe that, in every case, the construction of **F** requires about the same amount of loads in all strategies except for NR^{NV}, where it is almost double with respect to the others. For **J**, $NR^{W\mathbb{C}}$ requires reduced quantities of loads with respect to all the other strategies, followed by NR^{PC}. $NR^{W\mathbb{R}}$ obtains the same matrices of NR^{RC} with less loads, while $NR^{W\mathbb{R}}_{red}$ lies between them. Finally, NR^{NV} is the most load requiring strategy.

Table 5.2: Means of the flops required by the linear solver at each step of each strategy.

case	$\mathrm{NR}^{\mathrm{PC}}$	$\mathrm{NR}^{\mathrm{RC}}$	$\mathrm{NR}^{\mathrm{NV}}$	$\mathrm{NR}^{\mathrm{W}\mathbb{C}}$	$\mathrm{NR}^{\mathrm{W}\mathbb{R}}$	$\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$
1354peg.	387k	516k	2.45M	2.90M	487k	603k
1888rte	783k	702k	3.74M	4.32M	711k	1.08M
2000tex	1.59M	1.68M	28.1M	28.2M	1.67M	2.82M
2383wp	835k	1.36M	12.6M	14.1M	1.36M	1.65M
2746wp	976k	1.41M	17.3M	12.2M	1.41M	1.88M
2868rte	1.22M	1.25M	5.71M	6.23M	1.20M	1.89M
2869peg.	1.05M	1.39M	6.41M	7.19M	1.42M	1.81M
3375wp	1.39M	1.98M	16.5M	20.9M	1.92M	2.75M
6470rte	3.01M	4.02M	25.6M	30.3M	4.04M	6.33M
9241peg.	5.90M	6.89M	39.9M	54.1M	6.87M	10.2M
13659peg.	8.04M	8.82M	50.8M	52.1M	8.82M	13.0M

Linear System Solution

The second comparison study consists in counting the amount of *flops* (floating point operations) required by the linear solver of MATLAB to solve (5.4) in each step of each strategy.

To this purpose, each case study is solved through all the considered strategies, starting from the initial guess given in the MATPOWER file. The stop criteria has been set as the reaching of 100 iterations and, at each step, the flops needed to solve the linear system are collected. The mean of the required flops has been calculated and collected in Table 5.2.

Since all the introduced Jacobians are sparse matrices, MATLAB solves the associated linear systems with UMFPACK [115]. An estimate of the number of symbolic, numeric and solve flops is obtained by setting the debugging information of UMFPACK with the command `spparms('spumoni',2)`.

In all the cases, strategies with complex Jacobians ($\mathrm{NR}^{\mathrm{NV}}$, $\mathrm{NR}^{\mathrm{W}\mathbb{C}}$) require more flops for solving the linear system than the real-based ones ($\mathrm{NR}^{\mathrm{PC}}$, $\mathrm{NR}^{\mathrm{RC}}$, $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$, $\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$). This difference is due to heavier complex operators compared to real operators.

Among complex-based strategies, $\mathrm{NR}^{\mathrm{W}\mathbb{C}}$ requires generally an increased number of flops with respect to $\mathrm{NR}^{\mathrm{NV}}$, since the sparsity of $\mathbf{J}^w$ in (5.26) is higher than that of $\mathbf{J}$ in (5.18).

As for the strategies with real Jacobians, $\mathrm{NR}^{\mathrm{PC}}$ has the lowest required flops, followed by $\mathrm{NR}^{\mathrm{RC}}$ and $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$ (these last solve the same linear system). Despite the Jacobians of $\mathrm{NR}^{\mathrm{PC}}$ and $\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$ have the same size and sparsity, $\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$ is the worst among the real-based strategies. This because the linear

system of $\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$ is solved through the *unsymmetric* technique of UMFPACK, while in all the other strategies the *symmetric* technique is used.

Time Quantification

To quantify the elapsed times for the adopted case studies, the implementations in MATLAB of the strategies are executed in an environment characterized by an AMD(R) Opteron 6176 SE processor with fixed 2.3 GHz CPU, 12MB cache and 32GB RAM, equipped with an up to date GNU/Linux Operating System.

Since MATLAB is an interpreted language, the times can be worse than those of a compiled language. Nevertheless, linear algebra is multi-threaded by default in MATLAB and this feature is fully exploited in the adopted environment, so that the obtained times are anyhow competitive.

Distinct time values for the equations and Jacobian construction and for the linear system solution are calculated with the MATLAB functions `tic` and `toc` in the respective code sections. As for the flops count, each case study is solved with the reaching of 100 iterations as stop criterion, resulting in data sets of 100 measures for each time value. Table 5.3 shows the minimum of each set, since it is the value less influenced by non-related processes.

In each case, the results related to the construction of $\mathbf{F}$ confirm the theoretical estimations discussed in Section 5.4.1: all strategies require comparable times, except for $\mathrm{NR}^{\mathrm{NV}}$ that doubles the time.

The times for the evaluation of $\mathbf{J}$, instead, show some differences with respect to the estimations in Section 5.4.1. In particular, there are two main variations: first, $\mathrm{NR}^{\mathrm{NV}}$ is faster than expected, so that instead of being the worst strategy, it is the second best after $\mathrm{NR}^{\mathrm{W}\mathbb{C}}$; second, $\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$ is the slowest strategy, while theoretically it should have been better than $\mathrm{NR}^{\mathrm{RC}}$ and $\mathrm{NR}^{\mathrm{NV}}$. These differences are caused by the well-optimized matrix operations in MATLAB: $\mathrm{NR}^{\mathrm{NV}}$ requires more calculations, but they can be collected in only four matrix products; on the contrary, the process described in Section 5.3 leading to $\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$ requires an increased number of matrix products with respect to all the other strategies.

It is noteworthy that, despite they share the same results, the construction of the Jacobian (5.13) in $\mathrm{NR}^{\mathrm{RC}}$ is slower than that of the Jacobian (5.33) in $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$.

For the solving time, as seen in 5.4.1, the real-based strategies are the better ones and, in particular, $\mathrm{NR}^{\mathrm{PC}}$ is the fastest, followed by $\mathrm{NR}^{\mathrm{RC}}$ and $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$ (these last require almost the same time) and finally by $\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$ (that is disadvantaged by the UMFPACK *unsymmetric* solution technique). The

strategies that involve complex systems ($\mathrm{NR}^{\mathrm{NV}}$ and $\mathrm{NR}^{\mathrm{W}\mathbb{C}}$) respect exactly the estimations given by the number of required flops.

A major outcome from the above conducted experimental analysis is that the solving time accounts for most of the elapsed time, being one order of magnitude bigger than the other times. Then, $\mathrm{NR}^{\mathrm{PC}}$ is the fastest strategy, followed by $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$, $\mathrm{NR}^{\mathrm{RC}}$, $\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$ and finally by the complex ones.

However, it is pointed out that experimental results are strongly dependent on both the specific software implementation adopted (e.g., programming language, linked libraries, code optimization) and on its execution environment (computer architecture, such as threads and array processors [114] on multi-core or SIMD on many-core). Therefore, these aspects need to be taken into account when making the choice of the most appropriate strategy to employ.

In this respect, the value of conducting an analysis both on a theoretical basis and on a specific implementation setting is expected to be helpful from several perspectives. Among these, it could stimulate the development or more efficient implementations of theoretically promising strategies, targeting appropriate supporting technologies.

For example, investigating the adoption of many-core architectures, e.g., GPUs, looks a promising direction towards significantly reducing the linear system solution time so that strategies winning on reduced memory accesses (such as $\mathrm{NR}^{\mathrm{W}\mathbb{C}}$) could raise their competitiveness.

There exists a vast literature regarding the implementation of direct methods for sparse linear system solution that exploit both the host CPU and GPUs, such as the recent QR factorization presented in [116], and iterative methods, where the sparse matrix times vector subroutine is accelerated on GPUs [117].

At the moment, the MATLAB Parallel Computing Toolbox does not allow to accelerate direct methods for sparse matrices on GPUs and the fine tuning needed for the available iterative methods (i.e., the functions `gmres` and `bcg`) is left for future work.

5.4.2 Convergence rate

The ability of each strategy to achieve convergence is analyzed examining their behavior in solving the adopted cases, with randomly changed initial parameters according to three different scenarios: 1) random initial voltages, 2) random level of loads and 3) random R/X ratio.

In particular, each strategy is tested with 20 maximum iterations and tolerance 10^{-10} on 10 sets of 100 random initial states, generated through

normal distributions with changing standard deviation σ and mean depending on the scenario.

Since Jacobians of $\mathrm{NR}^{\mathrm{RC}}$, $\mathrm{NR}^{\mathrm{W}\mathbb{C}}$ and $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$ are similar matrices, the convergence rates of these strategies are always equal. Similarly, the linear system in $\mathrm{NR}^{\mathrm{W}\mathbb{R}}_{red}$ is equivalent to that of $\mathrm{NR}^{\mathrm{W}\mathbb{R}}$. Thus, only data concerning $\mathrm{NR}^{\mathrm{RC}}$ are shown.

For each value of σ in the considered ranges, the number of converged runs and of required steps to convergence, and information about the singular values of the Jacobian are collected to gain insights about convergence behaviors [118].

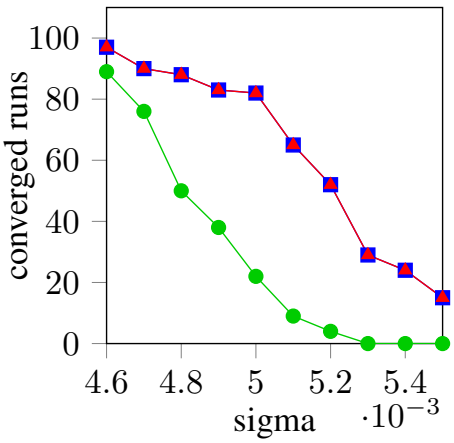
The singular values s of $\mathbf{J}$ correspond to the square root of the eigenvalues of $\overline{\mathbf{J}}^t\mathbf{J}$ and can be computed in a numerically stable way within MATLAB (through `svds`). As $\min(s)$ approaches 0 while $\max(s)$ increases, the conditioning number of $\mathbf{J}$ grows, producing an unreliable solution $\Delta\mathbf{X}$ of Equation (5.4) in NR. In addition, if many singular values of $\mathbf{J}$ are close to zero, then the correction $\Delta\mathbf{X}$ lies in a vector space with many almost parallel basis vectors. This implies that a small error in the computations can change considerably $\Delta\mathbf{X}$, so that the convergence of NR can go along an erratic path. Thus, the number of singular values that are smaller than a fixed threshold, i.e., $\#s < 0.01$ in the considered scenarios, represents a qualitative measure of the convergence behavior.

Initial Voltages

In the first convergence study, the initial voltages are randomly generated through a normal distribution with mean the voltages of the standard initial guess given in the MATPOWER file and variable σ.

In Fig. 5.1, on the left side, the convergence behavior for case6470rte is depicted and $\mathrm{NR}^{\mathrm{PC}}$ is clearly worse than the other strategies. On the right side, in non-converged runs, the Jacobian of $\mathrm{NR}^{\mathrm{PC}}$ has on average a greater number of small singular values and a bigger maximum singular value with respect to the other strategies. This behavior, characterizing most of the $\mathrm{NR}^{\mathrm{PC}}$ non-converged runs, implies that $\mathrm{NR}^{\mathrm{PC}}$ is more prone to follow erratic paths. The non-convergence of the other strategies can instead be linked to an oscillating behavior since the average $\#s < 0.01$ is small and the mean of $\max(s)$ remains close to 10^6 (an expected order of magnitude for $\max(s)$ of Jacobians related to transmission grids).

In Table 5.4, the total number of converged runs over the 1000 considered and the mean of the number of required steps to converge are shown. Except for case13659pegase, $\mathrm{NR}^{\mathrm{PC}}$ is in general the strategy that converges in a lower number of cases, while the other strategies have similar behaviors. Notice



	$min(s)$	$\#s<0.01$	$max(s)$
NR^{PC}	$3.06e-05$	$n.a.$	$2.97e+13$
NR^{RC}	$1.74e-03$	$n.a.$	$1.92e+06$
NR^{NV}	$1.72e-03$	$n.a.$	$1.71e+06$

Figure 5.1: Convergence behavior for case6470rte [3] with random initial voltages: the effect of changing σ on the number of converged runs (on the left) and the mean of data concerning singular values of the Jacobian when the strategies do not converge (on the right).

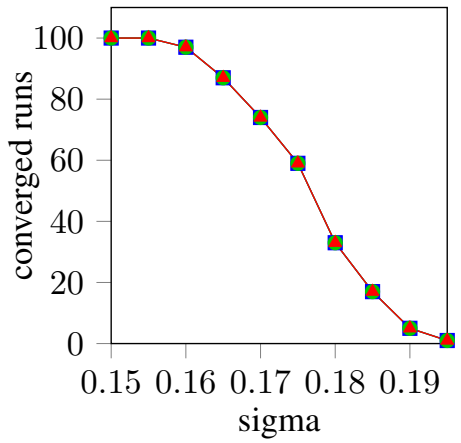
that, when NR^{PC} converges, it requires less steps than the other strategies.

Level of Loads

Instead of studying the strategies behavior with generic level of loads, a more unstable scenario of overloaded network is considered. Thus, active and reactive powers (P and Q) are increased by multiplying them for a random factor generated through the absolute value of a normal distribution with mean 1 and variable σ. When the electrical grid comprises thousands of nodes, even a change of 10% in the power factor can, depending how far the grid is from the nose point, promote convergence issues.

The common behavior of convergence is well represented in Fig. 5.2, where the collected data for case2869pegase are represented: on the left side it is shown that all the strategies converge in the same number of runs; the right side indicates that, as for the previous scenario, the singular values of the Jacobians in non-converged situations lead NR^{PC} to an erratic path and the other strategies to an oscillating behavior.

In Table 5.5 is clear that, in each case study, the total number of converged runs is the same for all strategies and the average number of steps required to convergence is reduced for NR^{PC} with respect to the other strategies.



	$min(s)$	$\#s<0.01$	$max(s)$
NR^{PC}	$2.41e-03$	6.27	$1.20e+12$
NR^{RC}	$1.01e-02$	0.67	$1.56e+07$
NR^{NV}	$4.10e-03$	1.04	$1.66e+05$

Figure 5.2: Convergence behavior for case2869pegase [3] with random initial voltages: the effect of changing σ on the number of converged runs (on the left) and the mean of data concerning singular values of the Jacobian when the strategies do not converge (on the right).

R/X Ratio

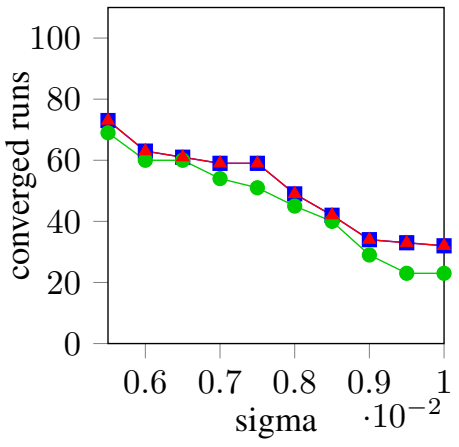
Finally, random R and X for each line are obtained, so that different values of the R/X ratio are tested. The random initial situations are generated with mean the values R and X of the MATPOWER file and variable σ.

As seen on the left of Fig. 5.3, related to case2383wp, generally NR^{PC} converges less than the other strategies. As in the previous scenarios, from the right side of Fig. 5.3, the behavior of non-converged runs is generally erratic for NR^{PC} and oscillatory for the other strategies.

In Table 5.6 is shown that NR^{PC} converges in a slightly minor number of runs and, again, requires generally less steps to reach convergence with respect to the other strategies.

5.5 Summary

Starting from the observation that the PFEs relate powers and voltages without explicitly referring to time, and the consequent possibility to choose the time scale of the analysis, triggered some interesting investigations. The implicit treatment of time allows to couple the PFEs to other equations, such as those characterizing the injection of real power produced by traditional or distributed generators, to form a model of the EI at the desired level of details. Other choices, different from those detailed in this chapter, can also



	$min(s)$	$\#s<0.01$	$max(s)$
NR^{PC}	$6.97e{-}03$	7.21	$3.93e{+}13$
NR^{RC}	$2.70e{-}02$	0.21	$6.11e{+}06$
NR^{NV}	$2.57e{-}03$	1.65	$6.23e{+}05$

Figure 5.3: Convergence behavior for case2383wp [3] with random initial voltages: the effect of changing σ on the number of converged runs (on the left) and the mean of data concerning singular values of the Jacobian when the strategies do not converge (on the right).

be possible. For instance, coupling the PFEs with a set of differential equations, resulting in a Differential Algebraic Equations system, can promote the study of voltage disturbance at high resolution; although an interesting topic, this is not addressed in this thesis. Among the possible couplings, the most important in the context of this thesis, excluding the one formalized by Equation (3.2), are the optimization problem expressed in Equation (3.4) and the one that will be introduced in Section 6.2.1. These couplings express the voltage control capabilities. It is then clear how important is, in the context of the thesis, to study efficient PFEs solution method.

As discussed in 5.1, the ability to couple the PFEs with other sets of equations come with a price: the complex formulation, directly linked to the physical laws governing electrical grids, is hard to manipulate from the analytical point of view (non-holomorphic set of equations) and difficult to directly handle inside a computer. The two straightforward approaches are: 1) fix two real coordinate systems, one for powers and the other for voltages, decompose the PFEs in terms of these coordinates and then solve the resulting set of real equations (Section 5.2.1); 2) double the number of complex equations and unknowns, and delegate the translation towards real variables to the most inner subroutine of the solution method (Section 5.2.2). In both cases, the NR method is employed, but in different analytical and geometric contexts. A third approach is then investigated in details in Section 5.3, namely the Wirtinger-based one. The novelty of the contribution is not in the high level idea of using Wirtinger calculus, but in observing the sym-

metries it produces and exploiting them to introduce: a new complex-based strategy, defined by Equation (5.26), a rewriting that ends up producing the same strategy as the one described in Section 5.2.1, and a reduced form, characterized by Equation (5.45). All the mentioned approaches have been studied in abstract terms and investigated from a computational point of view.

Working with the NR method, a relevant observation is that some of the described strategies have the same behavior in terms of convergence, others behave differently. Theoretical computational costs analysis and experimental results are employed to compare the strategies from the performance point of view.

None of the examined strategies resulted to be the best considering all the metrics, and then a trade-off between good convergence behavior and computational cost has to be found according to the requirements of a specific context. In particular, as will be discussed in Section 6.2.1, when dependability-related properties are evaluated through simulation, performance is a key feature, but it cannot compromise the possibility of estimating the system state due to not convergent sequence of iterations.

Table 5.3: Elapsed time in *ms* for evaluating **F**, **J** and solving the linear system (*sol*) at each step of each strategy

case	*ms*	$\mathrm{NR}^{\mathrm{PC}}$	$\mathrm{NR}^{\mathrm{RC}}$	$\mathrm{NR}^{\mathrm{NV}}$	$\mathrm{NR}^{\mathrm{W}\mathbb{C}}$	$\mathrm{NR}^{\mathrm{W}\mathbb{R}}$	$\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$
1354peg.	**F**	0.25	0.44	0.20	0.23	0.24	0.25
	J	1.33	1.35	1.90	1.00	1.76	2.42
	sol	13.82	19.58	15.97	20.68	16.25	15.48
1888rte	**F**	0.30	0.52	0.29	0.33	0.30	0.32
	J	1.75	1.68	2.20	1.20	2.15	3.20
	sol	17.61	27.05	18.73	29.45	18.73	23.11
2000tex	**F**	0.33	0.54	0.32	0.38	0.30	0.34
	J	1.84	1.80	2.53	1.32	2.35	3.34
	sol	24.47	57.02	26.92	61.96	27.49	32.74
2383wp	**F**	0.42	0.68	0.36	0.40	0.38	0.38
	J	2.18	2.15	2.94	1.56	2.75	4.29
	sol	26.60	48.66	33.01	51.41	32.83	33.56
2746wp	**F**	0.50	0.72	0.44	0.43	0.43	0.49
	J	2.55	2.40	3.42	1.73	3.21	4.63
	sol	26.93	61.03	34.89	51.34	32.80	36.36
2868rte	**F**	0.44	0.74	0.44	0.48	0.45	0.50
	J	2.67	2.52	3.26	1.81	3.17	4.78
	sol	31.64	41.54	29.13	45.26	28.23	34.53
2869peg.	**F**	0.59	0.81	0.47	0.49	0.41	0.57
	J	2.97	2.81	4.05	2.07	3.70	5.38
	sol	33.77	48.35	36.78	50.38	37.74	40.38
3375wp	**F**	0.60	0.85	0.50	0.51	0.49	0.57
	J	3.25	2.91	4.12	2.11	3.78	5.60
	sol	40.99	69.84	44.83	74.48	44.69	44.50
6470rte	**F**	1.13	1.59	0.93	0.91	0.99	1.09
	J	6.69	6.01	8.07	4.11	7.31	11.02
	sol	86.51	117.52	94.86	122.87	87.90	94.44
9241peg.	**F**	1.54	2.62	1.39	1.43	1.41	1.74
	J	10.88	10.50	15.98	7.33	12.63	18.86
	sol	136.65	191.70	156.40	210.77	157.60	165.84
13659peg.	**F**	2.03	3.60	1.86	2.08	1.95	2.23
	J	15.48	15.27	22.57	10.33	18.05	27.00
	sol	185.66	261.80	213.61	273.05	225.92	213.63

Table 5.4: Total number of converged runs and mean of required steps to convergence with different initial voltages

case	σ		NR^{PC}	NR^{NV}	NR^{RC}
1354peg.	0.012-0.030	#conv	371	661	661
		#steps	7.12	9.70	9.70
1888rte	0.0048-0.0066	#conv	618	588	588
		#steps	8.47	14.92	14.92
2000tex	0.046-0.064	#conv	132	484	484
		#steps	7.97	11.51	11.51
2383wp	0.009-0.018	#conv	208	372	365
		#steps	7.98	11.95	11.95
2746wp	0.009-0.018	#conv	476	572	572
		#steps	7.48	8.65	8.65
2868rte	0.0036-0.0054	#conv	114	750	748
		#steps	8.36	13.95	13.94
2869peg.	0.014-0.023	#conv	287	556	556
		#steps	7.95	10.95	10.95
3375wp	0.005-0.014	#conv	422	656	656
		#steps	7.57	10.17	10.17
6470rte	0.0046-0.0055	#conv	288	625	625
		#steps	8.15	17.52	17.52
9241peg.	0.0050-0.0068	#conv	418	550	550
		#steps	8.30	8.97	8.97
13659peg.	0.0030-0.0057	#conv	711	519	519
		#steps	7.45	13.99	13.99

Table 5.5: Total number of converged runs and mean of required steps to convergence with different levels of loads

case	σ		NR^{PC}	NR^{NV}	NR^{RC}
1354peg.	0.25-0.43	#conv	755	755	755
		#steps	5.77	7.52	7.52
1888rte	0.0070-0.0088	#conv	565	565	565
		#steps	6.28	6.74	6.74
2000tex	0.150-0.177	#conv	529	529	529
		#steps	6.74	7.62	7.62
2383wp	0.400-0.445	#conv	670	670	670
		#steps	8.24	10.62	10.62
2746wp	0.51-0.60	#conv	596	596	596
		#steps	6.87	7.45	7.45
2868rte	0.0048-0.0057	#conv	677	677	677
		#steps	6.77	7.39	7.39
2869peg.	0.150-0.195	#conv	573	573	573
		#steps	7.13	9.76	9.76
3375wp	0.180-0.225	#conv	456	456	456
		#steps	6.33	7.30	7.24
6470rte	0.0050-0.0068	#conv	656	656	656
		#steps	5.73	6.45	6.45
9241peg.	0.090-0.099	#conv	683	683	683
		#steps	7.30	11.29	11.29
13659peg.	0.001-0.004	#conv	200	200	200
		#steps	6.50	7.32	7.32

Table 5.6: Total number of converged runs and mean of required steps to convergence with different R/X ratios

case	σ		NR^{PC}	NR^{NV}	NR^{RC}
1354peg.	0.0001-0.0019	#conv	577	586	586
		#steps	4.82	5.09	5.10
1888rte	0.0020-0.0065	#conv	392	399	399
		#steps	5.60	6.87	6.87
2000tex	0.001-0.010	#conv	479	497	497
		#steps	4.45	4.98	4.98
2383wp	0.0055-0.0100	#conv	454	504	505
		#steps	6.47	8.23	8.23
2746wp	0.0005-0.0095	#conv	626	642	642
		#steps	5.34	5.97	5.97
2868rte	0.0015-0.0060	#conv	413	432	432
		#steps	5.66	7.45	7.45
2869peg.	0.0001-0.0019	#conv	494	519	519
		#steps	6.97	6.37	6.39
3375wp	0.0005-0.0050	#conv	604	619	619
		#steps	4.84	5.05	5.08
6470rte	0.0010-0.0055	#conv	478	483	483
		#steps	5.19	6.38	6.39
9241peg.	0.0001-0.0010	#conv	540	548	547
		#steps	7.00	6.35	6.35
13659peg.	0.0005-0.0050	#conv	459	451	451
		#steps	6.41	8.19	8.19

Chapter 6

Enhanced Smart Grid model

In this chapter the enhanced SG model will be presented, stressing out how both architecture design and EI state estimation have been influenced by the analysis discussed in Chapters 4 and 5.

Before discussing the details of the new developed models, some high level considerations are drawn, helpful to relate the basic SG framework in Chapter 3 with the enhanced SG framework. As already discussed, the need for improving the basic modeling approach was to improve in efficiency, thus triggering investigations and implementations of new supporting techniques, presented in the previous two Chapters. In this respect, the best solutions have been selected and exploited in the enhanced framework, as detailed in the subsequent sections. Their embedment implied substantial changes in the architecture of the overall enhanced framework, also reflected in the individual SAN models composing it. In particular, the adoption of the *Dependency-Aware Replication* (*DARep*) approach instead of the *State-Sharing Replication* (*SSRep*) approach, changed the structure of the overall model from a hierarchical composition of submodels through Rep and Join operators (as in Figure 3.8) in a flat composition of submodels through a Join operator (as in Figure 6.2). This resulted in a great advantage in terms of modular expansion of the overall framework and in easier management of the submodels connections. Therefore, the new SAN models, although implementing the same SG logical architecture described in Section 3.1, implement structure and behavior of modeled components exploiting an higher modularity.

Moreover, adoption of the new techniques for the PFEs implied updates algorithms in the pertinent SAN models, again differing from the corresponding ones in the basic modeling framework. As a result of these changes, the two frameworks, although showing the same power in representing and assessing SG scenarios, are significantly different in the specific SAN submodels.

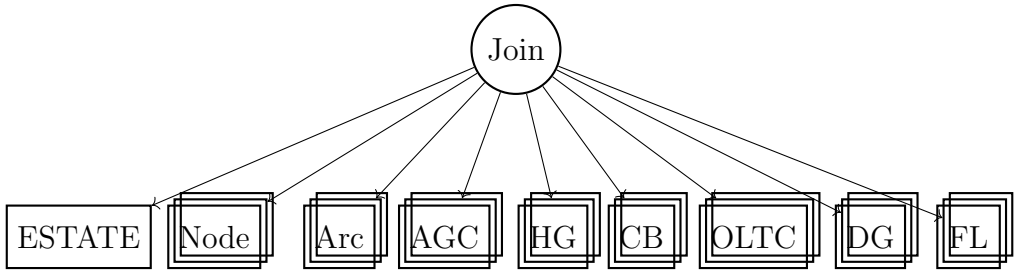


Figure 6.1: Enhanced SG model architecture.

6.1 Model architecture

Adopting *DARep* as the non-anonymous replication mechanism, the overall SG model is obtained as through a single *Join* of all the template models replicas. Figure 6.1 illustrates at very high level the resulting SG model architecture. Boxes in the figure represent the SG components for which template models are defined. They are almost the same as those described in Chapter 3, unless the MCS (AGC in the figure) that is now distributed to better represent SGs comprising a set of Microgrids (as shown in Figure 2.1b) each one with its own control. Multiple boxes enclosing the same component name represent the fact that a *population* of the same component type needs to be included in the model.

In order to define the architecture, the modeler has to write a XML file where all the template SAN models are listed, specifying which places are dependency-aware and their topologies. For the enhanced SG model a "pre-processing" phase has been considered: the XML is automatically generated. In particular, the case study is designed writing a description file in the MATPOWER format [64]; all the electrical parameters of the grid under analysis are encoded in the standard format whereas additional information about DGs, FLs, etc, and dependability related parameters, such as component MTTF, are written in newly designed fields. A template XML file, together with the mentioned description file, are the starting points to produce the architecture XML file, generated using the Perl Text::Xslate::Syntax::Kolon package [60].

In this way, the modeler can focus only on SAN templates design, as described in Section 6.2, and the description file, written in a "domain specific" format. The *DARep* script is in charge of the final model generation: a *Join* model and several atomic models, one for each replica of template models, are produced as described in Section 4.5.3 and depicted in Figure 6.2.

In the enhanced model, the EI is represented taking advantage of bus and branch template models, together with an additional template model called

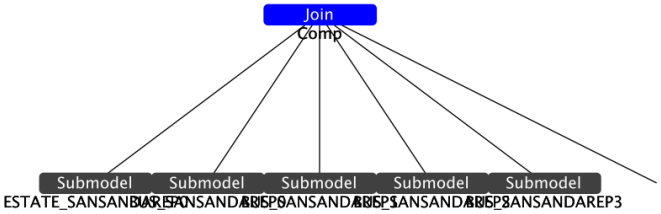


Figure 6.2: Example of Möbius *Join* produced by *DARep* for the enhanced SG model. Only a portion of the *Join* is depicted because it comprises hundreds of submodels.

ESTATE_SAN that is responsible for the state estimation procedure. The replica models share dependency-aware places according to the electrical grid topology, thus exploiting the low degree of interconnection among nodes.

6.2 Component modeling

In this section the template models are presented. Notice that the behavior of each template can be fully understood considering the fact that it is replicated and instantiated following the actual EI and ICT topologies. A preliminary distinction between template model roles can be done: there are template models that encode specific functionalities of components together with EI topology representation, in the following called *structural* template models, and template models that encode only behavioral aspects of components. The enhanced SG model comprises the following structural template atomic models:

- AGC_SAN, depicted in Figure 6.5, that is responsible for the MVGC failure and recovery,
- BUS_SAN, depicted in Figure 6.7, that represents a generic bus within the EI,
- BRANCH_SAN, depicted in Figure 6.6, the represents a generic power line,
- ESTATE_SAN, depicted in Figure 6.3, is responsible for the EI state estimation and the voltage control policy formalized by Algorithm 1. The *DARep* approach replicates only once ESTATE_SAN, producing ESTATESANSANDAREP0 depicted in Figure 6.4.

In addition, the following non-structural template models are considered:

- CB_SAN, depicted in Figure 6.8, represents the behavior and failure model of CBs. Once instantiated, a CB model can share information with the bus model where it is attached on. Information is shared also with ESTATESANSANDAREP0,

- DG_SAN, depicted in Figure 6.10, models DGs and, as for CBs, once instantiated can share information only with one bus model, the one it is attached to, and ESTATESANSANDAREP0,

- HG_SAN, depicted in Figure 6.12, models the interaction of the SG under analysis with other SGs,

- FL_SAN, depicted in Figure 6.11, models FLs similarly to CBs and DGs,

- OLTC_SAN, depicted in Figure 6.9, models OLTCs. An OLTC is attached to a specific power line, so the instantiated OLTC model shares information with the corresponding instance of BRANCH_SAN and with ESTATESANSANDAREP0.

Notice that NFLs are not considered at the SAN level but are encoded directly inside the description file following the MATPOWER format.

6.2.1 Description of ESTATE_SAN

The ESTATE_SAN SAN is the most important template atomic model considered within the enhanced SG model. In fact, once instantiated to form ESTATESANSANDAREP0, it shares places with all the other SAN template instances and it is responsible for the PFEs solution and the optimization problem. ESTATE_SAN is divided in four main parts:

- EI state estimation, comprising the places *Go* and *UpdateEState*, the instantaneous action *update*, the input gate *inUpdate* and the output gate *NewES*;

- scheduled and on-demand voltage control, comprising the places *SCHED* and *VOLT_CTRL*, the deterministic action *ScheduledVoltageCTRL*, the instantaneous activity *fireVoltageCTRL*, and the output gates *PromoteVoltageCTRL* and *VoltageCTRL*;

- information exchange, comprising the places *DG_CTRL_P*, *FL_CTRL_P*, *HG_CTRL_P*, *HG_PMax*, *HG_PMin*, *HG_QMax*, *HG_QMin*, *OLTC_T*, *CB_q*, *AGC_Status*, *BUS_Status* and *Branch_Status*;

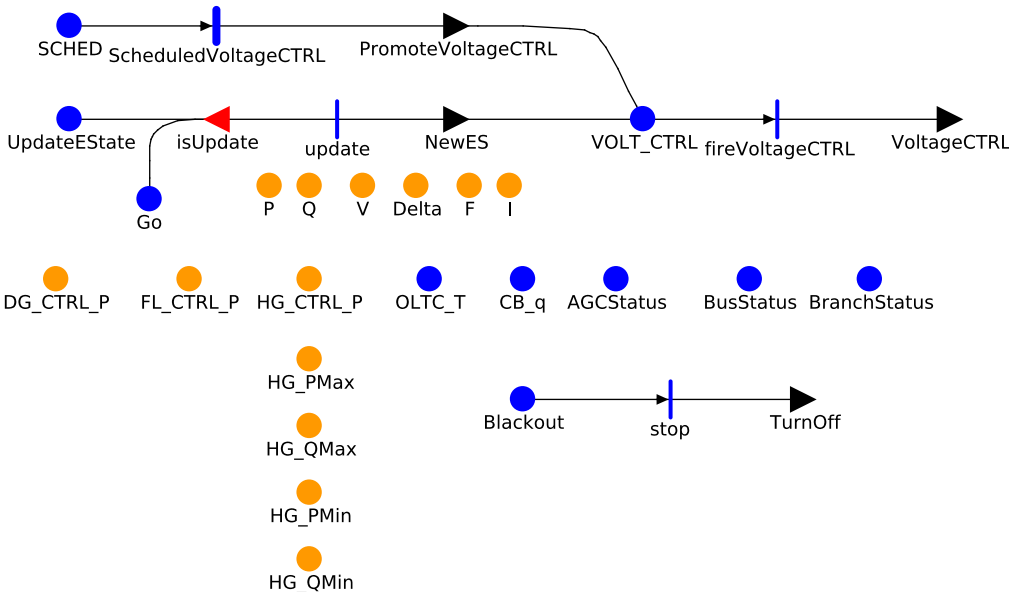


Figure 6.3: ESTATE_SAN.

- blackout management, comprising the place *Blackout*, the instantaneous action *stop* and the output gate *TurnOff*.

The place *Go* is shared among all template instances and at the beginning of the simulation contains one token. The simulation can proceed only if *Go* is not empty. Whenever an EI parameter changes, a token is inserted within place *UpdateEState*, also shared among all SANs, so that enabling the firing of *update*. Being instantaneous, *update* fires immediately and the PFEs solution routine, written inside *NewES*, is called. Consequently, the values inside *P*, *Q*, *V*, *Delta*, *F* (the power flowing in each branch) and *I* (the current flowing in each branch) are updated. This mechanism guarantees that each EI parameter change is an atomic operation and that the time spent by the simulator solving the PFEs does not affect the simulation time. In this way the performability measures described in Section 6.3, defined on the simulation time, are well defined. The EI parameters are stored within the places shared with the template components SANs, such as *DG_CTRL_P*, and whenever a failure or another event happens the places are updated.

The *fireVoltageCTRL* can fire only if there is a token inside *VOLT_CTRL*, and this can happen on a scheduled basis, according to the deterministic time specified in *ScheduledVoltageCTRL*, or on-demand, i.e., if there are buses with voltage out of bounds.

Within ESTATESANSANDAREP0 the instances of *AGC_Status*, *Bus_Status* and *Branch_Status* keep track of the SG topology, containing one

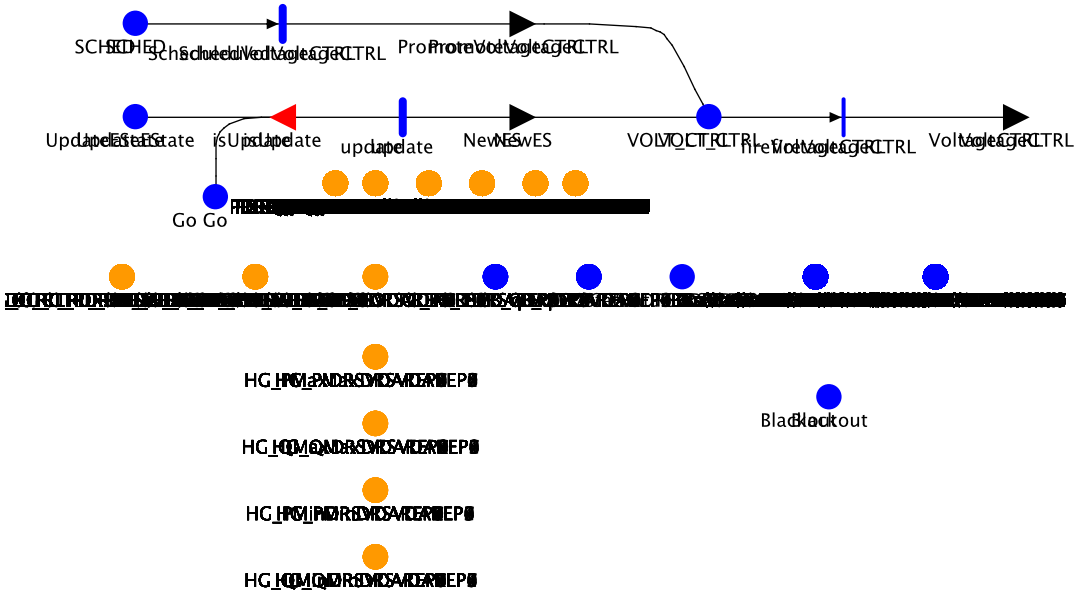


Figure 6.4: ESTATESANSANDAREP0. Here the Dependency-Aware places are superimposed.

token if active and zero tokens if failed. As discussed in Chapter 5, if the PFEs have no solution then the entire SG is considered in blackout, i.e. one token in placed inside *Blackout* and after the firing of *stop* the token inside *Go* is removed, causing the arrest of the simulation.

The numerical experiments presented in Chapter 5 guided the implementation of the PFEs solution routine following the Randell's *recovery block* paradigm [119]. In particular, after having observed that $\mathrm{NR}^{\mathrm{PC}}$ is fast but not always convergent, $\mathrm{NR}^{\mathrm{RC}}$ is almost as fast as $\mathrm{NR}^{\mathrm{PC}}$ but with a better convergence profile and $\mathrm{NR}^{\mathrm{NV}}$ is slow but sometimes converges when the other does not, the following scheme is adopted:

1. run $\mathrm{NR}^{\mathrm{PC}}$, check if there is convergence: if yes then stop, if not then roll back the EI state and continue,

2. run $\mathrm{NR}^{\mathrm{RC}}$, check if there is convergence: if yes then stop, if not then roll back the EI state and continue,

3. run $\mathrm{NR}^{\mathrm{NV}}$, check if there is convergence: if yes then stop, if not then stop in a failure state.

Notice that the PFEs routine is called several times inside the optimization problem solution and then a compromise between efficiency and accuracy is needed. In order to produce reliable implementations of the PFEs routines,

the code for NR^{PC} distributed together with MATPOWER [64] has been translated in C++, using the Octave [120] API.

The optimization problem that guides the voltage control capabilities is also revisited with respect to the one presented in Equation (3.4). In particular, after having observed in Section 3.7 that the unconstrained optimization of Equation (3.4) is inefficient, a new multi-objective optimization problem has been studied. The control actions considered for the enhanced SG model are the same as those discussed in Chapter 3, i.e., OLTC tap change, active power curtailment on DG, load shedding on FL and CB bank management, and also the voltage control policy of the MVGC remains the same, i.e., the one formalized in Algorithm 1. So, in this section the discussion of Section 3.3.1 is still valid and it is not repeated. Nevertheless, the way control actions are implemented is different.

The voltage control was designed as an unconstrained optimization problem with the unique objective function written in Equation (3.4). The unique objective function was the weighted sum of many pieces of the form

$$\big|\text{reference value} \;-\; \text{candidate value}\big|,$$

and the weights where chosen according to a predefined ordering so that reflecting different priorities. Minimizing the objective function implies minimizing each addend, according to its weight. Unfortunately, the addends are non-smooth functions because of the absolute value and the objective function is mixed-integer because it comprises discrete values, such as OLTC tap (tap), and continuous values, such as P^{inj}; thus, an heuristic approach was employed. The optimization problem of Equation (3.4) was solved using the *MO* paradigm offered by the metaheuristics tool ParadisEO [121]. In particular, a population of size 800 was considered.

Hence, a complete re-design has been carried out: instead of a single objective function, global optimization problem, the enhanced model considers the following multi-objective functions, constrained optimization problem.

objective functions:

$$\underset{\text{tap}_h}{\text{minimize}} \quad |^{\text{old}}\text{tap}_h - \text{tap}_h| \quad \text{for } h \in \text{OLTC} \tag{6.1}$$

$$\underset{\text{bank}_h}{\text{minimize}} \quad |^{\text{old}}\text{bank}_h - \text{bank}_h| \quad \text{for } h \in \text{CB} \tag{6.2}$$

$$\underset{P_h^{\text{inj}}}{\text{minimize}} \quad |P_h^{\text{av}} - P_h^{\text{inj}}| \quad \text{for } h \in \text{DG} \tag{6.3}$$

$$\underset{P_h^{\text{inj}}}{\text{minimize}} \quad |^{\text{old}}P_h^{\text{inj}} - P_h^{\text{inj}}| \quad \text{for } h \in \text{DG} \tag{6.4}$$

$$\underset{P_h^{\text{dis}}}{\text{minimize}} \quad |P_h^{\text{dem}} - P_h^{\text{dis}}| \quad \text{for } h \in \text{FL} \tag{6.5}$$

$$\underset{P_h^{\text{dis}}}{\text{minimize}} \quad |^{\text{old}}P_h^{\text{dis}} - P_h^{\text{dis}}| \quad \text{for } h \in \text{FL} \tag{6.6}$$

$$\underset{V_i}{\text{minimize}} \quad \mathbb{1}_{V_i > V_i^{max}}\Big(1\text{p.u.} + V_i - V_i^{max}\Big) + \\ + \mathbb{1}_{V_i < V_i^{min}}\Big(1\text{p.u.} + V_i^{min} - V_i\Big) \quad \text{for } i \in \text{PQ} \tag{6.7}$$

subject to:

$$S_i = E_i \sum_{k \simeq i} \overline{Y_{ik}^{\text{bus}}}\,\overline{E}_k \quad \text{for } i = 1, \ldots, n \tag{6.8}$$

$$P_{\text{ref}}^{\min} \leq P_{\text{ref}} \leq P_{\text{ref}}^{\max} \tag{6.9}$$

$$\text{Q}_{\text{ref}}^{\min} \leq \text{Q}_{\text{ref}} \leq \text{Q}_{\text{ref}}^{\max} \tag{6.10}$$

where:

$$|E_{f_h}| : |E_{t_h}| = \Big(1 + \frac{1.25}{100}\text{tap}_h\Big) : 1 \tag{6.11}$$

$$S_i = P_i + \jmath \text{Q}_i \quad \text{for } i = 1, \ldots, n \tag{6.12}$$

$$P_i = \mathbb{1}_{i \in \text{DG}} P_i^{\text{inj}} - \mathbb{1}_{i \in \text{FL}} P_i^{\text{dis}} - \mathbb{1}_{i \in \text{NFL}} P_i^{\text{load}} \tag{6.13}$$

$$\text{Q}_i = \mathbb{1}_{i \in \text{DG}} \text{Q}_i^{\text{inj}} - \mathbb{1}_{i \in \text{FL}} \text{Q}_i^{\text{dis}} + \\ - \mathbb{1}_{i \in \text{NFL}} \text{Q}_i^{\text{load}} + \mathbb{1}_{i \in \text{CB}} \text{Q}_{cb} \cdot \text{bank}_i \tag{6.14}$$

$$\text{Q}_i^{\text{inj}} = \frac{\sqrt{1-\rho^2}}{\rho} P_i^{\text{inj}} \tag{6.15}$$

$$\text{Q}_i^{\text{dis}} = \frac{\sqrt{1-\rho^2}}{\rho} P_i^{\text{dis}} \tag{6.16}$$

$$\text{tap}_i \in [-5, 5] \cap \mathbb{Z} \tag{6.17}$$

$$\text{bank}_i \in [0, 5] \cap \mathbb{Z} \tag{6.18}$$

$$P_i^{\text{inj}} \in [0, P_i^{\text{av}}] \tag{6.19}$$

$$P_i^{\text{dis}} \in [0, P_i^{\text{dem}}] \tag{6.20}$$

Equation (6.11) describes the proportion between voltages at the two ends of OLTC on line h. In particular, the voltage magnitude at the *from* end ($|E_{f_h}|$) and the *to* end ($|E_{t_h}|$) are related by a voltage drop of $1 + \frac{1.25}{100}\text{tap}_h$. When tap_h in increased by 1, the voltage drop increases by 1.25% of the nominal value of $|E_{f_h}|$. Thus, changes of tap_h have impact on Y^{bus}. Further details are in [3].

Equation (6.12) relates the complex power (S) to active power (P) and reactive power (Q). Equation (6.13) describes the equilibrium among the overall active power on the left hand side and P actually injected (P^{inj}), P actually dispatched to the flexible load (P^{dis}), and P dispatched to the inflexible load (P^{load}) on the right hand side. Considering the ρ as a given

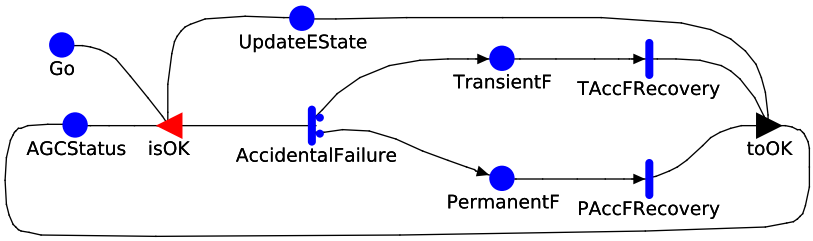


Figure 6.5: AGC_SAN.

constant, Equations (6.15) and (6.16) define Q in terms of P. Notice that P^{load} and Q^{load} are not related by a fixed ρ, in fact they are constants in the optimization problem. In Equation (6.14) is also considered the contribution of CBs. P^{inj} cannot exceeds P^{av}, and P^{dis} cannot exceeds P^{dem}, as described by Equations (6.19) and (6.20), respectively. tap and bank are discrete values bounded as in Equations (6.17) and (6.18).

Equation (6.8) imposes the power balance between supplied, demanded and lost in heat, namely Equation (5.2). Equations (6.9) and (6.10) impose bounds on P at the slack-bus (P_{ref}) and Q at the slack-bus (Q_{ref}).

The new optimization problem is solved using the *MOEO* paradigm offered by the tool ParadisEO [121]. In particular, a self-written implementation of the NSGA-II algorithm [122] is employed. The implementation of NSGA-II offered by ParadisEO does not tackle constrains, so it has been adapted to allow constraints-aware dominant sorting mechanism, already detailed in the original paper.

6.2.2 Description of AGC_SAN

The AGC_SAN SAN models the effect on the EI of ICT components failures/attacks. In the SAN depicted in Figure 6.5 only the part relative to failures is shown. In particular, the SG is divided in areas, each area is governed by an Area MVGC (AGC). When the exponentially distributed action *AccidentalFailure* fires one of two alternatives, with different probabilities, can happen: a transient failure of all the ICT in the area, i.e., a token within *TransientF*, or a permanent failure of all the ICT in the area, i.e. a token within *PermanentF*. After a transient failure, the AGC can return to the ok state only when the *TAccFRecovery* activity fires, whereas after a permanent failure a (long) deterministic period of time is employed in recovering the ICT in the area.

The core logic that describes the conditions under which an AGC can fail and the impact on its failure on the EI are defined within the input and out

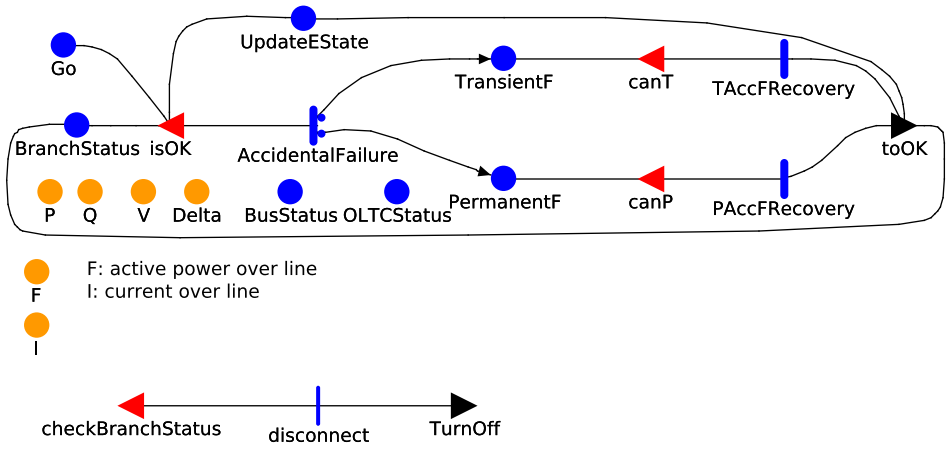


Figure 6.6: BRANCH_SAN.

gates *IsOk* and *toOk*, respectively.

The modeler can decide, according to the case study under analysis, different failure models just filling the mentioned gates with the appropriate C++ code.

As an example, consider the case in which the failure of the AGC implies that all the DGs in the area loose the connection with the area controller and then stop curtailing their powers, so that injecting too much, or curtailing all their powers, so that injecting 0. This kind of failure, as discussed in Sections 3.6.1 to 3.6.3, has a great impact on the electrical grid resilience.

6.2.3 Description of BUS_SAN and BRANCH_SAN

BUS_SAN and BRANCH_SAN have a very similar structure because, when instantiated, they represent the topology of the SG under analysis. In particular, *BusStatus* and *BranchStatus* contain one token if the corresponding bus or branch is working correctly and zero tokens otherwise. As for AGC_SAN, BUS_SAN and BRANCH_SAN can fail but there is an important difference: there are other electrical components attached on them. Thus, whenever a bus fails, all the electrical components attached to it have to fail instantaneously. This phenomenon is modeled with the instantaneous action *disconnect*, the output gate *TurnOff*, the input gate *checkBusStatus* in BUS_SAN and the input gate *checkBranchStatus* in BRANCH_SAN. If a the token inside *BusStatus* is removed then the corresponding instantaneous action *disconnect* fires and all the places *XStatus*, where *X* depends on the fact that BUS_SAN or BRANCH_SAN is considered, are set to zero. This mechanism guarantees that there is no electrical component alive if its support, a

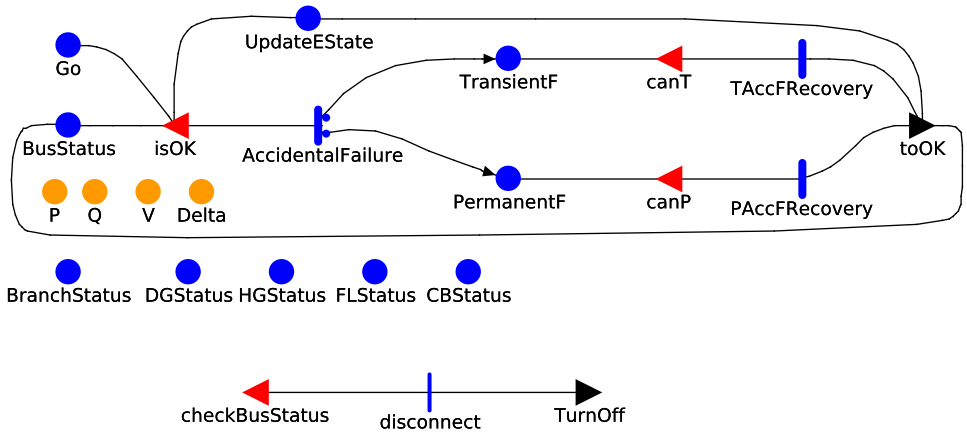


Figure 6.7: BUS_SAN.

bus or a branch, is dead.

The *DARep* approach permits to implement in a simple and clean way the disconnection mechanism: in fact, the places *XStatus* are dependency-aware SVs and are scanned using the *Deps*() functions. Notice that disconnecting a bus or a branch can produce islands in the electrical grid and then a blackout because, as mentioned in Chapter 5, MATPOWER is not capable to solve the PFEs in this case.

6.2.4 Description of CB_SAN and OLTC_SAN

CB_SAN and OLTC_SAN are similar to BUS_SAN and BRANCH_SAN except for the presence of additional information. In particular, CB_SAN comprises the place *CB_q* that, when the template model is instantiated, stores as many tokens as active banks in the corresponding CB. The number of tokens in *CB_q* can change only due to the following events:

- a firing of the voltage control in ESTATE_SAN, and in this case no actions are taken by CB_SAN,
- a failure of the AGC, and in this case the policy written within the output gate *ChangeCB_q* is followed.

Notice that the failure of AGC can have different impacts on different CBs, according to the *Index*() used in the instantiated CB_SAN. The OLTC_SAN is almost identical to the OLTC_SAN, of course it is attached to a branch instead of a bus.

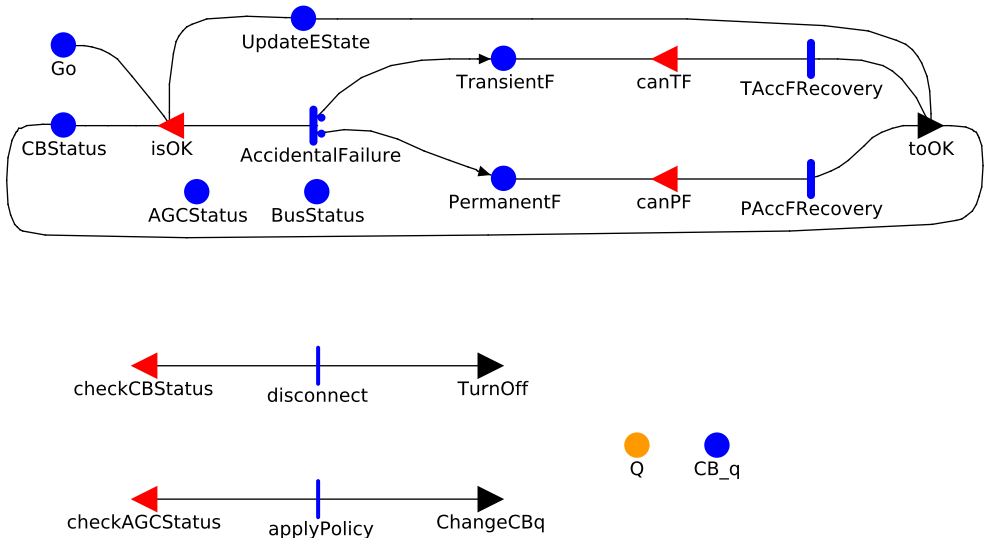


Figure 6.8: CB_SAN.

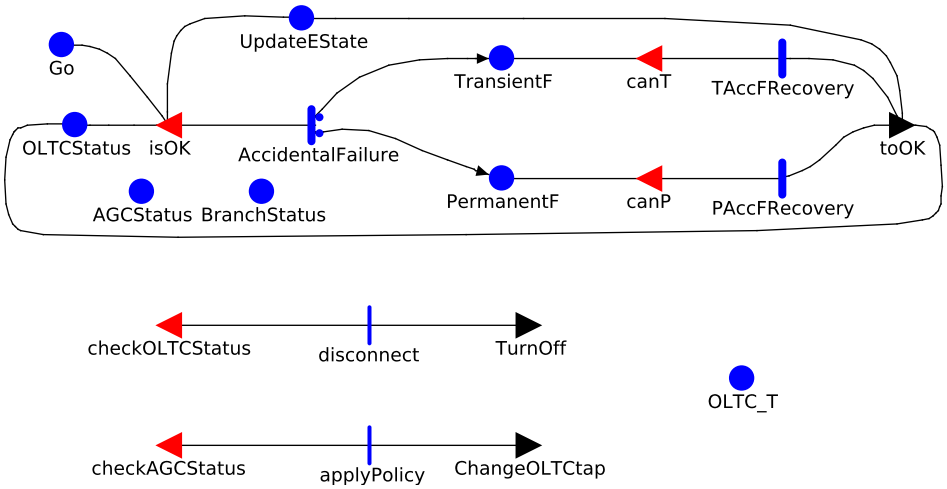


Figure 6.9: OLTC_SAN.

6.2.5 Description of DG_SAN and FL_SAN

DG_SAN and FL_SAN differ from the component templates previously described because DGs and FL, in addition to the mechanism employed in the other SANs, are characterized by the presence of P_i^{av}, P_i^{inj}, P_i^{dem} and P_i^{dis}. In particular, as already mentioned in Section 6.1, the modeler writes a description file where DGs and FLs power profiles are defined. These profiles are translated in static (constant) C++ vectors that represent the values that

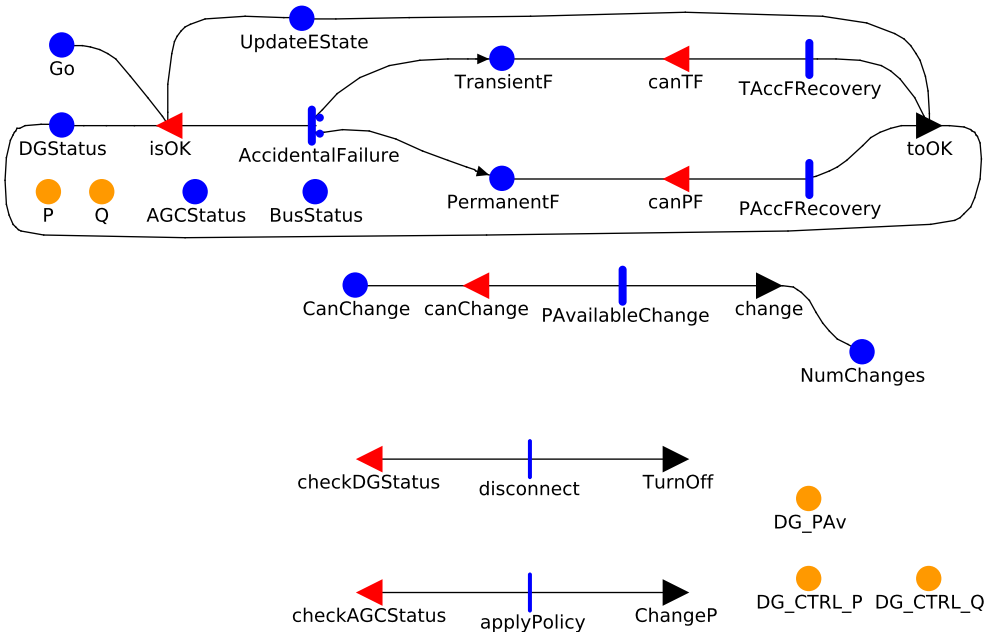


Figure 6.10: DG_SAN.

P_i^{av} and P_i^{dem} can assume, for DGs and FLs respectively.

In DG_SAN the mechanism responsible for the update of P_i^{av} is implemented by the places *CanChange* and *NumChanges*, the uniformly distributed action *PAvailableChange*, the input gate *canChange* and the output gate *change*. The i-th instance of DG_SAN maintains a local index inside *NumChanges* that indicates the element of the P_i^{av} array is currently stored in *DG_PAv*. When the action *PAvailableChange* fires the local index is incremented and a new value of P_i^{av} is stored in *DG_PAv*. In this way, the active power available at each DG is a stochastic process where the stochasticity is not in the values, that are predetermined, but in the time. The value of *DG_PAv* is also transcribed in *DG_CTRL_P* and a token is inserted in *UpdateEState*, so instantaneously the ESTATESANSANDAREP0 SAN check if its is an acceptable value. If it is not, i.e. if – after having solved the PFEs – there are out of bounds voltages, the optimization of the voltage control is launched and the curtailed power is written in *DG_CTRL_P*. This mechanism guarantees that both the instance of DG_SAN and ESTATESANSANDAREP0 are aware of the available power and the actually injected power.

FL_SAN is almost identical to DG_SAN, except for the fact that the update mechanism manages P_i^{dem} and P_i^{dis}, working on *FL_Preq* and *FL_CTRL_P*, instead of P_i^{av} and P_i^{inj}.

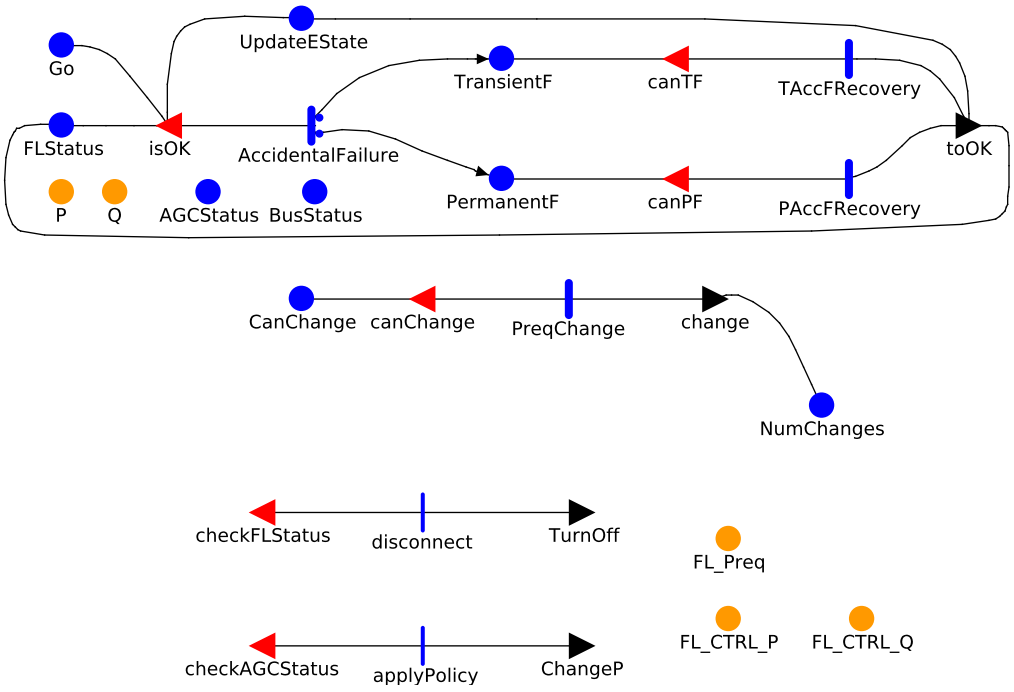


Figure 6.11: FL_SAN.

6.2.6 Description of HG_SAN

The template SAN model depicted in Figure 6.12 is responsible for the interaction between the SG under analysis and the other SGs.

At the moment, within the model only interactions at the EI level are considered, so there is no information exchange and the ICT is not involved. Indeed, the power exchange between one SG and the other SGs is an important feature of the distribution grid as it is foreseen in an imminent future, as depicted in Figure 2.1b, so it is fully modeled. The SG under analysis is modeled in great details, as discussed in the previous sections, whereas the other SGs are abstracted away, considering them as traditional generators (HGs).

During time, the amount of power required by the SG under analysis can change and, as for DGs and FLs, this is modeled considering a static vector of powers and a mechanism to read its values according to a stochastic process. In particular, the timed action *PQCChange* is uniformly distributed over an interval of fixed size, the place *Incr* is incremented whenever *PQCChange* fires and the input gate *PQConstrChange* updates the active power the SG can ask to its neighbours, writing on the place *HG_CTRL_P*. The HGs are modeled in ESTATE_SAN as *PV*-bus so the amount of active power is determined by the profile specified within the static vector, whereas the reactive

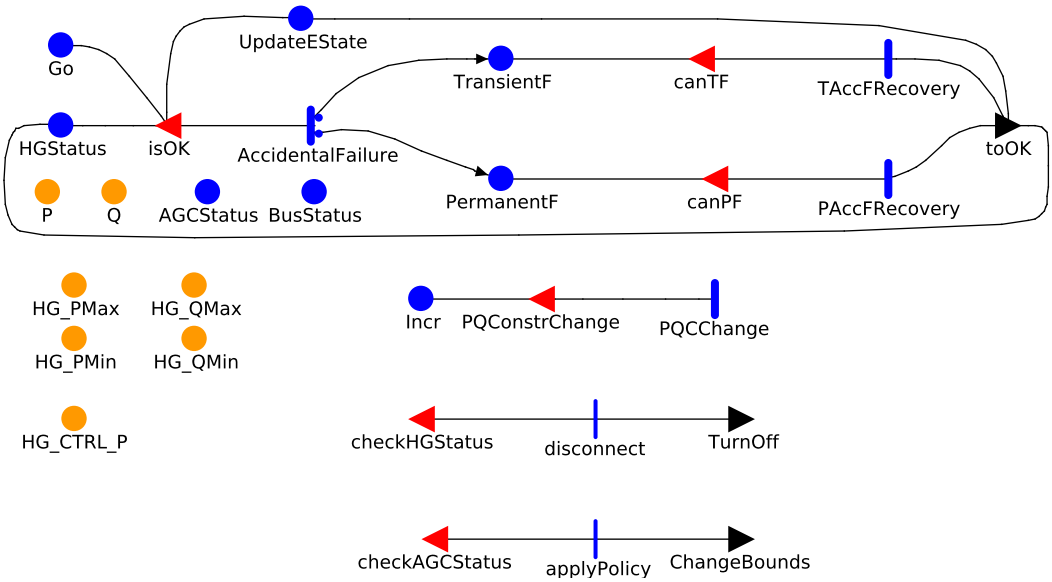


Figure 6.12: HG_SAN.

power is determined by the PFEs solution. In order to consider a dynamic interaction among SGs, the maximum and minimum amount of active and reactive powers can also change during time (modeled inside the same input gate *PQConstrChange*). Then it can happen that the SG under analysis requires/injects to the other SGs more reactive power than allowed, so reducing the QoS of the system.

6.3 Performability measures

Considering the formalism described in Section 2.5, and in particular Equations (2.1) and (2.2), in this section relevant performability measures are listed. All the places and actions mentioned so far can be employed to define performability measure, here only few examples are explicitly described. As already mentioned, the analysis considered in the thesis are mostly based on a period of time of one day. In addition to the measures defined in Sections 3.6.1 to 3.6.3, for Section 6.4 the following instantaneous and cumulated measure are considered.

- The probability that, during the day under analysis, the SG experienced a blackout. If the NR method does not converged during the solution of the PFEs then the system is considered in blackout, so – following the notation of Equation (2.1) – it is possible to formalize the

probability of blackout as in Equation (6.21):

$$\mathbb{P}^{\text{blackout}} = C(a)\, I^{a}_{\text{day}} \tag{6.21}$$

where $C(a)$ is non-zero only if a is the action stop in ESTATE_SAN, as formally described in Equation (6.22).

$$C(a) = \begin{cases} 1 \text{ if } a = \text{stop}, \\ 0 \text{ otherwise.} \end{cases} \tag{6.22}$$

- Considering FLs, after having noticed that the integral over time of active power is energy, formally

$$E_{\text{day}} = \int_0^{\text{day}} P(t)\, dt,$$

the amount of satisfied demand can be formalized as

$$E^{i}_{\text{sat}} = \int_0^{\text{day}} P^{\text{dis}}_{i}(t)\, dt, \tag{6.23}$$

whereas the amount of required energy is

$$E^{i}_{\text{req}} = \int_0^{\text{day}} P^{\text{dem}}_{i}(t)\, dt. \tag{6.24}$$

These measure can be expressed following the notation of Equation (2.2) as in Equation (6.25)

$$E^{i}_{\text{sat}} = \sum_{s} R^{i,\text{sat}}(s)\, J^{s}_{[0,\text{day}]}, \tag{6.25}$$

where $R^{i,\text{sat}}_{[0,\text{day}]}(s)$ is defined in Equation (6.26), in which $P^{\text{dis}}_{i}(s)$ is the active power dispatched to the i-th FL, when the stochastic process in in state s.

$$R^{i,\text{sat}}_{[0,\text{day}]}(s) = P^{\text{dis}}_{i}(s). \tag{6.26}$$

In particular, Equation (6.26) is defined on the basis of the fact that $P^{\text{dis}}_{i}(s)$ is stored within the place *FL_CTRL_P* in FL_SAN.

- The percentage of energy satisfaction can be considered a measure of QoS, as in Equation (6.27).

$$\text{Sat}^{i} = \frac{E^{i}_{\text{req}}}{E^{i}_{\text{sat}}}. \tag{6.27}$$

Similarly, it is possible to define, for DGs, the amount of energy that has been curtailed.

- Concerning the interaction among SGs, it is of particular importance to establish the quantity of reactive power Q that exceeds the upper or lower limit, *HG_QMax* and *HG_QMin* in Figure 6.12 respectively, during one day. There is no consensus among field experts [123] on how to measure the "complex energy", so here the simplest measure possible is considered, i.e., cumulating the out of bounds Q during one day:

$$E^{\text{ub}}_{\text{HG}} = \sum_{i \in \text{HG}} \int_0^{\text{day}} \left| \text{Q}_i - \text{HG\_QMax} \right| \mathbb{1}_{\text{Q}_i > \text{HG\_QMax}} \, dt, \tag{6.28}$$

  that can be easily expressed in terms of Equation (2.2) posing

$$R^i_{[0,\text{day}]}(s) = \begin{cases} \text{Q}_i > \text{HG\_QMax if } \text{Q}_i > \text{HG\_QMax in the marking } s, \\ 0 \text{ otherwise,} \end{cases} \tag{6.29}$$

  defined on the instance of HG_SAN attached to the i-th bus, and considering

$$R_{[0,\text{day}]}(s) = \sum_{i \in \text{HG}} R^i_{[0,\text{day}]}(s). \tag{6.30}$$

  Similarly, $E^{\text{lb}}_{\text{HG}}$ can be defined.

6.4 Case study

In order to test the performance of the enhanced SG model, a modular distribution grid has been chosen as the basis for the case study. The kernel of the EI is the *case33bw* distributed together with MATPOWER [3], depicted on the left side of Figure 6.13, here considered as one area of the SG. The EI of the SG under analysis has n_a areas, each comprising 33 buses. In the experiments, $n_a = 1, \ldots, 30$, so that the EI can hat at most 990 buses. Areas are interconnected by power lines. For the case $n_a = 2$, 9 power lines are introduced, as depicted in Figure 6.13 where $n_a = 2$. In addition to the NFL already present in *case33bw*, other electrical components are considered, in particular:

- each area has its own OLTC, placed on the branch between bus $i_a \cdot 33+1$ and bus $i_a \cdot 33 + 2$, for $i_a = 0, \ldots, n_a - 1$,

- each area has its own CB, placed on bus $i_a \cdot 33+7$, for $i_a = 0, \ldots, n_a - 1$,

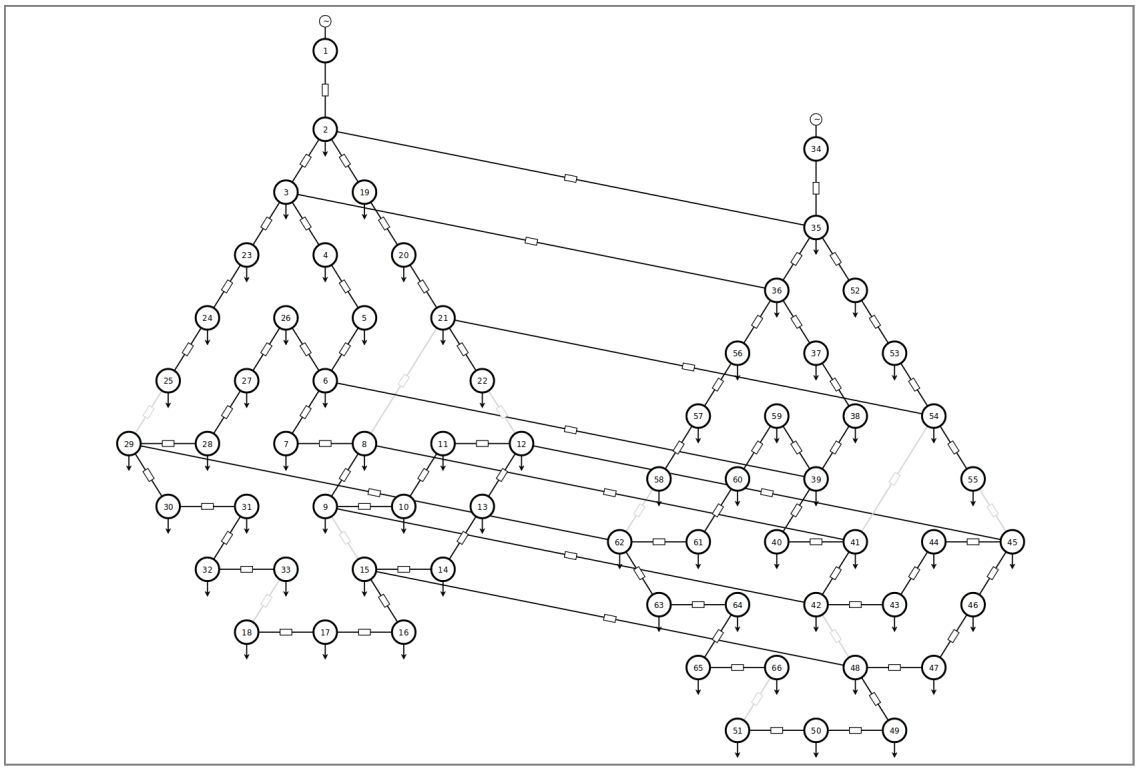


Figure 6.13: Structure of the EI for the scenario in which there are two areas. This picture has been produced using STAC [4].

- each area has a DG on bus $i_a \cdot 33 + 5$, for $i_a = 0, \ldots, n_a - 1$ with active power profile defined as *PW3MW* as show in Figure 6.14,
- each area has a DG on bus $i_a \cdot 33 + 11$, for $i_a = 0, \ldots, n_a - 1$ with active power profile defined as *PW500kW* as show in Figure 6.14,
- each area has a DG on bus $i_a \cdot 33 + 30$, for $i_a = 0, \ldots, n_a - 1$ with active power profile defined as *PW500kW* as show in Figure 6.14,
- each area has a FL on bus $i_a \cdot 33 + 18$, for $i_a = 0, \ldots, n_a - 1$ with active power profile defined as *FL500kW* as show in Figure 6.14,
- each area has a FL on bus $i_a \cdot 33 + 22$, for $i_a = 0, \ldots, n_a - 1$ with active power profile defined as *FL500kW* as show in Figure 6.14,
- each area has a FL on bus $i_a \cdot 33 + 25$, for $i_a = 0, \ldots, n_a - 1$ with active power profile defined as *FL1MW* as show in Figure 6.14,

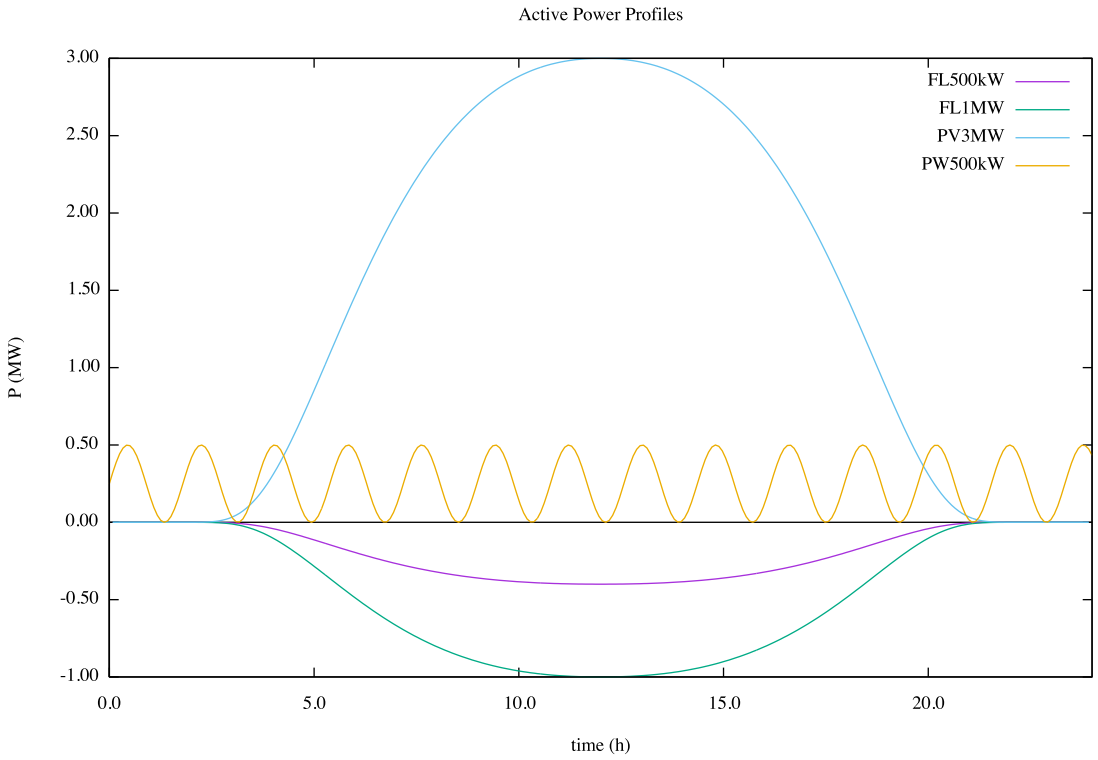


Figure 6.14: Active power profiles.

- each area has a FL on bus $i_a \cdot 33 + 33$, for $i_a = 0, \ldots, n_a - 1$ with active power profile defined as *FL500kW* as show in Figure 6.14.

Connections between the SG under analysis and other SGs are implemented trough HGs. In particular, on bus 1 it is attached a special HG, which is modeled as a $V\delta$-bus and serves as a reference for voltage magnitude and angle, and bus $i_a \cdot 33 + 1$, for $i_a = 1, \ldots, n_a - 1$ there are HGs modeled as PV-buses. Even if there are n_a areas, and then the enhanced SG model can comprise n_a different policies for the instances of AGC_SAN, as detailed in Section 6.2.2, for this case study a simple policy has been considered for all the areas: when one AGC fails, all the DGs and FLs under its influence maintain their injected and dispatched values, i.e. they stuck at the given value even if the optimization would require to change them.

In addition to the measures described in Sections 3.6.1 to 3.6.3, for this case study all the measure listed in in Section 6.3 are evaluated. In the next sections, though, only performance evaluations related to the basic and enhanced model are discussed, because the two are equivalent from the rep-

resentation power point of vies and the focus is on the ability to tackle huge SGs.

6.5 Comparison between basic and enhanced

In this section performance comparisons between the basic SG model discussed in Chapter 3 and the enhanced model described in this chapter are presented. Before commenting the tables of this chapter, the results shown in Tables 3.3 and 3.4 are recalled:

- the basic SG model was not able to overcome the initialization phase in a reasonable amount of time for $n = 165$ buses or more;
- even if it can overcome the initialization and start performing the simulation, the time needed for the analysis is infeasible for more than $n = 99$ buses;
- at increasing the number n of buses, the PFEs routine is called hundreds of thousands times and the percentage of non converged instances grows from 0 for $n = 33$ to $\frac{1.3 \cdot 10^5}{1.79 \cdot 10^5} = 0.726$.

The enhanced model is instead capable to perform the analysis in all the considered SG configurations, as shown in Table 6.1. Furthermore, the upper bound of $n = 990$ buses is not dictated by timing issues, as for the basic SG model, but instead by memory issues. In fact, as mentioned in Section 4.6.2, the implementation of *DARep* through a script exterior to Möbius produces thousands of SANs, each consuming a portion of RAM when the simulator is running. For example, the case study with $n = 660$ requires about 25Gbs of RAM on a machine equipped with the AMD Opteron 6176 CPU, 32Gb of RAM and running Ubuntu 14.04 with 60 of swappiness. All the experiment for the enhanced SG have been carried out on this machine except for the case $n = 990$ that requires about 40Gbs of RAM and it has been exercised on a machine equipped with an Intel Xeon E5530, 64Gb of RAM and running Ubuntu 14.04 with a swappiness of 60. The initialization time for the enhanced model is under 2 minutes even for the case $n = 990$.

6.6 Further analyzes on the enhanced model

A very important aspect that characterizes the enhanced model performance is the mean batch time. Here *batch time* refers to the computation time

Table 6.1: CPU initialization and mean batches time as n increases.

n	$t_{\text{basic\_init}}$ (s)	$\bar{t}_{\text{basic\_batch}}$ (s)	$t_{\text{enhanced\_init}}$ (s)	$\bar{t}_{\text{enhanced\_batch}}$ (s)	$\sigma_{\text{enhanced\_batch}}$ (s)
33	118.84	4460.57	0.15	846.04	13.21
66	1854.39	41371.5	0.85	9145.95	2586.98
330	NaN	NaN	18.03	1459.23	1353.10
660	NaN	NaN	65.17	1355.47	45.13
990	NaN	NaN	94.42	760.87	8.66

required by the machine to perform one simulation batch. Notice that *simulation time* has been defined as the time variable, in this case study spanning from 0s to 86400s (one day), and it is not related to performance but to the stochastic process time. The 5-th column of Table 6.1, marked as $\bar{t}_{\text{enhanced\_batche}}$, shows the mean time spent by the enhanced model in performing one batch, and the 6-th column the root mean square. It is interesting to notice that different electrical grids present different behaviors. In particular, for $n = 66$ and $n = 330$ the batch time can be considerably greater than the batch time of $n = 660$ and present more variation. This is due to the fact that in some of the batches the SG reaches quickly the blackout state whereas in other batches the SG is resilient to the considered failures and reaches the end of the simulation time (one day). Analyzing the simulation traces it has been possible to establish that all the blackouts were caused by the presence of islands within the EI caused by a few bus failures. The times

Table 6.2: Number of CTRL calls, PFEs subroutine calls, and among them how many do not converge, as n increases, for a simulation time of $24h$.

n	# CTRL calls	# NR^{PC} calls	# not converged
33	1	229	0
66	48	86,301	30
330	4	3,651	1,084
660	1	1,782	1,686
990	1	1,808	897

presented in Table 6.1 have been obtained exercising the enhanced SG model where only one PFEs solution strategy was employed, namely the NR^{PC}.

Choosing only one batch per case, i.e., selecting for $n = 33, 66, 330, 660, 990$ only one batch, the number of NR^{PC} calls are shown in Table 6.2. A vast

spectrum of possibilities can be observed, in particular:

- in the batch chosen for case $n = 33$, just one voltage control is called and all the PFEs calls have success,
- in the batch chosen for case $n = 66$, the failure of an electrical component starts a series of failures (cascading effect), but the presence of perfectly working control maintains the voltages in bounds. In order to maintain voltages in bounds, a great number of CTRL calls is required, likely the vast majority of PFEs calls reached convergence and the EI state is always correctly estimated,
- in the batch chosen for cases $n = 660$ and $n = 990$ only one CTRL call is performed, but almost all the PFEs calls in case $n = 660$ and about 50% of them in case $n = 990$ do not reach convergence. In particular, not only the PFEs calls performed during the solution of the control optimization problem but also those related to the EI state estimation do not converge, and then the SG goes to the blackout state.

Table 6.3: Convergence rate of *recovery block* (NR^{PC},NR^{RC},NR^{NV}) calls, and mean number of blackouts, as n increases.

n	NR^{PC} conv. rate	blackouts	*RB* conv. rate	blackouts
33	0.999922	0	1	0
66	0.99957	0	0.99977	0
330	0.441234	1	0.638052	0.5
660	0.071881	1	0.99946	0
990	0.008019	1	0.99141	0.5

It can be concluded that the larger the electrical grid is, the weaker the reliability of the PFEs subroutine is. Thus, implementing the EI state estimation subroutine following the recovery block paradigm was expected to improve the quality of the simulation results. The numerical results confirmed the expectations. In particular, Table 6.3 shows that the convergence rate, i.e., the number of converged PFEs calls divided by the number of total PFEs calls, of NR^{PC} quickly degrades whereas for the *reliability block* implementation of the EI state estimation the convergence rate remains always above 50%. In addition, for case $n = 330$ and $n = 660$ not only the convergence rate is improved but the simulation switches from reporting always a blackout to report it 50% of the times or never.

6.7 Summary

This chapter focused on two contributions: i) description of the enhanced SG model, based on exploiting the novel solutions presented in the previous two chapters, and ii) comparison of the basic SG model and the enhanced SG model in terms of efficiency indicators, to assess the achieved improvements and so demonstrate the relevance and the impact of the thesis work. In particular, in Section 6.1 the replacement of *SSRep* with *DARep* has been described. The change of compositional approach implied a rethinking of the model architecture, relationships among submodels and also of the input data. In fact, the enhanced model relies on an extended version of MATPOWER input format from which it extracts not only data about electrical components, but also information about topologies of interactions and components' non-functional properties, such as Mean Time to Failure and Mean Time To Recovery.

Reshaping the model architecture promoted the possibility to re-design all the submodels, and in Section 6.2 a detailed description of the enhanced submodels is provided. The developments presented in Chapter 5 are exploited in the ESTATE_SAN model, described in Section 6.2.1. Here a recovery block approach to the PFEs solution is adopted and a new optimization problem is defined to model the MCS logic.

In Section 6.4 a comparison analysis is carried out, between the basic and the enhanced SG models from the point of view of time required to accomplish their analyzes. To this purpose, a synthetic grid topology has been designed, able to be extended to represent distribution grids of increasing size. The goal of the analysis was to demonstrate the efficacy of the new techniques, developed with the purpose to enhance the ability of the framework to deal with realistic smart grid dimensions. Thus, although both frameworks have been applied to evaluate a number of performability related metrics to gather results about times characterizing their behavior, the values obtained for such metrics are not meaningful since the analyzed grids just represent a family of synthetically built grids of increasing size. Results from such comparison, presented in Section 6.6, demonstrate the improvements of the enhanced SG framework in tackling realistic grid scenarios. Exercising enhanced SG for the purpose of performability analysis, such as the studies discussed in Chapter Chapter 3 but on bigger size grids, is postponed to future work.

Chapter 7

Conclusions and outlook

This thesis addressed stochastic model-based analysis of SG with the goal of enhancing current practice in terms of ability to deal with large size grids topologies, as necessary when tackling realistic SG scenarios. The achieved contributions mainly consist in the definition and development of techniques to improve efficiency of evaluation from a resilience perspective.

Specifically, starting from a preliminary development of a model-based framework for SG evaluation already available at the ISTI SEDC Lab in Pisa, the research activity focused on:

- extensive application of the existing framework on a variety of grid topologies and failure scenarios, presented in Sections 3.6.1 to 3.6.3, which helped greatly to understand strengths but also limitations of the current solution and pointing out directions for refinement steps. In particular, efficiency was identified as the major obstacle to deal with grid topologies representative of realistic regional segments, so further advancements concentrated on this aspect;

- compositional operators adopted in the modular development of models were identified as a major weakness from efficiency viewpoint, and therefore as a research area where available solutions need improvements. In particular, new non-anonymous replication techniques for constructing the system model starting from template models of its components have been formalized, implemented and tested within the Möbius modeling framework. Among the available possibilities, the best from the point of view of performance, i.e., *Dependency-Aware Replication* (*DARep*), has been selected to replace the one adopted in the basic model, namely *State-Sharing Replication* (*SSRep*);

- state estimation computations were also identified as highly executed

operations and so another promising aspect for performance improvement. Several methods, all based on the Newton-Raphson method, to estimate the state of the electrical infrastructure have been studied and compared in terms of computational effort and convergence to a solution. In order to achieve a good balance between the limited computational cost of NR used by MATPOWER (NR^{PC}) and promising convergence behaviors of NR with rectangular coordinates (NR^{RC}) and NR described in [106] (NR^{NV}), a recovery block strategy has been selected;

- control logic, implemented as the solution of an optimization problem, requires several calls to the electrical state estimation process and then a fine tuning, aimed at minimizing the computational cost, has been performed;

- reformulation of the SG stochastic modeling framework has been accomplished, by introducing the best solutions among the new developed features into the starting basic models, thus generating an efficient enhanced model. Actually, the new developed techniques, as well as refinements of some modeling choices based on the lesson learned from exercising the basic models, led to an enhanced modeling framework that goes beyond the mere substitution of old features with new ones.

To demonstrate the fulfillment of the efficiency goal that guided the study on improvements, a comparison has been carried on between the basic and enhanced modeling frameworks, considering grid topologies of increasing size, from $n = 33$ nodes up to $n = 990$ nodes. Among major achievements pointed out through the comparison there is the fact that the initialization time of the basic model grows as $\mathcal{O}(n^4)$ and becomes infeasible already for $n = 99$, whereas the initialization time of the enhanced model grows according to the degree of interconnections within the electrical grid and remains always reasonably small. Not only the initialization time but also the batches time is considerably better in the enhanced model with respect to the basic model.

In addition, it is noticeable that some of the obtained results have relevance that goes beyond the SG context. Specifically, the new model composition operators to efficiently tackle non-anomymous replication are general solutions that fit any application context characterized by populations of non-anonymous components, as shown by the case study of Section 4.3. More than that, a completely new implementation of the *Rep* operator in Möbius has been proposed in order to tackle those models where non-anonymous replication is not required but a great number of shared SVs is needed, reducing the complexity from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^2)$, as discussed in Section 4.7.2.

Although the performed study led to such concrete outcomes recalled above, the tackled issues are still liable to improvements/extensions, with reference to both the SG sector specifically addressed in this thesis and the approaches themselves, seen as supporting features to a potentially much wider category of applications.

Focusing on the individual developed techniques, further research directions include:

- implementation of the developed modeling composition operators, especially the *DARep* approach, in a modeling environment different from Möbius. This would bring further feedback on the implications of the underlying technological choices provided by the adopted modeling and solution tool. A first step could be to compare the presented implementation of *DARep* with the recently developed Ontology Framework [**?**], investigating similarities and differences, and thinking at how to integrate the two concepts. Although Möbius is a widely used framework for dependability and performance analysis, it is not the only one; thus, extending the proposed replication mechanisms to other stochastic model-based frameworks would enlarge the population of modelers who would take advanage of them. One example can be GreatSPN [124, 125];

- an ongoing project is making native in Möbius the *DARep* operator, an in particular the auxiliary functions Index() and Deps() and the operator $\mathcal{D}$. This would certainly improve efficiency with respect to solutions that can be built by a tool utilizer on top of offered operators/features, as done so far. Compared to current *DARep* implementation, such a smart replicator operator would avoid resorting to the many files now needed, which require increasing (and possibly prohibitive) compilation time at increasing the number of replicas and the dependency degree they show;

- with reference to the PFEs solutions, further investigations on the considered approaches, for example exploiting both parallel implementations and properties of the matrices, are expected to improve the performance abilities of the proposed approaches. Also, potential applications of Wirtinger calculus to optimization problems in other application areas would be worth to investigate;

- preliminary investigations showed promising results as a consequence of having recasted performability measures to matrix functions, as presented in [17], opening new possibilities of applying *DARep* not only for simulation-based analysis but also to analytical solvers;

- the recovery block paradigm employed for enhancing performance and accuracy of the PFEs opens new research questions in the context of SG modeling and simulation. In particular, a quite basic implementation of this paradigm has been adopted and a careful study of its implications, compared with the adoption of more elaborated variants of recovery blocks and also with other paradigms, such as N-version programming, can guide further developments.

Instead, among the research directions that appear as interesting follow-on in the SG context, it is mentioned:

- to intensify the study of analysis scenarios in critical situations, more and more resembling realistic, complex contexts where multiple undesired events may occur, potentially putting in danger the ability of the distribution grid in delivering correct service. It could be also worth to extend the set of assessed analyzes indicators, to enhance the understanding on failure events implications from wider perspectives;

- following the improvements on the individual efficiency-promoting techniques, to further refine the SG modeling framework by conveniently embedding the most advanced solutions.

Chapter 8

List of acronyms and symbols

List of acronyms

AFI Abstract Functional Interface

AGC Area MVGC

AN Access Network

bus Bus-Bar

CB capacitor bank

CSRep Channel-Sharing Replication

DARep Dependency-Aware Replication

DER Distributed Energy Resource

DG distributed generator

DG Distributed Generator

EI Electrical Infrastructure

FL flexible load

HG traditional generator

HV High Voltage

ICT Information and Communication Technology

LC Local Controller

LVGC Low Voltage Grid Control

LV Low Voltage

MCS Monitoring and Control System

MCS Monitoring and Control System

MV-EI MV EI

MVGC Medium Voltage Grid Control

MV-MCS MV MCS

MV-MCS Medium Voltage Monitoring and Control System

MV Medium Voltage

MV Medium Voltage

MV-EI Medium Voltage Electrical Infrastructure

NFL non-flexible load

NR Newton-Raphson method

OLTC on-load tap changer

PFEs Power-Flow Equations

PN Petri Net

p.u. per unit

PVP photovoltaic power plant

QoS Quality of Service

slack-bus bus with fixed V and δ

SAN Stochastic Activity Network

SG Smart Grid

SPN Stochastic Petri Net

SSRep State-Sharing Replication

SV State Variable

WAN Wide Area Network

WP wind power plant

List of model names:

AGC_SAN Template SAN model representing a generic Area MVGC

BRANCH_SAN Template SAN model representing a generic line

BUS_SAN Template SAN model representing a generic bus

CB_SAN Template SAN model representing a generic CB

DG_SAN Template SAN model representing a generic DG

ESTATE_SAN Template SAN model that manages the EI state

ESTATESANSANDAREP0 SAN model representing the 0-th replica of ESTATE_SAN

FL_SAN Template SAN model that represent a generic FL

HG_SAN Template SAN model that represent a generic HG

LV_M Composed model representing the Smart Grid at the low voltage level

MV_ARC_M Template model representing a generic arc at medium voltage level

MV_ARC_M Template model representing a generic arc at medium voltage level

MV_DS_SAN Template SAN model representing a generic Distributed Generator at medium voltage level

MV_EI_INIT_SAN Atomic SAN model representing the initialization of the SAN models defined MV_EI_M

MV_EI_M Composed model representing MV-EI

MV_ESTATE_SAN Atomic SAN model representing the update of the state of Medium Voltage Electrical Infrastructure (MV-EI) directly triggered by failures (outages) in MV-EI

MVGC_ATTACK_SAN Template SAN model representing the malicious failures of a generic MVGC caused by an attack to the MVGC

MVGC_M Composed model representing a generic MVGC and the malicious failures of MVGC caused by an attack to MVGC

MVGCMs Composed model representing all the MVGC at medium voltage level

MVGC_SAN Template SAN model representing a generic MVGC

MV_LC_Ms Template model representing all the logical controllers LC at medium voltage level

MV_LC_SAN Template SAN model representing a generic logical controller LC

MV_MCS_M Composed model representing MV-MCS

MV_M Composed model representing the Smart Grid at the medium voltage level

MV_N1AN2_M Template model representing a generic arc of MV-EI with the associated starting and ending nodes

MV_N1AN2_Ms Composed model representing all the arcs of MV-EI with the associated starting and ending nodes

MV_NODE1_M Template model representing the generic node starting from a line at medium voltage level

MV_NODE1_M Template model representing the generic node starting from a line at medium voltage level

MV_NODE2_M Template model representing the generic node ending to a line at medium voltage level

MV_NODE2_M Template model representing the generic node ending to a line at medium voltage level

MV_PL_SAN Template SAN model representing a generic power line PL at medium voltage level

MV_SP_SAN Template SAN model representing the actuation of the new set points triggered by MVGC

OLTC_SAN Template SAN model that represent a generic OLTC

WAN_ATTACK_SAN Template representing the malicious failures of a generic Wide Area Network WAN caused by an attack to the WAN

WAN_SAN Template SAN model representing a generic Wide Area Network WAN

List of symbols:

node generic node component

buses set of buses

arc generic arc component

I line current

$\mathbf{I}$ vector of nodal currents

I^{bus} nodal current

Y line admittance

Y^{bus} bus admittance

$\mathbf{Y}$ bus admittance matrix

$\mathbf{G}$ bus conductance matrix

B line susceptance

$\mathbf{B}$ bus susceptance matrix

R line resistance

X line reactance

S complex power

$\mathbf{S}$ vector of complex powers

P active power

$\mathbf{P}$ vector of active powers

Q reactive power

$\mathbf{Q}$ vector of reactive powers

E complex voltage

$\mathbf{E}$ vector of complex voltages

V voltage magnitude

$\mathbf{V}$ vector of voltages magnitudes

δ voltage phase

$\boldsymbol{\delta}$ vector of voltages phases

$\mathbf{E}^x$ vector of voltages real part

$\mathbf{E}^y$ vector of voltages imaginary part

PQ-bus bus with fixed P and Q

PV-bus bus with fixed P and V

tap OLTC tap

Q_{cb} reactive power provided by each bank of the capacitor bank

bank number of active banks within the CB

ρ power factor

P_{ref} P at the slack-bus

$P_{\mathrm{ref}}^{\mathrm{min}}$ min P at the slack-bus bus

$P_{\mathrm{ref}}^{\mathrm{max}}$ max P at the slack-bus bus

$\mathrm{Q}_{\mathrm{ref}}$ Q at the slack-bus

$\mathrm{Q}_{\mathrm{ref}}^{\mathrm{min}}$ min Q at the slack-bus

$\mathrm{Q}_{\mathrm{ref}}^{\mathrm{max}}$ max Q at the slack-bus

P^{av} P available, produced by a DG

P^{inj} P actually injected

P^{dis} P actually dispatched to the flexible load

P^{load} P dispatched to the inflexible load

$\mathrm{Q}^{\mathrm{load}}$ Q dispatched to the inflexible load

n_{PQ} number of PQ-buses

n_{PV} number of PV-buses

n $n_{PQ} + n_{PV}$

PQ set of PQ-bus

OLTC set of line indexes that comprise an OLTC

CB set of bus indexes that comprise a CB

DG set of bus indexes that comprise a DG

FL set of bus indexes that comprise a FL

NFL set of bus indexes that comprise a NFL

$\mathbf{F}$ mismatch fuction

$\mathbf{J}$ Jacobian matrix

$\mathbf{X}$ unknowns

$\mathbf{u}$ generic vector

$\mathrm{NR}^{\mathrm{PC}}$ NR used by MATPOWER

$\mathrm{NR}^{\mathrm{RC}}$ NR with rectangular coordinates

W $\mathrm{NR}^{\mathrm{NV}}$ variables corresponding to voltages conjugates

$\mathbf{W}$ vector of $\mathrm{NR}^{\mathrm{NV}}$ W variables

$\mathrm{NR}^{\mathrm{NV}}$ NR described in [106]

$\mathbf{J}^{W}$ complex $\mathbf{J}$ determined by Wirtinger derivatives

$\mathrm{NR}^{\mathrm{W}\mathbb{C}}$ NR using Wirtinger calculus

$\mathbf{J}^{W\mathbb{R}}$ real $\mathbf{J}$ obtained from $\mathbf{J}^{W}$

$\mathrm{NR}^{\mathrm{W}\mathbb{R}}$ NR *new* method

$\mathrm{NR}_{red}^{\mathrm{W}\mathbb{R}}$ reduced NR *new* method

OLTC On Load Tap Changer

OPF Optimal Power Flow

P active power

PL Power Line

Q reactive power

Bibliography

[1] G. Iazeolla, *Principi e metodi di simulazione discreta.* Franco Angeli, 2010.

[2] J. D. D. Glover and M. S. Sarma, *Power System Analysis and Design*, 3rd ed. Pacific Grove, CA, USA: Brooks/Cole Publishing Co., 2001.

[3] R. D. Zimmerman, C. E. Murillo-Sanchez *et al.*, *MATPOWER: A MATLAB Power System Simulation Package*, version 6.0.

[4] http://immersive.erc.monash.edu.au/stac/, 2018.

[5] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster, "The Möbius framework and its implementation," *IEEE Trans. on Softw. Eng.*, vol. 28, no. 10, pp. 956–969, 2002.

[6] S. Chiaradonna, F. Di Giandomenico, and G. Masetti, "A stochastic modelling framework to analyze smart grids control strategies," in *Fourth IEEE International Conf. on Smart Energy Grid Eng.*, Oshawa, Canada, Aug. 2016.

[7] ——, "Analyzing the impact of failures in the electric power distribution grid," in *Seventh Latin-American Symp. on Dependable Comput.*, Cali, Colombia, Oct. 2016.

[8] G. Masetti, S. Chiaradonna, and F. Di Giandomenico, "Exploring equations ordering influence on variants of the newton-raphson method," in *Numerical Computations: Theory and Algorithms.* AIP Conference Proceedings, 2016.

[9] S. Chiaradonna, F. Di Giandomenico, and G. Masetti, "Efficient non-anonymous composition operator for modeling complex dependable systems," *Fast Abstract at EDCC2016 published on CoRR*, vol. abs/1608.05874, 2016. [Online]. Available: http://arxiv.org/abs/1608.05874

[10] G. Masetti, "Enhanced power grid evaluation through efficient stochastic model-based analysis," *arXiv preprint arXiv:1708.04576*, 2017, student paper at EDCC2017.

[11] G. Masetti, S. Chiaradonna, F. Di Giandomenico, B. Feddersen, and W. H. Sanders, "An efficient strategy for model composition in the möbius modeling environment," in 14^{th} *European Dependable Computing Conference*, 2018.

[12] G. Masetti, S. Chiaradonna, and F. Di Giandomenico, "Model-based simulation in Möbius: An efficient approach targeting loosely interconnected components," in *European Workshop on Performance Engineering*. Springer, 2017, pp. 184–198.

[13] S. Chiaradonna, F. Di Giandomenico, and G. Masetti, "A stochastic modeling approach for an efficient dependability evaluation of large systems with non-anonymous interconnected components," in *The* 28^{th} *Int. Symp. on Softw. Reliab. Eng.-IEEE*, Toulouse, France, Oct. 2017, pp. 46–55.

[14] ——, "On identity-aware replication in stochastic modeling for dependability analysis of large interconnected systems," *Submitted: IEEE Transactions on Reliability*, 2018.

[15] G. Masetti, S. Dutto, S. Chiaradonna, and F. Di Giandomenico, "Supporting CPS modeling through a new method for solving complex non-holomorphic equations," in *MODELSWARD conference, Special Session on domAin specific Model-based AppRoaches to vErificaTion and validaTiOn*, Funchal, Madeira - Portugal, 2018.

[16] S. Dutto, G. Masetti, S. Chiaradonna, and F. Di Giandomenico, "Enhancing newton-raphson method by wirtinger calculus for solving power-flow equations," *To appear in IEEE Transactions on Power Systems*, 2019, available online under the "Early access area on IEEE explore".

[17] G. Masetti and L. Robol, "Computing performability measures in Markov chains by means of matrix functions," *arXiv:1803.06322*, 2018.

[18] ——, "Analyzing a security and reliability model using krylov methods and matrix functions," Tech. Rep., 2018, technical report, ISTI-CNR.

[19] D. L. R. Idema, *Computational Methods in Power System Analysis*, ser. Atlantis Studies in Scientific Computing in Electromagnetics. Springer, 2014.

[20] X.-F. Wang, Y. Song, and M. Irving, *Modern Power System Analysis*. Springer, 2008.

[21] E. Commission, "Energy prices and costs in europe," https://ec.europa.eu/energy/sites/ener/files/publication/Energy%20Prices%20and%20costs%20in%20Europe%20_en.pdf, 2014.

[22] J.-C. Laprie, "Dependable computing and fault tolerance: Concepts and terminology," in *15th IEEE Int'l Symp. Fault-Tolerant Computing (FTCS-15)*, 1985, pp. 2–11.

[23] A. Avizienis, J. . Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[24] https://www.irriis.org, 2006.

[25] http://crutial.rse-web.it, 2008.

[26] https://tcipg.org, 2010.

[27] https://www.city.ac.uk/mathematics-computer-science-engineering/research/centre-for-software-reliability/research/research-projects/piafara-probabilistic-interdependency-analysis-framework, -data-analysis-and-on-line-risk-assessment, 2012.

[28] http://www.smartc2net.eu, 2015.

[29] R. Bloomfield, N. Chozos, and P. Nobles, "Infrastructure interdependency analysis: Introductory research review," *Adelard, London, document reference: d418/12101/3 issue 1*, 2009. [Online]. Available: http://www.csr.city.ac.uk/projects/cetifs.html

[30] R. Bloomfield, L. Buzna, P. Popov, K. Salako, and D. Wright, "Stochastic modelling of the effects of interdependency between critical infrastructures," in 4^{th} *Int. Workshop on Crit. Inf. Infrastruct. Secur. (CRITIS 2009)*, ser. LNCS, E. Rome and R. Bloomfield, Eds. Springer Berlin - Heidelberg, 2010, vol. 6027, pp. 201–212.

[31] A. Avritzer, F. Di Giandomenico, A. Remke, and M. Riedl, *Assessing Dependability and Resilience in Critical Infrastructures: Challenges and Opportunities*, 2012, pp. 41–63, resilience Assessment and Evaluation of Computing Systems.

[32] "Definition, implementation and application of a model-based framework for analyzing interdependencies in electric power systems," *International Journal of Critical Infrastructure Protection*, vol. 4, no. 1, pp. 24 – 40, 2011.

[33] W. H. Sanders and J. F. Meyer, "A unified approach for specifying measures of performance, dependability and performability," in *Dependable Computing for Critical Applications, Vol. 4 of Dependable Computing and Fault-Tolerant Systems*, A. Avizienis and J. Laprie, Eds. Springer Verlag, 1991, pp. 215–237.

[34] S. Chiaradonna and t. Di Giandomenico.

[35] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies," *IEEE Control Systems Magazine*, pp. 11–25, December 2001.

[36] P. Pourbeik, P. S. Kundur, and C. W. Taylor, "The anatomy of a power grid blackout," *IEEE Power and Energy Magazine*, pp. 22–29, September/october 2006.

[37] https://www.theguardian.com/us-news/2015/apr/13/oklahoma-city-bombing-20-years-later-key-questions-remain-unanswered, 2015.

[38] A. Babay, J. Schultz, T. Tantillo, and Y. Amir, "Toward an intrusion-tolerant power grid: Challenges and opportunities," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018.

[39] A. Bondavalli, S. Chiaradonna, and F. Di Giandomenico, "Model-based evaluation as a support to the design of dependable systems," in *Dependable Computing Systems: Paradigms, Performance Issues, and Applications*, H. B. Diab and A. Y. Zomaya, Eds. Wiley, 2005, pp. 57–86.

[40] D. M. Nicol, W. H. Sanders, and K. S. Trivedi, "Model-based evaluation: From dependability to security," *IEEE Trans. Depend. Sec. Comput.*, vol. 1, no. 1, pp. 48–65, 2004.

[41] K. S. Trivedi, G. Ciardo, M. Malhotra, and R. Sahner, "Dependability and performability analysis," in *Performance Evaluation of Computer and Communications Systems, LNCS 729*, L. Donatiello and R. Nelson, Eds. Sprinter-Verlag, 1993, pp. 587–612.

[42] G. Balbo, "Introduction to stochastic petri nets," in *Lectures on Formal Methods and Performance Analysis*, ser. LNCS. Springer Verlag, 2001, vol. 2090, pp. 84–155.

[43] K. S. Trivedi and A. Bobbio, *Reliability and Availability Engineering: Modeling, Analysis, and Applications*. Cambridge University Press, 2017.

[44] J. K. Muppala, A. S. Sathaye, R. C. Howe, and K. S. Trivedi, "Hardware and software fault tolerance in parallel computing systems," D. R. Avresky, Ed. Upper Saddle River, NJ, USA: Ellis Horwood, 1992, ch. Dependability Modeling of a Heterogeneous VAX-cluster System Using Stochastic Reward Nets, pp. 33–59. [Online]. Available: http://dl.acm.org/citation.cfm?id=162283.162302

[45] P. Buchholz, "Exact and ordinary lumpability in finite markov chains," *Journal of Applied Probability*, vol. 31, no. 1, p. 59–75, 1994.

[46] S. Derisavi, H. Hermanns, and W. H. Sanders, "Optimal state-space lumping in markov chains," *Inf. Process. Lett.*, vol. 87, no. 6, pp. 309–315, Sep. 2003.

[47] P. Buchholz, "Hierarchical markovian models: Symmetries and reduction," in *Performance Evaluation*, 1992, pp. 234–246.

[48] M. Lanus, L. Yin, and K. S. Trivedi, "Hierarchical composition and aggregation of state-based availability and performability models," *IEEE Transactions on Reliability*, vol. 52, no. 1, pp. 44–52, 2003.

[49] J. Rumbaugh, I. Jacobson, and G. Booch, *Unified Modeling Language Reference Manual, The (2Nd Edition)*. Pearson Higher Education, 2004.

[50] J. Hillston, *A Compositional Approach to Performance Modelling*. New York, NY, USA: Cambridge University Press, 1996.

[51] S. Donatelli, M. Ribaudo, and J. Hillston, "A comparison of performance evaluation process algebra and generalized stochastic petri nets," in *Proceedings 6th International Workshop on Petri Nets and Performance Models*, Oct 1995, pp. 158–168.

[52] L. Carnevali, L. Ridi, and E. Vicario, "Stochastic fault trees for cross-layer power management of wsn monitoring systems," in *2009 IEEE Conference on Emerging Technologies Factory Automation*, 2009, pp. 1–8.

[53] W. H. Sanders and J. F. Meyer, "Stochastic activity networks: Formal definitions and concepts," in *Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science, Berg en Dal, The Netherlands, July 3-7, 2000, Revised Lectures*, ser. LNCS, E. Brinksma, H. Hermanns, and J. P. Katoen, Eds. Springer-Verlag, 2001, vol. 2090, pp. 315–343.

[54] R. M. Fujimoto, *Parallel and Distributed Simulation Systems*. Wiley, 2000.

[55] M. Beccuti, G. Franceschinis, S. Donatelli, S. Chiaradonna, F. D. Giandomenico, P. Lollini, G. Dondossola, and F. Garrone, "Quantification of dependencies in electrical and information infrastructures: The crutial approach," in *2009 Fourth International Conference on Critical Infrastructures*, 2009, pp. 1–8.

[56] P. A. Lee and T. Anderson, *Fault Tolerance: Principles and Practice*, 2nd ed., J. C. Laprie, A. Avizienis, and H. Kopetz, Eds., 1990.

[57] T. Courtney, S. Gaonkar, K. Keefe, E. W. D. Rozier, and W. H. Sanders, "Möbius 2.3: An extensible tool for dependability, security, and performance evaluation of large and complex system models," in 39^{th} *Annu. IEEE/IFIP Int. Conf. on Dependable Syst. and Netw. (DSN 2009)*, Estoril, Portugal, June 2009, pp. 353–358.

[58] U. of Illinois Urbana-Champaign, "Mobius modeling environment," https://www.mobius.illinois.edu.

[59] S. Derisavi, P. Kemper, W. H. Sanders, and T. Courtney, "The möbius state-level abstract functional interface," in *Computer Performance Evaluation: Modelling Techniques and Tools*, T. Field, P. G. Harrison, J. Bradley, and U. Harder, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 31–50.

[60] L. Wall, *Programming Perl*, 3rd ed., M. Loukides, Ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2000.

[61] SmartC2Net Consortium, "Revised architecture & use cases," SmartC2Net Deliverable WP1 D1.2, September 2014.

[62] SmartC2net Consortium, "Comprehensive assessment: Fault & attack analysis and integrated simulation frameworks - preliminary," SmartC2Net Deliverable D5.2, September 2014.

[63] S. Chiaradonna, F. Di Giandomenico, and N. Murru, "On a modeling approach to analyze resilience of a smart grid infrastructure," in *Tenth European Dependable Comput. Conf. (EDCC 2014)*, Newcastle upon Tyne, UK, May 2014, pp. 166–177.

[64] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. on Power Syst.*, vol. 26, no. 1, pp. 12–19, 2011.

[65] B. Stott, "Review of load–flow calculation methods," *Proceedings of the IEEE*, vol. 62, no. 7, pp. 916–929, 1974.

[66] SmartC2Net Consortium, "Control framework and models," SmartC2Net Deliverable D4.1, September 2014.

[67] C. Masetti, "Revision of european standard en 50160 on power quality: Reasons and solutions," in *Proceedings of 14th International Conference on Harmonics and Quality of Power - ICHQP 2010*, 2010, pp. 1–7.

[68] J.-M. Muller, *Elementary Functions. Algorithms and Implementation.* Springer, 2006.

[69] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers," *ACM Trans. on Math. Softw.*, vol. 31, no. 3, pp. 363–396, 2005.

[70] P. Buchholz, "Exact and ordinary lumpability in finite Markov chains," *J. of Appl. Probab.*, vol. 31, no. 1, pp. 59–75, 1994.

[71] S. Derisavi, P. Kemper, and W. H. Sanders, "Symbolic state-space exploration and numerical analysis of state-sharing composed models," *Linear Algebra and its Applications, Special Issue on the Conf. on the Numerical Solution of Markov Chains 2003*, vol. 386, pp. 137–166, 2004.

[72] W. H. Sanders and R. S. Freire, "Efficient simulation of hierarchical stochastic activity network models," *Discrete Event Dynamic Systems*, vol. 3, no. 2, pp. 271–299, 1993.

[73] V. V. Lam, P. Buchholz, and W. H. Sanders, "A component-level path-based simulation approach for efficient analysis of large Markov models," in *37th Conf. on Winter Simulation*, M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, Eds., Orlando, Florida, 2005, pp. 584–590.

[74] L. Brenner, P. Fernandes, A. Sales, and T. Webber, "A framework to decompose GSPN models," in *Applications and Theory of Petri Nets 2005*, ser. LNCS, G. Ciardo and P. Darondeau, Eds. Springer-Verlag, 2005, vol. 3536, pp. 128–147.

[75] J. Hillston, *Compositional Markovian Modelling Using a Process Algebra*. Boston, MA: Springer US, 1995, pp. 177–196.

[76] C. Hirel, R. Sahner, X. Zang, and K. Trivedi, "Reliability and performability modeling using sharpe 2000," in *Computer Performance Evaluation.Modelling Techniques and Tools*, B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith, Eds. Springer Berlin Heidelberg, 2000, pp. 345–349.

[77] S. Chiaradonna, P. Lollini, and F. Di Giandomenico, "On a modeling framework for the analysis of interdependencies in electric power systems," in 37^{th} *Annu. IEEE/IFIP Int. Conf. on Dependable Syst. and Netw. (DSN 2007)*, Edinburgh, UK, June 2007, pp. 185–195.

[78] F. Flammini, *Critical Infrastructure Security: Assessment, Prevention, Detection, Response*. WIT Press, 2012.

[79] G. Masetti, S. Chiaradonna, and F. Di Giandomenico, "Model-based simulation in Mobius: an efficient approach targeting loosely interconnected components," in *Computer Performance Engineering: 13th European Workshop (EPEW)*, Berlin, Germany, Sep. 2017.

[80] G. Ciardo and K. S. Trivedi, "A decomposition approach for stochastic reward net models," *Perform. Eval.*, vol. 18, no. 1, pp. 37–59, 1993.

[81] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo, "Greatspn 1.7: Graphical editor and analyzer for timed and stochastic petri nets," *Perform. Eval.*, vol. 24, no. 1-2, pp. 47–68, Nov. 1995.

[82] M. A. Marsan, "Stochastic petri nets: An elementary introduction," in *Advances in Petri Nets 1989*, G. Rozenberg, Ed., 1990, pp. 1–29.

[83] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[84] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd ed. Chichester, UK: John Wiley and Sons Ltd., 2002.

[85] G. Clark and W. H. Sanders, "Implementing a stochastic process algebra within the möbius modeling framework," in *Process Algebra and Probabilistic Methods. Performance Modelling and Verification*, L. de Alfaro and S. Gilmore, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 200–215.

[86] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools," *Computer Science Review*, vol. 15-16, pp. 29 – 62, 2015.

[87] S. Bernardi, S. Donatelli, and A. Horvath, "Compositionality in the greatspn tool and its application to the modelling of industrial applications," in *University of Aarhus (Denmark)*, 2000, pp. 127–146.

[88] A. Benoit, L. Brenner, P. Fernandes, B. Plateau, and W. J. Stewart, "The peps software tool," in *Computer Performance Evaluation. Modelling Techniques and Tools*, P. Kemper and W. H. Sanders, Eds., 2003, pp. 98–115.

[89] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains*. John Wiley & Sons, Inc., 2006.

[90] K. S. Trivedi, *Probability and Statistics with Reliability, Queueing and Computer Science Applications*, 2nd ed. New York: John Wiley & Sons, 2002.

[91] https://www.perform.illinois.edu, 2018.

[92] S. Derisavi, P. Kemper, W. H. Sanders, and T. Courtney, "The Möbius state-level abstract functional interface," *Perform. Eval.*, vol. 54, no. 2, pp. 105–128, Oct. 2003.

[93] W. H. Sanders and J. F. Meyer, "Reduced base model construction methods for Stochastic Activity Networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 1, pp. 25–36, 1991.

[94] N. M. Josuttis, *The C++ Standard Library: A Tutorial and Reference*, 2nd ed. Addison-Wesley Professional, 2012.

[95] A. L. Williamson, "Discrete event simulation in the Mobius modeling framework," 1998.

[96] J. J. Grainger and W. D. J. Stevenson, *Power system analysis*. McGraw-Hill, Inc., 1994.

[97] C. T. Kelley, *Solving nonlinear equations with Newton's method*, ser. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003, vol. 1. [Online]. Available: http://dx.doi.org/10.1137/1.9780898718898

[98] A. Gomez-Exposito, A. Conejo, and C. Canizares, *Electric Energy Systems: Analysis and Operation*, ser. Electric Power Engineering Series. CRC Press, 2017.

[99] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003.

[100] D. Knoll and D. Keyes, "Jacobian-free newton–krylov methods: a survey of approaches and applications," *Journal of Computational Physics*, vol. 193, no. 2, pp. 357 – 397, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999103004340

[101] Y. Chen and C. Shen, "A jacobian-free newton-gmres(m) method with adaptive preconditioner and its application for power flow calculations," *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1096–1103, Aug 2006.

[102] R. Idema, D. J. P. Lahaye, C. Vuik, and L. van der Sluis, "Scalable newton-krylov solver for very large power flow problems," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 390–396, Feb 2012.

[103] T. Davis, *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006.

[104] J. B. Ward and H. W. Hale, "Digital computer solution of power-flow problems [includes discussion]," *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems*, vol. 75, no. 3, Jan 1956.

[105] H. Le Nguyen, "Newton-raphson method in complex form," *IEEE Trans. on Power Syst.*, vol. 12, no. 3, pp. 591 – 595, 1997.

[106] T. T. Nguyen and C. T. Vu, "Complex-variable Newton-Raphson load-flow analysis with FACTS devices," in *2005/2006 IEEE/PES Transmission and Distribution Conference and Exhibition*, 2006, pp. 183–190.

[107] J. E. Tate and T. J. Overbye, "A comparison of the optimal multiplier in polar and rectangular coordinates," *IEEE Trans. on Power Syst.*, vol. 20, no. 4, pp. 1667–1674, 2005.

[108] W. Wirtinger, "Zur formalen theorie der funktionen von mehr komplexen veränderlichen," *Mathematische Annalen*, vol. 97, no. 1, pp. 357–375, 1927.

[109] Z. Wang, B. Cui, and J. Wang, "A necessary condition for power flow insolvability in power distribution systems with distributed generators," *IEEE Trans. on Power Syst.*, vol. 32, no. 2, pp. 1440–1450, 2017.

[110] J. Bandler, M. El-Kady, and H. Gupta, "Practical complex solution of power flow equations," *IEEE Trans. on Circuits and Syst.*, vol. 29, no. 11, pp. 772–775, 1982.

[111] F. J. Gonzalez-Vazquez, "The differentiation of functions of conjugate complex variables: application to power network analysis," *IEEE Trans. on Educ.*, vol. 31, no. 4, pp. 286–291, 1988.

[112] A. Lee, "Centrohermitian and skew-centrohermitian matrices," *Linear Algebra Appl.*, vol. 29, Suppl. C, pp. 205–210, 1980.

[113] MathWorks, *MATLAB R2018a*, The Mathworks, Inc., Natick, Massachusetts, 2018.

[114] J. P. Hulskamp, S. Chan, and J. F. Fazio, "Power flow outage studies using an array processor," *IEEE Trans. on Power and App. Syst.*, no. 1, pp. 254–261, 1982.

[115] T. A. Davis, "Algorithm 832: UMFPACK V4.3 - an unsymmetric-pattern multifrontal method," *ACM Trans. on Math. Softw.*, vol. 30, no. 2, pp. 196–199, 2004.

[116] S. N. Yeralan, T. A. Davis, W. M. Sid-Lakhdar, and S. Ranka, "Algorithm 980: Sparse QR factorization on the GPU," *ACM Trans. on Math. Softw.*, vol. 44, no. 2, pp. 17:1–17:29, 2017.

[117] N. Bell and M. Garland, "Efficient sparse matrix-vector multiplication on CUDA," NVIDIA, Tech. Rep., 2008.

[118] Y.-Q. Shen and T. J. Ypma, "Newton's method for singular nonlinear equations using approximate left and right nullspaces of the jacobian," *Applied Numerical Mathematics*, vol. 54, no. 2, pp. 256 – 265, 2005, 6th IMACS.

[119] M. R. Lyu, *Software Fault Tolerance*, 1995.

[120] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring, *GNU Octave version 4.2.0 manual: a high-level interactive language for numerical computations*, 2016. [Online]. Available: http://www.gnu.org/software/octave/doc/interpreter

[121] S. Cahon, N. Melab, and E.-G. Talbi, "ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics"," *Journal of Heuristics*, vol. 10, no. 3, pp. 357–380, 2004.

[122] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[123] D. Jeltsema and G. Kaiser, "Active and Reactive Energy Balance Equations in Active and Reactive Time," ArXiv:1601.08207.

[124] M. Beccuti and G. Franceschinis, "Efficient simulation of stochastic well-formed nets through symmetry exploitation," in *2012 Winter Simulation Conference (WSC)*, Berlin, Germany, 2012, pp. 1–13.

[125] S. Baarir, M. Beccuti, D. Cerotti, M. D. Pierro, S. Donatelli, and G. Franceschinis, "The GreatSPN tool: recent enhancements," *ACM SIGMETRICS Perform. Eval. Rev., Spec. Issue on Tools for computer performance modeling and reliability analysis*, vol. 36, no. 4, pp. 4–9, 2009.