2010

A reconfigurable distributed multiagent system optimized for scalability

Summiya Moheuddin
West Virginia University

A RECONFIGURABLE DISTRIBUTED MULTIAGENT SYSTEM OPTIMIZED FOR SCALABILITY

by

Summiya Moheuddin

Thesis submitted to the College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

Approved by

Afzel Noore, PhD, Committee Chairperson

Muhammad A. Choudhry, PhD

Hany H. Ammar, PhD

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia

2010

Keywords: Multiagent System, Scalable Design, Spectral Clustering, Reconfiguration

Abstract

A RECONFIGURABLE DISTRIBUTED MULTIAGENT SYSTEM OPTIMIZED FOR SCALABILITY

by Summiya Moheuddin

This thesis proposes a novel solution for optimizing the size and communication overhead of a distributed multiagent system without compromising the performance. The proposed approach addresses the challenges of scalability especially when the multiagent system is large. A modified spectral clustering technique is used to partition a large network into logically related clusters. Agents are assigned to monitor dedicated clusters rather than monitor each device or node. The proposed scalable multiagent system is implemented using JADE (Java Agent Development Environment) for a large power system. The performance of the proposed topology-independent decentralized multiagent system and the scalable multiagent system is compared by comprehensively simulating different fault scenarios. The time taken for reconfiguration, the overall computational complexity, and the communication overhead incurred are computed. The results of these simulations show that the proposed scalable multiagent system uses fewer agents efficiently, makes faster decisions to reconfigure when a fault occurs, and incurs significantly less communication overhead. The proposed scalable multiagent system has been coupled with a scalable reconfiguration algorithm for an electric power system attempting to minimize the number of switch combination explored for reconfiguration. The reconfiguration algorithm reconfigures a power system while maintaining bus voltages within limits specified by constraints.

DEDICATION

To

The best parents anyone can have

My Father Qamar Moheuddin and My Mother Rashada Qamar

And

Their lifelong efforts for their children

ACKNOWLEDGMENTS

I would like to thank the people whose appreciation, help, and support have resulted into this thesis. Foremost, I would like to thank my advisor Dr. Afzel Noore for providing me with the opportunity to research under his supervision. He has been a source of constant support, inspiration, and guidance for me during the course of my degree. His feedback and constructive criticism has played a vital role in my intellectual development. I am also deeply indebted to my thesis committee members Dr. Muhammad A. Choudhry, Dr. Hany H. Ammar, and Dr. Ali Feliachi. This thesis would not have been possible without their valuable guidance and feedback. I also want to thank the US Department of Energy (DE-FC26-06NT42793) for partially sponsoring this research. I would like to thank my colleagues at the Advanced Power and Electricity Research Center for providing me with a conducive environment for research. I deeply appreciate the help and honest criticism that enabled me in improving my research methodology. I also want to express my gratitude for my friends who have helped me maintain a balance between work and life.

Finally, I want to thank my family for their unflinching support and love. I find myself short of words as I begin to thank my parents Qamar Moheuddin and Rashada Qamar. Whatever little that I am today and whatever I will ever achieve in my life will be because of my parent's relentless efforts. I want to thank my parents for providing me with the best opportunities, for believing in me, and for letting me follow my interests and dreams. I truly believe that all my accomplishments have been possible because of my parent's prayers, their guidance, and wisdom. They have always provided me with emotional support and reality checks during the hardest moments of my life. My brothers Ali Bahu and Qamar Zeb have always been my pillars of support. They are my source of strength and have taught me how to be a strong, motivated, and driven individual.

Table of Contents

Abstract ii

Dedication iii

Acknowledgments iv

LIST OF TABLES x

ABBREVIATIONS xi

Chapter 1 Introduction 1

1.1 Overview 1

1.2 MAS APPLICATIONS IN ELECTRIC POWER SYSTEMS 2

1.3 Problem Statement 4

1.4 Proposed Approach 4

1.5 Organization of the Thesis 4

Chapter 2 Literature Survey 6

2.1 Methods and Techniques for Power Distribution System Reconfiguration 6

Artificial Neural Networks 7

Heuristics 8

Expert Systems 9

Fuzzy Logic 11

Genetic Algorithms 12

2.2 Multiagent Systems for Power System Reconfiguration 13

2.3 Other Multiagent Applications in Electric Power Systems 15

Agents and Electric Power Markets 15

Multiagent Systems for Power System Voltage Control 17

Multiagent Systems for Power System Protection Coordination 18

Chapter 3 Agents and Multiagent Based Systems 20

3.1 Distributed Artificial Intelligence 20

3.2 Distributing Intelligence 20

3.3 What is an agent? 21

3.4 Multiagent Systems 23

3.5 Characteristics of Multiagent Systems 24

3.6 Levels of Organization 24

3.7 Characteristics of Applications Aimed by Multiagent Technology 25
3.8 Agent Cooperation 25
Methods of Cooperation 26
Advantages of Cooperation 26
3.9 Benefits of Multiagent Systems 27
Chapter 4 Spectral Clustering 29
4.1 Graph Notations 29
4.2 Graph Laplacians 30
Unnormalized Graph Laplacian 30
Normalized Graph Laplacian 31
4.3 Spectral Clustering 31
Similarity Graphs 32
The Process of Spectral Clustering 32
4.4 Spectral Clustering Algorithms 35
Unnormalized Spectral Clustering 35
Normalized Spectral Clustering – Algorithm 1 35
Normalized Spectral Clustering – Algorithm 2 36
Self Tuning Spectral Clustering 36
Chapter 5 Proposed Reconfigurable Multiagent Architectures 39
5.1 Testbed Power System – SCE's Circuit of the Future 39
5.2 Proposed topology-independent decentralized multiagent system architecture 41
Identification and reconfiguration of nodes affected by fault 41
5.3 Spectral Clustering Techniques for Power System Partitioning 45
Spectral clustering for partitioning a power system 46
5.4 Proposed Scalable Multiagent System Architecture 51
Functionality of the proposed scalable multiagent system 52
Identification and reconfiguration of nodes affected by fault 54
Identification of alternate source nodes 55
Implementation Details of the Proposed Multiagent System 57
Simulation Results 59
Chapter 6 Introducing Power System Constraints 64
6.1 Proposed Scalable Reconfiguration Algorithm 65

6.2 Implementation Details 66
6.3 Simulation Results 67
Example Scenario 68
Chapter 7 Conclusion and Future Work 81
7.1 Conclusion 81
7.2 Future Work 82
Appendix A 83
Algorithm for Topology-Independent Decentralized MAS Architecture 83
Algorithm for Scalable Multiagent System Architecture 85
Appendix B 88
Simulation results for the scalable multiagent system architecture 88
Simulation results for topology independent decentralized MAS Architecture 93
Comparison of simulation results for proposed scalable reconfiguration algorithm and brute force technique 98
References 104

LIST OF FIGURES

Figure 1 A weighted undirected graph 29
Figure 2 Weighted data graph 32
Figure 3 Generated Laplacian matrix 33
Figure 4 Applying spectral clustering 33
Figure 5 Eigen values and eigen vectors 34
Figure 6 Performing vertex mapping 34
Figure 7 Clustered graph 34
Figure 8 Southern California Edison's Circuit of the Future 40
Figure 9 Circuit of the Future represented as a graph 40
Figure 10 Graphical representation of SCE's Circuit of the Future and example fault scenarios 44
Figure 11 WSCC-9 bus system and its corresponding graph 48
Figure 12 Clustered WSCC-9 bus system and its corresponding graph 50
Figure 13 Assigning cluster agent CA_i to each cluster obtained in SCE's Circuit of the Future 52
Figure 14 Faults propagating between clusters in SCE's Circuit of the Future 55
Figure 15 Example fault scenarios in the SCE's Circuit of the Future 57
Figure 16 Agent communication in JADE during fault scenario 60
Figure 17 Comparison of communication overhead 60
Figure 18 Comparison of computational overhead 61
Figure 19 Comparison of performance with respect to buses affected by a fault 63
Figure 20 SCE's Circuit of the Future in PAT 67
Figure 21 Comparison of performance 68
Figure 22 Circuit of the Future after turning SW1 ON 69
Figure 23 Voltage profile after turning SW1 ON 69
Figure 24 Circuit of the Future after turning SW1 and SW7 ON 70
Figure 25 Votlage profile after turning SW1 and SW7 ON 71
Figure 26 Circuit of the Future after turning SW1 and SW2 ON 71
Figure 27 Voltage profile after turning SW1 and SW2 ON 72
Figure 28 Circuit of the Future after turning SW1 and SW4 ON 72
Figure 29 Voltage profile after turning SW1 and SW4 ON 73
Figure 30 Circuit of the Future after turning SW1 and SW6 ON 73
Figure 31 Voltage profile after turning SW1 and SW6 ON 74
Figure 32 Circuit of the Future after turning SW1 and SW5 ON 74
Figure 33 Voltage profile after turning SW1 and SW5 ON 75
Figure 34 Circuit of the Future after turning SW1 and SW3 ON 75
Figure 35 Voltage Profile after turning SW1 and SW3 ON 76
Figure 36 Circuit of the Future after turning SW1, SW5, and SW2 ON 77
Figure 37 Voltage profile after turning SW1, SW5, and SW2 ON 77
Figure 38 Circuit of the Future after turning SW1, SW5, and SW3 ON 78
Figure 39 Voltage profile after turning SW1, SW5, and SW3 ON 78

Figure 40 Circuit of the Future after turning SW1, SW5, and SW4 ON 79
Figure 41 Voltage profile after turning SW1, SW5, and SW4 ON..... 79

LIST OF TABLES

Table 1 Simulation results of selected fault scenarios in the topology-independent decentralized MAS .. 45
Table 2 Results obtained after applying spectral clustering on SCE's Circuit of the Future 51
Table 3 Simulation results of selected fault scenarios in the proposed scalable MAS 58

ABBREVIATIONS

ACL	Agent Communication Language
API	Application Programming Interface
AVR	Automatic Voltage Regulator
BA	Bus Agent
CA	Cluster Agent
CoF	Circuit of the Future
DG	Distributed Generator
FACTS	Flexible Alternative Current Transmission Systems
FIPA	Foundation for Intelligent Physical Agents
JADE	JAVA Agent Development Environment
MAS	Multiagent System
MEA	More Electric Aircraft
PA	Processor Agent
PAT	Power Analysis Toolbox
SA	Switch Agent
SCE	Southern California Edison
STATCOM	Static Synchronous Compensator
SVC	Static Var Compensator
SW	Switch
TCP	Transmission Control Protocol

Chapter 1 Introduction

1.1 Overview

A Multiagent system (MAS) is a collection of collaborating intelligent and autonomous computational entities called agents [1], perceiving the environment using sensors and acting upon it through actuators [2]. The flexibility and adaptability of multiagent systems make them attractive for several real world applications. Multiagent systems are suitable where scalability is an important requirement, especially where the number of entities to be monitored and controlled is large, processing is performed in a distributed and parallel environment, and there is an exponential growth in the computational complexity. This has been demonstrated by using the agent paradigm to build a biological model simulating polypeptide generation [3]. A combination of cluster computing and multiagent systems has also been used for achieving scalability. This concept is used in a distributed multiagent model for network-based financial computing and economic analysis [4], and for developing scalable software systems on heterogeneous networks [5].

The performance and scalability of a multiagent system is measured using indicators such as number of concurrent agents and associated tasks, organizational pattern of agents, coordination protocols used, and communication and computational overheads [6]. Coordination policies employed by agents are an important indicator of scalability as these are used to determine the total number of messages necessary to converge to a solution [7]. The authors also discuss additional dimensions along which a distributed solution using agent technology needs to scale, including total number of agents, size of data and agent diversity. In [8] the authors analyze the increase in communication load in terms of the number of messages, as the number of agents connected in a number of different topologies is increased. The concept of self building and self organizing MAS to achieve higher scale tolerance by defining scalability in terms of the total processing requirements for agents has been introduced in [9]. MAS that have fixed organizational structures are less scalable than those that adapt to the demands of an application. The issues related to scalability and communication overheads involved in model based

teamwork approaches using agents which do not rely on accurate models and further make use of dynamically evolving and overlapping sub teams are discussed in [10]. Another approach to scalability is to employ a locality model to limit the number of agent interactions thereby making an agent interact only with agents located in a local region [3].

While previous research has broadly focused on achieving scalability by inspecting the organizational patterns of multiagent communities, we investigate achieving scalability by optimally reducing the total number of agents in the multiagent system without compromising the performance. The reduction in the number of agents decreases the resource requirements, simplifies the agent system topology, reduces the communication overhead by limiting the number of possible interactions and minimizes the overall complexity of the system. The proposed scalable multiagent architecture is applied to power system reconfiguration as a case study.

1.2 MAS APPLICATIONS IN ELECTRIC POWER SYSTEMS

The evolution of electric power industry due to recent restructuring and deregulation has led to a major paradigm shift from centralized system towards decentralized solutions. Motivations for this change are discussed in [11]-[13]. Considering power system reconfiguration in particular, centralized solutions based on artificial neural networks, genetic algorithm and expert systems have been proposed [14], [15]. However, such solutions do not adequately address the requirements of modern power systems. Efficient monitoring, control, restoration and reconfiguration of power systems require distributed decentralized control. Multiagent systems have been applied for solving problems such as reconfiguration, restoration, fault identification, diagnosis and power system protection [16]-[19].

Most of the work employing multiagent systems for power system reconfiguration and restoration use an agent to represent an electric component such as buses, switches, and circuit breakers etc. [11], [20], [21]. A major problem with such solutions is the limited application especially for large scale power systems with hundreds of components. The size of the multiagent system used in addressing the problem and interactions among agents introduce challenges due to scalability.

Another issue that becomes more and more important as the size of target system increases is that of communication overhead. As discussed previously, communication and coordination policies used in a multiagent system can have serious impact on scalability. The issue of increased communication overhead has been discussed in a multiagent based algorithm for reconfiguration of ring structured shipboard power systems in [22], [23]. Similar concept for reconfiguration has been proposed for mesh structure power systems in [24]. Message passing is used for the identification of a ring structure to avoid accumulation of redundant information in the system. As the size of the power system increases, the communication overhead also increases. Huang et al. [25] introduce a purely decentralized approach to address this issue. The exponential increase in message or communication overhead with the number of loads in a power system and the resulting delay in decision making has been discussed in [26]. Their solution uses mobile agents for local data utilization and determining the impact of power system loads on power quality. Although such solutions show better performance than a centralized approach, further improvement can be achieved by optimizing the size of agent communities and their organizational patterns.

Recently, distributed intelligence has been applied to switch controls using the concept of teams and coaches [27], [28]. A team represents line segments bounded by switches, while coaches have the job to monitor and coordinate their teams and perform restoration by communicating with neighboring teams. This approach considers the availability of multiple sources, and coaches try to identify alternate sources that can supply the load in their team. The coaches move to team members to determine the state of team line segment. This design decision makes agents more susceptible to faults. Also, every node has to be configured for starting agents by providing hardware and software support. On detecting faults, team members communicate this information to other team members and to the coach.

There is also the need for a common place where information and measurements are collected and appropriately coordinated, even in approaches that are perfectly autonomous and distributed. This is especially useful in cases where one out of several possible options is to be selected based on some criteria. For example, in [20] junction agents act as coordinators/controllers for negotiation between different bus agents linked to the transmission line. A common coordinating entity has also been used in [29] where multiagent technology has been applied to the dynamic

reconfiguration of power systems by considering time varying loads. In [25] the use of a root agent to compute the net power of the system by collecting information from all its children is discussed. This root agent also acts as a central location where information is accumulated before any decisions are made.

1.3 Problem Statement

A major challenge of agent architectures for large scale power systems is scalability. Most existing architectures, associate an agent with an electronic component in the system. The idea is to have agents with local information communicate with each other to reach a solution. However, the amount of communication, coordination and other resource requirements for such solutions will soon outweigh the anticipated benefits of decentralization. So, no matter how promising the suggested multiagent system is, its size will always pose a problem. This triggers the need to find a trade-off between decentralization and size of the multiagent system. Efficient solutions would place a bound on the number of agents in the system.

1.4 Proposed Approach

In this research, we have developed clustering techniques to identify logical power system partitions such that agents can monitor these dedicated clusters or partitions rather than employ an agent to monitor each node or device. This reduces the resources utilized and the complexity of the system. Our proposed approach significantly reduces the number of agents in the system and the communication overhead while minimizing the system complexity and computational overhead. Furthermore, in the event of a fault, the affected nodes are quickly identified for reconfiguration.

1.5 Organization of the Thesis

The rest of this document is organized as follows,

Chapter 2 provides a literature review on the different methods applied for the reconfiguration of electric power systems. Power system reconfiguration using multiagent systems is discussed in detail.

Chapter 3 provides a detailed review on agents and multiagent based systems. Different architectures, communication protocols, interaction protocols, and general characteristics of multiagent systems are presented.

Chapter 4 provides a detailed review on spectral clustering. General theory as well as specific algorithms are presented.

Chapter 5 proposes a reconfigurable multiagent system architecture optimized for scalability. The proposed approach is discussed in detail. The feasibility of the proposed approach is illustrated using simulation results.

Chapter 6 introduces a scalable reconfiguration algorithm for electric power system coupled with the multiagent system. Voltage constraint for power system has been implemented.

Chapter 7 summarizes the work proposed in this thesis.

Chapter 2 Literature Survey

2.1 Methods and Techniques for Power Distribution System Reconfiguration

Distribution system reconfiguration is done for several reasons including scheduled maintenance, to deal with post fault line outages, and overloads etc. Reconfiguration is done by changing the on/off status of the power system switches so that the system can function under the present operating conditions.

When a fault occurs in a power system a protective relay detects it and trips the circuit breakers. A fault section is identified by reclosing the circuit breaker and sectionalizing switches which work to isolate or restore loads during a fault or maintenance. Finally, the load section between the bus and the fault section are energized again. After this the process of service restoration to the de-energized loads starts and is accomplished by transferring these loads to adjacent feeders [30].

Feeder reconfiguration involves opening and closing of switches thereby changing the system topology to achieve faster service restoration in case of faults and for minimizing system losses [14]. Effective fault location and service restoration algorithms can minimize the out of service area as well as the duration of the fault. Therefore efficient functioning of the power systems requires an effective service restoration plan [32].

Distribution system reconfiguration is also done for minimizing losses. Feeder load imbalance is one of the main reasons of line losses in the system. Reduction in losses can improve operational efficiency and reduce costs. Feeder reconfiguration is an effective method for loss reduction and achieving load balance [33].

Researchers have applied a wide variety of methods for power system reconfiguration including artificial neural networks [14], [32], genetic algorithms [14], [15], fuzzy logic [33], [45], heuristics [1-5], [13], expert systems [4], [6-11], dynamic programming [11], simulated annealing, and Petri nets etc. Some of the most common of these methods and techniques have been discussed below.

Artificial Neural Networks

A neuron is a brain cell that collects, processes, and disseminates electrical signals. A network of such neurons is what realizes the brain's information processing capacity [2]. Artificial Neural Networks, as the name suggests, tend to simulate this network of neurons in the human brain. Generally, neural computing involves the study of networks of adaptable nodes which through a process of learning from task examples store experiential knowledge and make it available for use [34]. Neural networks are complex nonlinear functions with many parameters. Their parameters can be learned from noisy data and they have been used for thousands of applications [2].

In [14] artificial neural networks have been applied for power system feeder reconfiguration. The overall objective of the system is to ensure minimal losses of active power. To achieve better performance the artificial neural network has been complemented with clustering techniques for the selection of best training set. Applying clustering techniques on the training set results in a more effective information source for the neural network. Therefore, the input layer of the neural network takes a reduced number of neurons for representing the distribution system. As output, the neural network provides a radial topology for the distribution system minimizing the losses.

Hsu and Huang have investigated techniques based on artificial intelligence for distribution system reconfiguration in [32]. Traditional optimization techniques can take longer to determine the set of switches to be operated upon for service restoration. Artificial neural networks and pattern recognition methodologies have been applied to effectively and efficiently determine a post fault service restoration plan. Feasibility of the proposed approaches has been tested by applying these on a distribution system of Taiwan Power Company. Inputs to the artificial neural network are loads of the lateral within the out of service area and the spare capacity of the supporting feeder. Feeder and lateral tie switches states are provided as outputs. The pattern recognition approach involves creating training patterns, storing these patterns in a database and retrieving them based on the pattern under study. The neural network approach provides a single restoration plan whereas several restoration plans are suggested by the pattern recognition approach.

Feeder reconfiguration using artificial neural networks has also been discussed in [35]. Neural networks have been employed to determine a system topology minimizing the power loss according to load pattern variations. The neural network uses the training set provided to determine the appropriate topology and therefore avoids the repetitive process of transferring the loads and estimating the power loss as done in the conventional methods. Two sets of neural networks have been used. The first set utilizes the load data and estimates the load levels for each zone. The second set uses these load levels as input to determine an appropriate system topology.

Heuristics

Heuristics are information based strategies for controlling problem solving processes. Heuristics also known as rules of thumb, educated guesses, or intuitive judgments are criteria or principles for choosing the most effective course of action among several alternatives. They represent compromise between two requirements, the need to make such criteria simple and at the same time as effective as possible [36]. Exact solutions for most complex problems require the evaluation of a large number of alternatives. For most applications the time required for an exact solution is unrealistic. Heuristics can be applied to such cases to reduce the number of evaluations and come up with a reasonable solution within reasonable time constraints.

Electric power system service restoration is a complex combinatorial optimization problem because there are many candidate switches in the distribution system that can be used. Heuristics help in devising a restoration plan faster by narrowing down the potential candidates [37], [38]. In order to devise quick restoration plans, it is important to avoid solving the combinatorial problem.

A non-combinatorial method for the restoration of a large scale distribution system has been proposed in [30]. Aoki et al. applied the dual effective gradient method in order to quickly restore a distribution system after fault. The proposed algorithm involves four basic steps. In the first step, de-energized loads are connected to an adjacent feeder. If this first step results in constraint violations, extra load transferred to the adjacent feeders are shifted to other feeders. If violations still prevail, the third stage involves load curtailment. Finally, in the fourth step new load transfer possibilities are analyzed to restore some or all of the curtailed loads.

A heuristic search algorithm has been applied for power system reconfiguration for restoration of power after the faulted zone has been isolated in [37]. The basis for the algorithm is provided by the heuristic rules compiled with the help of experienced operators at Taiwan Power Company. Any restoration algorithm should come up with a plan as quickly as possible, should restore as many loads as possible, and attempt to minimize the switching operations. Also, the post restoration system configuration should be as close to the original as possible. The system radiality should be retained and there should be no component overloads. The proposed algorithm is applied for service restoration of a Taiwan Power Company distribution system and provides efficient service restoration meeting all the aforementioned practical needs.

Heuristics have also been used for loss reduction in power systems through feeder reconfiguration. Feeder reconfiguration not only relieves the heavily loaded power system feeders by switching loads to other feeders but also improves voltage profile along feeders thereby reducing the overall power losses. The change in losses after a reconfiguration is performed can be determined from the load flow studies for the system configurations before and after the reconfiguration. However, performing load flow studies for all possible switching options is not practical for real time systems. Civanlar et al. in [39] developed a criterion to reduce the number of candidate options and a formula which does not require numerous load flow studies. Reduction in the number of switching options is achieved by applying an observation heuristic stating that loss reduction is achieved only when there is a significant voltage difference across the normally open tie switch and when transfer is done from the higher voltage drop side to the other side.

Expert Systems

Expert systems are programs which represent and apply factual knowledge of specific areas of expertise to solve problems [40]. An expert system is one that is capable of solving problems or giving advice in some knowledge rich domain. Although expert systems are considered intelligent yet they do not interact directly with any environment. To get information from the environment, an expert system relies on a user for information and provides the user with some feedback [1].

A knowledge based expert system emulates the reasoning process of a human expert in a particular domain. Expert systems can apply rules developed by human experts to solve problems using a heuristic approach. Power system equipment maintenance and operation as well as the collection and interpretation of large sets of data can be enhanced through an intelligent information processing approach. Expert systems can be applied to several power system functions including circuit configuration, restoration, dispatching, load flow studies, turbine generator analysis, alarm processing, fault location etc [38].

Expert systems seem a good choice for problems involving non-trivial reasoning and human knowledge. Many power system problems fit this criterion and solutions based on expert systems have been proposed. System reconfiguration for restoration and loss minimization is one of the examples. Liu et al. have proposed an expert system to aid operators during the restoration process to quickly restore power to as many customers as possible [42]. The expert system module for restoration utilizes a rule base of 180 rules to help operators in devising a restoration plan. Also, new rules have been proposed to determine a feeder reconfiguration ensuring loss minimization while maintaining a reliable system configuration. The restoration planner block of the expert system is capable of coming up with a set of breaker and switch actions that restore maximal outage area. This is done by the dividing the entire outage area into groups of zones based on feeder adjacency. Keeping the capacity and operating constraints in mind, it is attempted to supply each group from a neighboring feeder. If the system has not been restored fully, zone restoration is performed. Finally, if a zone has still not been restored load transfers are performed to a farther feeder. The functional block dealing with loss minimization finds appropriate switching operations to achieve the operational goal i.e. loss minimization.

A real time expert system acting as a restoration guide has been proposed in [43] to assist operators in quick fault identification and system restoration. The proposed system consists of two operational modes. The on-line mode identifies faulty equipment, performs restoration planning, and provides restorative operation guidance. The offline mode is used by operators to learn restorative strategies. The knowledge base for the power restoration function is formulated with the cooperation of power system operators. The knowledge base is used for the purpose of faulty equipment identification, overload removal planning, outage load charge planning, and restoration operation guidance.

Application of expert systems to power system problems relating to diagnosis, control, and design has also been discussed in [41]. In the context of distribution feeder reconfiguration a rule based system has been proposed. Other areas of application include rotating machinery diagnostic aid and capacitor placement aid.

Chang et al. have discussed a knowledge based software package for the analysis and control of distribution networks in [44]. An expert system shell is used for the interpretation of human knowledge.

Fuzzy Logic

Fuzzy set theory is a means of specifying how well an object satisfies a vague description. Fuzzy sets do not have sharp boundaries and the truth value for any proposition is a number between 0 and 1 rather than being just true or false. Two methodologies based on the fuzzy set theory are fuzzy logic and fuzzy control. Fuzzy logic is a method for reasoning with logical expressions describing membership in fuzzy sets. Fuzzy control is a methodology for constructing control systems in which the mapping between real valued input and output parameters is represented by fuzzy rules [2].

Fuzzy set theory and logic have been applied to solve several power system problems. A heuristic based network configuration approach for loss minimization using fuzzy notation has been proposed in [33]. A switch searching strategy has been proposed. One of the criteria used for searching tie switches is based on voltage drop across these switches. Line loss has been used as another index to determine which tie switch to close. Similarly, for the sectionalizing switches if the transferrable current is very close to the optimal transferrable current, the increased loss is minimized. The searching strategies have been formulated in fuzzy notation. The proposed algorithm adopts a heuristic approach to determine an appropriate set of switching operations.

Song et al. in [45] have proposed an improved evolutionary programming technique for reconfiguration to ensure loss reduction. The performance of the evolutionary process is speeded up using a fuzzy logic controller based on heuristic information.

Another coordinated system of fuzzy logic and evolutionary programming to perform power system reconfiguration for loss reduction has been proposed in [31].

Genetic Algorithms

A genetic algorithm is a variant of stochastic beam search and generates successor states by combining two parent states rather than modifying a single state. The genetic algorithm begins with a set of K randomly generated states called the population. Each state or individual is represented as a string over a finite alphabet most commonly a string of 0s and 1s. The algorithm maintains a large population of states and new states are generated by mutation and by crossover, combining pairs of state from the population [2].

A variant of genetic algorithm is the parallel genetic algorithm (PGA) that performs the algorithm in parallel. The PGA can be coarse grain or fine grain. The coarse grain PGA maps distributed sub population with some strings to several processes and strings are sometime exchanges between sub populations during optimization. On the other hand, the fine grain PGA maps each string to each process and string information is exchanged between each process. The coarse grain approach is characterized by a few computationally intensive processes with little communication demands while the fine grain approach uses a large number of processes with low computational requirements but high communication demands for coordination purposes [46].

Genetic algorithm has been applied for the optimal network reconfiguration of a power distribution system in [47]. System loading conditions and maximum system loading capacity is determined using the system load balancing index (LBI). Genetic algorithm attempts to determine an optimal network reconfiguration that would minimize the index value. The genetic algorithm is used to determine the status of both tie and sectionalizing switches ensuring constraints for voltage magnitude and line flows are met. The LBI values for the different network configurations are obtained and the one with the minimum value is selected as the optimal solution.

Power system reconfiguration using genetic algorithm has also been proposed in [48]. The genetic algorithm uses a vertex encoding/decoding mechanism to determine a network configuration that minimizes losses.

A coarse grain parallel genetic algorithm has been applied for power system service restoration in [46]. The PGA is implemented on a Transputer that is a multiple instruction stream and

multiple data stream processor. Genetic algorithm based reconfiguration of distribution networks has also been proposed in [49].

2.2 Multiagent Systems for Power System Reconfiguration

Reconfiguration is the main domain of study for power system survivability. Reconfiguration is required not only for the normal and continued operation of a power grid but also for service restoration. After a fault occurs, restoring a power system to an optimal target configuration is required for continued operation. Power system reconfiguration and restoration have been addressed using several different approaches. These include heuristics [1-5],[13], expert systems[4],[6-11], mathematical programming [11], soft computing [12], artificial neural networks, and genetic algorithm [14],[15]. However, such solutions do not adequately address the requirements of modern power systems. Efficient monitoring, control, restoration, and reconfiguration of power systems require distributed decentralized control. Multiagent technology has emerged as a suitable candidate to achieve this distribution of control. Considering power system reconfiguration and restoration in particular, significant research is being done to apply multiagent systems for addressing the problem.

A multiagent based algorithm for the dynamic reconfiguration of a power distribution network has been proposed in [29]. The algorithm divides a day into several time intervals, with each time interval being managed by a work agent. A solution set is produced by applying an improved hybrid particle swarm optimization algorithm to solve the static reconfiguration. Each work agent has a solution set and a coordination agent selects a solution from each work agent's set. These selected solutions are coordinated until the number constraint of switching operations is satisfied.

Multiagent systems have been used to develop a self reconfigurable electric power distribution system in [18]. A shipboard power system based on the zonal architecture is used as the test bed. The proposed multiagent system attempts to reconfigure a power system while maximizing the number of high priority loads served. The prototype system is composed of two types of agents namely zone agents and server agents. Each zone agent represents a power system zone. In the event of a fault, the zone agent negotiates with its neighboring zone agents to reconfigure the system. The goal is to serve as many high priority loads as feasible. The server agents serve as

the communication interface between the zone agents and the distribution system. When a fault occurs, the server agent sends a request message to the zone agent reporting the new status of the bus connection and asking for control actions. The zone agent receives this message and the reconfiguration process is started. When the fault is cleared the same procedure is followed but the zone agent analyzes the cost path and adjusts the bus connections appropriately. All agent negotiations during the reconfiguration process are performed using the *Contract Net* protocol.

A resilient real time agent based system for reconfiguring a power grid has been proposed in [50]. The proposed approach uses intelligent agent coordination to prevent potential outages from happening. Since a power grid requires both reactive and proactive behaviors hybrid layered agent architecture has been used. Different layers of the architecture capture the different desired behaviors. Three types of agents constitute the multiagent system.

Reliability coordination agent - models the behavior of reliability controllers.

Utility agent - models the behavior of most field remote terminal units and also acts as an interface between the utility e.g. load, generator, compensator etc. and the remote terminal units.

Reconfigurable device agent - embedded in each FACTS device to facilitate the selection of control function to be executed in real time.

The multiagent design attempts to achieve real time responses by pushing the data processing and analysis closer to the source.

Multiagent systems have also been applied for reconfiguring aircrafts' on board power systems. In [51] Xiangnan Bian et al. have proposed a multiagent system for power system reconfiguration and restoration. The multiagent system has been applied to a More Electric Aircraft (MEA) power system. MEA is recognized as the future trendsetter in the aerospace industry and imposes increasing demands on the on board electric power system such as high fault tolerance and reliability. The test system has a redundant power distribution layout. The agents in the proposed approach have been broadly classified into three categories namely, action agents, equipment agents, and facilitator agents. The action agents perform actions such as opening and closing circuit breakers. Equipment agents actually represent the power generation units and buses. Their main tasks include power adjustments at the generation units and monitoring buses. The facilitator or negotiating agents monitor the action and equipment agents.

These agents are organized into two levels of decision making. The action agents and the equipment agents work at the operational level while the facilitator agents work at the negotiation level and are capable of communicating with their neighbors.

The architecture explained above has also been applied for the optimal reconfiguration of the navy ship power system [52]. The multiagent system has been applied to study the optimization of configuration and reconfiguration problem.

Reconfiguration of a power system gets complicated in the presence of distributed generation units. This problem has been addressed in [53]. Multiagent technology has been applied to develop a coordination scheme between diesel unit and energy capacitor system.

Gualdrón et al. in [54] introduce the idea of a self reconfigurable electric power distribution system based on multiagent technology. The multiagent system has been applied to a power system based on zonal architecture having an AC backbone and DC zonal system. The agent system comprises of zone agents and server agents. Zone agents perform the reconfiguration while server agents manage the communication between zone agents and the simulated power distribution system. The middleware enabling this communication has been developed using TCP sockets.

2.3 Other Multiagent Applications in Electric Power Systems

Multiagent systems have been applied for solving problems such as reconfiguration, restoration, fault identification, diagnosis and power system protection [16]-[19].

Agents and Electric Power Markets

The recent deregulation of the electric power industry has led to large scale decentralized markets where each participating entity has only local information and the global system view is available to none. Multiagent systems naturally capture this decentralization and have increasingly been applied to model power markets. One such model has been proposed in [55]. The research has applied agent based decentralized approach for solving DC optimal flow problem. The optimal flow problem involves optimizing the electricity production and prices over the entire system and is usually solved using Newton's method. This approach works well for smaller power systems but does not scale well as the system size increases. As an alternative, agent based approach has been applied in [55]. The proposed agent system is composed of

several agents with each agent controlling a single electrical node or bus. Each agent solves its local problem based on local information and also uses information gathered through communication. Agents also communicate the values of some of their variables to other agents such that they can improve their solution. The shared information includes voltage angles, local marginal prices of the energy. Agents announce their prices, adjust their generation levels, and modify their own prices until system equilibrium is achieved.

Decreasing electricity costs through competition is one of the main objectives of electricity markets. Several entities are part of this process and are involved in negotiations to achieve their objectives. A multiagent model representing all such entities in a power market and the relationships among them has been proposed in [56]. The multiagent system model includes a market facilitator agent, buyer agents, seller agents, trader agents, a market operator agent, and a system operator agent. The agents in the system can develop their own decision rules and objectives. Buyers, sellers, and traders exhibit a strategic behavior for defining prices. These agents get involved in a negotiation process and based on the results obtained modify strategies for the next period. The agents have two different types of strategies, time dependent and behavior dependent. Time dependent strategies are used to adjust the price according to the time remaining until the end of the negotiation period. Behavior dependent strategies, on the other hand, define the price for the next period based on the results obtained in the previous ones.

Multiagent based power market models have also been used for the effective operation of the Microgrid. A microgrid is a small scale power system consisting of distributed small scale power facilities such as solar power, wind power, and micro turbine generators. In [57] an agent based operation mechanism in a power market environment has been proposed. Also, a multiagent system based on the proposed mechanism has been designed. The multiagent system uses Contract-Net protocol for cooperative distributed problem solving.

With the deregulation of the power market, power generating companies will compete to sell power in an auction market. In competitive markets forming coalitions i.e. binding agreements to coordinate market strategies are common practice in order to maximize payoffs. Human decision makers involved with such markets need assistance to determine potential coalitions that they can be part of. An agent based approach where agents perform such negotiations on behalf of their human counterparts has been proposed in [58], [59]. The negotiation protocol used by agents to pick coalition partners has been discussed in [58]. The protocol has been derived from

cooperative game theory but does not require trustworthy information exchange between the potential partners. Each agent is associated with a human counterpart and attempts to maximize the payoff for its player. To achieve this, all possible potential coalitions are simulated and their attractiveness is evaluated. The agent then negotiates with other agents. The negotiation process consists of two main steps, prioritizing potential partners list and bargaining with other agents. The solution obtained as a result is proposed to the human player and feedback is obtained. The agent uses this feedback to augment its knowledge base for future games. This negotiation model has been applied to a three agent game corresponding to a three-player power exchange in [59].

Multiagent Systems for Power System Voltage Control

Power system voltage control is done at three levels to prevent voltage deterioration and for ensuring better use of reactive power sources. At the primary level, rapid and random voltage variations are compensated. The control devices achieve this by maintaining output variables close to the reference values. Slow and large voltage variations are handled by making use of regional information at the secondary level. At the tertiary level, voltage control is done by solving some optimization problems utilizing the global information. Agent based solutions for automated control of voltage variations is an active research area in power systems. An intelligent agent based solution for the regulation of power system voltage profile has been proposed in [60]. Coordinated secondary voltage control of the power system has been achieved using agent technology. Each power system voltage controller such as the static Var compensator (SVC), automatic voltage regulator (AVR), and static synchronous compensator (STATCOM) is considered an agent. Electrically close voltage controllers in a region work together as a multiagent system to manage the regional voltage profile. Task sharing is coordinated using two different schemes. The first one is based on communication where each agent attempts to sort voltage violations on its own first. If it fails, it requests other agents in its multiagent system to eliminate the voltage violations. The agent receiving the request performs secondary voltage control by modifying the reference settings of their primary voltage control, ensuring its self interests are protected. A response is then sent to the request sender. The request and response process is repeated until the voltage violation is completely eliminated. The second method for coordinating task sharing is based on local estimation. Each agent monitors the voltage level not only at its location but also at the adjacent agent's location. If a voltage violation is detected,

irrespective of the agent location the agent changes the reactive power supply through its secondary voltage control to eliminate the voltage violation.

A multiagent based method to overcome shortcomings of conventional secondary voltage control has also been proposed in [61]. In the agent system, the primary power system voltage controllers are represented as a set of execution agents and the secondary voltage controller as a coordinator agent. The multiagent system works as a traditional secondary voltage control system under normal conditions. However, in case of a contingency the agents cooperate and coordinate their efforts to eliminate the voltage violations. The coordination and cooperation among agents is achieved using the Contract-Net protocol.

Multiagent Systems for Power System Protection Coordination

A modern electric power system cannot operate without protection. Protective equipments in a power system function to ensure post fault protection of a power system. This is done by isolating the faulted parts of the power system from the rest of the network. Protection is required to remove faulty elements of the power system as soon as possible to ensure overall system safety [62]. All parts of a power system need to be adequately protected, as the possible implications of delayed fault clearance or an incorrect protection operation can be disastrous. There are a variety of power system protection schemes available. Each scheme depends on factors such as location and importance of the power system, the component to be protected, resource availability, and utility policies etc [63].

Considerable research is being done on the application of multiagent systems to this area of electrical power engineering. Wan et al. in [64] have applied agent technology to the protection coordination of industrial power distribution systems. In case of a fault the area isolated by the protective device should be as small as possible. Only the protective device closest to the fault must operate. Remote protection should only be provided in case of a device failure wherein the next upstream device or device combination operates. When two devices act in this fashion they are said to be coordinated. The proposed multiagent based protection coordination system consists of relay agents, distributed generator agents, and equipment agents. The relay agents represent relays in the power system and detect relay misoperation, breaker failures, and DG connection status. These agents also perform back up protection but perform much better than

the traditional method. The distributed generator agents communicate with the relay agents to update them about connection status. Equipment agents represent different power system equipments, collect local information, and operate upon the equipments. These agents also communicate with the relay agents to provide protection and coordination functions. The overall relay coordination for power system protection is a result of the coordinated efforts of all these agents.

Wong and Kalam in [63] have proposed an agent based solution for power system protection. In their work, bus bar and line protection has been considered under typical constraints and specifications. The proposed architecture consists of a group of loosely coupled and decentralized problem solving agents. These agents are classified into three types namely initiator agents which interact with the user or external program, control agents which are responsible for coordinating the actions of different agents, and case agents which represent an expert system in a power system component requiring protection. Each agent has only limited knowledge and therefore, has to cooperate with other agents to combine their knowledge. The agents in the system specialize in specific power system components and generate only partial solution. These partial solutions are then coordinated to achieve the final solution. A blackboard is used as the communication medium and also for integrating partial solutions. Power systems where distributed generators are in function require specialized protection schemes. The penetration of distributed generators affects the fault current level and its characteristics thereby influencing the traditional protection arrangements. A multiagent based protection coordination scheme for a power system with distributed generation units has been proposed in [65]. A decentralized multiagent based protection scheme for distributed generation system has been proposed in [66]. The proposed multiagent system is capable of detecting high impedance faults, fault location, and load shedding. The system functionality is achieved mainly by relay agents which are simply digital relays. These agents interact with other agents and are capable of autonomous or cooperative behavior to ensure system protection.

Chapter 3 Agents and Multiagent Based Systems

3.1 Distributed Artificial Intelligence

Distributed Artificial Intelligence (DAI) is an amalgam of ideas and concepts from several disciplines including computer science, economics, sociology, and philosophy etc. As described in [1]

"DAI is the study, construction, and application of multiagent systems, that is, systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks."

3.2 Distributing Intelligence

Large scale industrial applications used for the management of extended networks such as telecommunications, transport, and distribution of power are inherently distributed in nature. Overall system functionality is achieved through the interaction of a large number of participating entities, distributed at several nodes of the network. Each component has only a partial view but the overall objective is accomplished through the efficient coordination of all actions. Any software system designed for the automation of such systems would be required to incorporate this intrinsic distribution. Following are a few of the many possible reasons motivating the distribution of intelligence and control for the management of large scale commercial applications.

1. Problems are physically distributed.
2. Problems are functionally distributed. It is not possible for a single individual/component to handle all the problems.
3. Complexity of the problem requires a distributed and local view. Problems are too extensive to be analyzed as a whole.
4. Distributing intelligence also enables flexibility. Systems can better incorporate changes in the structure or the environment.
5. The structure of today's networks also enforces a distributed view.

3.3 What is an agent?

Researchers working in the fields of Distributed Artificial Intelligence and multiagent systems have developed several notions about agents. Although there is a lot in common, yet there are some notable differences depending on the application domain. The literature contains numerous definitions of the term agent. Some of these definitions have been provided below.

"Agents are autonomous computational entities that can be viewed as perceiving their environment through sensors and acting upon their environment through effectors."[1]

"An agent is a computer system that is situated in some environment, and is capable of autonomous action, in this environment to meet design objective."

In [2] an agent is defined as,

"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors"

Weiss in [1] distinguishes agents from intelligent agents. While any computer system interacting autonomously with an environment can qualify as an agent, the criteria for intelligent agents are stricter. As defined by Weiss,

"An intelligent agent is one that is capable of flexible autonomous action in order to meet its design objectives."

The term flexibility as used in the above definition involves three things.

Reactivity: It is the capability of an intelligent agent to perceive its environment and provide timely responses to environmental changes.

Pro-activeness: Intelligent agents exhibit goal directed behavior. Dynamic environments where environmental parameters change a lot require more than simple reactivity from the agents.

Social ability: Intelligent agents are capable of interacting with other entities or agents. For an intelligent agent social ability is not limited to simple sharing of information. Rather, it can involve complex scenarios of negotiation and cooperation among agents in an attempt to achieve their design objectives.

Agents have also been defined in terms of their behavioral characteristics. In [67] an agent is defined as,

"Software components that can communicate with their peers by exchanging messages in an expressive agent communication language. While agents can be as simple as subroutines, typically they are larger entities with some sort of persistent control."

Also,

"An entity is a software agent if and only if it communicates correctly in an Agent Communication Language."

In [68] the term Softbot has been applied to describe an agent.
"A softbot is an agent that interacts with a software environment by issuing commands and interpreting the environment's feedback."

A softbot possesses sensors and effectors in the form of commands. It is through these commands that the agent or softbot changes the environment and obtains information about it. A softbot as described in [68],

a) exhibits goal directed behavior,
b) can process natural languages,
c) is capable of moving and cloning,
d) continuously functions in its environment,
e) improves overtime through learning and adaptation,
f) interacts with other agents present in its environment,
g) is capable of using the same knowledge for multiple tasks, and
h) plans actions to achieve its goals. The execution of these is then monitored by the softbot and in case of problems error recovery is initiated.

Agents have also been characterized using mentalistic notions. As defined in [69],

"An agent is an entity whose state is viewed as consisting of mental components (e.g. beliefs, capabilities, choices, and commitment."

According to Ferber [70] an agent is a physical or virtual entity

a) which is capable of acting in an environment,
b) which can communicate directly with other agents,
c) which is driven by a set of tendencies (in the form of individual objectives or of a satisfaction/survival function which it tries to optimize),
d) which possesses resources of its own,
e) which is capable of perceiving its environment (but to a limited extent) ,
f) which has only a partial representation of this environment (and perhaps none at all),
g) which possesses skills and can offer services,
h) which may be able to reproduce itself, and
i) whose behavior tends towards satisfying its objectives, taking account of the resources and skills available to it and depending on its perception, its representations, and the communications it receive.

In [71] an agent has been defined as,

"Autonomous agents are computational systems that inhabit some complex, dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed."

According to Maes [71], depending upon their environment, agents can take a variety of forms. The range varies from autonomous robots in real environments, animated agents in simulated physical environments, to software agents in the digital world.

Smith et al. in [72] use agents to solve the end user programming problem i.e. enabling non-professional users to program computers. According to them,

"An agent is a persistent software entity dedicated to a specific purpose."

Several other definitions of agents can be found in [73], [74].

3.4 Multiagent Systems

The increasing networking of computers and the added complexity of the environments require more than a single agent to achieve system objectives. Although it is possible for a single agent to exist and be useful, generally agents work in groups or societies.

A Multiagent system (MAS) is a collection of collaborating intelligent and autonomous computational agents [1], perceiving the environment using sensors and acting upon it through actuators [2]. A Multiagent system is a type of distributed artificial intelligence system in which several agents coordinate their knowledge and activities and reason about the process of coordination.

No single agent is capable to achieve all system objectives. Therefore, agents share their skills and cooperate to solve problems that are beyond their individual capabilities. Since each agent is a separate thread of control, computations can be performed in parallel resulting in performance gains. A multiagent system also provides modularity, fault tolerance through redundancy, and brings together the knowledge and expertise of multiple experts.

3.5 Characteristics of Multiagent Systems

As discussed in [75] some of the defining characteristics of multiagent systems are:

1 Each agent has incomplete capabilities or information required for solving the problem at hand.
2 Global system control does not exist.
3 Data is decentralized.
4 Computation is asynchronous.

3.6 Levels of Organization

As defined in [70],

"An organization can be defined as an arrangement of relationships between components or individuals which produces a unit, or system, endowed with qualities not apprehended at the level of the components or individuals."

Three levels of organization can be observed in a multiagent system

Micro-social Level: This level is more interested in the interactions between agents and the kind of connections that exit between two agents or a small number of agents.

Groups: This level studies the differentiation of the roles and activities of the agents, the emergence of organization structures, and the aggregation of agents during the formations of these structures.

Global Societies: Dynamics of a large number of agents together with the general structure of the system and its evolution is considered at this level.

3.7 Characteristics of Applications Aimed by Multiagent Technology

Multiagent technology attempts to develop solutions that go beyond the conventional data processing techniques resulting in highly flexible and adaptive distributed software. Multiagent systems have been applied to a variety of domains including industrial manufacturing, production processes, analysis of business processes, electronic entertainment, and analysis of social phenomena. Applications using multiagent systems have one or several of the features noted below.

The application involves a significant degree of distribution. Different kinds of distributions taken into account are noted below. Application components and data can be

1. Geographically distributed.
2. Temporally distributed.
3. Semantically distributed.
4. Functionally distributed.

The complexity of the application cannot be addressed by a single, large, and centralized system.

3.8 Agent Cooperation

To fully harness the potential of multiagent technology, there has been an increasing interest in cooperative agents i.e. agents working towards a common goal. As discussed in [75] planning for a single agent involves devising a set of actions considering goals, capabilities, and environment constraints. On the other hand, planning for a multiagent system, in addition to these, considers the constraints imposed by the relationships that exist among agents and how each agent's actions are affected because of these. Therefore, it is important to understand the dynamics of how agents cooperate towards the accomplishment of a common goal.

"Cooperation is coordination among non-antagonistic agents." [1].

Methods of Cooperation

The different methods of cooperation employed by agents in a community to achieve overall system objectives have been listed below.

Grouping and Multiplication: This method draws the agents physically closer together. It is facilitated by creating either a homogeneous unit in space or by providing agents with the perception that they are side by side through a communication network.

Communication: Agents use this method of cooperation to "talk" to other agents and benefit from the knowledge possessed by other agents.

Specialization: Agents become more and more adapted to their task through the process of specialization.

Collaboration through Sharing: Agents are said to be collaborating when they work together on a common task. Collaborating agents distribute tasks, information, and resources to realize overall system functionality.

Coordination of Actions: Agent communities are always involved in a number of supplementary tasks not directly related to the job at hand. The purpose of such tasks is to ensure accomplishment of productive tasks under favorable conditions.

Conflict Resolution: Arbitration and negotiation are the two methods that are used for resolving conflicts in an agent community. Conflicts can arise because of a number of reasons including incompatible goals and insufficient resources.

Advantages of Cooperation

Cooperation enables agents to work together in a group yielding individual benefits for an agent and collective benefits for the entire agent community. Some of these benefits are listed below:

1 Cooperation enables the agent community to accomplish tasks otherwise impossible for an individual agent.

2 When agents work together on a task in a collaborative manner, productivity of each improves.
3 Agent cooperation can increase the number of tasks accomplished in a given amount of time.
4 Cooperation can reduce the time required for carrying out a task.
5 Cooperation can improve the use of available resources.

3.9 Benefits of Multiagent Systems

Multiagent technology has progressed from being considered only a research discipline to finding its way into commercial applications for solving real world problems. Multiagent systems offer a novel way for the analysis, design, and implementation of complex software systems. Some of the desirable properties and benefits offered by this technology have been noted below.

1. Multiagent systems are suitable for developing, understanding, and managing open, distributed, heterogeneous, large scale, dynamic, and flexible architectures.
2. Multiagent systems do not impose any a priori structure, thus allowing more flexibility. Almost all real world systems and applications are too complex to be completely characterized and precisely described. The flexibility offered by multiagent systems makes them a suitable match for such applications.
3. Multiagent systems capture the intelligence and interaction depicted by natural intelligent systems such as humans. Therefore, multiagent systems can offer insights and understanding about the dynamics of the interactions among natural intelligent beings, their organization into groups and societies to achieve improvement.
4. Multiagent systems take into account the adaptation and evolution necessary for achieving overall system objectives. They are distributed in nature and allow for easy integration, appearance, and disappearance of agents.
5. Multiagent systems can be powerful successors to the object oriented paradigm based on their ability to combine local behaviors with autonomy.
6. Multiagent systems enable distributed decision making. [1] , [70]
7. Harnessing the power of parallelism, multiagent systems can offer significant speed up and efficiency.

8. Multiagent systems can be seen as more reliable and robust. Being composed of numerous interacting and intelligent components or agents, failure of one or a few agents does not mean overall system failure.
9. Multiagent systems offer better scalability. Agents with similar or enhanced functionalities can always be introduced to deal with a larger problem.
10. It is possible to reuse agents participating in one multiagent system in several different application scenarios.

Chapter 4 Spectral Clustering

This research uses spectral clustering and therefore an overview of the spectral graph theory and the process of spectral clustering is discussed. However, before discussing the details of spectral clustering a general overview of commonly used graph notions is presented.

4.1 Graph Notations

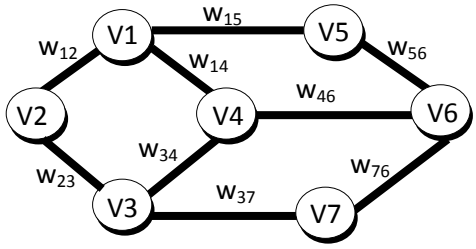

Figure 1 A weighted undirected graph

In this section several graph notations that have been used in this research are described. As shown in Figure 1 let

$\boldsymbol{G(V, E)}$ be an undirected weighted graph representing n data points.

$\boldsymbol{V = [V_1, V_2, \dots, V_n]}$ is the vertex set.

$\boldsymbol{E}$ denotes edges connecting vertices.

$\boldsymbol{w_{ij} \geq 0}$ is the non negative edge weight between vertex i and j.

If $\boldsymbol{w_{ij} = 0, V_i \, and \, V_j}$ are not connected.

$\boldsymbol{W = \left[w_{ij}\right]_{i,j=1,\dots,n}}$ is the weighted adjacency matrix.

$\boldsymbol{d_i = \sum_{j=1}^{n} w_{ij}}$ is the degree of a vertex $V_i \in V$.

- $\boldsymbol{D} = \begin{bmatrix} \boldsymbol{d1} & & \\ & \boldsymbol{d2} & \\ & & \ddots \\ & & & \boldsymbol{dn} \end{bmatrix}$ is the degree matrix with the degree of each vertex on the diagonal.

4.2 Graph Laplacians

The main tool used by any spectral clustering algorithm is the Laplacian matrix [76]. Spectral graph theory studies these matrices and analyzes the spectrum of the matrix representing a graph. Spectrum is defined as the eigen vectors of a graph, ordered by the magnitude of their corresponding eigen values. There are several different laplacian matrices with distinct characteristics.

Unnormalized Graph Laplacian

Consider an undirected weighted graph G, with non negative weight or adjacency matrix W and the degree matrix D. An unnormalized laplacian matrix is defined as [76],

$$\boldsymbol{L = D - W}$$

Some important properties of this matrix include:

1. L is symmetric and positive semi-definite.
2. The smallest eigen value of L is 0, and the corresponding eigen vector is the constant vector 1.
3. L has n non negative real valued eigen values

$$\boldsymbol{0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n}$$

4. For every vector $f \in R^n$ we have

$$\boldsymbol{\acute{f}Lf = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2}$$

5. The multiplicity of the eigen value 0 of L equals the number of connected components in the graph.

Normalized Graph Laplacian

A normalized laplacian matrix is defined as

$$\boldsymbol{L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}}$$

Important properties of this matrix are noted below:

1. L_{sym} is symmetric and positive semi-definite.
2. L_{sym} has n non negative real valued eigen values

$$\boldsymbol{0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n}$$

3. 0 is an eigen value of and L_{sym} the corresponding eigen vector is $\boldsymbol{D^{\frac{1}{2}}1}$. $\mathbf{1}$ is the constant one vector.
4. For every vector $f \in R^n$ we have

$$\boldsymbol{\acute{f} L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2}$$

5. The multiplicity of the eigen value 0 of L_{sym} equals the number of connected components in the graph.

4.3 Spectral Clustering

Clustering is the process of identifying the underlying structure in data and determining groups of similar behavior [77]. It is used for exploratory data analysis in fields such as computer science, biology, and social sciences. Spectral Clustering is a popular modern clustering algorithm based on spectral graph theory. Spectral clustering is a specialized technique for data analysis. Spectral clustering often outperforms traditional clustering approaches, is very easy to implement, and can be solved by standard linear algebra methods.

Similarity Graphs

Any clustering algorithm attempts to divide data points into groups such that points within the same group are similar to each other and dissimilar from those in other groups. Similarity graphs are a nice representation for data points to be analyzed [76]. Given a set of data points P,

$$\boldsymbol{P} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}]$$

and some notion of similarity between the data points i and j,

$$\boldsymbol{S_{ij}} \geq \boldsymbol{0}$$

then each vertex in the similarity graph represents a point in the data set. Two vertices are connected when the similarity between the two is positive or greater than a certain threshold. S_{ij} then becomes the edge weight. After obtaining the similarity graph representation of data the goal of the clustering algorithm is to find a graph partition such that the edge weights within a group are maximized and weights between groups are minimized.

The Process of Spectral Clustering

A general stepwise outline of the spectral clustering process is provided below.

1. Obtain a weighted undirected graph
    - Translate a set of data points into a weighted graph.
    - Each data point is represented by a vertex in the corresponding graph.
    - Graph connectivity emulates the connections between the data points.
    - Edge weights are determined based on some criteria.

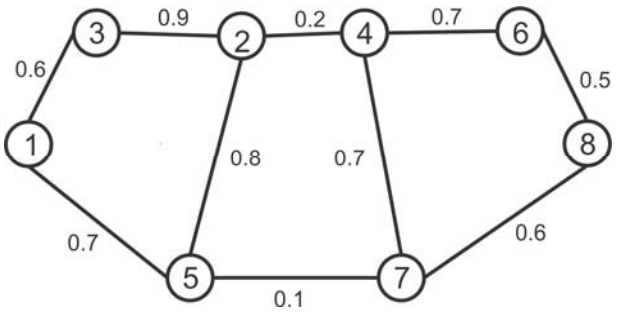

Figure 2 Weighted data graph

2. Compute the graph Laplacian matrix

$$\begin{array}{c} \\ X1 \\ X2 \\ X3 \\ X4 \\ X5 \\ X6 \\ X7 \\ X8 \end{array} \begin{bmatrix} & & & & & & & \\ 1.3 & 0 & -0.6 & 0 & -0.7 & 0 & 0 & 0 \\ 0 & 1.9 & -0.9 & -0.2 & -0.8 & 0 & 0 & 0 \\ -0.6 & -0.9 & 1.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.2 & 0 & 1.6 & 0 & -0.7 & -0.7 & 0 \\ -0.7 & -0.8 & 0 & 0 & 1.6 & 0 & -0.1 & 0 \\ 0 & 0 & 0 & -0.7 & 0 & 1.2 & 0 & -0.5 \\ 0 & 0 & 0 & -0.7 & -0.1 & 0 & 1.4 & -0.6 \\ 0 & 0 & 0 & 0 & 0 & -0.5 & -0.6 & 1.1 \end{bmatrix}$$

Figure 3 Generated Laplacian matrix

3. Apply Spectral Clustering

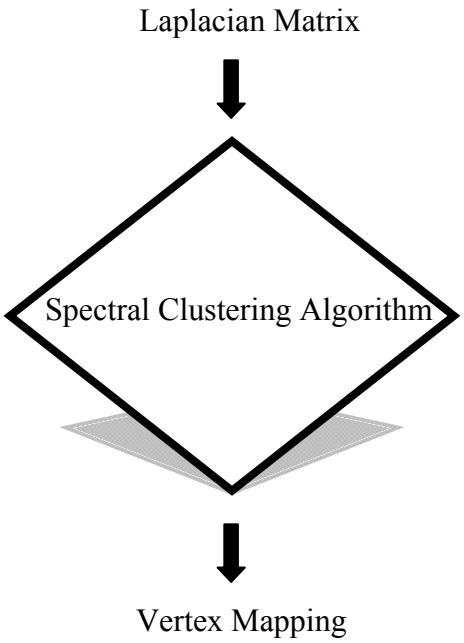

Figure 4 Applying spectral clustering

4. Compute Eigen Values and Eigen Vectors to determine final vertex mapping

Eigen Values

$$\begin{bmatrix} 3.2 \\ 2.6 \\ 1.7 \\ 1.4 \\ 1.3 \\ 1.2 \end{bmatrix}$$

Eigen Vectors

$$\begin{bmatrix} -0.3 & -0.1 & 0.5 & 0.5 & -0.2 & -0.1 \\ -0.6 & -0.1 & -0.4 & -0.4 & 0.2 & 0.1 \\ 0.5 & 0.1 & 0.4 & -0.6 & -0.1 & 0.1 \\ 0.2 & -0.6 & 0.1 & 0.1 & 0.4 & 0.4 \\ 0.5 & 0.1 & -0.6 & 0.4 & 0.1 & -0.1 \\ -0.1 & 0.4 & -0.1 & 0.1 & -0.5 & 0.6 \\ -0.1 & 0.5 & 0.2 & 0.1 & 0.6 & -0.4 \\ 0 & -0.3 & -0.1 & -0.2 & -0.4 & -0.6 \end{bmatrix}$$

Figure 5 Eigen values and eigen vectors

$$\begin{matrix} X1 \\ X2 \\ X3 \\ X4 \\ X5 \\ X6 \\ X7 \\ X8 \end{matrix} \begin{bmatrix} -0.2 \\ 0.2 \\ -0.1 \\ 0.4 \\ 0.1 \\ -0.5 \\ 0.6 \\ -0.4 \end{bmatrix}$$

Figure 6 Performing vertex mapping

5. Partition the original graph into clusters of related data

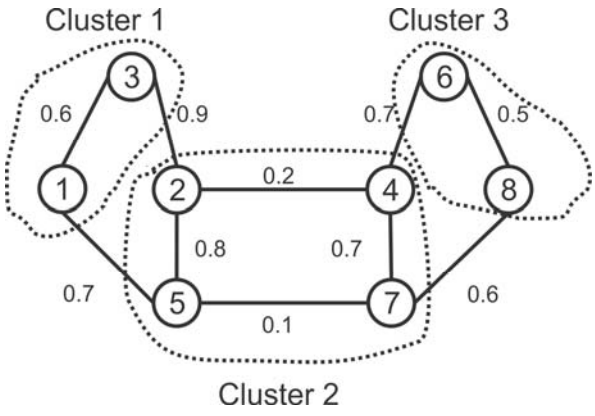

Figure 7 Clustered graph

4.4 Spectral Clustering Algorithms

A brief overview of some spectral clustering algorithms is presented in this section.

Unnormalized Spectral Clustering

An unnormalized spectral clustering algorithm has been presented in [76].

Input:

- A similarity matrix containing the pair wise similarities between data points
- The number K of clusters to be constructed.

Step 1: Construct a similarity graph and obtain its weighted adjacency matrix.

Step 2: Compute the unnormalized Laplacian matrix.

Step 3: Compute the first K eigen vectors of the laplacian matrix.

Step 4: Let U be the matrix containing the eigen vectors.

Step 5: Let Y_i be the vector corresponding to the i^{th} row of U.

Step 6: Cluster the points in $Y_{i,\, i = 1, \ldots, n}$ with the K-means algorithm into K clusters.

Normalized Spectral Clustering – Algorithm 1

Shi and Malik presented a normalized spectral clustering algorithm in [78]

Input:

- A similarity matrix containing the pair wise similarities between data points
- The number K of clusters to be constructed.

Step 1: Construct a similarity graph and obtain its weighted adjacency matrix.

Step 2: Compute the unnormalized Laplacian matrix.

Step 3: Compute the first K eigen vectors $u_1, u_2, \cdots u_k$ of the generalized problem

$$Lu = \lambda Du.$$

Step 4: Let U be the matrix containing the eigen vectors.

Step 5: Let Y_i be the vector corresponding to the i^{th} row of U.

Step 6: Cluster the points in $Y_{i,\ i = 1,\ \ldots,\ n}$ with the K-means algorithm into K clusters.

Normalized Spectral Clustering – Algorithm 2

Ng et al. presented a spectral clustering algorithm using the normalized Laplacian matrix [79]

Input:

- A similarity matrix containing the pair wise similarities between data points
- The number K of clusters to be constructed.

Step 1: Construct a similarity graph and obtain its weighted adjacency matrix.

Step 2: Compute the normalized Laplacian matrix.

Step 3: Compute the first K eigen vectors $u_1, u_2, \cdots u_k$ of the Laplacian matrix.

Step 4: Let U be the matrix containing the eigen vectors.

Step 5: Construct the matrix T using the formula,

$$\boldsymbol{t_{ij} = u_{ij} / \left(\sum_k u_{ik}^2 \right)^{\frac{1}{2}}}$$

Step 6: Let Y_i be the vector corresponding to the i^{th} row of T.

Step 7: Cluster the points in $Y_{i,\ i = 1,\ \ldots,\ n}$ with the K-means algorithm into K clusters.

Self Tuning Spectral Clustering

The self tuning spectral clustering algorithm has been proposed by Zelnik-Manor and Perona [80]. The algorithm uses local scales to ensure better clustering specially in cases when the data includes multiple scales or the clusters are placed in a cluttered background. Also, the use of structure of the eigen vectors to automatically determine the number of clusters has been suggested.

Step1: Compute a local scale for each point in the data set using the distance from the K^{th} neighbor,

$$\boldsymbol{\sigma_i = d(x_i, x_k)}$$

Step2: Compute the locally scaled affinity matrix, where affinity between a pair of points is defined as,

$$\boldsymbol{\widehat{A}_{ij} = \exp\left(\frac{-d^2(x_i, x_j)}{\sigma_i \sigma_j}\right)}$$

Step3: Create a diagonal matrix D such as,

$$\boldsymbol{d_{ii} = \sum_{j=1}^{n} \widehat{A}_{ij}}$$

Step4: Construct the normalized affinity matrix,

$$\boldsymbol{L = D^{-\frac{1}{2}} \widehat{A} D^{-\frac{1}{2}}}$$

Step5: Determine the C largest eigen vectors of L and construct the matrix X containing these vectors. C is the largest number of group possible.

Step6: Recover the rotation R which best aligns the columns of matrix X with the canonical coordinate system using the incremental gradient descent scheme. Let Z be the matrix obtained as a result i.e.

$$\mathbf{Z = XR}$$

Step7: Grade the cost of the alignment for each group number, up to C using the following cost function

$$\boldsymbol{M_i = max_j\, Z_{ij}}$$

$$\boldsymbol{J = \sum_{i=1}^{n} \sum_{j=1}^{C} \frac{Z_{ij}^2}{M_i^2}}$$

Step8: Set the final group number C_{best} to be the largest group number with minimal alignment cost.

Step9: Take the alignment result Z of the top C_{best} eigenvectors and assign the original point to cluster c if and only if,

$$\boldsymbol{max_j}\left(\boldsymbol{Z_{ij}}^{\boldsymbol{2}}\right) = \boldsymbol{Z_{ic}}^{\boldsymbol{2}}$$

Chapter 5 Proposed Reconfigurable Multiagent Architectures

In this research, we have developed clustering techniques to identify logical power system partitions such that agents can monitor these dedicated clusters or partitions rather than employ an agent to monitor each node or device. This reduces the resources utilized and the complexity of the system. Our proposed approach significantly reduces the number of agents in the system and the communication overhead while minimizing the system complexity and computational overhead. Furthermore, in the event of a fault, the affected nodes are quickly identified for reconfiguration. Before developing the scalable architecture a decentralized MAS architecture was developed to understand the challenges and requirements of a scalable reconfigurable multiagent architecture. The agents in this decentralized MAS use only local information and the MAS is topology-independent. We use the decentralized MAS as a baseline to compare the performance of the proposed scalable MAS and the time taken to reconfigure in the event a fault is detected. Simulations for the proposed approaches were performed on a prototype distribution system, Circuit of the Future (CoF) developed by Southern California Edison (SCE).

5.1 Testbed Power System – SCE's Circuit of the Future

Circuit of the Future project began in 2003 as an attempt to develop a test platform for incorporating new technologies to increase reliability, improve service quality, and control customer costs. The circuit was designed to test a variety of equipment types, protection schemes, and communication strategies. As shown in Figure 8, the CoF has a single substation with three main feeders. To enable flexible re-routing of power flow the feeders have been connected through seven normally open switches. There are 14 loads with a total real power demand of 24 MW and reactive power load of 12.96 MVar. There are 14 capacitor banks for reactive power (Var) support and two distributed generators for providing real power and reactive power. Figure 9 is a graphical representation of the Circuit of the Future (CoF). Every bus in the circuit is translated into a vertex in the graph. There are 66 buses in the CoF represented by the 66 vertexes of the graph. The edges in the graph emulate the physical connectivity between the buses. The seven normally open switches in the circuit are represented by dotted lines. Vertex/Node 1 represents the power source in the circuit.

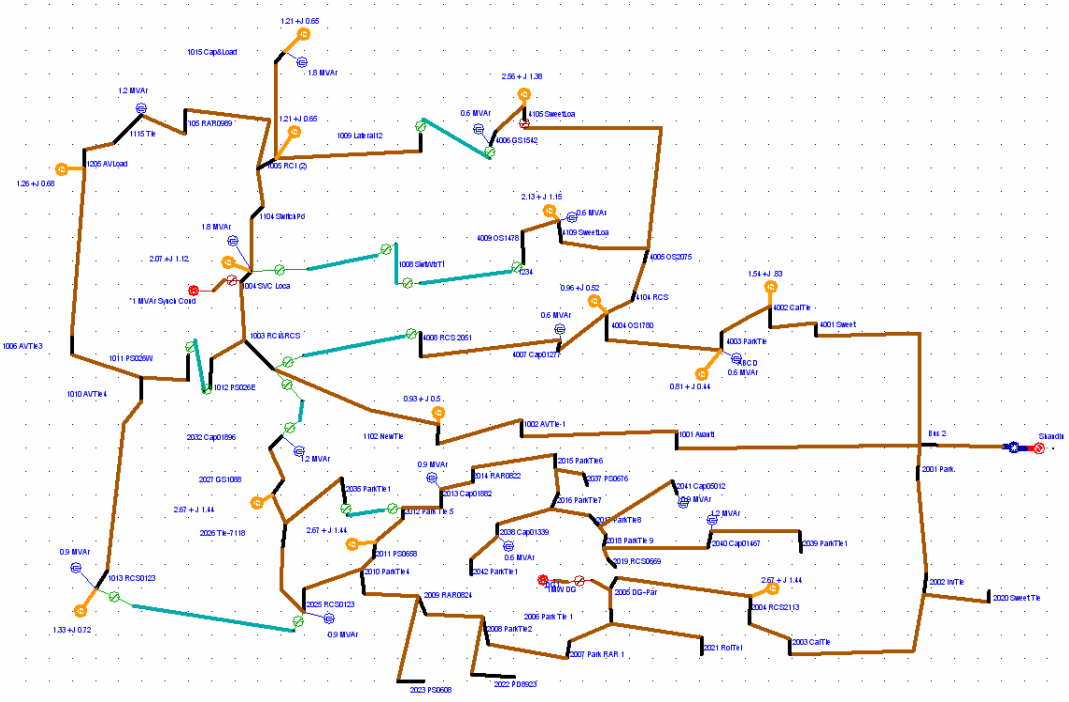

Figure 8 Southern California Edison's Circuit of the Future

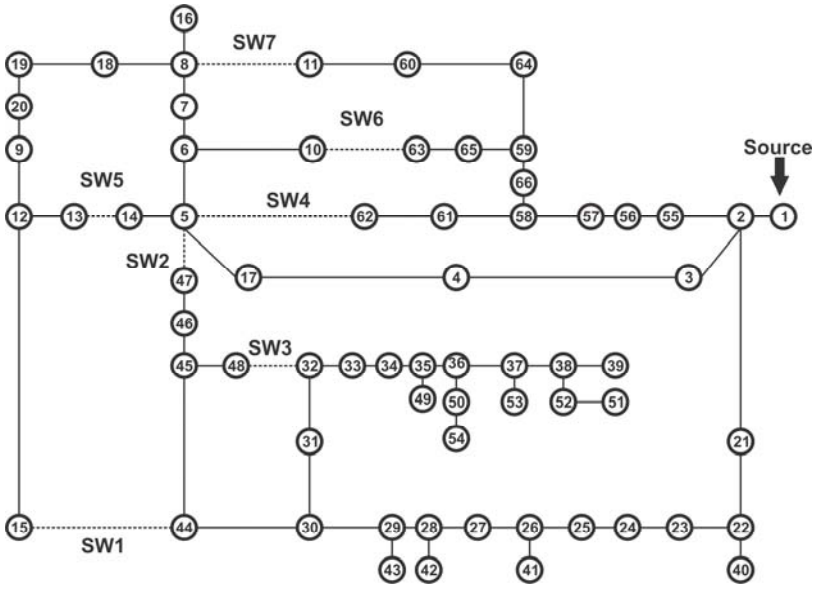

Figure 9 Circuit of the Future represented as a graph

In the next section details about the topology independent decentralized multiagent system have been proposed.

5.2 Proposed topology-independent decentralized multiagent system architecture

The proposed topology-independent MAS uses three types of agents and their functions are briefly described. *Bus Agent (BA)* represents buses in the power system. Agents representing neighboring buses are defined as neighbors in the corresponding multiagent system. As a completely decentralized approach, only local information is used by these agents. The agents have no topology information and only neighboring agents in the system are allowed to communicate with each other.

Processor Agent (PA) represents a common place in the multiagent system where information from other agents is accumulated. PA is the common coordinating entity in the system. It is used to collect the dynamic operational profile of the system and identify nodes that are affected in the system in the event of a fault. *Switch Agent (SA)* monitors and controls switches in the power system and performs the task of turning a switch on or off.

The proposed topology-independent decentralized MAS was applied to the *Circuit of the Future* (COF) system, developed by Southern California Edison (SCE) [81]. There are a total of 66 buses and 7 switches in the system requiring 66 *bus agents* and 7 *switch agents*.

Identification and reconfiguration of nodes affected by fault

As soon as a *bus agent* detects a loss of power, it starts communicating with its neighboring agents about the problem. A message is sent to each neighbor, to determine the status of power availability. If the neighboring agents also do not have power, they forward the message further to their neighbors. As the messages are forwarded, agents append their identifiers to the message. When the message reaches an agent whose corresponding bus has power or reaches an agent whose identifier is already in the message content, the forwarding stops. The flow of message also stops at the *terminal nodes* (TN) as shown in Figure 10. We define *terminal nodes* as buses corresponding to agents that have no neighbors except the sender of the message. When a message reaches a *terminal node*, the corresponding agent appends its identifier to the message content and forwards it to the *processor agent* (PA). The information about different paths a message took generates the knowledge of all nodes or buses involved in a fault. Therefore, every

bus agent where the message flow stops, forwards the final message it receives to the *processor agent*. The *processor agent* then processes the message received and compiles a list of all the agents affected by the fault.

While disseminating information about the fault, if a bus has a neighboring switch, the corresponding *bus agent* sends a message to the *bus agent* at the other end of the switch to determine if it has power. If the other *bus agent* informs that its corresponding bus has power, by turning the switch on, the de-energized area can be supplied power. A message is sent to the *processor agent* suggesting the switch position as a potential choice for reconfiguration. However, there can be situations where buses at both ends of the switch are without power. Such a switch is ruled out as a choice for reconfiguration. The *processor agent* compiles a list of all the suggested switch positions and then selects a switch or a possible combination for reconfiguration. *Switch agents* corresponding to these selected switches control the final switching operation for reconfiguration. The coordination of different tasks by all the agents involved in fault identification and reconfiguration is described in Algorithm I. A detailed explanation of every step in Algorithm 1 is provided in Appendix A.

Algorithm I

Algorithm for the Topology-Independent Decentralized MAS

(Function and coordination of agents after a fault is detected between Bus i and Bus j)

Bus Agent

BAj communicates with neighbors to disseminate fault information

An agent BAx receives a message and performs the following tasks:

IF BAx has power and a neighboring switch THEN

Inform sender that BAx is energized

ELSE

    IF BAx has neighboring switch THEN

    Check if bus at other end of switch is energized

        IF bus is energized THEN

        Suggest switch position to PA

ENDIF

ENDIF

IF BAx is at terminal node THEN

Send an intermediate list of agents affected by a fault to PA

ELSE

Add self to the list of agents affected by the fault

FOR all elements in the list of neighbors

Send the list generated in previous step

ENDFOR

ENDIF

Processor Agent

Determines all agents affected by the fault

Determines all potential switches for reconfiguration

Selects a switch for reconfiguration

Switch Agent

Performs the final task of changing the switch status

Using the circuit of the future testbed the algorithm is described for various fault scenarios and the interaction among agents to perform reconfiguration is discussed. With Bus 1 as the source node, suppose there is a fault between Bus 5 and Bus 17 as indicated by Fault 1 in the Figure 10. *Bus agent* at Bus 5 i.e. BA5 senses that it has no power and forwards the message to its neighbors BA17, BA6 and BA14. Since BA17 has power, the message flow stops there. The other neighbors of BA5 keep forwarding the message by adding their ID. When the message reaches BA6 it has two possible paths. One is to BA10 and the other is to BA7. The message flow continues as normal along the path through BA7. On reaching BA10, since it is a *terminal node* the message flow stops. At this point, BA10 forwards the message to the PA. The same steps take place at Buses 16, 15, 13, 14, since they are all *terminal nodes*. After consulting BA63, BA10 suggests to PA the neighboring switch SW6 as a possible option for reconfiguration. BA15, BA5, and BA8 also recommend their neighboring switches SW1, SW2

and SW4, SW7 respectively to PA. However, SW5 between Buses 13 and 14 does not make the list of possible switches for reconfiguration as buses at both end of the switch are de-energized.

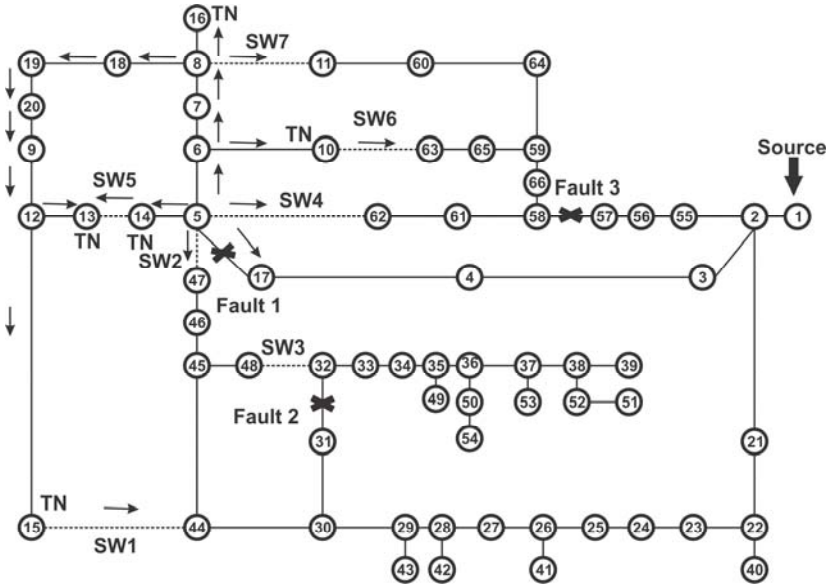

Figure 10 Graphical representation of SCE's Circuit of the Future and example fault scenarios

After *bus agents* have finished communicating, the *processor agent* will have acquired knowledge of the nodes affected by the fault. The *processor agent* then uses this information and suggested switch positions to direct the *switch agents* to perform the switching operation for reconfiguration.

All possible single fault scenarios for the test system are simulated using the proposed topology-independent decentralized multiagent system. Some example scenarios and results are listed in Table 1. Although the same number of nodes is involved in Fault 1 and Fault 2 in Table 1, the number of messages passed is different. The reason is that the communication involved in any fault scenario has two main components. The first component pertains to the messages for propagating the information about the fault. This happens between the *bus agents*. The second component is the communication between the *bus agents* and the *processor agent*. This depends on the number of *terminal nodes* affected by a fault and the number of nodes having neighboring switches. Therefore, although the first communication component in both the faults is the same, it is the second component that makes the difference. An analysis of the proposed topology-

independent decentralized MAS architecture and the agent interactions shows that the size of the multiagent system is very large. The agents in the system mostly perform simple message forwarding tasks while consuming resources.

Table 1 Simulation results of selected fault scenarios in the topology-independent decentralized MAS

Fault No.	Source Node	Destination Node	Nodes Affected	No. of Nodes Affected	Messages Passed	Time Taken (ms)	Suggested Switch Positions for
1	17	5	5, 6, 7, 8, 10, 16, 18, 19, 20, 9, 12, 13, 14, 15	14	49	40.7	SW-4, SW-2, SW-6, SW-7, or SW-1
2	31	32	32, 33, 34, 35, 36, 49, 50, 54, 37, 53, 38, 52, 39, 51	14	31	31.5	SW-3
3	57	58	65, 11, 60, 64, 59, 66, 58, 61, 63, 62	10	29	46.7	SW-4, SW-6, or SW-7

Also, the large communication overhead is due to communication between agents when a fault is detected and to determine the affected nodes. Depending on the number of nodes involved in a fault, this overhead can cause significant delays in decision making. After the affected nodes have been identified, a switching position is selected for reconfiguration. While the proposed approach works well for power systems of smaller size, for larger power systems we need a scalable solution to perform effective reconfiguration when a fault occurs. The design of a proposed scalable MAS architecture in presented in the next section.

5.3 Spectral Clustering Techniques for Power System Partitioning

In this section we propose a spectral clustering algorithm to logically partition the power network into clusters of connected buses. In the algorithm we select a clustering parameter that represents the notion of electric distances. To tune the clustering algorithm for partitioning a power system and to give importance to electrical properties of the system, the bus impedance matrix is used to acquire the necessary information for clustering. Agents are

then assigned to each cluster or partition. This modified approach scales well as the size of the power system increases by reducing both the total number of agents in the system and the communication overhead. However, to gain these benefits, the agents in the clusters acquire additional knowledge about the topology in their cluster.

By partitioning a power system network into strongly connected components significant performance gains are obtained. It has been shown that many control actions and small disturbances impact only a small portion of the system [82]. Also, power system applications that require load-flow calculations and contingency analysis take more time as the size of the matrices increases. So dimensionality reduction by focusing only on the most affected portion of the network can be highly efficient. We first represent the power system as a weighted graph $G(V, E)$ where every bus in the power system is defined as a vertex/node. The edges of the graph represent the connection between different buses in the power system and the edge weights are based on the electrical distances between different buses in the power system. It is important to take into consideration the electrical properties of the clusters, if a partitioning algorithm is to be applied to a power system graph. We have chosen the electrical distances between different buses as the clustering parameter. These distances are obtained from the bus impedance matrix, *Zbus*. Each element in the symmetric *Zbus* matrix indicates the electrical closeness of two buses in a power system. The closer the buses in the power system lower the value of the corresponding *Zbus* elements. In our clustering algorithm, the final edge weights are obtained by inverting the absolute values of the *Zbus* matrix so that the edge weights are higher when the buses are closer and vice versa.

Spectral clustering for partitioning a power system

Clustering is the process of identifying the underlying structure in data and determining groups of similar behavior [77]. Spectral clustering algorithm is based on the spectral graph theory. The spectrum of a matrix representing a graph is analyzed by computing the eigen vectors of a graph, and ordering by the magnitude of their corresponding eigen values. Spectral clustering algorithms usually outperform traditional clustering algorithms [76]. The clustering algorithm used in the proposed architecture is based on the work done by Zelnik-Manor and Perona [80] and Ng et al. [79]. Their work introduces local scales to improve clustering performance and the

automatic determination of number of groups that naturally exist in the data by exploiting the eigen vector structure. We briefly present a general outline of the spectral clustering algorithm.

First, a matrix representation of the large scale structure is constructed. Eigen values and eigen vectors of the matrix are calculated. These eigen values and vectors provide global information about the structure of the matrix and its connectivity. Also, a mapping of data points to a lower dimensional representation is performed based on one or more eigen vectors. Finally, the data points are assigned to different clusters. We tailor the spectral clustering algorithm for partitioning a power system. The modified algorithm is used on a WSCC-9 bus system [83].

Let S represent the set of buses in the power system. Then,

$$\boldsymbol{N = |S|}$$

The N x N adjacency matrix is,

$$\boldsymbol{A = [A_{ij}]}$$

$$\boldsymbol{Z = [Zbus_{ij}]}$$

Elements of the adjacency matrix are calculated

$$\boldsymbol{a_{ij} = \frac{1}{abs(Z_{ij})}}$$

After obtaining the adjacency matrix, the N x N diagonal degree matrix is obtained using

$$\boldsymbol{D(i,j) = \sum_{j=1}^{N} A(i,j)}$$

The spectral clustering algorithm uses the normalized Laplacian matrix defined as,

$$\boldsymbol{\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}}$$

$\hat{A}$ contains all the information necessary to perform spectral clustering. The eigen vectors and eigen values of this matrix are used to determine the specified number of groups K, in the original power system. The K largest eigen vectors of $\hat{A}$ are then used to obtain an N x K matrix, V . The rows of V are renormalized to have unit length generating an N x K matrix Y. Each row

of *Y* acts as a point in the *K* dimensional space. K-means algorithm is then applied on this matrix for clustering.

The buses in the WSCC-9 bus power system are transformed into 9 vertices of a graph. The edges in the graph represent the connectivity between the buses. The WSCC-9 bus system and the corresponding graph are shown in Figure 11. We first construct the adjacency matrix *A* from the weighted graph. There are a total of nine buses in the power system. This results in a 9 x 9 adjacency matrix representing the adjacency between the buses of the power system.

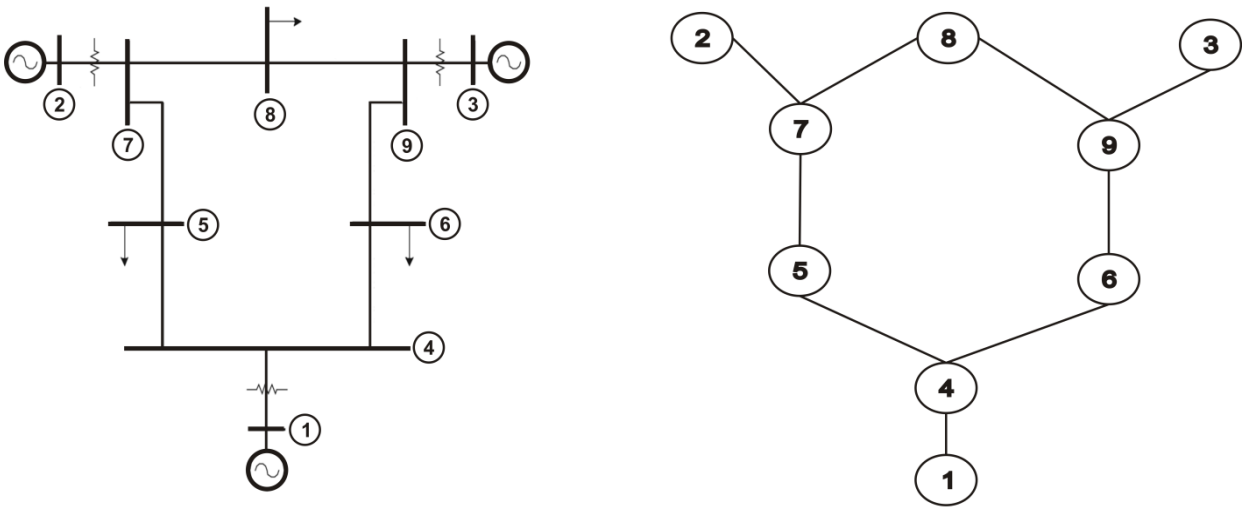

Figure 11 WSCC-9 bus system and its corresponding graph

In our proposed algorithm we modified the adjacency matrix to represent the notion of electric distances [84]. A bus in a power system can be electrically closer to another, although not directly connected. To give importance to such electrical properties and to fully capture the information provided by the *Zbus*, the modified adjacency matrix we used has a different structure. The resulting 9 x 9 adjacency matrix *A* for the WSCC-9 bus system is shown. It can be observed that the diagonal entries in the matrix *A* are the largest in each row since a bus is electrically closest to itself. We have experimentally verified that this choice of adjacency matrix based on electrical properties of the system generated better results. The 9 x 9 diagonal degree matrix D is obtained from the adjacency matrix. Finally, using both the adjacency matrix and the degree matrix, we obtain the 9 x 9 Laplacian matrix $\hat{A}$. The matrix $\hat{A}$ for the WSCC-9 bus system is shown. The eigen vectors and eigen values of the Laplacian matrix are calculated.

From these eigen vectors the 9 x 3 matrix V is constructed, where v_1, v_2, and v_3 are the three largest eigen vectors of $\hat{A}$. The structure of the eigen vectors of the Laplacian matrix encodes the structure of the resulting cluters.

$$\boldsymbol{A} = \begin{array}{c} \\ Bus\ 1 \\ Bus\ 2 \\ Bus\ 3 \\ Bus\ 4 \\ Bus\ 5 \\ Bus\ 6 \\ Bus\ 7 \\ Bus\ 8 \\ Bus\ 9 \end{array} \begin{bmatrix} Bus\ 1 & Bus\ 2 & Bus\ 3 & Bus\ 4 & Bus\ 5 & Bus\ 6 & Bus\ 7 & Bus\ 8 & Bus\ 9 \\ 1.6039 & 1.3134 & 1.3082 & 1.4702 & 1.3930 & 1.3873 & 1.3134 & 1.3014 & 1.3082 \\ 1.3134 & 1.6144 & 1.3362 & 1.3134 & 1.3443 & 1.3010 & 1.4664 & 1.3986 & 1.3362 \\ 1.3082 & 1.3362 & 1.6001 & 1.3082 & 1.3001 & 1.3378 & 1.3362 & 1.3758 & 1.4630 \\ 1.4702 & 1.3134 & 1.3082 & 1.4702 & 1.3930 & 1.3873 & 1.3134 & 1.3014 & 1.3082 \\ 1.3930 & 1.3443 & 1.3001 & 1.3930 & 1.4697 & 1.3380 & 1.3443 & 1.3156 & 1.3001 \\ 1.3873 & 1.3010 & 1.3378 & 1.3873 & 1.3380 & 1.4682 & 1.3010 & 1.3062 & 1.3378 \\ 1.3134 & 1.4664 & 1.3362 & 1.3134 & 1.3443 & 1.3010 & 1.4664 & 1.3986 & 1.3362 \\ 1.3014 & 1.3986 & 1.3758 & 1.3014 & 1.3156 & 1.3062 & 1.3986 & 1.4639 & 1.3758 \\ 1.3082 & 1.3362 & 1.4630 & 1.3082 & 1.3001 & 1.3378 & 1.3362 & 1.3758 & 1.4630 \end{bmatrix}$$

$$\boldsymbol{D} = \begin{array}{c} \\ Bus\ 1 \\ Bus\ 2 \\ Bus\ 3 \\ Bus\ 4 \\ Bus\ 5 \\ Bus\ 6 \\ Bus\ 7 \\ Bus\ 8 \\ Bus\ 9 \end{array} \begin{bmatrix} Bus\ 1 & Bus\ 2 & Bus\ 3 & Bus\ 4 & Bus\ 5 & Bus\ 6 & Bus\ 7 & Bus\ 8 & Bus\ 9 \\ 12.399 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 12.454 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 12.396 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12.265 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12.198 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 12.165 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12.276 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12.237 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12.228 \end{bmatrix}$$

$$\hat{\boldsymbol{A}} = \begin{array}{c} \\ Bus\ 1 \\ Bus\ 2 \\ Bus\ 3 \\ Bus\ 4 \\ Bus\ 5 \\ Bus\ 6 \\ Bus\ 7 \\ Bus\ 8 \\ Bus\ 9 \end{array} \begin{bmatrix} Bus\ 1 & Bus\ 2 & Bus\ 3 & Bus\ 4 & Bus\ 5 & Bus\ 6 & Bus\ 7 & Bus\ 8 & Bus\ 9 \\ 0 & 0.129 & 0.129 & 0.116 & 0.120 & 0.120 & 0.129 & 0.129 & 0.129 \\ 0.129 & 0 & 0.127 & 0.129 & 0.124 & 0.128 & 0.116 & 0.120 & 0.127 \\ 0.129 & 0.127 & 0 & 0.129 & 0.128 & 0.124 & 0.127 & 0.122 & 0.116 \\ 0.116 & 0.129 & 0.129 & 0 & 0.120 & 0.120 & 0.129 & 0.129 & 0.129 \\ 0.120 & 0.124 & 0.128 & 0.120 & 0 & 0.123 & 0.124 & 0.125 & 0.128 \\ 0.120 & 0.128 & 0.124 & 0.120 & 0.123 & 0 & 0.128 & 0.126 & 0.124 \\ 0.129 & 0.116 & 0.127 & 0.129 & 0.124 & 0.128 & 0 & 0.120 & 0.127 \\ 0.129 & 0.120 & 0.122 & 0.129 & 0.125 & 0.126 & 0.120 & 0 & 0.122 \\ 0.129 & 0.127 & 0.116 & 0.129 & 0.128 & 0.124 & 0.127 & 0.122 & 0 \end{bmatrix}$$

$$\boldsymbol{V} = \begin{array}{c} \\ Bus\ 1 \\ Bus\ 2 \\ Bus\ 3 \\ Bus\ 4 \\ Bus\ 5 \\ Bus\ 6 \\ Bus\ 7 \\ Bus\ 8 \\ Bus\ 9 \end{array} \begin{bmatrix} \boldsymbol{V_1} & \mathbf{V_2} & \mathbf{V_3} \\ -0.3344 & -0.4768 & 0.0161 \\ -0.3344 & 0.2936 & 0.4534 \\ -0.3344 & 0.2724 & -0.4998 \\ -0.3344 & -0.4768 & 0.0161 \\ -0.3316 & -0.2469 & 0.2048 \\ -0.3318 & -0.2499 & -0.2063 \\ -0.3344 & 0.2936 & 0.4534 \\ -0.3312 & 0.3181 & 0.0641 \\ -0.3344 & 0.2724 & -0.4998 \end{bmatrix}$$

A closer look at the second column corresponding to v_2 shows the structure of the resulting clusters or partitions. Buses that will be placed in the same cluster have very similar values, differing by a small fraction. All buses having negative eigen values in eigen vector v_2 will be grouped in the same cluster. Cluster 1 includes Buses 1, 4, 5, and 6. Similarly Cluster 2 includes Buses 3 and 9, and Cluster 3 includes Buses 2, 7, and 8. Figure 12 illustrates the results of the clustering process. The proposed spectral clustering algorithm was extended to the Southern California Edison's *Circuit of the Future* (CoF), which is a larger system compared to the WSCC-9 bus system [84]. As shown in Figure 9 there are a total of 66 buses in this system [81]. A 66 x 66 Laplacian matrix is obtained. As before, the eigen values are used to group the 66 buses into 5 clusters. Table 2 summarizes the details of the five clusters in the CoF after applying the proposed spectral clustering algorithm.

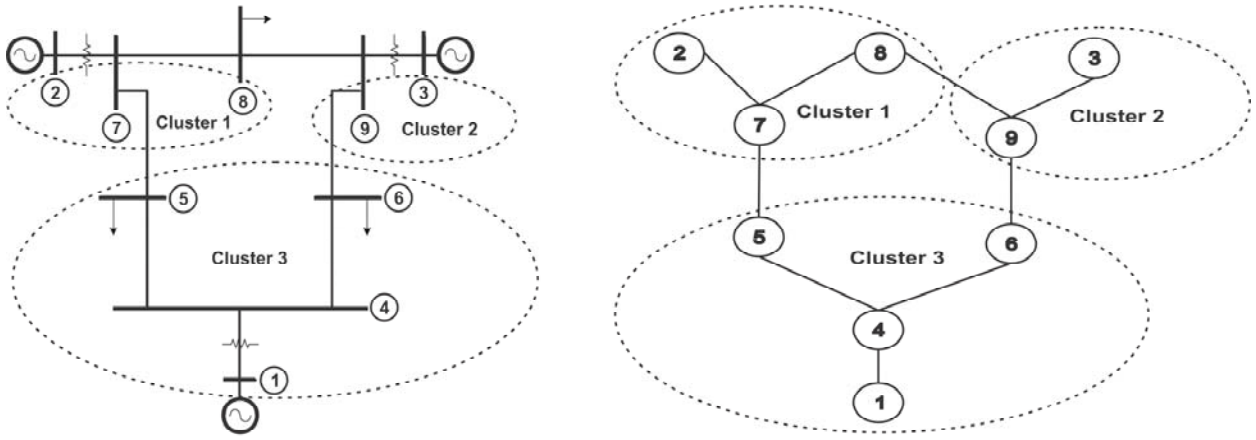

Figure 12 Clustered WSCC-9 bus system and its corresponding graph

Table 2 Results obtained after applying spectral clustering on SCE's Circuit of the Future

Cluster No.	Total Bus Count	Buses in the Cluster
Cluster 1	22	1, 2, 3, 55, 56, 57, 58, 61, 62, 66, 59, 65, 63, 64, 60, 11, 21, 22, 40, 23, 24, 25
Cluster 2	16	4, 17, 5, 6, 14, 7, 10, 8, 16, 18, 19, 20, 9, 12, 13, 15
Cluster 3	13	47, 46, 45, 44, 48, 30, 29, 28, 43, 42, 27, 26, 41
Cluster 4	6	31, 32, 33, 34, 35, 49
Cluster 5	9	36, 50, 54, 37, 53, 38, 52, 39, 51

5.4 Proposed Scalable Multiagent System Architecture

In this section we describe a proposed scalable multiagent architecture using fewer agents . The proposed MAS architecture uses two types of agents. *Cluster Agent* is assigned to each cluster or partition instead of assigning an agent to each bus as proposed in the decentralized topology-independent MAS. This results in a smaller agent system. We designate these agents as *cluster agents* or CAs [84]. Figure 13 shows the 5 *cluster agents* CA1-CA5 and 7 reconfiguration switches, SW1-SW7. These *cluster agents* can communicate with each other and work in coordination to perform different functions. By dividing the large system into logical clusters and assigning these to different agents we reduce the size and complexity of the problem for each agent, thereby making the load flow calculations, monitoring, reconfiguration, or

restoration efficient and manageable. *Switch Agent* turns a reconfiguration switch on or off based on the chosen path and given constraints.

Functionality of the proposed scalable multiagent system

Each CA has knowledge of the topology of the nodes or buses in the cluster it is monitoring. It is provided in the form of a matrix which contains the weighted adjacency information of only the buses in the cluster associated with an agent. For example, there are 16 buses in the cluster associated with CA2, and so a 16 x 16 symmetric matrix is provided to the associated CA.

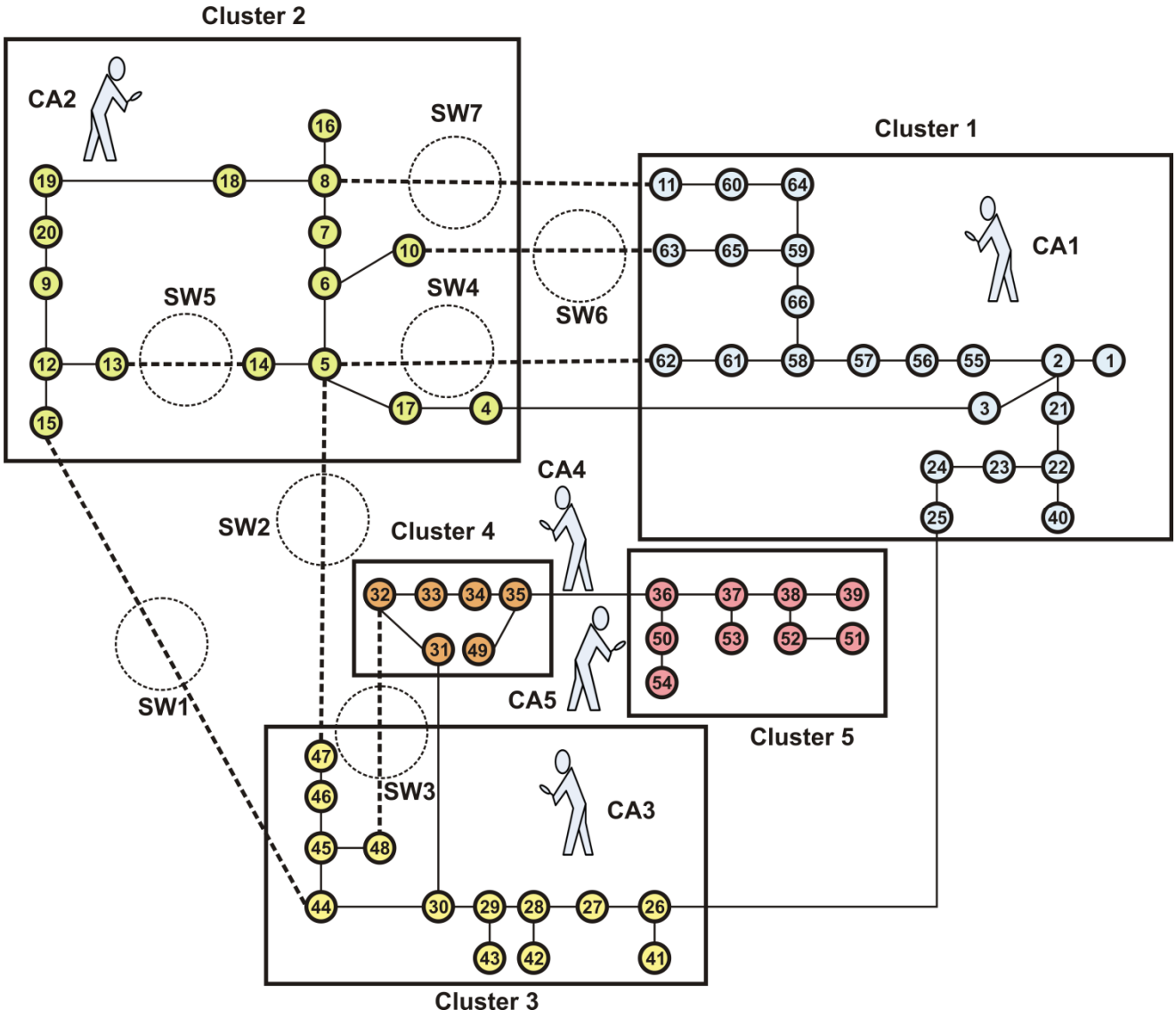

Figure 13 Assigning cluster agent CA_i to each cluster obtained in SCE's Circuit of the Future

Thus each agent has a lower dimensional matrix compared to the actual system size. The overall algorithm describing the interaction between agents in the proposed scalable MAS is described in Algorithm II [84]. A detailed explanation of every step in Algorithm II has been provided in Appendix A.

The edge weights forming entries of the weighted adjacency matrix can represent different constraints in the system. Once a fault is detected, the CAs can identify possible solutions and an optimal solution is selected based on these constraints. For example, edge weights can represent the power flow between each node. These edge weights can also be assigned the values of voltage drop between nodes. In this manner we can identify paths which have minimum voltage drops. Also, priorities can be encoded into these edge weights. For example, if certain paths in the power system are known to be more reliable than others, this information can be encoded in the edge weights such that their priority is reflected in the decision making process. A cost function is used to identify an optimal solution for reconfiguration, depending on system constraints. We define the cost function as,

$$\boldsymbol{Cost} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{K}} \boldsymbol{C_i W_i}$$

Algorithm II

Algorithm for the Proposed Scalable MAS

Step 1: Obtain a weighted graph G(V,E) from the power system

Step 2: Apply spectral clustering algorithm to obtain K partitions

Step 3: Assign agents to clusters obtained

Interaction of cluster agent CA_j with a faulty cluster agent CA_i

Step 4: CA_j sends a list of potential source nodes and associated costs to CA_i

For a cluster agent with fault

Step 4a: Dynamically update model of cluster topology

Step 5: CA_i communicates with its neighboring *cluster agents* for finding a solution

Step 6: CA_i determines nodes affected by the fault

Step 7: CA_i performs analysis of each alternate source node Si in parallel

Step 8: CA_i evaluates potential reconfiguration solutions from neighbors

Step 9: CA_i determines a feasible best solution based on constraints

Step 10: CA_i communicates with selected *switch agent* to perform the final reconfiguration

where W_i represents the weight of each edge in the path, K is the length of the path and C_i is a Cost Adjustment Factor, which models different constraints. For example, it can be used to give importance to load priorities when both a high priority load and a low priority load need power at the same time. C_i can be adjusted to minimize the cost of the path to the high priority load making sure it is served before any other load in the system, or C_i can be adjusted to select a more reliable path by reducing the overall cost of the path. Since our aim for this case study is the demonstration of effectiveness of *cluster agents* interaction, we consider simple fault scenarios resulting in loss of line. A fault is modeled as a loss of line between two buses in the power system where certain nodes in the power system are unsupplied.

Identification and reconfiguration of nodes affected by fault

When a fault is first detected in a particular cluster, the *cluster agent* associated with that cluster updates its model of cluster topology. It then determines the nodes affected by the fault. The agent determines the minimal cost of paths between its current source node and all other nodes in its graph structure. The agent also finds the parent nodes of all the nodes involved in the shortest path from the source node. In this manner an agent acquires knowledge of the paths and the associated costs for all nodes in its cluster. After application of this algorithm, all the nodes for which the cost of the path is greater than a maximum value are identified as those affected by the fault. Certain faults can propagate from one cluster to neighboring clusters. Under such circumstances, the *cluster agent* associated with the cluster where the fault actually occurs first identifies a list of neighbors which can be potentially affected by the current fault in the system. The *cluster agent* then informs these neighbors about the problem. The neighboring *cluster agents* also update their model of the cluster topology and pass this information to its neighbors which can be affected by the newly introduced fault in their cluster. All affected *cluster agents*

get involved in the process of reconfiguration in parallel [84]. Figure 14 shows such a scenario where a fault in Cluster 1 propagates to Cluster 3, Cluster 4, and Cluster 5. The affected buses in each cluster are highlighted in gray. The four *cluster agents* affected by the fault between Bus 21 and Bus 22 in Cluster 1, work together to reach a decision for reconfiguration.

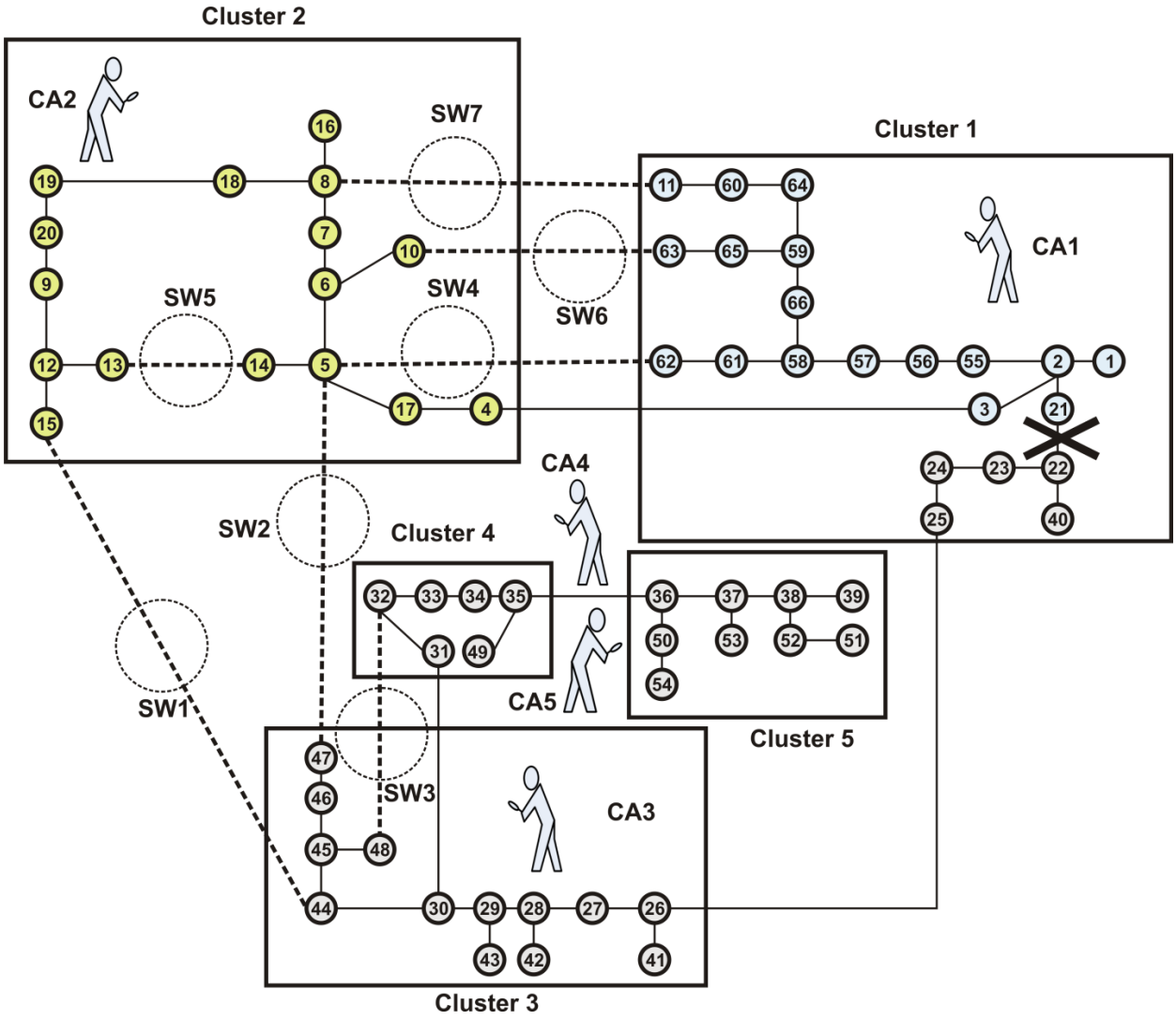

Figure 14 Faults propagating between clusters in SCE's Circuit of the Future

Identification of alternate source nodes

For a cluster, a source node is defined as any node which is connected to a node in some other cluster. These are called source nodes because they have the potential to provide power to the de-

energized area by switching on the nearest reconfiguration switch. The agent acquires the position of source nodes at start up. Some of these source nodes will be at a location and will have connectivity, such that they can never provide power to the de-energized buses. For a particular source node, these will be the buses un-reachable from it based on the new network connectivity information after the fault. So the agent tries to rule out all such source nodes. The *cluster agent* for the affected cluster also considers the option of internal reconfiguration if it has switches inside its cluster. Meanwhile, agents in the neighboring clusters also perform their computations for finding the best route to redirect the power to the faulty area. It is important to note that since each agent is a separate thread, all these computations are performed in parallel. The neighboring agents determine the cost from their currently active source node to the nodes which can potentially provide power to the affected cluster. This information is shared with the CA in the affected cluster. Since time is critical in such situations, to reach a decision quickly the CA determines the nodes which cannot act as source node under the current fault and rules them out. For the rest of the nodes, the *cluster agent* chooses the source node with the minimal cost to the affected nodes. In this manner the agent with the help of its neighbors determines the optimal path for reconfiguration, after a fault is detected. The switching action is then performed to energize the unsupplied buses through the newly configured path. As an example, let us assume there is a fault between Bus 7 and Bus 8, in the cluster assigned to CA2 as shown in Figure 15 [84]. When the fault is detected, CA2 communicates with its neighbors CA1 and CA3, to initiate the task of identifying possible reconfigurable paths and computing their associated costs. CA2 determines the nodes affected by the fault. The affected nodes are highlighted in gray in Figure 15. The neighboring agents perform their calculations to suggest alternate routes and associated costs to CA2. CA2 determines the feasibility of different source nodes. Notice in this case that switching on SW2, SW4, or SW6 cannot solve the problem since they will not be able to supply power to the affected nodes. So CA2 rules these out. The potential alternate source nodes are identified as Node 11 in Cluster 1, and Node 44 in Cluster 3. The third option involves internal switching within Cluster 2 between Nodes 13 and 14. Out of these options the agent selects the one with the minimum cost. We simulated all possible single faults for SCE's *Circuit of the Future*. Some example scenarios of faults in each cluster of the scalable multiagent architecture are analyzed.

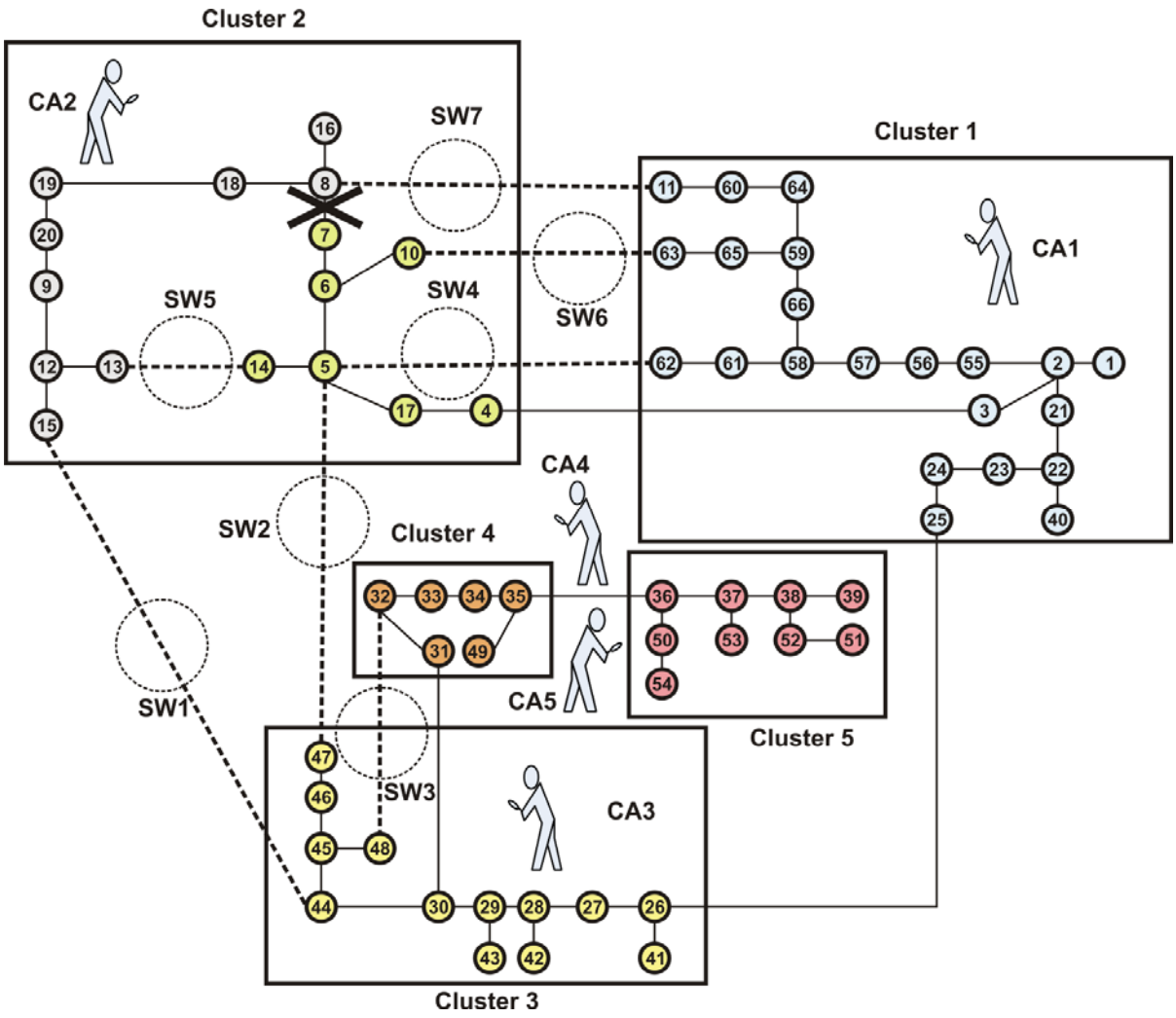

Figure 15 Example fault scenarios in the SCE's Circuit of the Future

The number of messages passed, possible switches needed for reconfiguration and the associated cost, and the time taken to arrive at a solution are summarized in Table 3[84].

Implementation Details of the Proposed Multiagent System

The proposed multiagent system is implemented using JADE (Java Agent Development Environment). The JADE framework simplifies the development and run-time execution of multiagent applications through a FIPA (Foundation for Intelligent Physical Agents) compliant middle-ware. It allows distribution of the agent platform across multiple machines and allows

Table 3 Simulation results of selected fault scenarios in the proposed scalable MAS

Cluster No.	Source Node	Destination Node	Clusters Affected by a Fault	Alternate Source Nodes	Switch for Reconfiguration	Cost	Messages Passed	Time (ms)
C1	2	55	C1	8 10 5	SW7 SW6 SW4	5 4 2	4	24.9
C1	1	2	C1, C2, C3, C4, C5	None	NA	NA	20	35.8
C2	7	8	C2	14 11 44	SW5 SW7 SW1	3 10 5	4	17.1
C3	27	28	C3, C4, C5	15 5	SW1 SW2	11 2	12	36.1
C4	31	32	C4, C5	48	SW3	7	5	32.9
C5	36	50	C5	None	NA	NA	2	14.2
Between C1 and C2	3	4	C2	62 11 63 47 44	SW4 SW7 SW6 SW2 SW1	7 10 9 8 5	4	23.5

controlling the configuration through a remote GUI. A homogeneous set of APIs (Application Programming Interfaces) is provided that do not depend on the underlying network or the Java language version. A simple set of APIs hide the complexity of the middleware. The communication infrastructure used by agents in the proposed MAS is provided by JADE. JADE follows FIPA standards which enable agents written in different languages and running on different platforms to communicate with each other. JADE messages adhere to ACL (Agent

Communication Language) standards. An ACL message has several attributes such as *performative*, *receiver*, *sender*, and *content*. *Performative* attribute defines the type of the message and agents in the implemented MAS take different actions depending upon the message type. For example, all *cluster agents* suggesting alternate sources to a faulty *cluster agent* set the *performative* attribute of the messages sent to *ACLMessage.PROPOSE*. The faulty *cluster agent* on receiving this type of message knows that the content contains proposals sent by neighbors for reconfiguration [85].

A snapshot of agent communication in JADE during a fault scenario is shown in Figure 16. The agent communication was captured using another feature provided by JADE i.e. the *sniffer agent*. A *sniffer agent* can intercept ACL messages during agent communication and displays them graphically using a notation similar to UML sequence diagrams. It is also a good debugging tool for agent societies as it helps in observing the message exchange between agents. Another, important feature offered by JADE which has been implemented in the proposed MAS is the concept of *behaviours*. It is common practice in agent programming to have several concurrent active tasks within an agent. One approach for implementing these concurrent activities is to use Java thread programming. However, since Java threads are not designed for large scale parallelism this approach proves to be inefficient. JADE *behaviours* offer an efficient alternative for implementing concurrent tasks within an agent. The proposed MAS uses this feature of JADE as agents are concurrently involved in many different tasks. Examples of these tasks include communication with neighbors, identification of nodes affected by a fault, identification of alternate sources, etc. Also, each agent has several *behaviours* running in parallel. For example each of the CAs has a cyclic *behaviour*, which executes continuously for the lifetime of the agent and receives messages indicating a fault in the respective cluster. These messages are sent by a dedicated agent in the system, which receives this information from the graphical user interface during simulation. *Cluster agents* also have *behaviours* for performing the analysis required for the identification of alternate source nodes.

Simulation Results

SCE's *Circuit of the Future* was used to simulate all possible single faults for both the topology-independent decentralized MAS architecture and the scalable MAS architecture.

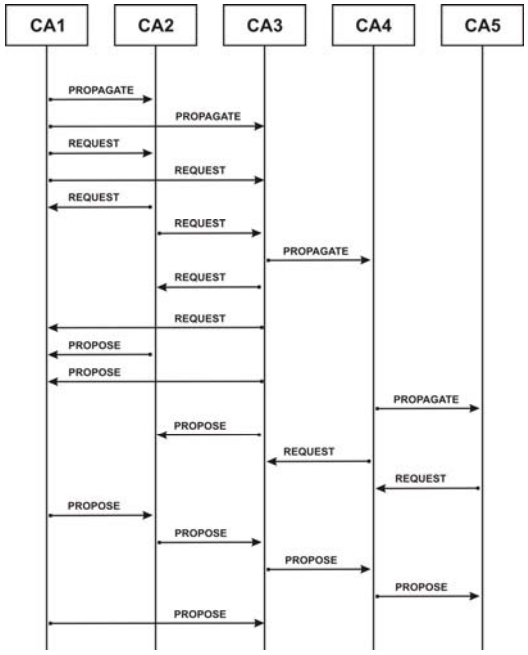

Figure 16 Agent communication in JADE during fault scenario

The results obtained from these simulations are shown in Figure 17, Figure 18, and Figure 19 [84].

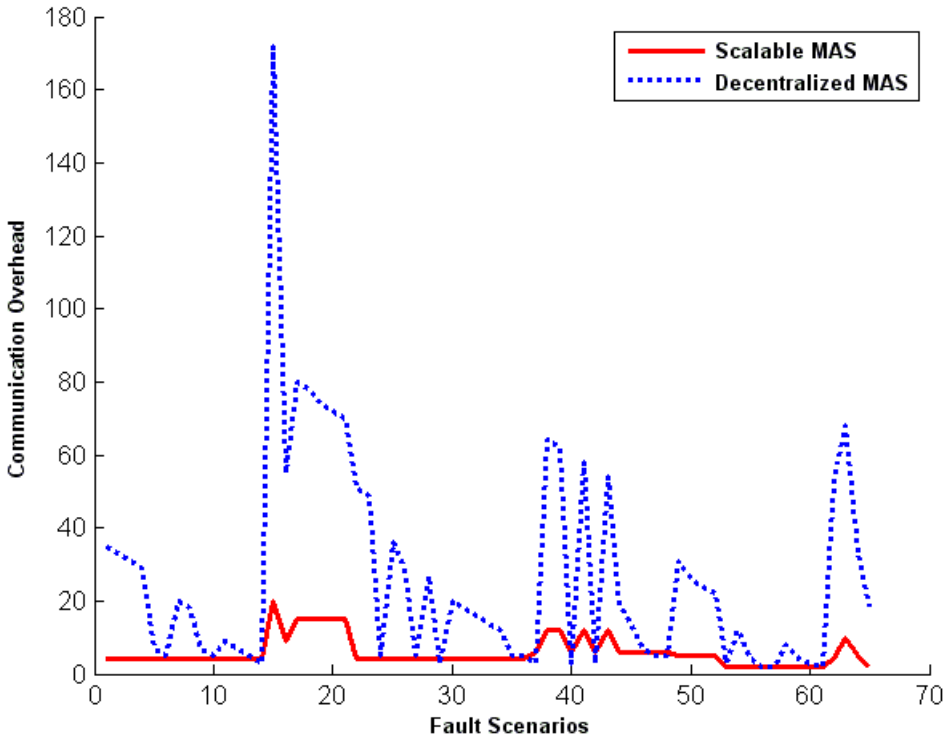

Figure 17 Comparison of communication overhead

Figure 17 presents a comparison of the two MAS approaches in terms of the communication overhead. We define communication overhead as the total messages passed in the agent community to reach a decision. It can be clearly seen in the figure that the proposed scalable MAS experiences far less communication overhead to reach the same decision as compared to the completely decentralized MAS. The average communication overhead for the scalable MAS is approximately 6 messages. On the other hand, for the decentralized MAS the average communication overhead is 27 messages. Also, the scalable MAS performance is mostly uniform, with few occasional spikes. A more uniform performance can help to model the communication infrastructure for the system in a way that optimizes performance. On the other hand, communication overhead for the decentralized approach shows a more irregular pattern. Considering the worst case scenario, for a fault between Bus 1 and Bus 2 in the test system, the communication overhead incurred in the topology-independent decentralized MAS is 172 messages, while for the scalable MAS it is significantly lower (20 messages).

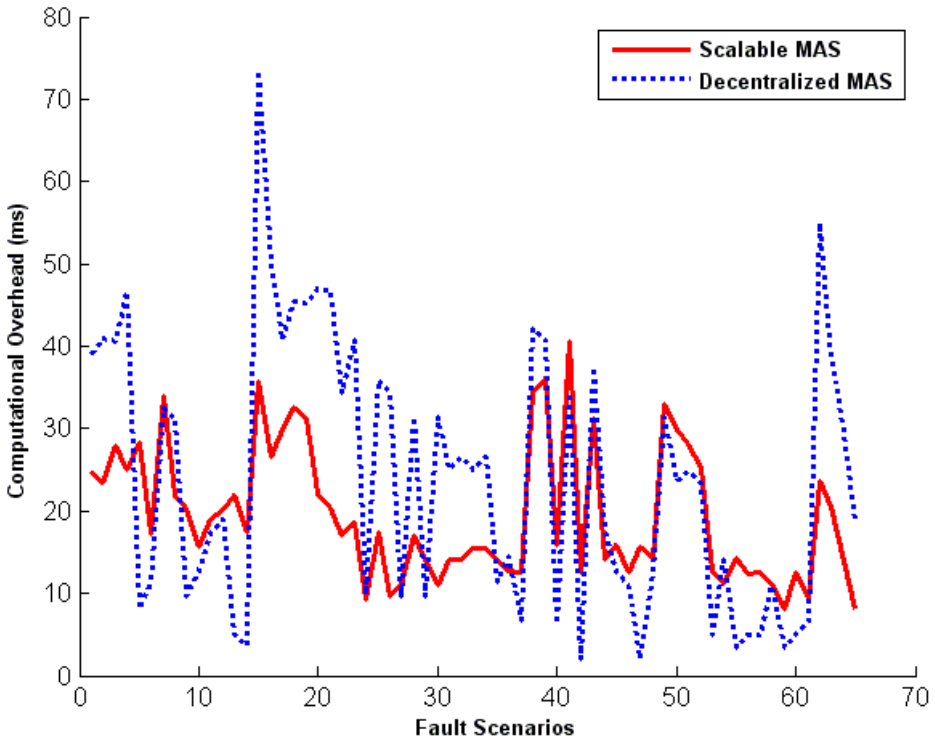

Figure 18 Comparison of computational overhead

Next, we compare the two approaches in terms of the computational overhead. For the purpose of these simulations we define the computational overhead as the total time taken by the MAS to arrive at a decision to perform reconfiguration in the event of a fault. Multiple simulations of all possible single fault scenarios were performed to obtain a comprehensive profile of the total time taken by each MAS. The results are shown in Figure 18. It can be seen that the scalable MAS takes less total time to reach a decision. The average time taken by the scalable MAS to reach a decision is 19.47 ms, while for the decentralized MAS it is 23.92 ms. It is important to note, that not only does the scalable MAS reach a decision quickly but also, the decision is more informed. Each *cluster agent* in the proposed scalable MAS performs a complete analysis of the situation by identifying good and bad alternate sources and using the input from neighbors the cluster agent selects an optimal switching position based on the cost function and external constraints. The decentralized MAS on the other hand takes the time to propagate the fault information and any viable switching position is selected. Although the *cluster agents* in the proposed scalable MAS are performing more work, yet the overall system is designed such that the total time taken is less. The worst case scenario in terms of the computational overhead for the topology-independent decentralized MAS is 73.40 ms, while for the proposed scalable MAS it is 40.60 ms.

Finally, we investigate the relationship between the total number of buses affected by a fault and the performance for each of the proposed approaches. The observations are summarized in Figure 19. When the number of buses affected by a fault is small the scalable approach does incur some overhead. However, when the number of buses affected by a fault increases, the proposed scalable MAS outperforms the topology-independent decentralized MAS.

We have also made some general observations about the proposed scalable MAS. Since buses affected by a fault such as loss of line in a power system are those which are physically close, they are most likely to fall in the same cluster. This reflects that the proposed choice of electrical distance as the clustering parameter is very pragmatic. One problem related to this choice is that the electrical distances do not incorporate the system dynamics.

By partitioning the entire system into clusters and assigning an agent to each cluster, task sharing is achieved which reduces the problem complexity. In comparison to the topology-independent decentralized MAS, the proposed scalable MAS uses fewer number of agents and results in

reducing the resource consumption and communication overhead. Also, the entire process of communication is more deterministic and less error prone. As the number of agents involved in the communication increases, the entire decision making process becomes more susceptible to faults and errors due to lost messages or problems with the communication network. Based on the simulation results and our observations and analysis, the proposed scalable MAS architecture is robust and performs well for large scale power systems [84].

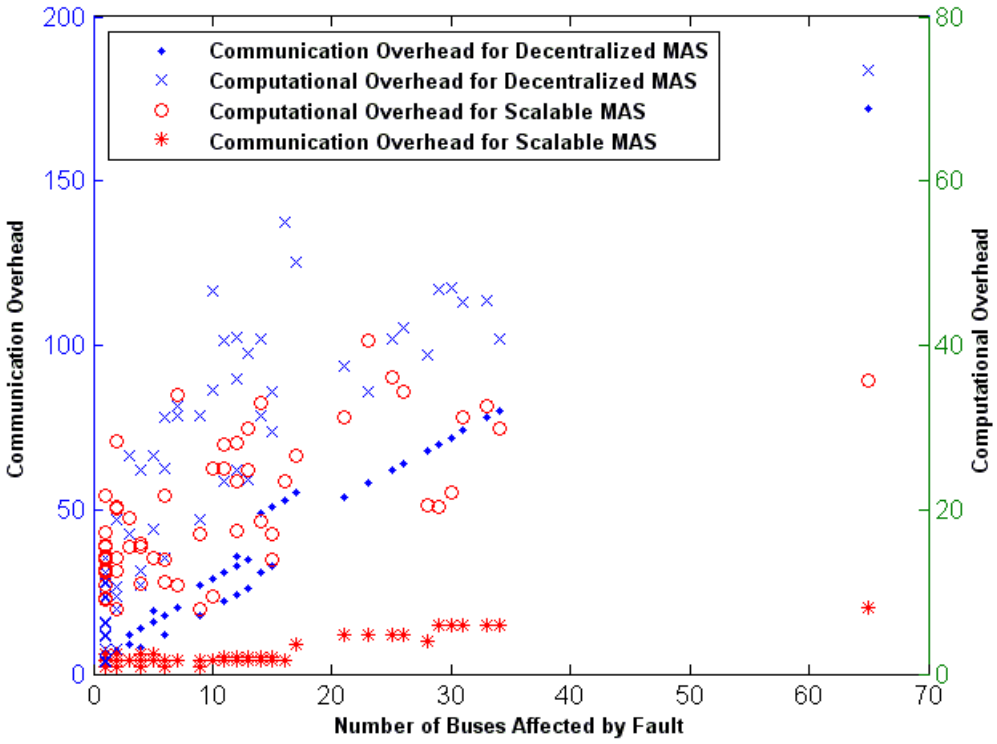

Figure 19 Comparison of performance with respect to buses affected by a fault

Chapter 6 Introducing Power System Constraints

The proposed scalable multiagent system suggests initial switch positions for reconfiguration. Considering the simplified case where the purpose of the reconfiguration algorithm is to simply restore power to the nodes affected by some fault, using any one of the suggested switch position would restore power to the affected area. However, power systems function under a set of constraints that have to be met to ensure proper functioning. One of the most important of these constraints is maintaining a good voltage profile. The main objective of any reconfiguration algorithm is to isolate the faulty part of the network and restore the network with good voltage profile. A reconfiguration algorithm, working under the constraint of maintaining a good voltage profile, has been proposed in this research work. The multiagent system suggests initial switch positions which have been prioritized based on the cost function. The next step in the reconfiguration process is to determine if using a switch or combination of switches satisfies the bus voltage constraint as specified below,

$$\boldsymbol{V_{min} \leq V \leq V_{max}}$$

For the purpose of this work,

$$\boldsymbol{V_{min} = 0.90\ p.u.\ (volt)}$$

and

$$\boldsymbol{V_{max} = 1.10\ \ p.u.\ (volt)}$$

When a fault in the power system leads to a low unacceptable voltage profile in a significant area, the objective is to restore the system such that acceptable voltages are maintained at all buses. The topology of the power system needs to be altered in a way that brings the voltage at each bus within the specified limits. The simplest way of achieving this objective is to apply brute force technique and evaluate each possible switch combination until an acceptable voltage profile is obtained. For the purpose of this research the brute force approach was applied to develop a baseline for comparison purposes. However, exhaustively searching for a switch or set of switches is not feasible considering time constraints as well as the large sizes of present day power systems.

6.1 Proposed Scalable Reconfiguration Algorithm

In this work a reconfiguration algorithm has been proposed that attempts to minimize the number of switching combinations explored and therefore speeds up the process of reconfiguration. The algorithm begins by using the set of switches proposed by the multiagent system. Notice, the set of switches that the multiagent system proposes is a subset of the actual number of switches in the power system. The multiagent system only proposes switching operations that appear feasible under the given fault conditions. Therefore, the reconfiguration algorithm starts off with a limited number of switches.

All possible single faults were generated for SCE's Circuit of the Future testbed. Extensive studies of voltage profiles were performed for each fault. The primary objective of these studies was to come up with a set of rules that could help in minimizing the number of switching combinations. The studies focused on the following,

1 Number of buses affected by each fault.
2 The bus voltages, after initial reconfiguration is performed using the switch positions suggested by the multiagent system.
3 The clusters to which buses' violating the voltage limits belong.

Detailed analysis of the data generated revealed that the behavior of the voltage profile, after initial reconfiguration, had a relation with the number of buses affected by the fault. For the fault cases where the number of buses affected was small, a single switching operation was enough to meet the voltage constraint. In other cases with larger number of affected buses, a combination of switches was required. Therefore, we set a threshold for our algorithm. The behavior of the algorithm changes based on the value of the threshold. The threshold is defined as,

$$\boldsymbol{\theta = 10}$$

where θ is the number of buses affected by the fault.

If the number of buses affected is less than the threshold, the algorithm takes the switch positions suggested by the multiagent system and comes up with a system configuration that meets the specified constraints. However, if the number of affected buses is greater than or equal to the threshold, the algorithm attempts to optimally find a switch combination to ensure all constraints

are met. The process starts by closing the first open switch as suggested by the multiagent system. Next, the algorithm looks at the voltage profiles and determines which buses have voltages below the specified limits. Once this information is obtained, the next step is to determine the clusters to which these buses belong. At the end of this step a list of clusters experiencing the voltage violation is formed. An analysis of the data generated revealed that switches closest to the cluster affected by a fault contribute less towards bringing the voltage profile within limits. Therefore, the cluster currently experiencing the fault is moved down the list. After compiling this list, the next step is to determine which switches in these clusters can be turned on to improve the voltage profile. After having compiled a list of alternate switches, the next step is to prioritize these based on the cost function given below.

$$\boldsymbol{Cost} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{K}} \boldsymbol{C_i W_i}$$

where W_i represents the weight of each edge in the path, K is the length of the path and C_i is a Cost Adjustment Factor, which models different constraints.

Cost analysis is performed for each of the alternate switches and a prioritized list of switches is generated. Based on the switch priorities, simulations are performed by turning on each suggested switch or switch combination. The effect of the combination on the voltage profile is examined. At the end of this step, a final feedback is provided to the algorithm in the form of the number of buses experiencing voltage violation. The switch combination giving the least number of buses with voltages below limit is selected. This process continues until a good voltage profile is obtained.

6.2 Implementation Details

The proposed algorithm has been implemented using Power Analysis Toolbox (PAT). PAT is a power system simulation environment developed in MATLAB/Simulink. It is a flexible and modular tool for load flow, transient and small signal analysis [86]. The simulation package can be used as a design and analysis tool for complex interactive power systems. The simulation package allows advanced vectorized system modeling and applied non-linear control leading to robust control of electric power system dynamics. The huge amount of information required to describe the power system components is hierarchically organized into the PAT data structure. To perform load flow analysis, a load flow module extracts this stored information. The load

flow problem is then solved using a Newton-Raphson algorithm [86]. PAT also provides function to graphically represent the power network. Figure 20 shows the circuit of the future as represented in PAT.

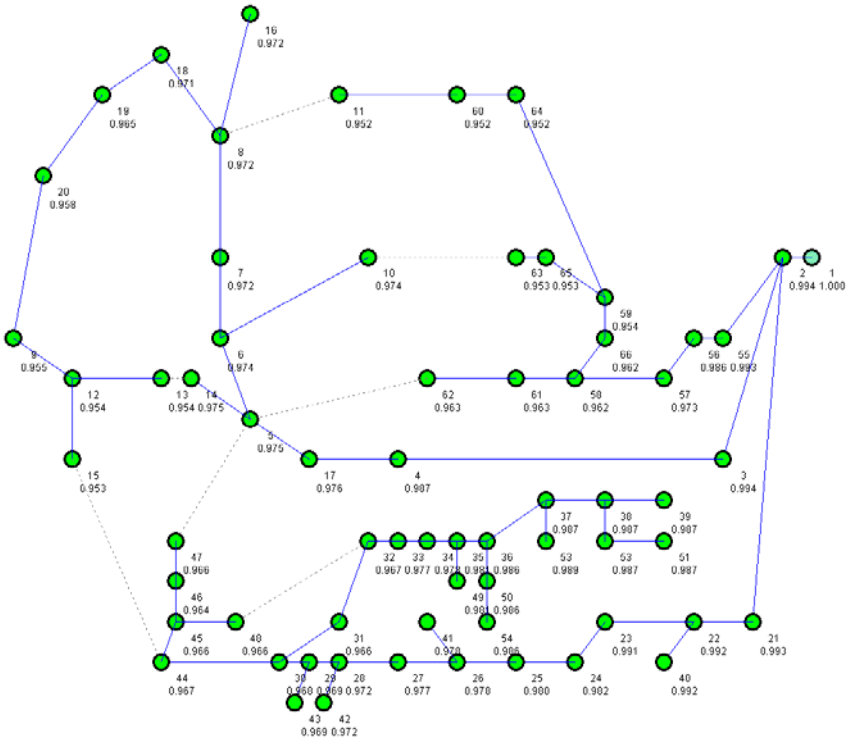

Figure 20 SCE's Circuit of the Future in PAT

Bus 1, colored in blue for differentiation purposes, is the source node. The figure also includes the identifier as well as the voltage magnitude at each bus. Solid lines represent power transmission lines whereas, the normally open switches in SCE's Circuit of the Future are represented using dotted lines. As depicted in the figure, under normal operating conditions the voltage at each bus satisfies the constraint.

6.3 Simulation Results

The proposed algorithm was applied to the Circuit of the Future testbed and the results obtained showed a marked improvement as compared to randomly selecting switch combinations. A comparison of the performance of the two approaches is presented in Figure 21. The plot clearly illustrates that the proposed scalable approach performs better than the brute force technique. The proposed algorithm adjusts itself based on the feedback provided, and hence the

improvement in performance.. The performance of the proposed algorithm is also very predictable and uniform. Detailed simulation results are included in Appendix B.

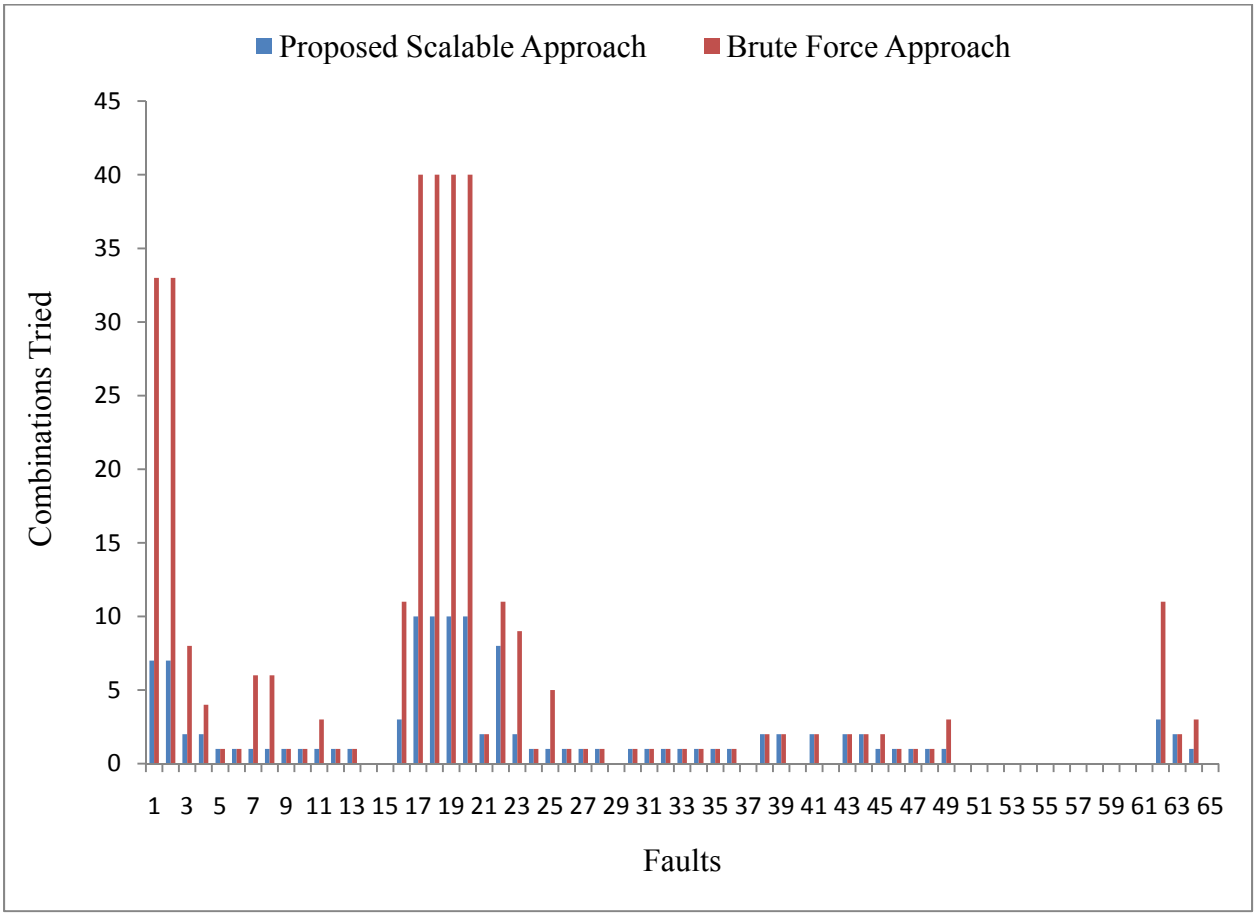

Figure 21 Comparison of performance

Example Scenario

The *Circuit of the Future* was used to simulate all possible single faults for brute force technique and the proposed scalable reconfiguration algorithm. In this section, the working of the algorithm is described using an example scenario. Let us assume there is a fault between Bus 21 and Bus 22 in Cluster 1 as shown in Figure 22. However, this fault propagates and affects buses in cluster 3, 4, and 5. As a first step, the multiagent system suggests the switch positions that can be used for initial reconfiguration. For this particular fault scenario, the agents in the scalable

multiagent system identify SW1 and SW2 as the potential alternate sources. Since, the suggested switch positions have been prioritized based on the cost function SW1 is selected to perform the initial reconfiguration. Figure 22 and Figure 23 show the resulting power system and the voltage profile after turning SW1 on.

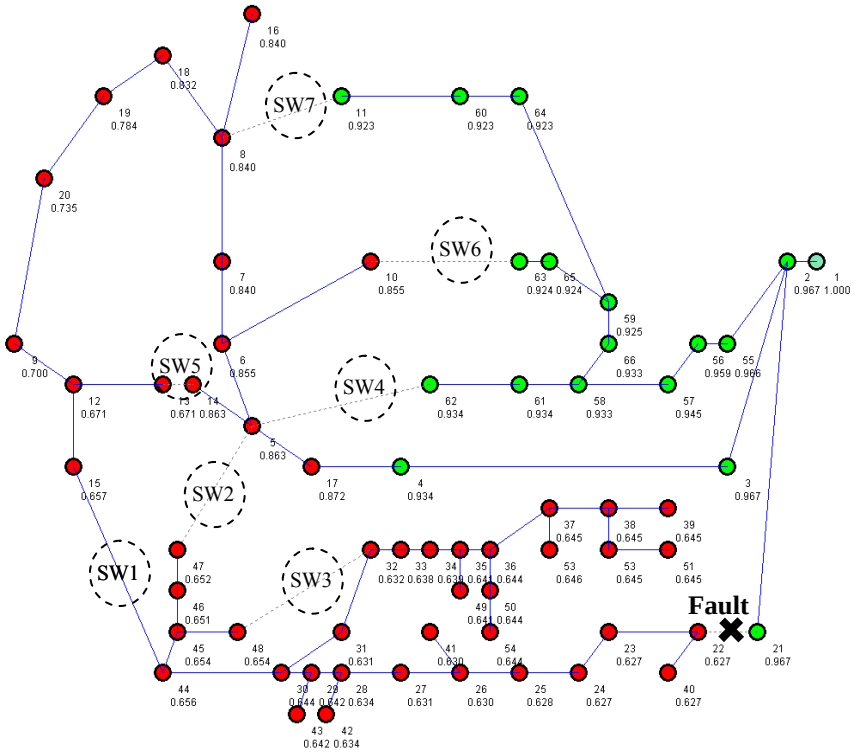

Figure 22 Circuit of the Future after turning SW1 ON

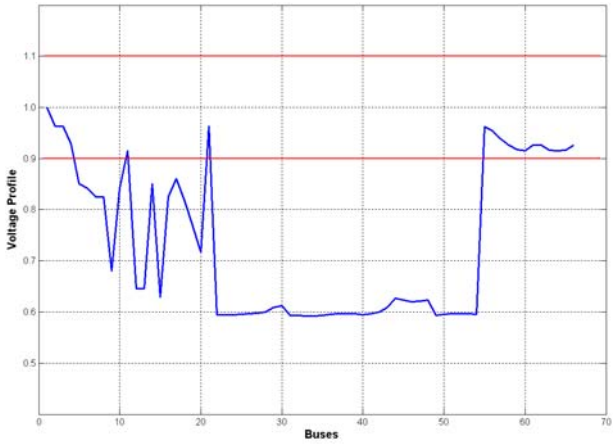

Figure 23 Voltage profile after turning SW1 ON

Although, power has been restored to the affected area, a total of 48 buses in the power system violate the voltage limits. The buses violating the voltage limits have been highlighted in red as shown in Figure 22. Corresponding voltage profile is shown in Figure 23.

Next, based on the buses experiencing the voltage violation a list of potential alternate switches is compiled. For the fault scenario being discussed, SW2, SW3, SW4, SW5, SW6, and SW7 are identified as the alternate switches. Next, a cost analysis is performed for each switch to obtain a prioritized list of switch combinations. This list of switch combinations to bring the voltages within the specified limits is given below,

$$\begin{matrix} & \text{First Switch} & \text{Second Switch} \end{matrix}$$
$$\begin{bmatrix} 1 & 7 \\ 1 & 2 \\ 1 & 4 \\ 1 & 6 \\ 1 & 5 \\ 1 & 3 \end{bmatrix}$$

Based on this matrix the first combination to try is SW1 and SW7. After turning these switches ON a total of 53 buses in the system highlighted in red in Figure 24 experience voltage violation.

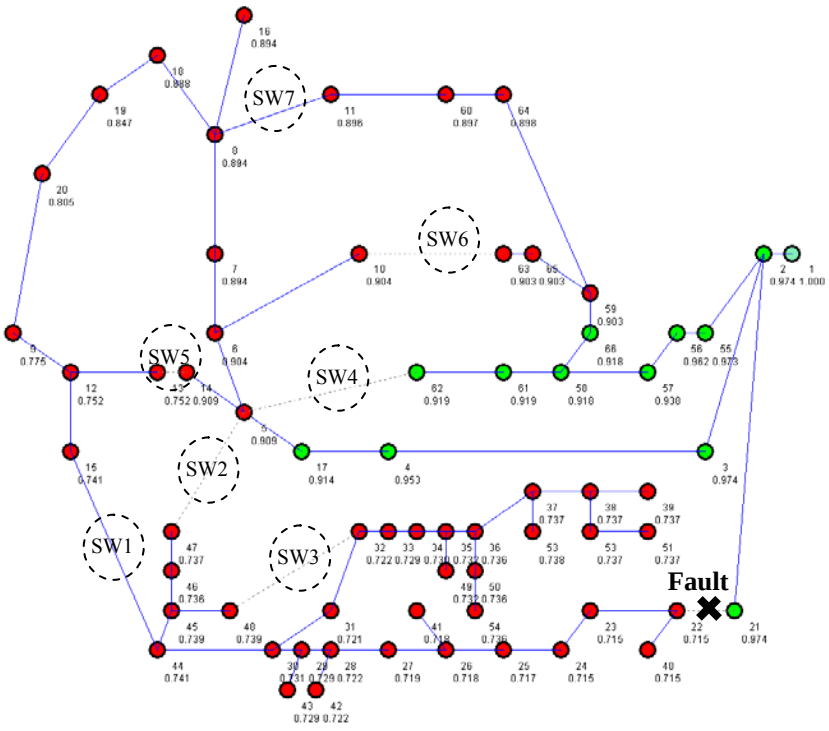

Figure 24 Circuit of the Future after turning SW1 and SW7 ON

Figure 25 presents the voltage profile of the system after SW1 and SW7 are turned on. It can be seen that voltage magnitude for most of the buses in the system is below the acceptable voltage limit of 0.9 p.u. represented by the solid red line across the plot.

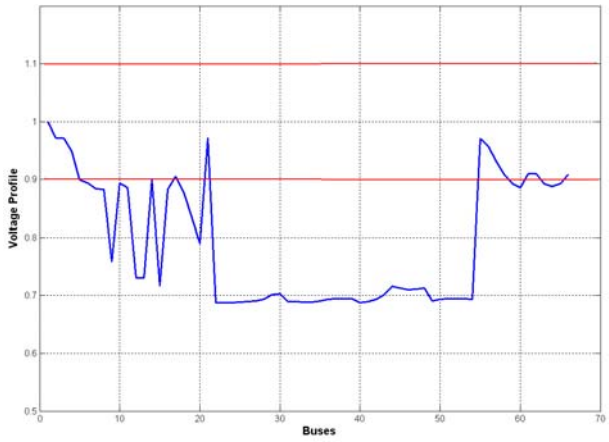

Figure 25 Votlage profile after turning SW1 and SW7 ON

Based on the prioritized list of switches next the combination of SW1 and SW2 is tried. The 36 buses violating the voltage constraint are highlighted in red in Figure 26.

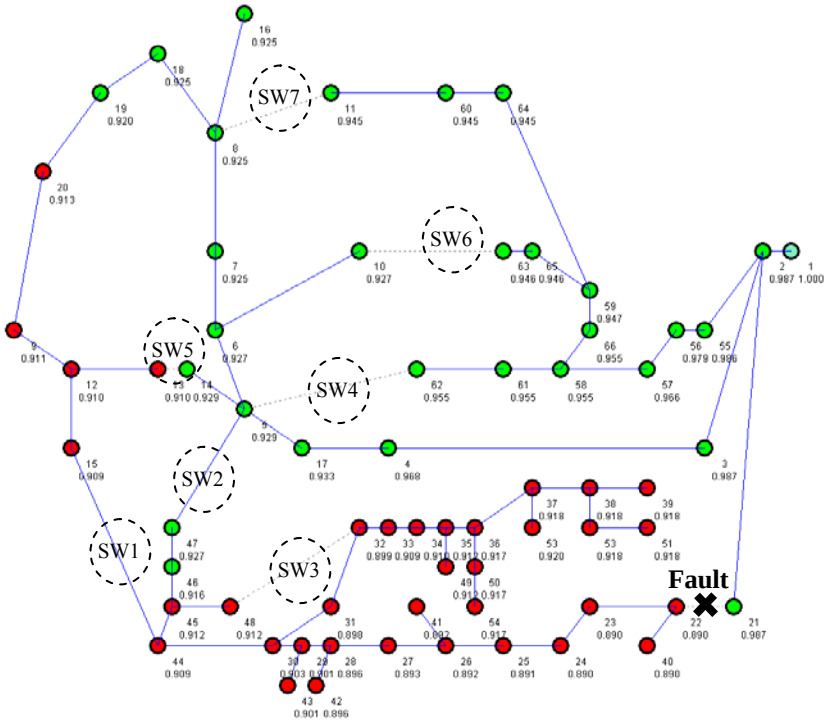

Figure 26 Circuit of the Future after turning SW1 and SW2 ON

The resulting voltage profile is shown in Figure 27 with 36 buses below the 0.9 p.u. limit.

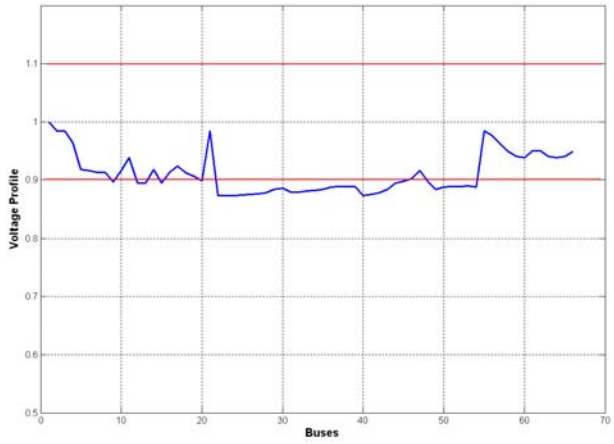

Figure 27 Voltage profile after turning SW1 and SW2 ON

Since the voltage constraint is not met yet, the next combination is tried. Results obtained after turning SW1 and SW4 on are illustrated in Figures 28 and 29.

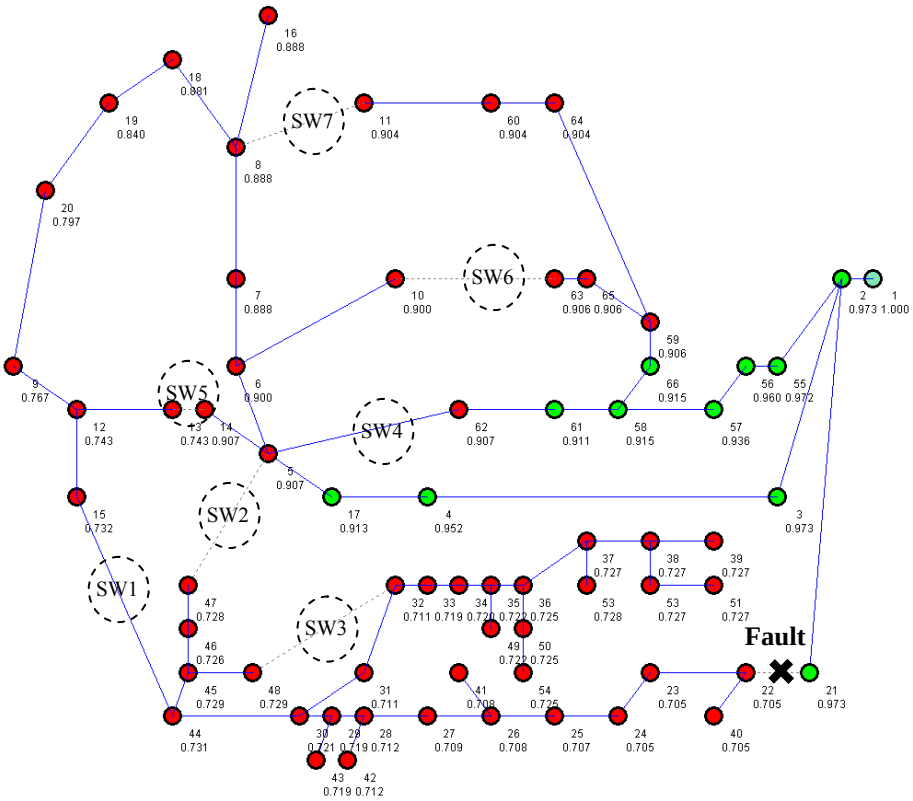

Figure 28 Circuit of the Future after turning SW1 and SW4 ON

54 buses experience voltage violation, with the voltage magnitude at these buses far below V_{min}.

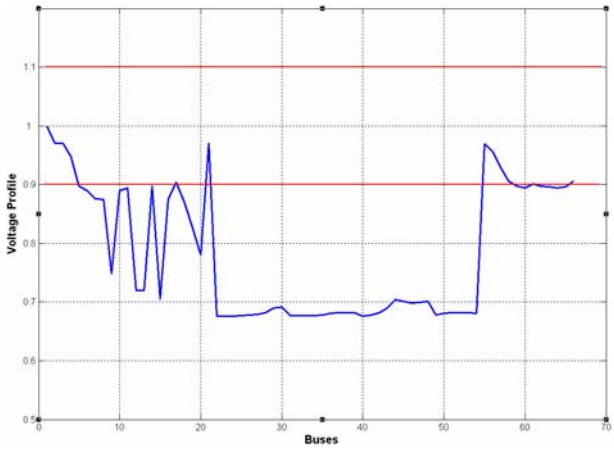

Figure 29 Voltage profile after turning SW1 and SW4 ON

The next combination in the prioritized list of switch combinations is SW1 and SW6. 53 buses in the system violate the voltage constraint and are highlighted in red in Figure 30.

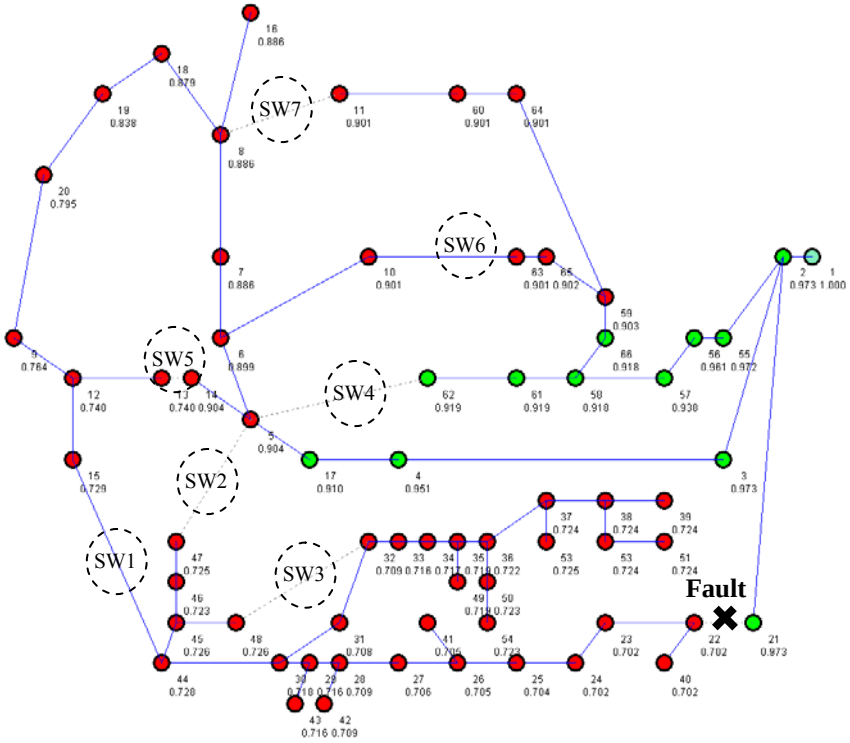

Figure 30 Circuit of the Future after turning SW1 and SW6 ON

Figure 31 illustrates similar results in the form of a voltage profile plot. It can be seen that most of the buses in the plot have voltage magnitude below the solid red line representing V_{min}.

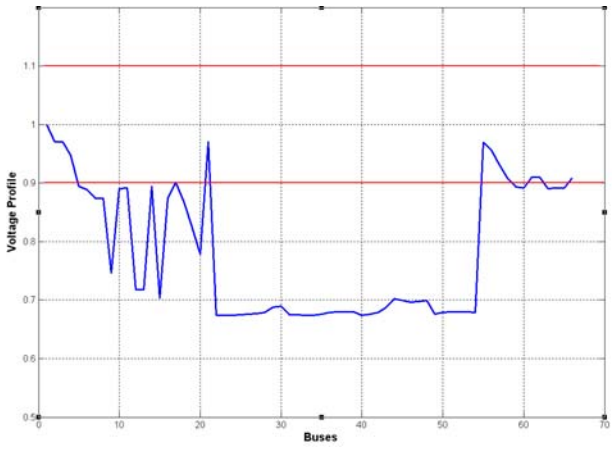

Figure 31 Voltage profile after turning SW1 and SW6 ON

Next, SW1 and SW5 are turned ON to analyze the resulting system and the voltage profiles. As shown in Figures 32 and 33, 22 of the 66 buses experience voltage violation.

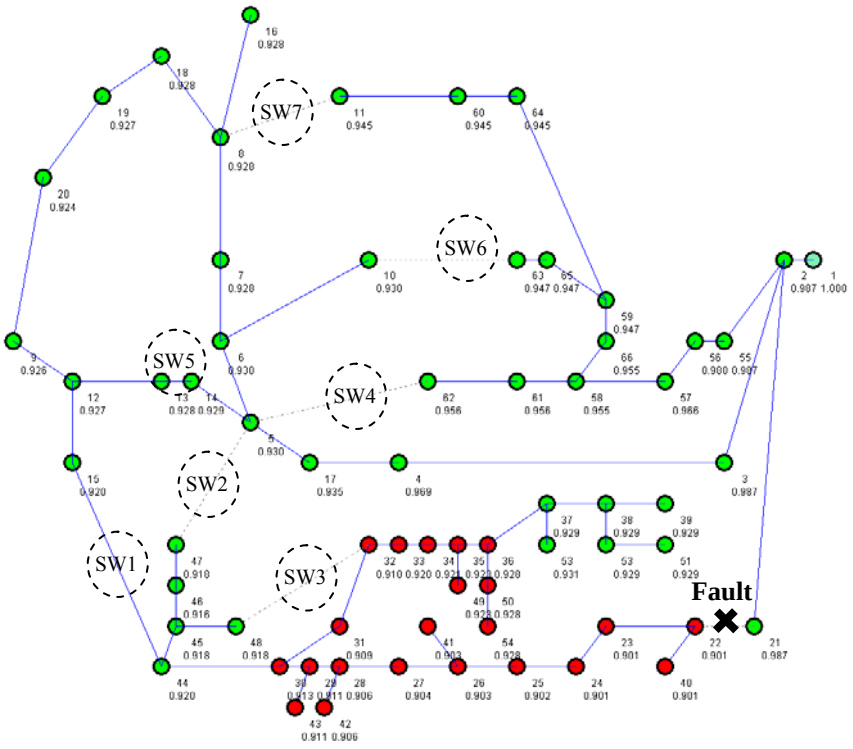

Figure 32 Circuit of the Future after turning SW1 and SW5 ON

Note in Figure 33, although the buses are experiencing voltage violation the magnitude is closer to 0.9 p.u. Turning ON SW1 and SW5 not only reduces the number of buses violating voltage constraint but also brings the magnitude at these buses closer to V_{min}.

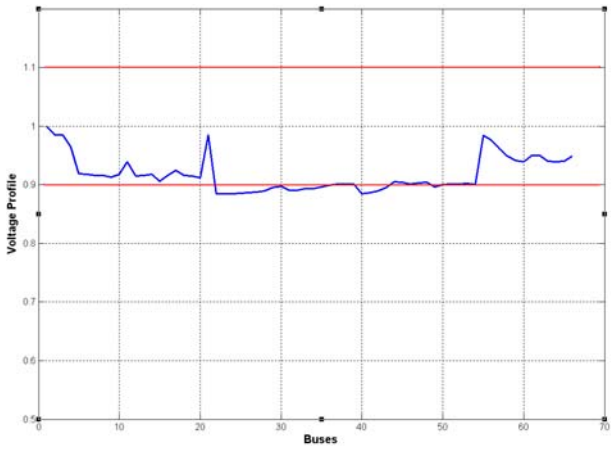

Figure 33 Voltage profile after turning SW1 and SW5 ON

As the next step in the analysis of different switch combinations and the resulting voltage profiles, SW1 and SW3 are turned ON. 48 buses in the system violate the voltage constraint. The results obtained are presented in Figures 34 and 35.

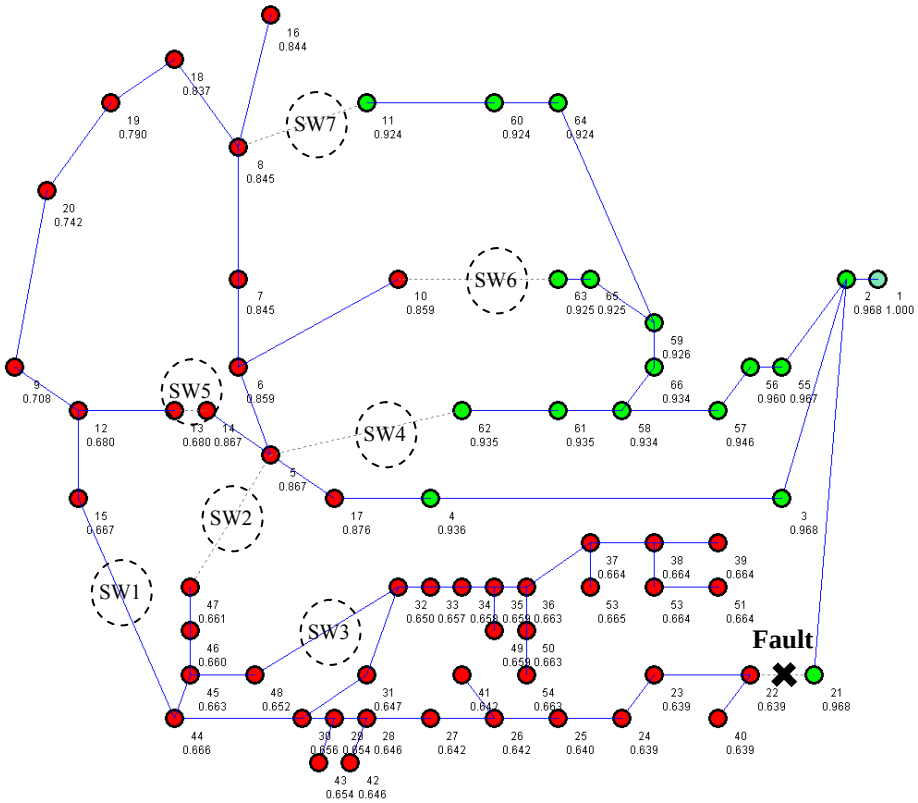

Figure 34 Circuit of the Future after turning SW1 and SW3 ON

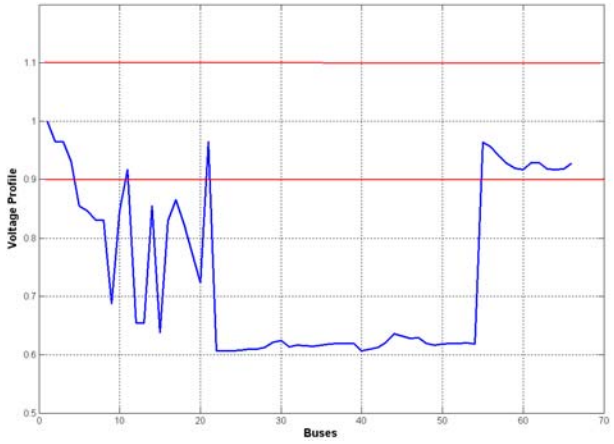

Figure 35 Voltage Profile after turning SW1 and SW3 ON

None of the proposed 2 switch combinations bring the bus voltages within the specified limits. Although some combinations bring the bus voltages closer to V_{min}, yet the voltage constraint is not satisfied. The results obtained after exploring all proposed 2 switch combinations have been summarized in the matrix below,

$$\begin{bmatrix} Switch & Switch & \#\ of\ buses\ violtaing\ voltage\ constraint \\ 1 & 7 & 53 \\ 1 & 2 & 36 \\ 1 & 4 & 54 \\ 1 & 6 & 53 \\ 1 & 5 & 22 \\ 1 & 3 & 48 \end{bmatrix}$$

The algorithm now attempts to explore 3 switch combinations. Since in the previous step switch combination SW1 and SW5 resulted in the least number of buses violating the voltage constraint, it is selected for the next step.

The first 3 switch combination to be tried is SW1, SW 5, and SW2. This results in 12 buses where the voltage is not within the limits specified. These buses are highlighted in red in Figure 36.

Figure 37 shows the voltage profile obtained after trying the first 3 switch combination. As illustrated in the figure, only a few buses in the system violate the voltage constraint. Also, the voltage magnitude at these buses is only slightly below 0.9 p.u.

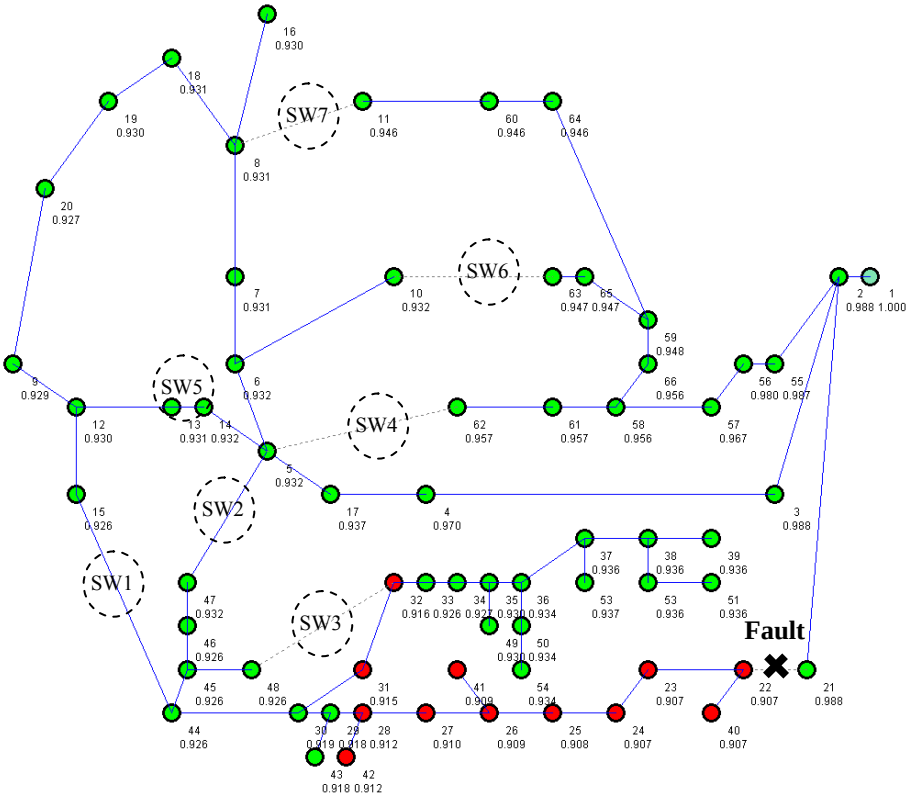

Figure 36 Circuit of the Future after turning SW1, SW5, and SW2 ON

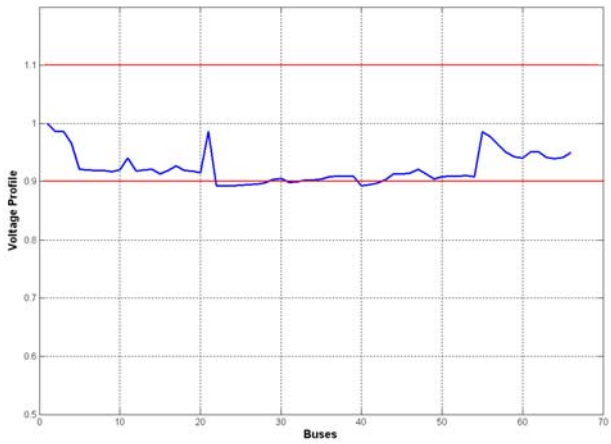

Figure 37 Voltage profile after turning SW1, SW5, and SW2 ON

Next the combination SW1, SW5, and SW3 is tried. The voltage profile is worse than the previous case with a total of 18 buses experiencing voltage violation. Figures 38 and 39 illustrate the results obtained.

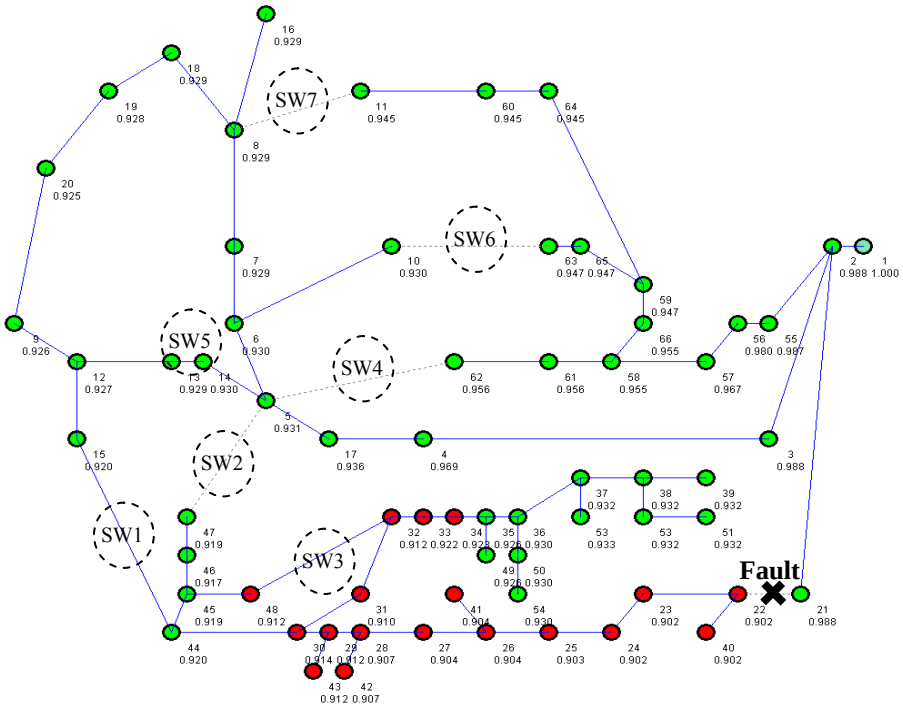

Figure 38 Circuit of the Future after turning SW1, SW5, and SW3 ON

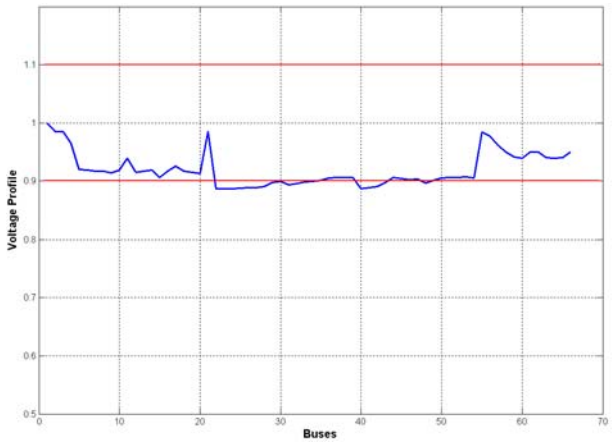

Figure 39 Voltage profile after turning SW1, SW5, and SW3 ON

The final step in the reconfiguration is turning SW1, SW5, and SW4 ON. As illustrated in Figure 40 trying this combination satisfies the voltage constraint bringing all bus voltages within the limits specified. All buses in the figure are green in color representing good voltage magnitudes at all buses after reconfiguration.

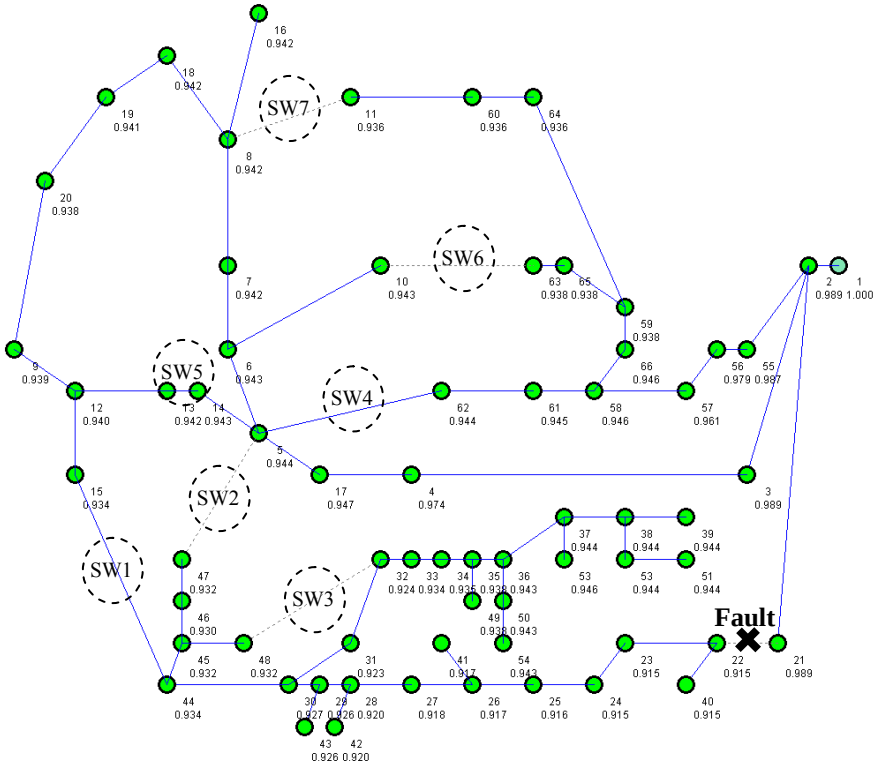

Figure 40 Circuit of the Future after turning SW1, SW5, and SW4 ON

Figure 41 shows the voltage profile obtained after turning SW1, SW5, and SW4 on. It can be seen that all buses have voltages within the limits specified. This satisfies the voltage constraint and therefore, the reconfiguration process stops. At the end of system reconfiguration power is restored to all buses with a good voltage profile.

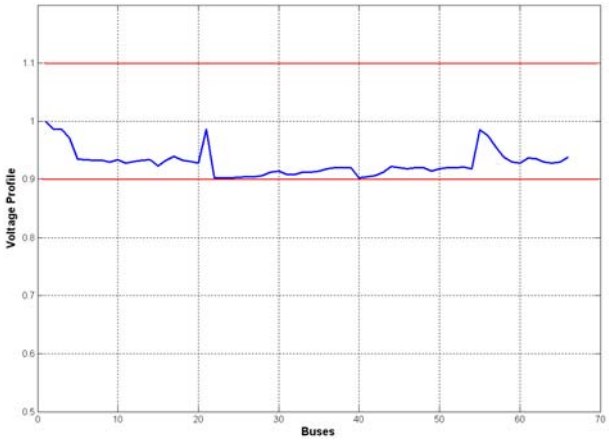

Figure 41 Voltage profile after turning SW1, SW5, and SW4 ON

Hence the proposed reconfiguration algorithm explores fewer combinations. This is achieved by providing a feedback to the algorithm in the form of the number of buses violating the voltage constraint. The algorithm adjusts its functionality based on the feedback and outputs a switch combination that satisfies the constraints specified.

Chapter 7 Conclusion and Future Work

7.1 Conclusion

Although multiagent systems and other distributed approaches offer better scalability as compared to traditional centralized approaches, yet efficient solutions based on these technologies need to scale well. Most existing decentralized multiagent system solutions for power system applications associate an agent with each component in the system. Although such solutions offer complete decentralization of control and are generally topology independent, yet for large scale systems the number of agents in the MAS poses scalability challenges. Also, the amount of computations, communication, and coordination required for these solutions triggers the need to find a trade-off between decentralization and size of the multiagent system. Reduction in the total number of agents in a multiagent system decreases the resource requirements, simplifies the agent system topology, reduces the communication overhead by limiting the number of possible interactions, and minimizes the overall complexity of the system. The proposed scalable MAS addresses these issues by limiting the total number of agents in the system, and reducing the computational and the communication overhead. Furthermore, in the event of a fault, the affected nodes are quickly identified for reconfiguration. Depending on the system constraints, an optimal solution for reconfiguration is identified using the cost function. However, to gain these benefits, the agents in the clusters acquire additional knowledge about the topology of their cluster. Simulation results show that the proposed scalable MAS experiences significantly less computational and communication overhead as compared to the topology independent decentralized MAS. Also, the performance of the scalable MAS improves as the number of nodes affected by a fault increases, making it more attractive for large scale power systems. The proposed scalable MAS has been coupled with a scalable reconfiguration algorithm. The reconfiguration algorithm reduces the number of switch combinations to be explored in order to reconfigure a power system. The power system is assumed to be working under voltage constraints requiring voltage at each bus to be within $\pm$ 10% of the standard voltage i.e. 1.0 p.u. The results obtained show that the proposed scalable approach performs better than the random switch selection approach making it attractive for commercial power systems.

7.2 Future Work

This research is the first step towards the application of techniques from different fields including mathematics, electrical engineering, and computer science. It provides a sound basic structure with several opportunities for improvement and extension. Some of these are listed below.

1. In this research work, the analysis to find the effects of a fault and to determine an optimal combination of switches is performed online in real time. Time is of the essence in post fault reconfiguration of the power system. To guarantee timely operations and reduce the online computational burden, artificial intelligence based learning techniques can be used. Most artificial intelligence techniques require training data. Data generated during offline studies can be used for this purpose. Data mining and machine learning techniques can also be applied for knowledge and data acquisition. The learned knowledge can then be used to generate faster results during real world operation.
2. The clustering parameter used for the proposed work did not include the power system operational dynamics for finding partitions. The resulting clusters were static and did not change with changes in the system topology or operational conditions. Incorporating this information into the algorithm for dynamic clustering can result in more efficient solutions.
3. Simulation results were based on comprehensively simulating all single fault scenarios. Analysis of the system behavior in case of multiple complex faults needs to be performed.
4. There is a need to develop efficient interfaces between agent development platforms and existing power system simulation environments for real time simulation studies.
5. In our proposed work, good voltage profile was obtained using a combination of switches. In real world power systems, voltage control is performed mostly by reactive power sources such as shunt capacitors, synchronous generators, static VAR compensators etc. As future work, these sources can be added to the power system as well as represented in the multiagent architecture for effective voltage control.

Appendix A

Algorithm for Topology-Independent Decentralized MAS Architecture

Step 1: Introduce fault between BA_i and BA_j .

Step 2: Initiate message passing sequence

BA_j senses loss of power

BA_j retrieves list of neighboring BAs.

BA_j creates a message

message.performative = ACLMessage.INFORM

message.content = BA_j's agent identifier (AID)

message.receiver = each neighboring BA

Step 3: During message passing sequence

For Bus Agents

BA_x receives a message from BA_y.

BA_x checks message performative

If performative is ACLMessage.Request

Creates a message

message.performative = ACLMessage.PROPOSE

message.content = Sensed value of power at the associated bus

message.receiver = BA_y.

If performative is ACLMessage.PROPOSE

Retrieves received message's content

If content > 0

Creates a message

message.performative = ACLMessage.PROPOSE

message.content = Switch position between BA_x and BA_y

message.receiver = Processor Agent (PA)

If content < 0

Creates a message

message.performative = ACLMessage.PROPOSE

message.content = null.

message.receiver = Processor Agent (PA)

If performative is ACLMessage.INFORM

BA_x senses its own power

If power = = 0

Retrieves message content to form a list of AIDs

If AID of BA_x is not in the list

If has BA_x has neighboring switches

Creates a message

message.performative = ACLMessage.REQUEST

message.content = "Need Power"

message.receiver = Each BA_z on other side of switch

BA_x retrieves list of neighbors

If only neighbor of BA_x is BA_y.

BA_x Extracts list of AID from message content

Appends its own AID to the list

Creates a message

message.performative = ACLMessage.INFORM

message.content = List of AIDs

message.receiver = PA

If BA_x has other neighbors

BA_x Extracts list of AID from message content

Appends its own AID to the list

Creates a message

message.performative = ACLMessage.INFORM

message.content = List of AIDs

message.receiver = Each neighbor of BA_x

For Processor Agent (PA)

PA receives a message

If performative is ACLMessage.PROPOSE

PA compiles a list of suggested switch positions from message

If performative is ACLMessage.INFORM

PA compiles a list of buses affected by the fault from message

Step 4: After message passing sequence completes

PA randomly selects a switch position from the compiled list.

PA creates a message

message.performative = ACLMessage.INFORM

message.content = "Perform Switching"

message.receiver = Switch agent associated with the chosen switch

Step 5: Final Switching

Switch agent receives the message and performs the final switching operation.

Algorithm for Scalable Multiagent System Architecture

Step 1: Obtain a weighted graph G(V,E) from the power system

Obtain Z-bus for the power system

Convert power system into a weighted graph

Each bus in the power system translates to a graph vertex

Each transmission line in the power system is a graph edge

Inverted absolute values of the Z-bus become edge weights

Step 2: Apply spectral clustering algorithm to obtain K partitions

Compute the adjacency (W) and Degree matrix of the graph G

Compute the normalized Laplacian matrix $\hat{A} = D^{-\left(\frac{1}{2}\right)} W D^{-\left(\frac{1}{2}\right)}$

Compute the N x K matrix $X = [x1, x2, \ldots, xk]$, where $x_1, x_2, \ldots, x_k$ are the K largest Eigen vectors of $\hat{A}$

Compute the N x K matrix Y, by renormalizing rows of X to have unit length

Use K-means algorithm to cluster. Each row of Y acts as a point in the K dimensional space

Finally cluster the original data points to obtain the final clusters

Step 3: Introduce agents to the clusters obtained in the previous step

Assign a cluster agent CA_i to each cluster i.

Provide an M x M weighted adjacency matrix encoding topology of the cluster i to CA_i, where M is the number of buses in the cluster i

For a cluster agent CA_j helping another faulty cluster agent CA_i

Step 4: Receive message from CA_i

CA_j determines the cost associated with each node in its cluster that can supply power to the fault cluster

CA_j compiles a list of potential source nodes and associated costs

CA_j creates a message

message.performative = ACLMessage.PROPOSE

message.content = list of potential source nodes and associated costs

message.receiver = CA_i

For a cluster agent with fault

Step 4A: Introduce fault in cluster i

CA_i is informed of the fault

CA_i updates its model of cluster topology after the fault

Step 5: CA_i informs its neighboring CAs about problem in its cluster

CA_i creates a message

message.performative = ACLMessage.REQUEST

message.content = "Help"

message.receiver = each neighbor of CA_i

Step 6: CA_i performs analysis of situation and tries to find a potential solution

Computes nodes affected by fault

Determines if internal reconfiguration is possible

Introduces parallel behaviors for each possible alternate source node

Step 7: Analysis of each alternate source node S_i is performed in parallel

Determines if all affected nodes are supplied by changing current source node to S_i

If yes, the S_i is added to the list of good sources

Otherwise, S_i is added to the list of bad sources

Step 8: Determine final feasible solution

After analysis of all alternate sources has been performed proposals from neighboring CAs are evaluated

CA_i computes a list of potential source nodes and associated costs proposed by neighbors

CA_i checks if a proposed source node is in the list of good sources compiled earlier

If yes, it is added to the list of final feasible sources

Finally CA_i selects the option with the minimum cost from the list of feasible sources

Step 9: CA_i informs appropriate switch agent to perform the final reconfiguration

Appendix B

Simulation results for the scalable multiagent system architecture

Source Node	Destination Node	Nodes Affected	No. of Nodes Affected	Messages Passed	Time Taken (ms)	Switch Positions Suggested
2	55	11,55,56,57,58,59,60,61,62,63, 64,65,66	13	35	39.1	SW-4, SW-6, SW-7
55	56	11,56,57,58,59,60,61,62,63,64, 65,66	12	33	40.9	SW-4, SW-6, SW-7
56	57	11,57,58,59,60,61,62,63,64,65, 66	11	31	40.6	SW-4, SW-6, SW-7
57	58	11,58,59,60,61,62,63,64,65,66	10	29	46.7	SW-4, SW-6, SW-7
58	61	61,62	2	7	7.8	SW-4
61	62	62	1	5	11	SW-4
58	66	11,59,60,63,64,65,66	7	20	32.7	SW-6, SW-7
66	59	11,59,60,63,64,65	6	18	31.2	SW-6, SW-7
59	65	63,65	2	7	9.4	SW-6
65	63	63	1	5	12.4	SW-6
59	64	11,60,64	3	9	17.1	SW-7
64	60	11,60	2	7	18.7	SW-7
60	11	11	1	5	4.7	SW-7
22	40	40	1	3	3.1	none
1	2	2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24, 25,26,27,28,29,30,31,32,33,34, 35,36,37,38,39,40,41,42,43,44, 45,46,47,48,49,50,51,52,53,54, 55,56,57,58,59,60,61,62,63,64, 65,66	65	172	73.4	none

Source Node	Destination Node	Nodes Affected	No. of Nodes Affected	Messages Passed	Time Taken (ms)	Switch Positions Suggested
2	3	3,4,5,6,7,8,9,10,12,13,14,15,16, 17,18,19,20	17	55	50.1	SW-1, SW-4, SW-2, SW-6, SW-7
2	21	21,22,23,24,25,26,27,28,29,30, 31,32,33,34,35,36,37,38,39,40, 41,42,43,44,45,46,47,48,49,50, 51,52,53,54	34	80	40.7	SW-2, SW-1
21	22	22,40,23,24,25,26,41,27,28,42, 43,29,30,31,32,33,34,35,36,49, 50,54,37,53,38,52,39,51,44,45, 48,46,47	33	78	45.5	SW-2, SW-1
22	23	23,24,25,26,41,27,28,42,43,29, 30,31,32,33,34,35,36,49,50,54, 37,53,38,52,39,51,44,45,48,46, 47	31	74	45.3	SW-2, SW-1
23	24	24,25,26,41,27,28,42,43,29,30, 31,32,33,34,35,36,49,50,54,37, 53,38,52,39,51,44,45,48,46,47	30	72	47	SW-2, SW-1
24	25	25,26,41,27,28,42,43,29,30,31, 32,33,34,35,36,49,50,54,37,53, 38,52,39,51,44,45,48, 46,47	29	70	46.9	SW-2, SW-1
4	17	5,6,7,8,9,10,12,13,14,15,16, 17,18,19,20	15	51	34.3	SW-1, SW-4, SW-2, SW-6, SW-7
17	5	5,6,7,8,9,10,12,13,14,15,16,18, 19,20	14	49	40.7	SW-1, SW-4, SW-2, SW-6, SW-7
5	14	14	1	5	9.5	SW-5
5	6	6,7,8,9,10,12,13,15,16,18,19,20	12	36	36	SW-5, SW-1,

Source Node	Destination Node	Nodes Affected	No. of Nodes Affected	Messages Passed	Time Taken (ms)	Switch Positions Suggested
						SW-6, SW-7
6	7	7,8,9,12,13,15,16,18,19,20	10	29	34.6	SW-5, SW-1, SW-7
6	10	10	1	5	9.3	SW-6
7	8	8,9,12,13,15,16,18,19,20	9	27	31.4	SW-5, SW-1, SW-7
8	16	16	1	3	9.3	none
8	18	9,12,13,15,18,19,20	7	20	31.5	SW-5, SW-1
18	19	9,12,13,15,19,20	6	18	25	SW-5, SW-1
19	20	9,12,13,15,20	5	16	26.6	SW-5, SW-1
20	9	9,12,13,15	4	14	24.9	SW-5, SW-1
9	12	12,13,15	3	12	26.5	SW-5, SW-1
12	13	13	1	5	11.1	SW-5
12	15	15	1	5	14.2	SW-1
26	41	41	1	3	6.4	none
26	27	27,28,42,43,29,30,31,32,33,34, 35,36,49,50,54,37,53,38,52,39, 51,44,45,48,46,47	26	64	42.2	SW-2, SW-1
27	28	28,42,43,29,30,31,32,33,34,35, 36,49,50,54,37,53,38,52,39,51, 44,45,48,46,47	25	62	40.8	SW-2, SW-1
28	42	42	1	3	6.1	none
28	29	43,29,30,31,32,33,34,35,36,49, 50,54,37,53,38,52,39,51,44,45, 48,46,47	23	58	34.4	SW-2, SW-1
29	43	43	1	3	1.6	None
29	30	30,31,32,33,34,35,36,49,50,54, 37,53,38,52,39,51,44,45,48,46, 47	21	54	37.4	SW-2, SW-1

Source Node	Destination Node	Nodes Affected	No. of Nodes Affected	Messages Passed	Time Taken (ms)	Switch Positions Suggested
30	44	44,45,48,46,47	5	19	17.6	SW-3, SW-2, SW-1
44	45	45,46,47,48	4	14	12.6	SW-3, SW-2
45	46	46,47	2	7	10.7	SW-2
46	47	47	1	5	1.5	SW-2
45	48	48	1	5	12.4	SW-3
31	32	32,33,34,35,36,49,50,54,37,53, 38,52,39,51	14	31	31.5	SW-3
32	33	33,34,35,36,49,50,54,37,53,38, 52,39,51	13	26	23.7	none
33	34	34,35,36,49,50,54,37,53,38,52, 39,51	12	24	24.9	none
34	35	35,36,49,50,54,37,53,38,52,39, 51	11	22	23.4	none
35	49	49	1	3	4.7	none
36	37	37,53,38,52,39,51	6	12	14.1	none
36	50	50,54	2	4	3.1	none
50	54	54	1	2	4.7	none
37	53	53	1	2	4.6	none
37	38	38,52,39,51	4	8	10.9	none
38	52	52,51	2	4	3	none
52	51	51	1	3	4.7	none
38	39	39	1	2	6.3	none
3	4	4,5,6,7,8,9,10,12,13,14,15, 16,17,18,19,20	16	53	54.9	SW-1, SW-4, SW-2, SW-6, SW-7
25	26	26,41,27,28,42,43,29,30,31,32, 33,34,35,36,49,50,54,37,53,38, 52,39,51,44,45,48,46, 47	28	68	38.9	SW-2, SW-1

Source Node	Destination Node	Nodes Affected	No. of Nodes Affected	Messages Passed	Time Taken (ms)	Switch Positions Suggested
30	31	31,32,33,34,35,36,49,50,54, 37,53,38,52,39,51	15	33	29.5	SW-3
35	36	36,50,54,37,53,38,52,39,51	9	18	18.8	none

Simulation results for topology independent decentralized MAS Architecture

Source Node	Destination Node	Nodes affected	No. of nodes affected	Messages passed	Time Taken (ms)	Switch Positions Suggested
2	55	11,55,56,57,58,59,60,61,62, 63,64,65,66	13	35	39.1	SW-4, SW-6, SW-7
55	56	11,56,57,58,59,60,61,62,63, 64,65,66	12	33	40.9	SW-4, SW-6, SW-7
56	57	11,57,58,59,60,61,62,63,64, 65,66	11	31	40.6	SW-4, SW-6, SW-7
57	58	11,58,59,60,61,62,63,64,65, 66	10	29	46.7	SW-4, SW-6, SW-7
58	61	61,62	2	7	7.8	SW-4
61	62	62	1	5	11	SW-4
58	66	11,59,60,63,64,65,66	7	20	32.7	SW-6, SW-7
66	59	11,59,60,63,64,65	6	18	31.2	SW-6, SW-7
59	65	63,65	2	7	9.4	SW-6
65	63	63	1	5	12.4	SW-6
59	64	11,60,64	3	9	17.1	SW-7
64	60	11,60	2	7	18.7	SW-7
60	11	11	1	5	4.7	SW-7
22	40	40	1	3	3.1	none
1	2	2,3,4,5,6,7,8,9,10,11,12,13, 14,15,16,17,18,19,20,21,22, 23,24,25,26,27,28,29,30,31, 32,33,34,35,36,37,38,39,40, 41,42,43,44,45,46,47,48,49, 50,51,52,53,54,55,56,57,58, 59,60,61,62,63,64,65,66	65	172	73.4	none
2	3	3,4,5,6,7,8,9,10,12,13,14,15, 16,17,18,19,20	17	55	50.1	SW-2, SW-4, SW-6, SW-7,

Source Node	Destination Node	Nodes affected	No. of nodes affected	Messages passed	Time Taken (ms)	Switch Positions Suggested
						SW-1
2	21	21,22,23,24,25,26,27,28,29, 30,31,32,33,34,35,36,37,38, 39,40,41,42,43,44,45,46,47, 48,49,50,51,52,53,54	34	80	40.7	SW-1, SW-2
21	22	22,40,23,24,25,26,41,27,28, 42,43,29,30,31,32,33,34,35, 36,49,50,54,37,53,38,52,39, 51,44,45,48,46,47	33	78	45.5	SW-1, SW-2
22	23	23,24,25,26,41,27,28,42,43, 29,30,31,32,33,34,35,36,49, 50,54,37,53,38,52,39,51,44, 45,48,46,47	31	74	45.3	SW-1, SW-2
23	24	24,25,26,41,27,28,42,43,29, 30,31,32,33,34,35,36,49,50, 54,37,53,38,52,39,51,44,45, 48,46,47	30	72	47	SW-1, SW-2
24	25	25,26,41,27,28,42,43,29,30, 31,32,33,34,35,36,49,50,54, 37,53,38,52,39,51,44,45,48, 46,47	29	70	46.9	SW-1, SW-2
4	17	5,6,7,8,9,10,12,13,14,15,16, 17,18,19,20	15	51	34.3	SW-4, SW-2, SW-6, SW-7, SW-1
17	5	5,6,7,8,9,10,12,13,14,15,16, 18,19,20	14	49	40.7	SW-4, SW-2, SW-6, SW-7, SW-1
5	14	14	1	5	9.5	SW-5
5	6	6,7,8,9,10,12,13,15,16,18,	12	36	36	SW-7, SW-6,

Source Node	Destination Node	Nodes affected	No. of nodes affected	Messages passed	Time Taken (ms)	Switch Positions Suggested
		19,20				SW-1, SW-5
6	7	7,8,9,12,13,15,16,18,19,20	10	29	34.6	SW-7, SW-1, SW-5
6	10	10	1	5	9.3	SW-6
7	8	8,9,12,13,15,16,18,19,20	9	27	31.4	SW-7, SW-1, SW-5
8	16	16	1	3	9.3	none
8	18	9,12,13,15,18,19,20	7	20	31.5	SW-1, SW-5
18	19	9,12,13,15,19,20	6	18	25	SW-5, SW-1
19	20	9,12,13,15,20	5	16	28.2	SW-1, SW-5
20	9	9,12,13,15	4	14	24.9	SW-5, SW-1
9	12	12,13,15	3	12	26.5	SW-1, SW-5
12	13	13	1	5	11.1	SW-5
12	15	15	1	5	14.2	SW-1
26	41	41	1	3	6.4	none
26	27	27,28,42,43,29,30,31,32,33, 34,35,36,49,50,54,37,53,38, 52,39,51,44,45,48,46,47	26	64	42.2	SW-1, SW-2
27	28	28,42,43,29,30,31,32,33,34, 35,36,49,50,54,37,53,38,52, 39,51,44,45,48,46,47	25	62	40.8	SW-1, SW-2
28	42	42	1	3	6.1	none
28	29	43,29,30,31,32,33,34,35,36, 49,50,54,37,53,38,52,39,51, 44,45,48,46,47	23	58	34.4	SW-1, SW-2
29	43	43	1	3	1.6	None
29	30	30,31,32,33,34,35,36,49,50, 54,37,53,38,52,39,51,44,45, 48,46,47	21	54	37.4	SW-1, SW-2

Source Node	Destination Node	Nodes affected	No. of nodes affected	Messages passed	Time Taken (ms)	Switch Positions Suggested
30	44	44,45,48,46,47	5	19	17.6	SW-1, SW-3, SW-2
44	45	45,46,47,48	4	14	12.6	SW-2
45	46	46,47	2	7	10.7	SW-2
46	47	47	1	5	1.5	SW-2
45	48	48	1	5	12.4	SW-3
31	32	32,33,34,35,36,49,50,54,37, 53,38,52,39,51	14	31	31.5	SW-3
32	33	33,34,35,36,49,50,54,37,53, 38,52,39,51	13	26	23.7	none
33	34	34,35,36,49,50,54,37,53,38, 52,39,51	12	24	24.9	none
34	35	35,36,49,50,54,37,53,38,52, 39,51	11	22	23.4	none
35	49	49	1	3	4.7	none
36	37	37,53,38,52,39,51	6	12	14.1	none
36	50	50,54	2	4	3.1	none
50	54	54	1	2	4.7	none
37	53	53	1	2	4.6	none
37	38	38,52,39,51	4	8	10.9	none
38	52	52,51	2	4	3	none
52	51	51	1	3	4.7	none
38	39	39	1	2	6.3	none
3	4	4,5,6,7,8,9,10,12,13,14,15, 16,17,18,19,20	16	53	54.9	SW-4, SW-2, SW-6, SW-7, SW-1
25	26	26,41,27,28,42,43,29,30,31, 32,33,34,35,36,49,50,54,37,5 3,38,52,39,51,44,45,48,46,47	28	68	38.9	SW-1, SW-2

Source Node	Destination Node	Nodes affected	No. of nodes affected	Messages passed	Time Taken (ms)	Switch Positions Suggested
30	31	31,32,33,34,35,36,49,50,54, 37,53,38,52,39,51	15	33	29.5	SW-3
35	36	36,50,54,37,53,38,52,39,51	9	18	18.8	none

Comparison of simulation results for proposed scalable reconfiguration algorithm and brute force technique

		Proposed Scalable Reconfiguration Algorithm		Brute Force Technique
Source Node	**Destination Node**	**Switch Combination Suggested**	**No. of Combinations Tried**	**No. of Combinations Tried**
2	55	SW7, SW1, SW4	7	33
55	56	SW7, SW1, SW4	7	33
56	57	SW7, SW1	2	8
57	58	SW7, SW1	2	4
58	61	SW4	1	1
61	62	SW4	1	1
58	66	SW7	1	6
66	59	SW7	1	6
59	65	SW6	1	1
65	63	SW6	1	1
59	64	SW7	1	3

		Proposed Scalable Reconfiguration Algorithm		Brute Force Technique
Source Node	**Destination Node**	**Switch Combination Suggested**	**No. of Combinations Tried**	**No. of Combinations Tried**
64	60	SW7	1	1
60	11	SW7	1	1
22	40	none	0	0
1	2	none	0	0
2	3	SW4, SW1	3	11
2	21	SW1, SW5, SW4	10	40
21	22	SW1, SW5, SW4	10	40
22	23	SW1, SW5, SW4	10	40
23	24	SW1, SW5, SW4	10	40
24	25	SW1, SW5	2	2
4	17	SW2, SW4, SW1	8	11
17	5	SW2, SW1	2	9

		Proposed Scalable Reconfiguration Algorithm		Brute Force Technique
Source Node	**Destination Node**	**Switch Combination Suggested**	**No. of Combinations Tried**	**No. of Combinations Tried**
5	14	SW5	1	1
5	6	SW5	1	5
6	7	SW5	1	1
6	10	SW6	1	1
7	8	SW5	1	1
8	16	none	0	0
8	18	SW5	1	1
18	19	SW5	1	1
19	20	SW5	1	1
20	9	SW5	1	1
9	12	SW5	1	1
12	13	SW5	1	1

		Proposed Scalable Reconfiguration Algorithm		Brute Force Technique
Source Node	**Destination Node**	**Switch Combination Suggested**	**No. of Combinations Tried**	**No. of Combinations Tried**
12	15	SW1	1	1
26	41	none	0	0
26	27	SW1SW2	2	2
27	28	SW1SW2	2	2
28	42	none	0	0
28	29	SW1SW2	2	2
29	43	none	0	0
29	30	SW1SW2	2	2
30	44	SW2	2	2
44	45	SW2	1	2
45	46	SW2	1	1
46	47	SW2	1	1

		Proposed Scalable Reconfiguration Algorithm		Brute Force Technique
Source Node	**Destination Node**	**Switch Combination Suggested**	**No. of Combinations Tried**	**No. of Combinations Tried**
45	48	SW3	1	1
31	32	SW3	1	3
32	33	none	0	0
33	34	none	0	0
34	35	none	0	0
35	49	none	0	0
36	37	none	0	0
36	50	none	0	0
50	54	none	0	0
37	53	none	0	0
37	38	none	0	0
38	52	none	0	0

		Proposed Scalable Reconfiguration Algorithm		Brute Force Technique
Source Node	**Destination Node**	**Switch Combination Suggested**	**No. of Combinations Tried**	**No. of Combinations Tried**
52	51	none	0	0
38	39	none	0	0
3	4	SW4SW1	3	11
25	26	SW1SW2	2	2
30	31	SW3	1	3
35	36	none	0	0

References

[1] G. Weiss, *Multiagent systems a modern approach to distributed artificial intelligence*, The MIT Press, 1999.

[2] S. Russell, and P. Norvig, *Artificial intelligence: a modern approach*, Prentice Hall, 1995.

[3] F. Avellaneda, C. Bustacara, J.P. Garzon, and E. Gonzalez, *Implementation of a molecular simulator based on a multiagent system,* in Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology, pp. 117-120, 2006.

[4] J. Yen, A. Chung, H. Ho, B. Tam, R. Lau, M. Chua, and K. Hwang, *Collaborative and scalable financial analysis with multi-agent technology*, in Proceedings of the 32nd Annual Hawaii International Conference, vol. Track5, 1999.

[5] K.P. Chow, and Y.K. Kwok, *On load balancing for distributed multiagent computing*, in IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 8, pp. 787-801, 2002.

[6] L.C. Lee, H.S. Nwana, D.T. Ndumu, and P.D. Wilde, *The stability, scalability and performance of multi-agent systems*, in BT Technology Journal, vol. 16, no. 3, pp. 94-103, 1998.

[7] O.F. Rana, and K. Stout, *What is scalability in multi-agent systems*, in Proceedings of the fourth international conference on Autonomous agents, pp. 56-63, 2000.

[8] S.H. Nwana, and L.C. Lee, *Stability, fairness and scalability of multi-agent systems*, in International Journal of Knowledge-Based Intelligent Engineering Systems, vol. 3, pp. 2-3, 1999.

[9] P.J. Turner, and N.R. Jennings, *Improving the scalability of multi-agent systems*, in Proceedings of 1st International Workshop on Infrastructure for Scalable Multi-Agent Systems, pp. 246-262, 2000.

[10] P. Scerri, Y. Xu, E. Liao, J. Lai, and K. Sycara, *Scaling teamwork to very large teams*, in Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 888-895, 2004.

[11] K. Huang, *Shipboard power system reconfiguration using multi agent system*, Ph.D. dissertation, The Florida state university, 2007.

[12] G.A. Taylor, M.R. Irving, P.R. Hobson, C. Huang, P. Kyberd, and R.J. Taylor, *Distributed monitoring and control of future power systems via grid computing*, in IEEE Power Engineering Society General Meeting, 2006.

[13] D.A. Cartes, and S.K. Srivastava, *Agent applications and their future in the power industry*, in IEEE Power Engineering Society General Meeting, pp. 1-6, 2007.

[14] H. Salazar, R. Gallego, and R. Romero, *Artificial neural networks and clustering techniques applied in the reconfiguration of distribution systems*, in IEEE Transactions on Power Delivery, vol. 21, no. 3, pp. 1735-1742, 2006.

[15] T. Brunner, W. Nejdl, H. Schwarzjirg, and M. Sturm, *On-line expert system for power system diagnosis and restoration*, in Intelligent Systems Engineering, vol. 2, no. 1, pp. 15-24, 1993.

[16] C.C. Liu, J. Jung, G.T. Heydt, V. Vittal, and A.G. Phadke, *Strategic power infrastructure defense (SPID) system a conceptual design*, in IEEE Control Syst. Mag., vol. 20, no. 4, pp. 40-52, 2000.

[17] L. Liu, K.P. Logan, D.A. Cartes, and S.K. Srivastava, *Fault detection, diagnostics, and prognostics: software agent solutions* , in IEEE Transactions on Vehicular Technology, vol. 56, no. 4, pp. 1613-1622, 2007.

[18] J.G. Gomez-Gualdron, M. Velez-Reyes, and L.J. Collazo, *Self-reconfigurable electric power distribution system using multi-agent systems*, in IEEE Electric Ship Technologies Symposium, pp. 180-187, 2007.

[19] I.S. Baxevanos, and D.P. Labridis, *Implementing multiagent systems technology for power distribution network control and protection management*, in IEEE Transactions on Power Delivery, vol. 22, no. 1, pp. 433-443, 2007.

[20] T. Nagata, Y. Tao, H. Sasaki, and H. Fujita, *Decentralized approach to power system restoration by means of multi-agent approach,* in Bulk Power System Dynamics and Control - VI, 2004.

[21] K. Huang, S.K. Srivastava, D.A. Cartes, and M. Sloderbeck, *Intelligent agents applied to reconfiguration of mesh structured power systems*, in International Symposium on Antennas and Propagation, pp. 298-304, 2007.

[22] K. Huang, D.A. Cartes, and S.K. Srivastava, *A multiagent-based algorithm for ring-structured shipboard power system reconfiguration*, in The International Conference on System, Man and Cybernetics, vol. 1, pp. 530-535, 2005.

[23] K. Huang, D.A. Cartes, and S.K. Srivastava, *A multiagent-based algorithm for ring-structured shipboard power system reconfiguration* , in IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 37, no. 5, pp. 1016-1021, 2007.

[24] K. Huang, S. Sanjeev, and D. Cartes, *Decentralized reconfiguration for power systems using multi agent system*, in Proceedings of the 1st Annual 2007 IEEE Systems Conference, pp. 253-258, 2007.

[25] K. Huang, S.K. Srivastava, D.A. Cartes, and M. Sloderbeck, Intelligent agents applied to reconfiguration of mesh structured power systems, in International Conference on Intelligent Systems Applications to Power Systems, pp. 1-7, 2007.

[26] F. Ponci, and A.A. Deshmukh, A mobile agent for measurements in distributed power electronic systems, in IEEE Instrumentation and Measurement Technology Conference, pp. 870-875, 2008.

[27] D. Elizalde, D. Staszesky, and M. Meisinger, Use of distributed intelligence for reliability improvement using minimum available distribution assets, in IEEE/PES Transmission and Distribution Conference and Exposition, pp. 1-6, 2006.

[28] D.M. Staszesky, Use of virtual agents to effect intelligent distribution automation , in IEEE Power Engineering Society General Meeting, 2006.

[29] Z. Li, X. Chen, K. Yu, B. Zhao, and H. Liu, A novel approach for dynamic reconfiguration of the distribution network via multi-agent system, in 3rd International Conference on Deregulation and Restructuring and Power Technologies, pp. 1305-1311, 2008.

[30] K. Aoki, K. Nara, M. Itoh, T. Satoh, and H. Kuwabara, *A New Algorithm for Service Restoration in Distribution Systems*, in IEEE Transactions on Power Delivery, vol. 4, no. 3, pp. 1832 – 1839, 1989.

[31] G.S. Wang, P.Y. Wang, Y.H. Song, and A.T. Johns, *Co-ordinated system of fuzzy logic and evolutionary programming based network reconfiguration for loss reduction in distribution systems*, in Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, vol. 3, pp. 1838-1844,1996.

[32] Y.Y. Hsu, and H.M. Huang, *Distribution system service restoration using the artificial neural network approach and pattern recognition method*, in *IEE* Proceedings on Generation, Transmission and Distribution, vol. 142, no. 3, pp. 251-256, 1995.

[33] W. Lin, H. Chin, and G. Yu, *An effective algorithm for distribution feeder loss reduction by switching operations*, in *IEEE Transmission and Distribution Conference*, vol. 2, pp. 597-602, 1999.

[34] I. Aleksander, and H. Marton, *An introduction to neural computing*, Chapman & Hall, 1992

[35] H. Kim, Y. Ko, and K.H. Jung, *Artificial neural-network based feeder reconfiguration for loss reduction in distribution systems*, in IEEE Transactions on Power Delivery, vol. 8, no. 3, pp. 1356-1366, 1993.

[36]J. Pearl, *Heuristics Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, 1984.

[37] Y.Y. Hsu, H.M. Huang, H.C. Kuo, S.K. Peng, C.W. Chang, K.J. Chang, H.S. Yu, C.E. Chow, and R.T. Kuo, *Distribution system service restoration using a heuristic search approach*, in IEEE Transactions on Power Delivery, vol. 7, no. 2, pp. 734-740, 1992.

[38] T. Taylor, and D. Lubkeman, *Implementation of Heuristic Search Strategies for Distribution Feeder Reconfiguration*, IEEE Transactions on Power Delivery, vol. 5, no. 1, pp. 239-246, 1990.

[39] S. Civanlar, J.J. Grainger, H. Yin, and S.S.H. Lee, *Distribution feeder reconfiguration for loss reduction*, in IEEE Transactions on Power Delivery, vol. 3, no. 3, pp. 1217-1223, 1988.

[40] A. Konar, *Computational intelligence principle, techniques, and applications*, Springer, 2005.

[41] T. Taylor, and D. Lubkeman, *Applications of knowledge-based programming to power engineering problems*, in IEEE Transactions on Power Systems, vol. 4, no. 1, pp. 345-352, 1989.

[42] C.C. Liu, S.J. Lee, and S.S. Venkata, *An expert system operational aid for restoration and loss reduction of distribution systems*, in IEEE Transactions on Power Systems, vol. 3, no. 2, pp. 619-626,1988.

[43] K. Hotta, H. Nomura, H. Takemoto, K. Suzuki, S. Nakamura, and S. Fukui, *Implementation of a real-time expert system for a restoration guide in a dispatching center*, in IEEE Transactions on Power Systems, vol. 5, no. 3, pp. 1032-1038, 1990.

[44] G. Chang, J. Zrida, and J.D. Birdwell, *Knowledge-based distribution system analysis and reconfiguration*, in IEEE Transactions on Power Systems, vol. 5, no. 3, pp. 744-749, 1990.

[45] Y.H. Song, G.S. Wang, A.T. Johns, and P.Y. Wang, *Distribution network reconfiguration for loss reduction using fuzzy controlled evolutionary programming*, in IEE Proceedings on Generation, Transmission and Distribution, vol. 144, no. 4, pp. 345-350,1997.

[46] Y. Fukuyama, and C. Hsaio-Dong, *A parallel genetic algorithm for service restoration in electric power distribution systems*, in Proceedings of IEEE International Conference on Fuzzy Systems, vol. 1, pp. 275-282, 1995.

[47] P. Ravibabu, K. Venkatesh, and C.S. Kumar, *Implementation of genetic algorithm for optimal network reconfiguration in distribution systems for load balancing*, in IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering, pp. 124-128, 2008.

[48] J. Wang, A. Luo, Q. Mingjun, and L. Maojun, *The improved clonal genetic algorithm & its application in reconfiguration of distribution networks*, in IEEE PES Power Systems Conference and Exposition, vol. 3, pp. 1423-1428, 2004.

[49] Y. Hong, and S. Ho, *Genetic algorithm based network reconfiguration for loss minimization in distribution systems*, in IEEE Power Engineering Society General Meeting, vol. 1, pp. 486-490, 2003.

[50] H. Qi, W. Zhang, and L.M. Tolbert, *A Resilient Real-Time Agent-Based System for a Reconfigurable Power Grid*, in Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems, pp. 43-48, 2005.

[51] X. Bian, Q. Ma, and Y. Zhou, *A Multi-Agent Approach to the More Electric Aircraft Power System Reconfiguration*, in ICMA International Conference on Mechatronics and Automation, pp. 978-982, 2007.

[52] J.A. Momoh, and O.S. Diouf, *Optimal Reconfiguration of the Navy Ship Power System using Agents*, in IEEE PES Transmission and Distribution Conference and Exhibition, pp. 562-567, May 2006.

[53] Y. Qudaih, and T. Hiyama, *Reconfiguration of power distribution system using multi agent and hierarchical based load following operation with energy capacitor system*, in International Power Engineering Conference, pp. 223-227, 2007.

[54] J.G. Gomez-Gualdron, and M. Velez-Reyes, *Simulating a Multi-Agent based Self-Reconfigurable Electric Power Distribution System*, in IEEE Workshops on Computers in Power Electronics, pp. 1-7, 2006.

[55] J.C. Jacobo, D.D. Roure, and E.H. Gerding, *An agent-based electrical power market*, in Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1655-1656 2008.

[56] I. Praça, A. Andrade, H. Morais, M. Cardoso, C. Ramos, and Z. Vale, *A multi-agent system for the support of producer coalition formation in electricity markets*, in Proceedings of the 2008 ACM Symposium on Applied Computing, 2008.

[57] K. Hak-Man, and T. Kinoshita, *Multiagent system for Microgrid operation based on power market environment*, in Telecommunications Energy Conference, pp. 1-5, 2009.

[58] V. Krishna, and V.C. Ramesh, *Intelligent agents for negotiations in market games. I. Model [electricity supply]*, in International Conference on Power Industry Computer Applications, pp. 388-393, 1997.

[59] V. Krishna, V.C. Ramesh, *Intelligent agents for negotiations in market games. 2. Application*, in *IEEE Transactions on Power Systems,* vol. 13, no. 3, pp. 1109-1114, 1998.

[60] H.F. Wang, *Multi-agent co-ordination for the secondary voltage control in power-system contingencies*, in IEE Proceedings on Generation, Transmission and Distribution, vol. 148, no. 1, pp. 61-66, 2001.

[61] S. Gehao, J. Xiuceng, and Z. Yi, *Optimal Coordination For Multi-Agent Based Secondary Voltage Control In Power System*, in Transmission and Distribution Conference and Exhibition, pp. 1-6, 2005.

[62] G.S.H. Jarrett, and M. Kaufmann, Power *System Protection*, Institution of Engineering and Technology, 2007.

[63] S.K. Wong, and A. Kalam, *Distributed intelligent power system protection using case based and object oriented paradigms*, in International Conference on Intelligent Systems Applications to Power Systems, pp. 74-78, 1996.

[64] H. Wan, K.K. Li, and K.P. Wong, *An multi-agent approach to protection relay coordination with distributed generators in industrial power distribution system*, in Industry Applications Conference, vol. 2, pp. 830- 836, 2005.

[65] Z. Chen, and W. Kong, *Protection coordination based on multiagent for distribution power system with distribution generation units*, 2008.

[66] Z. Xiangjun, K.K. Li, W.L. Chan, and S. Sheng, *Multi-Agents based Protection for Distributed Generation Systems*, in Proceedings of the IEEE International Conference on Electric Utility Deregulation, Restructuring and Power Technologies, vol. 1, pp. 393–397, 2004.

[67]M.R. Genesereth, and S.P. Ketchpel, *Software agents*,in ACM Communications,vol. 37, no.7, 1994.

[68] O. Etzioni, N. Lesh, and R. Segal, *Building softbots for UNIX,* In Spring Symposium, pp. 9–16, 1994.

[69]Y. Shoham, *Agent-oriented programming*, in Artificial Intelligence Journal, *vol.* 60, no. 1, pp. 51-92, 1993.

[70] J. Ferber, *Multi-Agent Systems An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, 1999.

[71] P. Maes, *Artificial life meets entertainment: lifelike autonomous agents*, ACM Communications, vol. 38, no. 11, pp. 108-114, 1995.

[72] D.C. Smith, A. Cypher, and J. Spohrer, *KidSim: programming agents without a programming language*, ACM Communications, vol. 37, no. 7, pp. 54-67, 1994.

[73] J.S. Rosenschein, and M.R. Genesereth, Deals among rational agents, in Distributed Artificial Intelligence, Morgan Kaufmann Publishers Inc, pp. 227-234, 1988.

[74] S. Franklin, and A. Graesser, *Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents*, in Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages, pp. 21-35, 1997.

[75] N.R. Jennings, K. Sycara, and M. Wooldridge, *A roadmap to agent research and development*, in Autonomous Agents and Multi-Agent Systems, 1998.

[76] U. Luxburg, A tutorial on spectral clustering, in Statistics and Computing, vol. 17, no. 4, pp. 395-416, 2007.

[77] S.E. Schaeffer, Graph clustering, in Computer Science Review, vol. 1, no. 1, pp. 27-64, 2007.

[78] J. Shi, and J. Malik, *Normalized cuts and image segmentation*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 888 – 905, 2000.

[79] A. Ng, M. Jordan, and Y. Weiss, On spectral clustering: analysis and an algorithm, in Advances in Neural Information Processing Systems 14, 2001.

[80] L. Zelnik-Manor, and P. Perona, Self-tuning spectral clustering, Advances in Neural Information Processing Systems, 2004.

[81] S.L. Hamilton, C.K. Vartanian, M.E. Johnson, and A. Feliachi, K. Schoder, and P. Hines, Circuit of the future: interoperability and SCE's DER program, Bulk Power System Dynamics and ControlVII. Revitalizing Operational Reliability 2007 iREP Symposium, pp. 1-9, 2007.

[82] N. Muller, and V.H. Quintana, A sparse eigenvalue-based approach for partitioning power networks, in IEEE Transactions on Power Systems, vol. 7, no. 2, pp. 520-527, 1992.

[83] P.M. Anderson, and A.A. Fouad, Power system control and stability, IEEE Press, 2002.

[84] S. Moheuddin, A. Noore, and M. Choudhry, A reconfigurable distributed multiagent system optimized for scalability, in International Journal of Computational Intelligence, vol. 5, no. 1, pp. 60-71, 2009.

[85] F.L. Bellifemine, G. Caire, and D. Greenwood, Developing multi-agent systems with JADE, Wiley, 2007.

[86] K. Schoder, A. Hasanovic, A. Feliachi, *PAT: a power analysis toolbox for MATLAB/Simulink*, IEEE Transactions on Power Systems, vol. 18, no. 1, pp. 42- 47, 2003.