APPLICATION OF PROBABILISTIC METHODS FOR EFFECTIVE AND RELIABLE OPERATION OF ELECTRICAL AND ELECTROMECHANICAL SYSTEMS

Yerzhigit Bapin

A thesis is submitted in partial fulfilment of the requirements of Nazarbayev University for the degree of Doctor of Philosophy in Science, Engineering and Technology

School of Engineering and Digital Sciences
Department of Mechanical and Aerospace Engineering
Nazarbayev University

53 Kabanbay Batyr Avenue,
Nur-Sultan, Kazakhstan, 010000

Supervisor:
Vasileios Zarikas, Ph.D.
Department of Mechanical and Aerospace Engineering

March, 2021

Abstract

This PhD thesis presents novel system control methods that can be utilized for effective and reliable operation of electric grids and passenger elevators. First of all, this study introduces a new spinning reserve allocation optimization technique that takes into account load and renewable power generation, inter-zonal conventional power generating capacity and demand response. Using the bivariate Farlie-Gumbel-Morgenstern probability density function, the framework presented in this thesis utilizes a new method to simulate the power generation of wind farms. In addition, the presented framework uses a Bayesian Network (BN) algorithm to fine-tune the spinning reserve allocation based on previous hours' actual unit commitment, as well as the hour and day types.

The model proposed in this study has been tested on the IEEE Two-Area Reliability Test System (RTS) to quantify the effect of the bivariate wind prediction model and the Bayesian network-based Reserve Allocation Adjustment Algorithm (RAAA) on reliability and cost-effectiveness of the power grid. The findings show that combining a bivariate wind forecast model with RAAA improves power grid stability by 2.66 percent while lowering overall system running costs by 1.12 percent.

Secondly, the present work introduces an algorithm with an objective of optimal dispatching control of passenger lifts. The algorithm utilizes the data received from video cameras and dispatches the elevator cars based on the passenger count. The proposed algorithm utilizes the information on the number of people and dispatches the lifts with an objective to move the maximum number of passengers to the desired building levels within the minimum amount of time. In addition, the algorithm considers each person's size and whether or not they have luggage. To account for uncertainty in image acquisition, the algorithm assigns the probability weights to the number of people who are waiting for a lift and riding the lifts. The main purpose of the algorithm is to minimize the following performance metrics: average travel time (ATT), average journey time (AJT) and average waiting time (AWT).

The suggested algorithm works well in situations of limited traffic sizes, according to a test case scenario conducted on a ten-story office building having four elevator cars (less than 200 people). In a scenario with large up-peak high intensity traffic, the proposed algorithm primarily underperforms. The proposed algorithm's best output was seen in situations with random inter-floor passenger movement. In scenarios of changing traffic intensity and size ATT increased by 39.94 percent and 19.53 percent, respectively.

Table of Contents

List of Tables

List of Illustrations

Acronyms and Definitions

TSO – Transmission System Operator

ISO – Independent System Operator

SO – System Operator

PP – Power Plant

PG – Power Grid

EM – Energy Market

EC – Electricity Consumer

R - Retailer

DR – Demand Response

ELNS – Expected Load Not Served

EENS – Expected Energy Not Supplied

LOLP – Loss of Load Probability

LOL – Loss of Load

DLC – Direct Load Control

IEEE – Institute of Electrical and Electronics Engineers

ASDR – Ancillary Service Demand Response

SCUC – Security Constrained Unit Commitment

MILP – Mixed Integer Linear Programming

RTS – Reliability Test System

SCED – Security Constrained Economic Dispatch

ESRMC - Energy and Spinning Reserve Market Clearing

AI – Artificial Intelligence

BN – Bayesian Network

EGC – Elevator Group Control

DisCo – Electricity Distribution Company

WIC – Wholesale Industrial Customer

REP – Renewable Energy Producer

FOR – Forced Outage Rate

COPT – Capacity Outage Probability Table

ORR – Outage Replacement Rate

PDF – Probability Density Function

CDF – Cumulative Density Function

RV – Random Variable

ICG – Interconnected Conventional Generator

DRP – Demand Response Provider

DT – Day Type

NDE – Net Demand Error

HT – Hour Type

CPD - Conditional Probability Distribution

CPT - Conditional Probability Table

VE – Variable Elimination

EM – Expectation Maximization

MARC - Missing at Random Completely

MAR – Missing at Random

USD – United States Dollar

MW – Megawatt

MWh – Megawatt-hour

Preface

Electricity plays a key role in our lives. At the beginning of its age, the electrical energy was mostly used for lighting, heating, cooking and running electric motors. Controlling of the electric devices, at that time, was mostly manual with little to no automation. The development of electronic industry led to increasing automation of electric and electromechanical devices and machinery.

Nowadays, almost all devices that use electricity are automated. The conventional automation logic is based on deterministic or conditional rules and criteria. Conventional electrical and electromechanical systems are not capable of accounting for uncertainties that arise during their operation. Often this leads to ineffective and unreliable work of these systems.

The rapid growth of computer technology bolsters the development of new and highly sophisticated algorithms that can be implemented to control and operate electrical and electromechanical systems. For instance, implementation of control strategies based on probabilistic methods or artificial intelligence can be used to reduce uncertainty and increase reliability of electrical grids, or increase efficiency of conventional passenger lifts.

The objective of this work is to test the following research hypothesis:

> ***Conventional operation and control strategies employed in power grids and passenger elevator systems can be improved, in terms of reliability and effectiveness, by implementation of probabilistic and machine learning algorithms based on the Bayesian inference.***

This PhD thesis focuses on the development and implementation of novel probabilistic, machine learning methods based on Bayesian inference to improve reliability and effectiveness of electrical grids and elevator systems.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Vasileios Zarikas for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Mehdi Bagheri and Prof. Nick Papanikolaou, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

My sincere thanks also goes to Dr. Luis R. Rojas-Solórzano and all administrative staff of Nazarbayev University School of Engineering and Digital Sciences for providing help and support throughout my journey this Ph.D programm.

Finally, my biggest thanks to my family for all the support you have shown me through this research. Thanks to my son, sorry for not spending enough time with you, and to my wife Gulvira, thanks for all your support, without which I would have stopped these studies a long time ago.

Author’s Declaration

I declare that the research contained in this thesis, unless otherwise formally indicated within the text, is the original work of the author. The thesis has not been previously submitted to this or any other university for a degree, and does not incorporate any material already submitted for a degree.

Yerzhigit Bapin

Date

Chapter 1 – Introduction

It should be noted that the main goal of this thesis is to present novel system control methods based on probabilistic and machine learning algorithms that could be utilized to increase the effectiveness and reliability of electrical and electromechanical systems.

Obviously, the definition of "*electrical system"* is more or less clear, and most of the readers would probably define it as a power grid with producers and consumers of electrical energy that are linked with each other by means of electric wires. However, the term *"electromechanical system"* is more general, and most of the readers would not think of a particular electromechanical system if they asked to do so. Electromechanical systems can be anything that consists of electrical and mechanical parts. Therefore, it is important to clarify what kind of systems exactly were considered in this work and what are the reasons for that.

As mentioned above, in this work the term *"electrical system"* is defined as a power grid system which is used to supply electricity to the people. The electromechanical systems, in this work, are represented by the passenger elevator systems, since both power grids and passenger elevators have common issues that could be solved or mitigated by a single approach. The following table specifies the issues that are common for both power grids and passenger elevators.

Table 1 – Comparison of power grid and passenger elevator systems

Problem	**Power Grid System**	**Passenger Elevator System**
High Uncertainty	Due to Renewables and Load	Due to Traffic Flow
Low Reliability/Dependability	Not Enough Reserve Capacity	Not Enough Information
Nonoptimal Dispatching	Nonoptimal Dispatch of Electric Energy	Nonoptimal Dispatch of Elevator Cars

First of all, the operation and control of both, power grids and passenger elevator systems can be improved if we find the ways to reduce the uncertainty that is caused by different system components. In power grids, the uncertainty, to a large extent, is caused by renewable power and load, and to a small extent by generator outages. In passenger elevators, most of the uncertainty comes from the passenger traffic. The primary source of this problem comes down to our inability to accurately predict future events, such as renewable power generation, load fluctuations, number of passengers wishing to go to a certain floor.

Second issue that arises during the operation of power grids and passenger elevators is related to their reliability or dependability. This issue is particularly significant in the power grid operation, since reduced reliability of a power grid can negatively affect our lives. Reliable operation of a power grid can be ensured by having enough operating reserve capacity, but this problem becomes more complex when we ask ourselves a question *"how much is enough"*?

Finally, optimization of resource dispatching is another issue that is common to both power grids and passenger elevators. In the context of this work, we focus on solving the problem of optimal allocation of spinning reserves in the power grids and optimization of dispatching of lifts in passenger elevator systems.

In connection to this, the rest of this work is focused on application of the approach that is based on probabilistic methods and Bayesian Inference for optimal allocation of spinning reserves in power grids and optimal dispatching control of passenger elevator systems.

1.1. Electric Grid Systems

In the pursuit of sustainable development many countries attempt to reduce the use of fossil fuels by transitioning to the technologies that are more ecologically friendly and less wasteful. There is no doubt that, that smart grid technologies along with renewables will play a key role in such transition. Within the past ten years the overall renewable capacity in the world has been doubled [1]. The long-term expectation for the total share of renewables in the total power generation is that it will reach 40 percent by 2040 [2]. Nevertheless, increasing the penetration levels of renewable energy is a challenging task. The main issue with integration of renewable energy into the grid system is that it is extremely stochastic and difficult to predict, it is therefore, the accommodation of high share of renewable power will need a flexibility of the grid system from the technical as well as operational perspectives.

The production-consumption equilibrium is normally secured by the transmission system operators (TSOs), sometimes referred to as independent system operators (ISOs), in conventional power grids by regulating the output of the power generation plants and/or by limiting the electricity consumption of large industrial consumers. In vertically integrated energy markets, such as the energy market of Kazakhstan, this is often achieved by directive control from the TSO side. In vertically integrated energy markets the production,

transmission and distribution of electricity is done by monopolies; thus, the main feature of such market structure is that it is highly regulated.

On contrary, in liberal energy markets, the supply-demand equilibrium is ensured through the market mechanisms by placing proper price signals. In this market structure the producers of electricity bid on an auction, and the bidders with lowest price get to sell their electricity to the consumers.

In the Ancillary service market, the market participants sell such services as spinning reserve, supplemental reserve, voltage regulation black-start and so on.

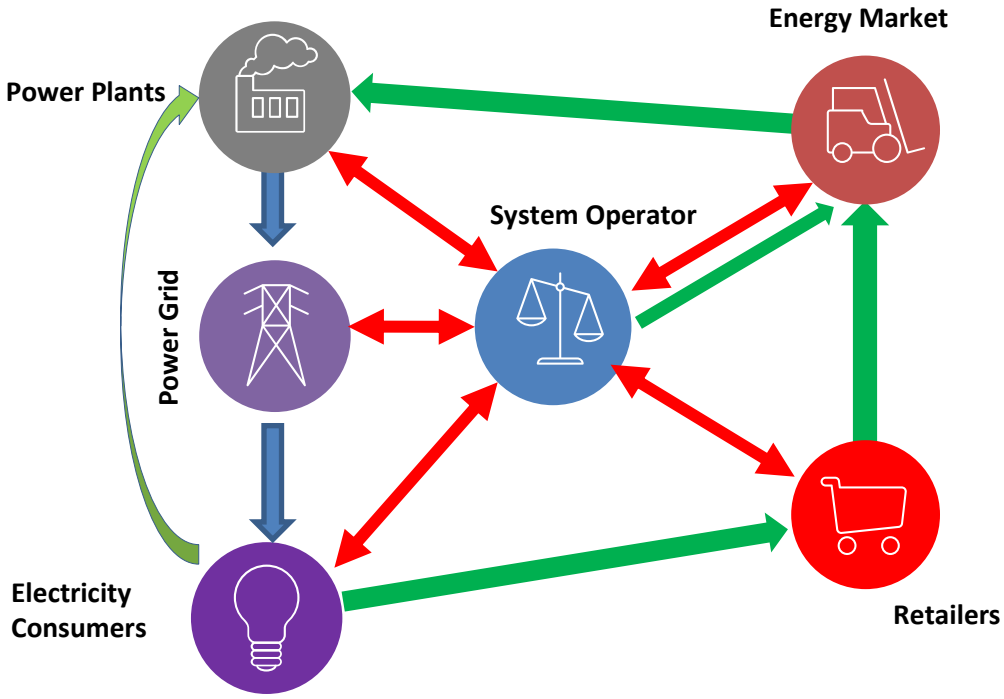


Figure 1. Interaction of the system operator with market entities

Figure 1 represents simplified schematic of how the system operator interacts with the market entities. The blue arrows represent the flow of services (electricity or ancillary services). As can be seen, the electricity is produced by the **Power Plants (PP)** and delivered to the **Electricity Consumers (EC)** through the **Power Grid (PG).** ECs use electricity for commercial or residential purposes. Besides the production of electrical energy, PPs provide ancillary services, which are procured by the **System Operator (SO).**

Green arrows represent the financial settlement between the market entities. In this scheme, ECs pay their money to **Retailers (R)** who purchase electricity on the energy market from PPs. It should be noted that in hybrid energy markets (i.e. the markets that

have features from both vertically integrated and liberal market structures), electricity procurement can be done directly between PP and EC via bilateral contracts. Similarly, SO purchase ancillary services from PPs through the energy market.

Finally, the red arrows represent the flow of information and commands from and to the system operator. The main purpose of this flow is to ensure power balance between energy production and load and continuous supply of electrical energy to consumers.

Traditional market structure becomes irrelevant when we introduce renewable energy and smart-grid technologies into the grid. The supply-demand equilibrium in smart grids can be partly accomplished by the use of demand response (DR) services. The development of economic incentives and the placement of appropriate technical solutions will help to implement DR programs [3]. Nowadays, different forms of DR schemes are in place to encourage people to reduce their energy demand at times of contingency or capacity shortages. Consumers are normally well compensated for voluntarily reducing their energy intake, which encourages them to engage in DR activities on a regular basis. In this manner, DR systems will help both grid operators and energy users.

Nevertheless, smooth integration of renewables and DR into existing grid infrastructure and market will require the introduction of novel methods of grid control and operation. Particularly, tremendous care must be taken when it comes to the adequacy and stability of electric grids. Nowadays, the main grid security quantification methods dominating in the power industry are deterministic and probabilistic. The prior approach requires having operating reserve capacity in the amount of the largest unit in feed or the percentage of peak demand.

Although, evaluation of power grid security based on deterministic methods does not take into account the uncertainties that occur during the grid operation, great deal of existing techniques are based on deterministic approach. The relative flexibility and lower input data criteria of deterministic reliability assessment approaches are the key reasons for their widespread use [4]. In the other hand, probabilistic reliability estimation is more sophisticated and necessitates comprehensive knowledge about system characteristics such as generator failure rates, load and renewable prediction faults, and so on.

The ability to capture system uncertainties and determine the magnitudes and consequences of these uncertainties on the function of power systems is one of the benefits

of probabilistic approaches over deterministic methods [3]. As a result, the probabilistic reliability criteria treat events based on their likelihood of occurrence and severity [5].

1.2. Conventional Passenger Elevators

In 2018, nearly 4.22 bln. people, or 55,3% of the world's population, lived in urban settlements [5]. People's natural tendency to migrate from rural to urban areas, along with global population growth, would end up in about 6.68 bln. people, which is equal to 68,4 percent of the world's population, living in urban settlements by the year of 2050 [5]. Considering all of the above, we should concentrate on reshaping traditional practices toward more sustainable electricity usage, pollution avoidance, and improved urban service efficiency. Given this view, the use of intelligent and environmentally friendly technology is critical for the long-term viability of urban settlements.

Some cities have already started an appealing, but difficult, journey to widespread acceptance of modern smart technologies. The transition to a smart city is a long-term process that necessitates significant improvements in both infrastructure and policy mechanisms. Since the timely and effective flow of people and goods has a positive impact on the growth and prosperity of cities, transportation system transformation is one of the key elements in terms of technical modernization. Many cities around the world have now adopted innovative public transportation and automobile traffic control strategies.

The city of Pittsburgh in the United States, for example, has introduced the Scalable Urban Traffic Control (SURTRAC) program. The main purpose of SURTRAC is to analyze information from the street intersections and take appropriate actions (change traffic lights) to improve traffic conditions in real time [6]. SURTRAC managed to minimize the average time vehicles spent in the traffic jams by 41%, reduce the number of exits by 31%, the total journey time by 26%, and car pollution by 21% [7].

For complete transition towards an intelligent urban settlement, smart technologies must be integrated into entire technological framework of the urban settlement as well as the building infrastructure, i.e. ventilation, heating, building transportation etc. Nowadays the interest of the research community is focused in means of public transportation inside the buildings, particularly elevator systems, since intelligent technologies integrated in the building infrastructure could result in an identical effect as smart solutions implemented in urban settlement's transportation systems.

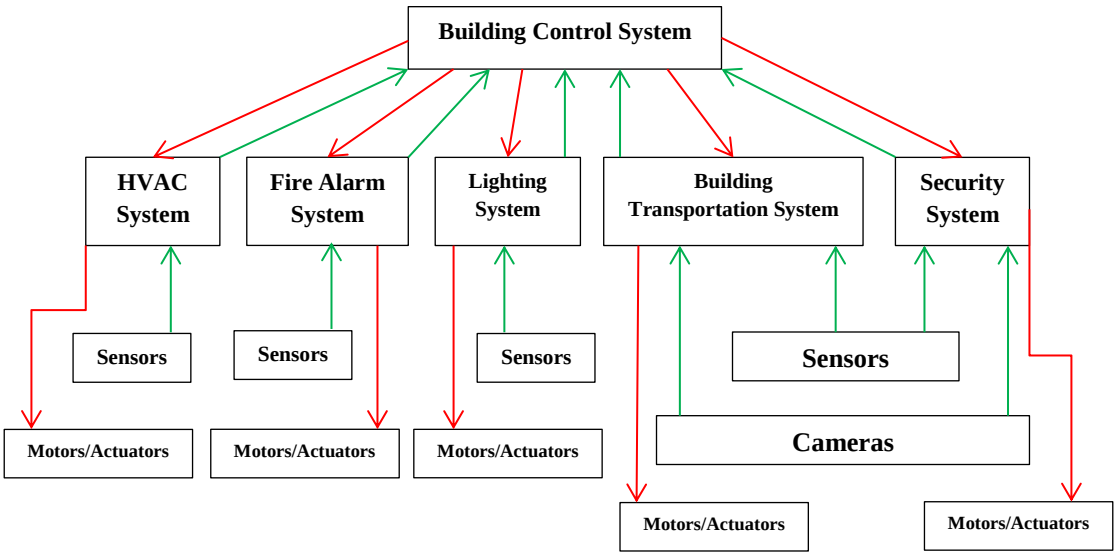


Figure 2. Simplified schematic of smart building control system

Figure 2 represents simplified schematic of smart buildings' control system. The green arrows represent the flow of information, while red arrows represent the flow of control commands. The building control system receives the information from sensors (temperature, humidity, motion sensor etc.) placed inside and outside of the building and from surveillance cameras. Based on this information, the building control system makes decisions and sends the action commands to the devices (motors and actuators) that control the operation of building's HVAC, Fire Alarm, Security, Lighting and Vertical Transportation systems.

Chapter 2 – Literature Review

This chapter presents a review of existing state-of-the-art developments related to application of probabilistic methods in estimation of spinning reserve requirements for electric grids and controlling the group of elevators using the information on the number of passengers obtained from surveillance cameras.

2.1. Electric Grid Systems

The interest in probabilistic methods of power system reliability assessment has been increasing with the growth of stochastic generation, mainly presented by wind and solar power. Various reserve estimation approaches, considering stochastic generation, have been proposed during the last few years. Figure 3 represents existing power system reliability assessment approaches. There are two distinct methods used to account for stochasticity in the grid system adequacy and reliability quantification. One method is to set an upper limit for reliability indices that assess the expected loss of load or capacity. Another way is by including an economic penalty into objective function. For example, Bouffard et al [8] suggest a hybrid deterministic/probabilistic reliability criterion by placing an upper limit on the loss of load probability (LOLP) and expected load not served (ELNS). Setting a reference value for the risk indexes will minimize computational intensity and also reduce the risk of finding suboptimal unit commitment schedules in some situations [9]. However, bounded reliability metrics models can struggle to find a global optimum.

The papers by Ortega-Vazques et al [10] and Liu et al [11] propose methodologies for computation of operating reserves based on the cost/benefit analysis. In these studies the minimization of overall system costs, which include unit operational expenses and load curtailment costs, determines the optimum spinning reserve requirements. Somewhat similar approach, based on the cost/benefit analysis has been proposed by Bapin et al [12]. The methodology considers uncertainties due to load and wind forecast errors and utilizes the equivalent assisting unit method to account for interconnection power flows. Certainly, the methodologies described above provide for more adequate estimation of operating reserve requirements compared to deterministic reliability assessment methods, yet emergence of new technologies and market mechanisms call for development of new reserve estimation methodologies.

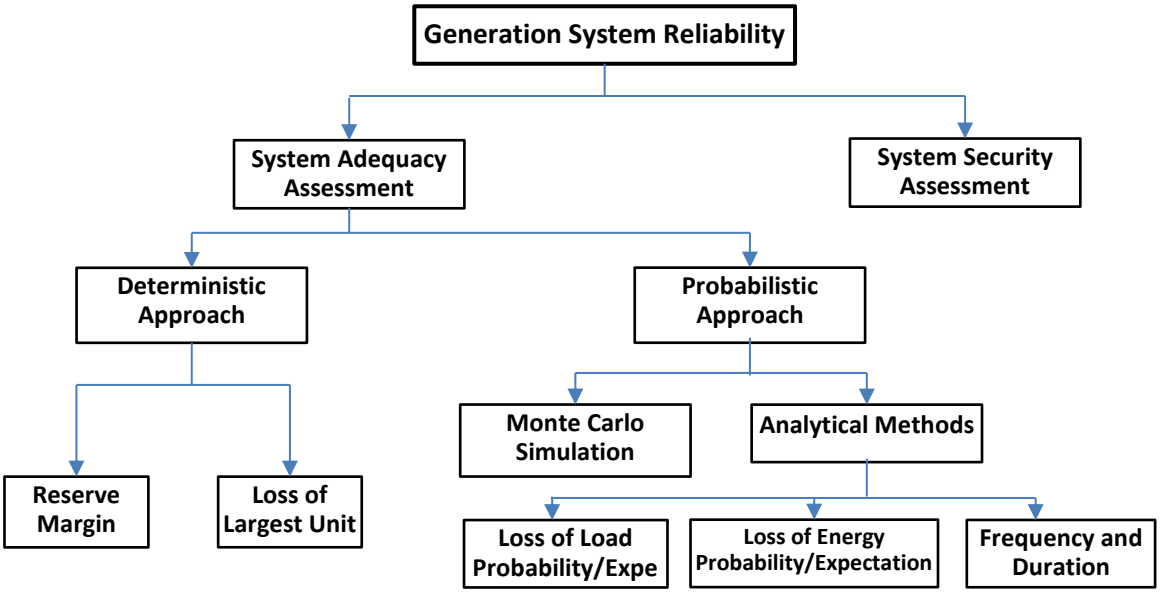


Figure 3. Power system reliability assessment methods

Despite the fact that demand response service is considered as a new technology in the electric power sector, much study has been conducted on how to incorporate DR into the energy market structure. The Demand Response Reserve Project (DRRP) built by the New England Independent System Operator (ISO-NE) is presented in a study by Burke et al [13]. DRRP's aim was to incorporate demand response into the energy market as ancillary service by delivering forward and operating reserve services. Nevertheless, the paper did not specify how ISO-NE assessed grid stability during DRRP deployment. A technique for quantification of the spinning reserve capacity using the Direct Load Control (DLC) software is suggested in a study by Shayesteh et al [14]. DLC is a demand management system that allows the utility to detach contracted consumers from the grid in the event of a power outage. The method was put to the test on an IEEE-14 bus system to see if the DLC program's implementation influenced the amount of Energy Not Supplied. It should be noted that only three out of ten potential contingency situations were examined, and simultaneous contingencies were not included. Without providing any justification, the DLC participation scenarios were set to 15 percent, 30 percent, and 50 percent. Parvania et al [15] presents a probabilistic model that allows to schedule reserves provided by DR in the Ancillary Service Demand Response (ASDR) market. Only outages of generating units and transmission lines, as well as load prediction error, were factored into the formula. The security constrained unit commitment (SCUC) problem was optimized in two stages using a mixed-integer linear program (MILP). The proposed method was applied on a six-bus system and the IEEE one area RTS. Both systems were believed to be completely disconnected from one another. Somewhat identical

methodology was developed in the paper by Khorsand et al [16]. However, unlike [15] this study neglects the uncertainty caused by load and renewable prediction errors, the system stochastics are given only as random outages of generation system. Karangelos et al [17] present a security-constrained joint forward reserve and energy market clearing mechanism that includes demand response reserves. The method utilizes SCUC formulation developed in their previous studies [18] and [19]. The reliability criteria in this SCUC formulation include the procurement of spinning reserves supplied by generating units as well as demand side reserves to withstand a series of contingencies. The approach applies to discrete processes and ignores stochastics due to renewable and load prediction errors. In their study, Liu et al [20] provide a comprehensive demand response model where the demand side stability was optimized in both reserve and energy markets. The demand response model was incorporated into the model for estimating operating reserves presented in [11]. Two-stage SCUC, in which the reserve and energy markets are cleared in the same time, is used to assess the optimum spinning reserve, like DR reserve. In order to minimize computational complexity, the methodology ignores multiple order contingencies. Olamaei et al [21] propose a DR model of deterministic security criteria that is embedded into a traditional SCUC. The model infers continuous elasticity of demand with linear relationship between price and demand for energy. The model is put to the test on an IEEE one-area RTS to see how it will reduce overall system costs. In their study, Darvish et al [22] integrate incentive-based and time-based DR systems into a stochastic optimization scheduling model that only considers producing unit failure as a stochastic input. The energy demand and prices are modeled in this analysis in the same way as they were in [21]. The incentive-based demand response program is combined with the Dynamic Economic Dispatch (DED) problem in a model proposed by Abdi et al [23]. For incentive-based DR systems, the analysis builds non-linear models of receptive load. The Random Drift Particle Swarm Optimization algorithm is used to solve the combination of DED and DR model. In this study, the spinning reserve needs were calculated using the traditional *N-1* security criterion. Babu et al [24] propose Truncated State Space (TSS) based reliability assessment method that incorporates DR. The truncation of the state space is defined as elimination of systems states with low probability of occurrence to reduce computational complexity of the problem. The methodology considers generation and transmission hierarchical levels of a power system and tested on IEEE one area RTS. Silva et al [25] suggest a model for estimating spinning reserves in power systems containtnig renewable-energy sources. Rather than quantifying spinning reserve conditions, the model focuses on factors of stability. The proposed approach's numerical

efficiency is accomplished by the use of the cross entropy (CE) concept. CE's main idea is to adjust outage replacement rates (ORR) for system components depending on their value to grid stability. A capacity factor and a multi-state short term Markov model are used to model the volatility associated with renewable energy. The former captures primary electricity source intermittency, while the latter captures random outages in renewable units. The paper by Shayesteh et al. [26] provides a cost-benefit analysis-based approach for calculating the reserves needs of integrated grid networks. To minimize the total number of buses in the power grid, the suggested model employs the radial-equivalent-independent approach. The optimization of reserve requirements is performed using either security constrained unit commitment (SCUC) or security constrained economic dispatch (SCED). Reddy et al [27] suggest an energy and spinning reserve market clearing (ESRMC) approach that includes both fossil and wind power generators. To model the uncertainty associated with wind power and load, the proposed method employs parametric assumptions. A genetic algorithm is used to solve the multi-objective optimization problem, with the goal of lowering the device risk level and overall cost. A strategy for co-optimized spinning reserve and energy markets, incorporating provision of DR from commercial and industrial customers, is proposed in reference [28]. The suggested approach employs a standard unit commitment with a deterministic reliability criterion of 10% of net demand.

In comparison to deterministic approaches, the methods discussed above provide a sufficient estimate of spinning reserve requirements. However, as emerging technology and business mechanisms arise, we are forced to develop new reserve calculation methods that can be extended to smart grids. AI algorithms based on Bayesian networks, for example, can be seen as a promising way.

Bayesian networks (BNs) are probabilistic models that can encode some form of information, whether definite or unknown. As a result, they have a mathematically sound method for establishing an information representation system. Furthermore, it is the only way to make choices that is mathematically compatible. BNs may be used in conjunction with supervised or unsupervised learning methods to make intelligent forecasts based on historical evidence. Another possible application of BNs is the exploration of causal relations.

While BNs have been used in a number of engineering fields [29], [30], [31] only a few studies have used them to measure power system efficiency or to probabilistically quantify power generation reserve potential. One of the first works to use BNs to assess the

stability of a power grid is Reference [32]. The method employs BNs to represent system states based on data from its modules, such as generating unit availability, power lines, and so on. The states of the grid system are expressed by the Loss of Load binary variable, which signals that the generation system is unable to satisfy demand. The technique assesses a power system's stability in terms of LOLP which can be extended to interconnected grids. A reliability evaluation approach based on the BN approximate inference algorithm is proposed in reference [33]. To construct the BN representation of a grid structure, the approach employs a fault tree graph and the bucket elimination algorithm. A BN-based power system reliability evaluation approach is proposed in reference [34], which uses training data to establish the BN of a power system. The Monte-Carlo data sampling technique is used to produce the training data in this process. The technique assesses a power system's reliability in terms of LOL and cannot be applied to integrated grids.

2.2. Conventional Passenger Elevators

Current cutting-edge elevator dispatching control approaches are primarily motivated by the following pair of goals. The first goal is to reduce energy demand while maintaining the same degree of comfort and usability. Second goal is to reduce the passenger wait time by optimizing the elevator dispatch method. Not so long ago, the achievement of these goals was almost impossible. Currently this is not a problem due to development of advanced artificial intelligence (AI) video-assisted elevator control systems, which has been made possible by recent increases in computing capacity.

In a paper by Shofield et al [35], the authors present one of the first works explaining the use of video cameras in traditional passenger elevators. However, this research is not focusing on solving the elevator optimization problem, it is only restricted to the issue of counting the passengers. Similarly, research studies presented in references [36], [37], [38], [39] focus on various methods of usage of video cameras in elevator control systems that are not primarily aimed at strengthening elevator dispatch strategies.

Other studies suggest using video cameras to improve elevator monitoring and dispatch to some extent. For example, Ding et al [40] propose the use of security cameras to minimize overcrowding during emergency evacuations, while Mohamudally et al [41] propose the use of in-car cameras to assess the elevator car's usable capacity and cut down the number of unwanted stops.

Despite the fact that the use of data received from the video/image processing systems placed inside and outside of the lift cabins can greatly enhance the efficiency of traditional elevator systems, it is still not obvious how effectively use these data in order to upgrade the vertical transportation systems. Kim et al [42] suggest an elevator control scheme that uses data collected by cameras placed outside of the elevators to change the elevator allocation program by implementing generic algorithm. In this work, the number of traveling people is estimated using data from cameras. The route forecasting method is fairly straightforward, and it is built on the premise that the travelers who come after the one who last pressed the elevator call button move in similar direction. Based on the published outcomes, the data received from the security cameras led to minimization of the overall wait period by more than eight percent while the estimated average crowding was fifty five percent. According to the findings of this work, using cameras in situations of reduced and intense traveler loads (e.g., less than forty five percent or more than seventy five percent of overall capacity) ended up in negligible waiting period reductions.

Zhang et al [43] preset a method to optimally allocate elevators that work in groups that incorporates data provided by the traveler identification and monitoring system. The Unscented Kalman Filter is used to map passenger motion, while Haar-like feature-based passenger recognition technique is used in the algorithm. The primary aim of this algorithm is to reduce the elevator system's power usage and passenger wait time. Chou et al [44] proposes another elevator monitoring algorithm that makes use of the information gathered by the hallway cameras. The study considers total count of travelers queuing for lifts, their travel route, and lift capacity in order to reduce passenger wait times. Before being sent to a conventional elevator control system for dispatch, the data is evaluated using a Region Based Convolutional Neural Network.

In order to adapt to current passenger traffic conditions, the above studies use video tracking devices to estimate the count of travelers as well as to forecast the travelers' routs. The biggest flaw of these methods is that they do not allow for the uncertainties that come from the crowd of travelers as well as from the systems involved in video and image acquisition/processing. Although the traveler states are interpreted as deterministic in these experiments, it is clear that the proposed systems' forecasts cannot be absolutely correct.

A paper by Provan [46] was one of the pioneers in describing the application of BNs in lift dispatching control strategies. In this work, the author proposed a Bayesian Network structure for stochastic discrete-event control applications. In comparison to standard probabilistic finite state machines, the study argues that the hybrid variable-based Bayesian

interpretation of stochastic discrete-event systems presented in this work is simpler and effective. Cheng et al [47] suggest a BN enhanced learning method for efficient lift dispatching control systems in their paper. The method presented in this work constructs a low-dimension abstract state space to reduce the state space of the dispatch optimization problem. To perform reasoning and derive quantities for every abstract state, the authors use a BN. The proposed algorithm's final step involved the use of a neural network to determine the optimum state-action value function based on the BN's performance. The paper assumes a 20^{th} order truncated Erlang distribution for elevator load time.

Chapter 3 – Methodology

3.1. Electric Grid Systems

The basic structure of the proposed spinning reserve assessment model is presented in Figure 4. In the initial step, its main purpose is to determine all possible system states by combining the information on the generation system and net demand. Next, the model performs the two-stage optimization in order to find the most optimal power generation and reserve schedules. Finally, it performs adjustment of previously calculated reserve schedules taking into consideration different parameters.

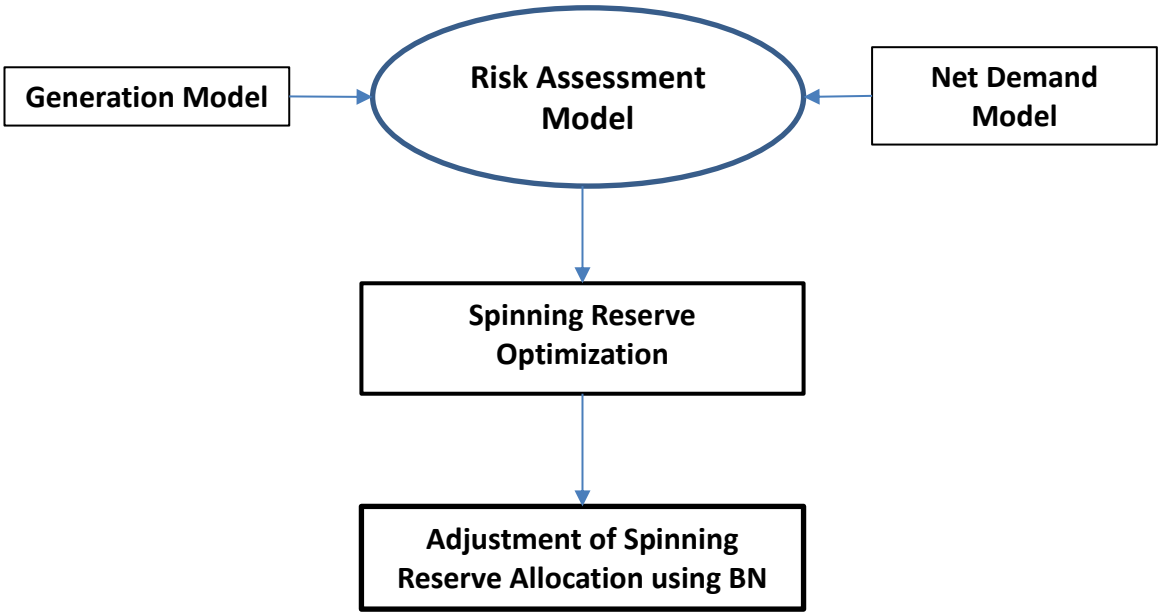


Figure 4. Spinning Reserve Assessment Model Structure

The model consists of the following submodels:

Generation System Model – determines the generation system states and creates the matrix, also referred to as the Capacity Outage Probability Table, specifying the generation system's available capacity and associated state probability as well as the cumulative probability.

Net Demand Model – quantifies the net electricity demand for the whole operating horizon. Net demand is simply a matrix, consisting of arrays (one array for each hour of an operating horizon), the arrays contain the predicted values of net demand and associated probabilities of occurrence. This approach is based on the parametric assumption that the errors associated with prediction of load and renewable power output follows a particular probability distribution. The continuous net demand data is then discretized using a seven-

interval approximation technique [12], which devides the continuous net demand dataset into seven equal segments as shown on Figure 5.

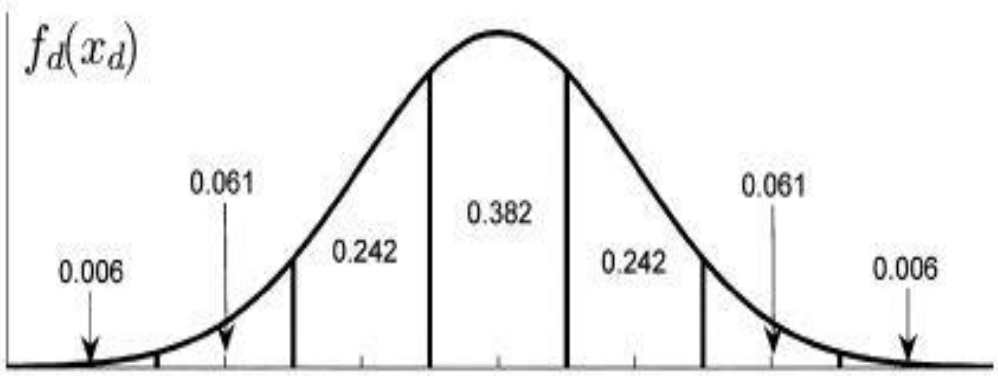


Figure 5. Discretization of the Normal distribution using seven-interval approximation

Risk Assessment Model – calculates the risk indices associated with each possible system state for the whole operating horizon.

Spinning Reserve Optimization – performs a two-stage optimization. The first stage optimization is conducted in order to optimize the power generation schedules. The spinning reserve schedule is improved in the second stage with the aim of lowering the cost of running the spinning reserve as well as the cost of load shedding.

Reserve Allocation Adjustment Algorithm – applies Bayesian inference that is based on the expert knowledge and actual statistical data to perform adjustment of previously calculated spinning reserve schedules.

The proposed model is carried out in three phases. The flowchart of the proposed model is presented in Figure 6.

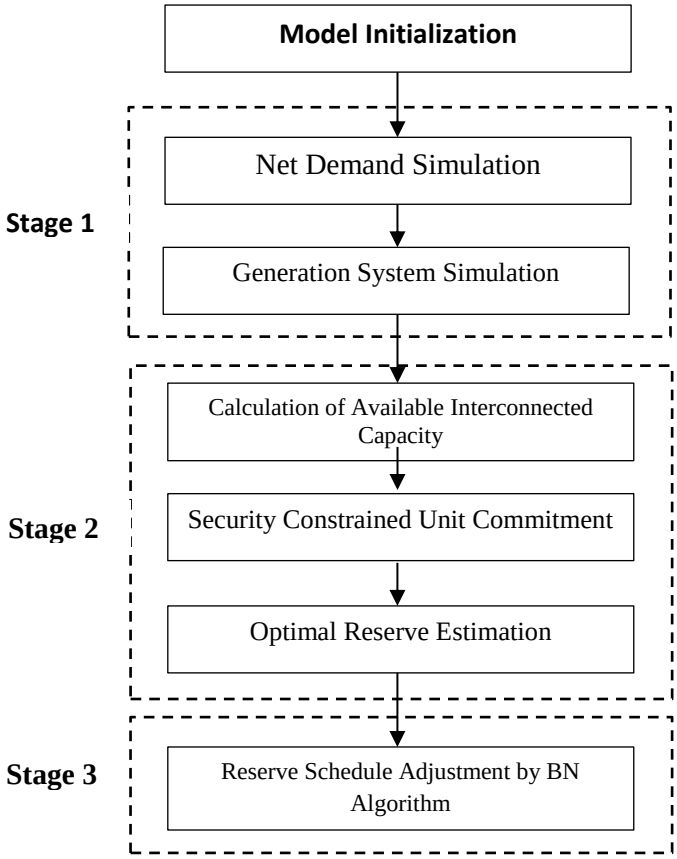


Figure 6. Flowchart of the proposed model

During the first stage, the stability of the power system under consideration is assessed without regard to its interconnection with other networks. The assisting power systems' Capacity Outage Probability Tables (COPT) are obtained using a recursive algorithm and inserted into the supported system's COPT in the second stage. The reliability assessment is done in terms of the Expected Energy Not Supplied (EENS), which is reliability metric or index for possible power supply shortfalls to customers. As a result, the amount of spinning reserves needed is determined depending on the system's level of reliability and the power available at any given time. In the final stage, the BN-based Reserve Allocation Adjustment Algorithm is used to modify reserve schedules based on intra-hour actual performance. Below is a brief summary of the measurements performed during the first, second, and third stages.

3.1.1. Market Clearing Process

This part describes the multi interval market clearing mechanism, in which the network operator purchases electric resources and ancillary services from market participants on a day-ahead basis.

The procurement process begins with **Wholesale Industrial Consumers (WIC)** and electricity **Distribution Companies (DisCo)** submitting day-ahead load predictions to the

grid operator, as well as power production forecasts from **Renewable Energy Producers (REP)**. Renewables are funded by benefits such as priority dispatch and feed-in tariffs in this pricing strategy, which means that REPs are excluded from fair bidding. . Figure 7 depicts the multi interval market clearing process.

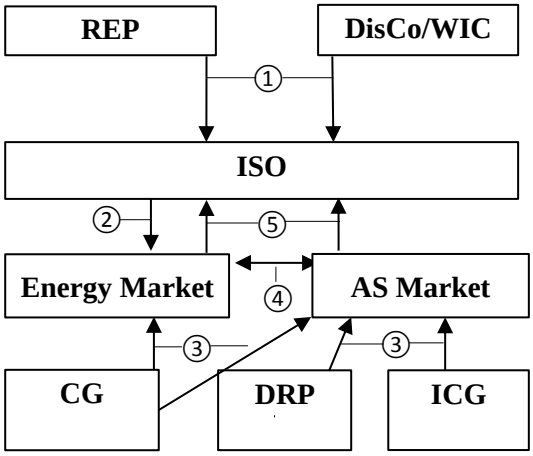


Figure 7. Electricity market clearing scheme

The network operator submits the aggregated net demand to the trading platform in the second stage to begin the trading phase. It's worth noting that, since the current model relies on estimating spinning reserve requirements, the definition of the market architecture is limited to electric energy and ancillary service procurement. Market participants apply their price-quantity proposals after the net demand information is released. Intra-zonal **Conventional Generators (CGs)** can participate in the energy and ancillary service markets, while **Inter-Zonal Conventional Generators (ICGs)** and DRPs can participate in the ancillary service market by supplying up-spinning reserve capacity.

Finally, the network operator selects the winning bids and performs a two-stage stochastic SCUC to calculate energy generation and spinning reserve schedules. Both goods (electric energy and ancillary services) are treated as competing assets in this market design, so market clearing for both is done at the same time.

3.1.2. Generation System Model

The capacity outage probability table (COPT) is used in the proposed method to model the states of the generation system. A two-state Markovian representation of a random process is used to calculate the generation system states. A generating unit in a two-state Markov method may be in one of two states: fully operational and ready to generate energy or have spinning backup, or inaccessible due to a technical issue.

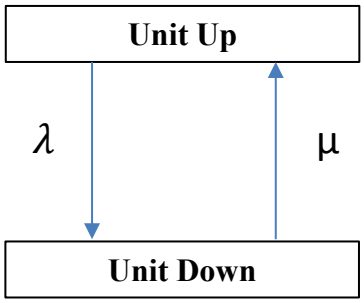


Figure 8. A power generation unit's two-state Markov model

The risk of limited power due to generating unit unavailability, also known as the outage replacement rate, is determined by the rate at which the unit transitions from an operating to a failure state, as well as the rate by which it is replaced. The probability of finding a two-state generating unit not operating due to a technical issue at any specific time ***t*** (assuming that the unit was successfully operating at the beginning of operating horizon) is given by the following expression:

$$p_{(down)} = \frac{\lambda}{\lambda+\mu} - \frac{\lambda}{\lambda+U} e^{-(\lambda+\mu)t} \quad (1)$$

where ***U*** is the ratio of the unit's down time to the total time (down time + up time). ***U*** is often referred to as the ***Forced Outage Rate (FOR)***.

The chance of locating unit *i* on outage, ignoring the unit's recovery phase and assuming an exponential distribution of the time to failure, is equivalent to $1 - e^{-\gamma_i T}$ where $\boldsymbol{\gamma_i}$ is the i^{th} unit's failure rate and t is the lead time [4].

The lowercase indices denote the variable number (e.g., $\boldsymbol{P_i}$ where $\boldsymbol{i}$ = 1, 2, 3 …), whereas the uppercase indices denote the variable term (e.g. P_RE, where RE stands for renewable energy). Furthermore, capital ***P*** denotes the generation of thermal and renewable energy units, while lowercase ***p*** denotes probability.

The recursive algorithm is used to calculate the generation system state, which is given by [4]:

$$p_{m,i} = (1 - ORR_i)\, p_{m-1} + (ORR_i)\, p_{m-1}(P_m - P_i) \quad (2)$$

where $\boldsymbol{p_{m-1}}$ and $\boldsymbol{p_{m,i}}$ are the combined probabilities of generation system state ***m*** before and after unit ***i*** is added, $\boldsymbol{ORR_i}$ is the outage replacement rate of unit ***i***, $\boldsymbol{P_i}$ and $\boldsymbol{P_m}$ are the installed capacity of unit ***i*** and the service outage at state ***m***.

3.1.3. Net Demand Model

Renewable and load prediction errors can have an almost identical effect on power grid stability as an unforeseen outage of thermal units [45], [46]. Because of the incentives intended to encourage the production of renewable energy, and because load and renewable projections are unrelated, renewable energy is treated as a negative load [8]. As a result, the forecasted value of net demand is given by:

$$D^t = L^t - P^t_{RE} \tag{3}$$

$\boldsymbol{P^t_{RE}}$ is the forecasted combined solar and wind power generation at time interval $\boldsymbol{t}$, where $\boldsymbol{L^t}$ is the forecasted electricity demand. The net demand prediction is the product of including the real net demand with a margin of error [10]:

$$D^t_F = D^t_A + e^t_D = L^t - P^t_W - P^t_{PV} = L^t_A - P^t_{W,A} - P^t_{PV,A} + e^t_W + e^t_{PV} \tag{4}$$

where $\boldsymbol{D^t_A}$, $\boldsymbol{P^t_{W,A}}$,and $\boldsymbol{P^t_{PV,A}}$ are the real values of net demand, wind, and solar power production at time interval $\boldsymbol{t}$, and $\boldsymbol{e^t_D}$, $\boldsymbol{e^t_W}$ and $\boldsymbol{e^t_{PV}}$ are the predicted errors of net demand, wind, and solar power output at time interval $\boldsymbol{t}$.

3.1.3.1. Load Model

Year after year, load estimation approaches have improved. Modern load prediction methods are extremely advanced, yielding extremely precise forecasts. Because of the repeatability of load and the sophistication of load prediction techniques, it is reasonable to assume that the standard deviation of the load forecast error is linearly proportional to the real load, with the proportionality of this relationship depending on the forecasting framework's precision.

3.1.3.2. Wind Power Generation Model

The most common approach to model renewable power output utilized in reliability assessment studies is to use a parametric probability density function (PDF). The PDF is selected based on the model's timeframe and application. The Gaussian [8]. [9], [10], Weibull [47], [48], [49], mixed distribution based on Laplace and normal distributions [50], Beta [51], hyperbolic [52] and the Levy **α**-stable distributions [53] are among the many PDFs used in power system research.

The Weibull marginal PDF is often used in wind speed simulation. The use of bivariate PDF is what makes this work special. The suggested bivariate PDF depicts the combined wind speed and trajectory distribution.

The power output of a wind turbine is calculated based on its power curve given by the following relation [54]:

$$P_W = \begin{cases} P_f(v) & v_{\min} \le v < v_R \\ P_R & v_R \le v < v_{\max} \end{cases} \quad (5)$$

where $\boldsymbol{P_R}$ is a wind turbine's maximum power output at wind speed $\boldsymbol{v_R}$, $\boldsymbol{v_{min}}$ and $\boldsymbol{v_{max}}$ are the cut-in and cut-off wind speeds, respectively. $\boldsymbol{P_f(v)}$ is a function of wind velocity $\boldsymbol{v}$ and power $\boldsymbol{P}$. Wind speeds that are below the mean and above the limit generate no electricity. The regimes below the minimum and above the maximum wind speeds result in zero power generation. In this study, the *Power-coefficient based model* is utilized, which is given by [54]:

$$P_f(v) = \frac{1}{2}\rho_{air} \cdot A_{WT} \cdot C_{eq} \cdot v^3 \quad (6)$$

where $\boldsymbol{\rho_{air}}$ is the air density, which is supposed to be constant (1.225 kg/m3), $\boldsymbol{A_{WT}}$ is the swept area of a wind turbine rotor, and $\boldsymbol{C_{eq}}$ is the dimensionless power coefficient quivalent, which is assumed to be 0.4 [54].

Traditionally, univariate PDFs have been used in parametric wind forecast models, but these models lack the ability to account for parameters that can reduce precision. The bivariate PDFs, which are made up of two marginal distributions, can be used to solve this problem. In terms of the goodness-of-fit criterion, study [55] shows that bivariate models yield more reliable results than univariate models. The Farlie-Gumbel-Morgenstern model with a mixture of two Weibull distribution functions is used in this analysis to propose a bivariate model that considers wind velocity and direction [55]. Equations (7) and (8) give the bivariate Farlie-Gumbel-Morgenstern PDF and CDF, respectively. [55]:

$$f_{V,\theta}(v,\theta) = f_V(v) f_\theta(\theta)\{1 + r_{V\theta}[2F_V(v) - 1][2F_\theta(\theta) - 1]\} \quad (7)$$

$$F_{V,\theta}(v,\theta) = F_V(v) f_\theta(\theta)\{1 + r_{V\theta}[1 - F_v(V)][1 - F_\theta(\theta)]\} \quad (8)$$

where $\boldsymbol{V}$ and $\boldsymbol{\theta}$ are the random variables that describe wind speed and direction, respectively, and $\boldsymbol{r_{v\theta}}$ is the statistical correlation parameter between $\boldsymbol{V}$ and $\boldsymbol{\theta}$ that is given by the equation below [55]:

$$r_{V\theta}^2 = \frac{r_{Vc}^2 + r_{Vs}^2 - 2r_{Vc}r_{Vs}r_{cs}}{1 - r_{cs}^2}, \; -\frac{1}{3} \le r_{V\theta} \le \frac{1}{3} \quad (9)$$

where

$$r_{Vc} = \tau_{V\theta}[(v_1, \cos\theta_1), ..., (v_n, \cos\theta_n)] \quad (10)$$

$$r_{Vs} = \tau_{V\theta}[(v_1, \sin\theta_1), ..., (v_n, \sin\theta_n)] \quad (11)$$

$$r_{cs} = \tau_{\theta\theta}[(\cos\theta_1, \sin\theta_1), ..., (\cos\theta_n, \sin\theta_n)] \quad (12)$$

where $\boldsymbol{v_i}$ and $\boldsymbol{\theta_i}$ are realizations of wind speed in ***m/s*** and angular direction in ***rad***, respectively, and $\boldsymbol{\tau}$ denotes the Pearson product-moment correlation.

3.1.3.3. Solar Power Generation Model

The power output of a PV module is determined by solar radiation, as well as the module's working temperature and parameters [56]. The Beta distribution [47], [48] which is expressed as shown in equation (13), is used to model the solar radiation rate:

$$f_b(I) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \times I^{(\alpha-1)} \times (1-I)^{(\beta-1)},\ 0 \le I \le 1, \alpha \ge 0, \beta \ge 0 \quad (13)$$

where $\boldsymbol{f_b(I)}$ is the PDF of the solar radiation intensity $\boldsymbol{I}$ and $\boldsymbol{\Gamma}$ represents the Gamma function [48]:

The shape parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ and are expressed as follows [47]:

$$\beta = (1-\mu)\left(\frac{(\mu+\mu^2)}{\sigma^2} - 1\right) \quad (14)$$

$$\alpha = \mu\left(\frac{\mu+\mu^2)}{\sigma^2} - 1\right) \quad (15)$$

where $\boldsymbol{\sigma}$ and μ are the standard deviation and mean of the solar radiation rate respectively. Provided the strength of solar radiation, the power output of a PV module can be calculated using the following expression [57]:

$$P_{PV} = I \cdot A_{PV} \cdot \eta_{PV} \cdot \eta_{mppt} \cdot \cos\theta_s \quad (16)$$

where $\boldsymbol{A_{PV}}$ is the total area of PV modules, $\boldsymbol{\eta_{PV}}$ is the PV module conversion performance, $\boldsymbol{\eta_{MPPT}}$ is the maximum power point tracking efficiency, and $\boldsymbol{\theta_S}$ is the solar irradiance angle.

3.1.4. Risk Assessment Model

To assess the efficiency of the power grid, the current approach uses the Expected Energy Not Supplied (EENS) model. Generally speaking, EENS is the product of the amount of unserved energy due to the generator outage and the probability of this event happening (Figure 9).

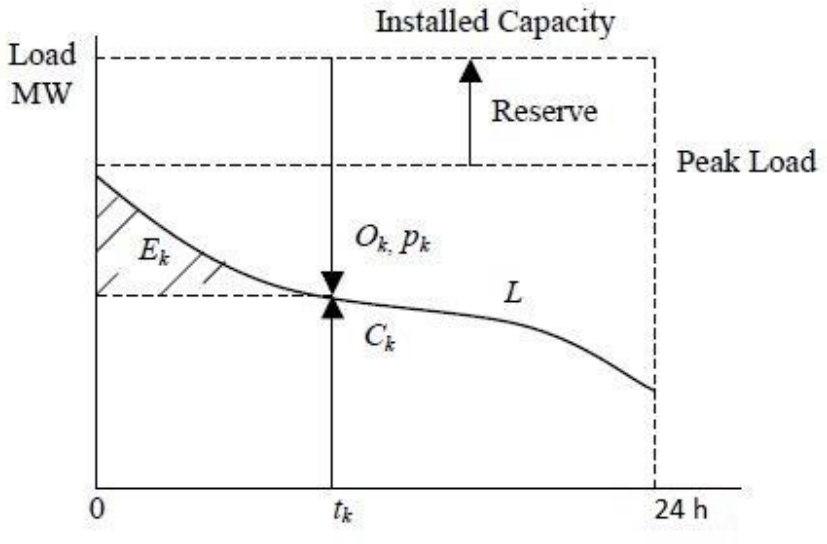


Figure 9. Graphical representation of EENS [4]

In other words, the sum of energy that would have been undelivered to consumers in the event of a contingency is denoted by EENS, which is given by the following expression:

$$EENS^t = \begin{Bmatrix} \sum_{m=1}^{M} \left(CE_m^t\right) q_m, \; for \; CE \geq 0; \\ 0 \qquad otherwise \end{Bmatrix} \tag{17}$$

where $\boldsymbol{CE_m^t}$ is the sum of curtailed energy during time interval $\boldsymbol{t}$ at a power grid state $\boldsymbol{m}$, as provided by the expression:

$$CE_m^t = \sum_{s=1}^{S} (D_s^t - \sum_{i=1}^{I} P_{i,m}^t) q_s \tag{18}$$

where $\boldsymbol{P_{i,m}^t}$ is the available power of a fossil unit $\boldsymbol{i}$ over a time interval $\boldsymbol{t}$ and state $\boldsymbol{m}$. The states of net demand and generation system availability scenarios are represented by $\boldsymbol{q_s}$ and $\boldsymbol{q_m}$, respectively. The total number of traditional units, net demand scenarios, and generation system states are represented by $\boldsymbol{I}$, $\boldsymbol{S}$ and $\boldsymbol{M}$, respectively.

3.1.5. Power Generating Capacity of Assisting Systems

Since network interconnection increases power system stability, it is normal for a grid system to have interconnections with adjacent grids. Due to higher system inertia and operational reserve power, interconnected networks are less vulnerable to short-term disruptions [58].

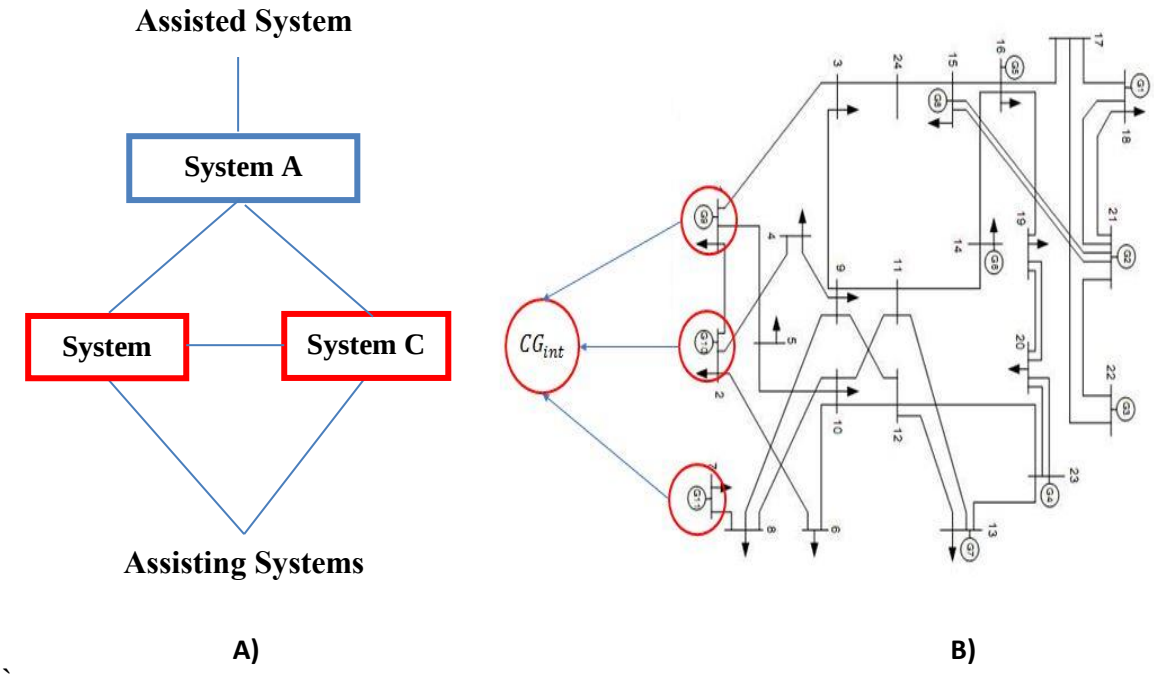


Figure 10. Assisting and Assisted Systems (A); multistate interconnected power generating unit (B)

This research considers available spinning reserve power of interconnected grid systems using a method called the equivalent assisting unit defined in [4] and [12]. The systems that receive and provide spinning reserve assistance are referred to as the assisted and assisting systems, respectively, in this method. The assisting system's spinning reserves are incorporated by using multi-state inter-zonal units in the COPT of the supported system using the recursive technique mentioned above. However, calculations cannot take into account all available inter-zonal spinning reserve power. The tie-line power limits restrict the overall value of inter-zonal spinning reserve, which can be determined using the equation below [12]:

$$IR^{\max} = \begin{cases} R^{t}_{total} & for\ IR^{t}_{total} < B^{\max} \\ B^{\max} & for\ R^{t}_{total} \geq B^{\max} \end{cases} \tag{19}$$

where $\boldsymbol{IR^{t}_{total}}$ and $\boldsymbol{B^{max}}$ calculated using the formulas below:

$$IR^{t}_{total} = \sum_{j=1}^{J} (IR^{inst}_{j} - IR\,e^{t}_{j} - IR\,r^{t}_{j,}) \tag{20}$$

$$B^{\max} = \sum_{l}^{L} B^{\max}_{l} \tag{21}$$

where $\boldsymbol{IRe_j^t}$ and $\boldsymbol{IRr_j^t}$ are the power dedicated for energy output and spinning reserve given by inter-zonal unit $\boldsymbol{j}$, respectively, at time interval $\boldsymbol{t}$. $\boldsymbol{IR_j^{inst}}$ is the interconnected unit $\boldsymbol{j's}$ installed power, $\boldsymbol{B_l^{max}}$ is the transmission line $\boldsymbol{l's}$ overall transmission capacity, and $\boldsymbol{J}$ and $\boldsymbol{L}$ are the cumulative number of interconnected systems and transmission lines, respectively.

3.1.6. Demand Response

Implementing demand response services will also help to increase grid system stability. In reality, ISOs all over the world have already adopted a variety of DR systems. In general, DR programs encourage energy users to participate in the ancillary service market by agreeing to reduce their electricity usage as required by the ISO in exchange for financial compensation. Reference [59] provides a basic overview of the most popular DR systems. DRPs, like other ancillary utility sector players, have up-spinning reserve on a competitive basis under the proposed market framework. The energy and power costs are included in the "price" portion of DRPs' price-quantity deals, whereas the "quantity" component specifies the maximum amount of load reduction, which must be greater than or equal to the minimum amount defined by ISO, as well as the maximum time for this reduction. The cost-function for DRP is derived using linear approximation in the same way that it was for traditional power.

3.1.7. Security Constrained Unit Commitment

3.1.7.1. Objective Function

A mixed-integer linear problem is used to model the two-stage stochastic unit commitment. The base-case scenario, which results in the most possible unit commitment, is the state of the system with all units available. The measurements performed during the first stage are aimed at determining the system's lowest operating expense. The sum of the costs of electricity generation, reserve provision, and load shedding is the net cost of system operation [60]:

$$C_{total} = C_E + C_R^{CG} + C_R^{ICG} + C_R^{DRP} + C_{LS} \tag{22}$$

where $\boldsymbol{C_E}$ is the power generation cost, $\boldsymbol{C_R^{CG}}$ is inter-zonal conventional units cost of spinning reserve, $\boldsymbol{C_R^{ICG}}$ is the intra-zonal conventional units' cost of spinning, $\boldsymbol{C_R^{DRP}}$ is the demand response providers' cost of spinning and $\boldsymbol{C_{LS}}$ is the cost of load curtailment. The following equation gives the cost of energy generation:

$$C_E = \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{n=1}^{N}(\lambda_{i,n}^{t} P_{i,n}^{t} + C_{i\min}^{t} u_i^t + CS_i^t) \quad (23)$$

where $\boldsymbol{\lambda_{i,n}^{t}}$ is the slope of the $\boldsymbol{n^{th}}$ segment of the $\boldsymbol{i^{th}}$ unit's piecewise linear cost function at time interval $\boldsymbol{t}$, $\boldsymbol{P_{i,n}^{t}}$ is the $\boldsymbol{i^{th}}$ unit's $\boldsymbol{n^{th}}$ segment power production (MW) during time interval $\boldsymbol{t}$, $\boldsymbol{C_{i,min}^{t}}$ is the $\boldsymbol{i^{th}}$ unit's minimum operating cost at time interval $\boldsymbol{t}$, $\boldsymbol{u_i^t}$ is the binary indicator of the $\boldsymbol{i^{th}}$ unit's status at time interval $\boldsymbol{t}$ (0 – power generation mode is off, 1 – power generation mode is on), $\boldsymbol{CS_i^t}$ is the $\boldsymbol{i^{th}}$ unit's start-up cost during time interval $\boldsymbol{t}$. The following equation gives the cost of spinning reserve offered by CGs:

$$C_R^{CG} = \sum_{t=1}^{T}\sum_{i=1}^{I}(Cup_i^t Rup_i^t + Cdw_i^t Rdw_i^t) \quad (24)$$

where $\boldsymbol{Cup_i^t}$ is the cost for provision of up-spinning reserve during time interval $\boldsymbol{t}$ by the $\boldsymbol{i^{th}}$ unit, $\boldsymbol{Rup_i^t}$ is the amount of up-spinning reserve (MW) during time interval $\boldsymbol{t}$ provided by the $\boldsymbol{i^{th}}$ unit, $\boldsymbol{Cdw_i^t}$ is the $\boldsymbol{i^{th}}$ unit's cost for provision of down-spinning reserve during time interval $\boldsymbol{t}$, $\boldsymbol{Rdw_i^t}$ is the amount of down-spinning reserve (MW) during time interval $\boldsymbol{t}$ provided by the $\boldsymbol{i^{th}}$ unit. The following equation gives the cost of spinning reserve offered by ICGs:

$$C_R^{ICG} = \sum_{t=1}^{T}\sum_{j=1}^{J} CIR_j^t \; IR_j^t \quad (25)$$

where $\boldsymbol{CIR_j^t}$ is the cost of power produced during time interval $\boldsymbol{t}$ by the $\boldsymbol{j^{th}}$ interconnected unit, $\boldsymbol{IR_j^t}$ is the amount of reserve during time interval $\boldsymbol{t}$ provided by the $\boldsymbol{j^{th}}$ interconnected unit. The following equation gives the cost of spinning reserve supplied by DRPs:

$$C_R^{DRP} = \sum_{t=1}^{T}\sum_{k=1}^{K} CDR_k^t \quad (26)$$

where $\boldsymbol{CDR_k^t}$ is the cost of spinning reserve during time interval $\boldsymbol{t}$ provided by the $\boldsymbol{k^{th}}$ DRP. The variable $\boldsymbol{C_{LS}}$ is given by:

$$C_{LS} = \sum_{t=1}^{T}\sum_{m=1}^{M}(\sum_{i=1}^{I}\sum_{n=1}^{N}\lambda_{i,n}^{t} r_{i,n,m}^{t} + \sum_{j=1}^{J}\sum_{n=1}^{N}\lambda_{j,n}^{t} ir_{j,n,m}^{t} + \sum_{k=1}^{K}\sum_{n=1}^{N}\lambda_{k,n}^{t} dr_{k,n,m}^{t} + VOLL \times CE_m^t) \quad (27)$$

where $\boldsymbol{r_{i,n,m}^{t}}$ is the deployed spinning reserve from the $\boldsymbol{n^{th}}$ block of energy offer by the $\boldsymbol{i^{th}}$ unit in the $\boldsymbol{m^{th}}$ scenario during time interval $\boldsymbol{t}$, $\boldsymbol{ir_{j,n,m}^{t}}$ is the deployed up-spinning reserve from the $\boldsymbol{n^{th}}$ block of energy offer provided by the $\boldsymbol{j^{th}}$ inter-zonal unit in the $\boldsymbol{m^{th}}$ scenario during time interval $\boldsymbol{t}$, $\boldsymbol{dr_{k,n,m}^{t}}$ is the deployed spinning reserve from the $\boldsymbol{n^{th}}$ block of energy offer by the $\boldsymbol{i^{th}}$ unit in the $\boldsymbol{m^{th}}$ scenario during time interval $\boldsymbol{t}$, $\boldsymbol{CE_m^t}$ is the amount of

energy curtailed due to a shortage of power generating capacity in the m^{th} scenario during time interval t, $VOLL$ is the value of lost load the constant showing the price of electricity supply interruption.

3.1.7.2. First-Stage Optimization Constraints

The objective function is minimized by applying the restrictions set out in the following equations.

Supply and demand equilibrium:

$$(D^{t}_{F} - CE^{t}) = \sum_{i=1}^{I} (P_i^{t} u^{t}_{i} + Rup_i^{t} u^{t}_{i} - Rdw^{t}_{i} u^{t}_{i}) + \sum_{j=1}^{J} IR^{t}_{j} u^{t}_{j} + \sum_{k=1}^{K} DR^{t}_{k} u^{t}_{k} \quad (28)$$

where, DR_k^t is the load reduction by the k^{th} demand response provider.

Power generating unit power balance:

$$P^{inst}_{i} - P^{t}_{i} \geq Rup^{t}_{i} \quad (29)$$

$$P^{t}_{i} - P^{\min}_{i} \geq Rdw^{t}_{i} \quad (30)$$

where, P_i^{inst} and P_i^{min} are the installed capacity and minimum power output of an i^{th} unit.

The constraints for inter-zonal units and DRPs are given by the equations below:

$$IR^{inst}_{j} - IR^{t}_{j} \geq Rup^{t}_{j} \quad (31)$$

$$DR^{\max}_{k} - DR^{\min,t}_{k} \geq Rup^{t}_{k} \quad (32)$$

where DR_k^{max} is the maximum curtailment level of a k^{th} demand response provider.

3.1.7.3. Second-Stage Optimization Constraints

The following relationships define the second-stage constraints.

The balance of spinning reserve capacity:

$$\sum_{k=1}^{K} (D^{t}_{k} - dr^{t}_{k,m} - CE^{t}_{k,m}) = q_m \sum_{i=1}^{I} (P^{t}_{i} + rup^{t}_{i,m} + rdw^{t}_{i,m}) + \sum_{j=1}^{J} ir^{t}_{j,m} \quad (33)$$

where $rup_{i,m}^t$ and $rdw_{i,m}^t$ are the deployed up and down spinning reserves by the i^{th} unit in the m^{th} scenario during time interval t respectively.

DRP's load limits:

$$0 \leq L^{t}_{k,m} \leq L^{t}_{k,\max} \quad (34)$$

Spinning reserve limits:

$$0 \le r_{i,up,m}^{t} \le q_{m}^{t} R_{i,up}^{t} \tag{35}$$

$$0 \le r_{i,dw,m}^{t} \le q_{m} R_{i,dw}^{t} \tag{36}$$

ICG and DRP reserve constraints:

$$0 \le ir_{j,\mathrm{m}}^{t} \le q_{m} IR_{j}^{t} \tag{37}$$

$$0 \le dr_{k,m}^{t} \le q_{m} DR_{k}^{t} \tag{38}$$

The cost of spinning reserve provision and the socioeconomic cost of load disruption are counterbalanced to optimize spinning reserve requirements.

3.1.8. Reserve Allocation Adjustment Algorithm (RAAA)

The algorithms focused on Bayesian Networks have piqued the attention of power system researchers and engineers [61] among a wide range of machine-learning algorithms. The Bayes' law, as the name implies, is founded on the following expression:

$$p(A \mid B) = \frac{p(B \mid A)\, p(A)}{p(B)} \tag{39}$$

where ***p(B)*** is the normalizing constant and ***p(A)*** is the *prior* probability of an occurrence of event ***A***. Before any knowledge about event ***B*** is considered, the prior likelihood ***p(A)*** may be thought of as an initial expectation about event ***A***. The posterior probability, also known as the conditional probability, represents the likelihood of ***A*** happening if ***B*** has already occurred. Similarly, ***p(B|A)***, also known as the *posterior* probability, is the conditional probability of ***B*** happening if ***A*** already happened.

The Bayesian inference method can be used to change prior probabilities based on new information using Bayes' law. In section 3.2.3, the interested reader can find more details on Bayesian networks and Bayesian inference.

The proposed BN rescheduling algorithm is based on previous work [63], with a few changes as detailed below. This algorithm's main goal is to fine-tune intraday reserve allocation based on actual net demand and unit commitment plan realization. Given the actual data from the previous hour *(t-1)*, the algorithm updates the intra-hour reserve allocation. The proposed algorithm also considers factors such as the hour type (peak/non-peak hours) and day type (weekday/weekend/holiday days) random variables, which can affect reserve allocation accuracy. Throughout the scheduling horizon, the modification

process is repeated hour by hour. Figure 11 depicts a condensed version of the proposed BN.

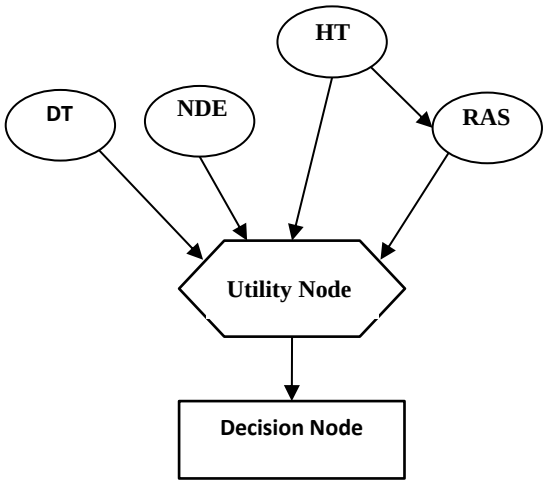


Figure 11. BN of the Proposed Algorithm.

3.1.8.1. Probabilistic Nodes.

The following random variables are represented by probabilistic nodes, which are represented by elliptical-shaped figures:

Hour Type (HT) is a binary random variable that indicates whether the hour is peak or non-peak. The amount of stress placed on power systems during peak load hours is extremely high; as a result, the probability of supply-demand mismatch due to capacity deficit during these hours is very high. The allocation of additional spinning reserve power by the network operator will help solve this issue. During peak hours, the proposed reserve rescheduling algorithm assigns a higher probability of allocating an additional 10% of spinning reserve energy, while off-peak hour reserve allocation is unaffected.

Another binary random variable is **Day Type (DT)**, which indicates whether the scheduling horizon is a weekday, weekend, or holiday. The algorithm gives holiday and weekend days a higher chance of receiving an additional 10% of spinning reserves, whereas weekday reserve allocation is unaffected.

The real net demand prediction error is represented by the **Net Demand Error (NDE)**, which is a continuous random variable. Based on the real net demand prediction error and a fixed threshold, the node assigns a higher probability of positive (increase) or negative (decrease) spinning reserve power change. It is fair to set the threshold equal to the standard deviation of net demand prediction error for the purposes of this analysis. According to [62], the IEEE RTS load forecast error must be modeled using a normal distribution with a standard deviation of 5%, but the threshold was set to 7% of forecasted demand to account for renewable forecast variability. The priors of states with NDE

deviations less than 7% are given a probability weight of 0.25, while the priors of states with NDE deviations greater than 7% are given a probability weight of 0.125:

$$p\,(NDE) = \begin{Bmatrix} 0.25,\ D_F^{t-1}\cdot 7\% > NDE^{t-1} > -D_F^{t-1}\cdot 7\% \\ 0.125,\ D_F^{t-1}\cdot 7\% < NDE^{t-1} < -D_F^{t-1}\cdot 7\% \end{Bmatrix} \tag{40}$$

The most important aspect of the probabilistic nodes is the reasoning that is used to update the evidence. In mathematic terms, the modification of spinning reserves is expressed as follows.

Positive ten percent change:

$$p\,(R_U^t \mid NDE^{t-1}) = \begin{Bmatrix} 0.8,\ (NDE^{t-1}) > 7\%\ D_F^{t-1} \\ 0.2,\ otherwise \end{Bmatrix} \tag{41}$$

where $\boldsymbol{R_U^t}$ is the ten percent positive change of spinning reserve at time interval ***t.***

Positive five percent change:

$$p\,(R_u^t \mid NDE^{t-1}) = \begin{Bmatrix} 0.8,\ 5\%\ D_F^{t-1} < NDE^{t-1} \le 10\%\ D_F^{t-1} \\ 0.2,\ otherwise \end{Bmatrix} \tag{42}$$

where $\boldsymbol{R_u^t}$ is the five percent positive change of spinning reserve at time interval ***t***.

Negative ten percent change:

$$p\,(R_D^t \mid NDE^{t-1}) = \begin{Bmatrix} 0.8,\ (NDE^{t-1}) > 10\%\ D_F^{t-1} \\ 0.2,\ otherwise \end{Bmatrix} \tag{43}$$

where $\boldsymbol{R_D^t}$ is the ten percent negative change of spinning reserve at time interval ***t***.

Negative five percent change:

$$p\,(R_d^t \mid NDE^{t-1}) = \begin{Bmatrix} 0.8,\ 5\%\ D_F^{t-1} < NDE^{t-1} \le 10\%\ D_F^{t-1} \\ 0.2,\ otherwise \end{Bmatrix} \tag{44}$$

where $\boldsymbol{R_d^t}$ is the five percent negative change of spinning reserve at time interval ***t***. For all other scenarios, the chances of adjusting the spinning reserve are zero. The example in Table 2 expands on the above-mentioned conditional dependencies.

Table 2 - Calculation of the amount of spinning reserve change based on real net demand

Var	Forecasted/Observed value, MW	Change/Difference, MW	Change/Difference, %
D_A^{t-1}	1 467	129	10.47
D_F^{t-1}	1 328		
R_F^t	268	27	10
R_{Adj}^t	295		

In this case, the difference between real and forecasted net demand is greater than 7% of forecasted net demand, so equation (41) must be used in further calculations. The algorithm will allocate the probability of increasing previously determined spinning reserve by 295 MW to 0.8 in this case, according to equation (41). The adjustment process is not complete at this point; the Decision Node will make the final decision on the adjustment stage

Another continuous random variable is **Reserve Allocation Smoothening (RAS)**, which represents the disparity between spinning reserves reserved for ***t-1***, ***t***, and ***t+1*** time intervals. Provided the gap between the spinning reserves of ***t-1*** and ***t+1,*** **RAS** prioritizes increasing (positive smoothening) or decreasing (negative smoothening) the spinning reserve potential for time span ***t***. The positive and negative smoothening are expressed mathematically as follows.

Five percent positive smoothening:

$$p\left(R_u^t \mid R^{t-1}, R^t, R^{t+1}\right) = \begin{Bmatrix} 0.8, & R^{t+1} > 1.1R^t \ \ \& \ \ R^{t-1} > 1.1R^t \\ 0.2, & otherwise \end{Bmatrix} \tag{45}$$

Five percent negative smoothening:

$$p\left(R_d^t \mid R^{t-1}, R^t, R^{t+1}\right) = \begin{Bmatrix} 0.8, & R^{t+1} < 0.9R^t \ \ \& \ \ R^{t-1} < 0.9R^t \\ 0.2, & otherwise \end{Bmatrix} \tag{46}$$

The RAS likelihood is manually set according to the user's preferences. It was set to 0.8 for the purposes of this analysis.

Table 3 - Calculation of spinning reserve change based on forecasted reserve requirements

Var	SR Allocation, MW	Difference, MW	Difference, %	Probability
R_F^{t-1}	232	36	13,43	-
R_F^t	268	-	-	-
R_F^{t+1}	256	8	4,48	-

R_{Adj}^{t}				0.2

The example in Table 3 shows how to calculate the conditional probability of smoothing based on forecasted reserve allocation. The discrepancy between initial reserve allocation measured for time intervals *t* and *t-1* is the first step in the smoothing process. Similar calculations must be performed for time intervals *t* and *t+1*. In this particular case, R_F^t greater than R_F^{t-1} and R_F^{t+1}, thus the equation (45) must be applied. According to the equation (45) the probability of increasing reserve requirement by 5% would be set to be equal to 0.2.

3.1.8.2. Utility Node

The utility nodes, in general, provide knowledge on expectations for results and their immediate ancestors. The proposed algorithm's utility node stores knowledge about alternative combinations of immediate predecessors' states based on data from the probabilistic nodes. The reserve adjustment judgment is based on the weight coefficients, which reflect the strength of each combination's effect on the final decision. A portion of the conditional dependency table is presented in Table 4.

Table 4 - Conditional dependency table for utility nodes

Day Type	**Holiday/Weekend**							
NDE	**5% Positive Change**				**10% Negative Change**			
RAS	5% Positive		5% Negative		5% Positive		5% Negative	
Hour Type	NP	P	NP	P	NP	P	NP	P
Value	8	9	3	4	5	4	7	5

3.1.8.3. Decision Node

The decision node is typically used to model the alternatives open to a decision maker. The decision node's goal in the proposed algorithm is to find optimal decisions given the parameters mentioned above. The following set of decisions is given by the proposed algorithm:

1. Maintain the original reserve requirement;
2. Increase the reserve requirement by five percent;
3. Increase the reserve requirement by ten percent;
4. Reduce the reserve requirement by five percent;
5. Reduce the reserve requirement by ten percent.

3.2. Conventional Passenger Elevators

It's worth noting that all of the lift systems considered in this work are standard lifts employing a collective control strategy with two call buttons, one for each movement direction. More detailed information on the elevator system

3.2.1. Elevator Group Control using the Nearest Car Control Policy

The failure of modern classical EGC algorithms to account for uncertainties involved with passenger traffic is their key flaw. Let's focus our attention at the traditional Nearest Car (NC) control policy as an example. The NC method is appropriate for two-three lifts in a seven or more storey building [63]. When the call is registered, NC continues to look for the most effective way to allocate lifts until the call is completed. In order to find the best solution for serving the call NC calculates the Figure of Suitability (FS) in accordance with the rules below:

a) If an elevator is heading towards a passenger and both its and the passenger's chosen directions are similar, NC provides the elevator a location bias, which can be interpreted as if the elevator was one level closer to the caller. The Figure of Suitability (*FS),* in this case, is determined as indicated in equation (48):

$$FS = (N+1) - (d-1) = (N+2) - d \tag{47}$$

where $(N+1)$ is the building's total count of floors, and d is the distance between the caller and the lift expressed in floors.

b) If lift is assigned to move in the direction of the caller whose chosen direction is adverse to the lift direction, *FS* is calculated as shown below:

$$FS = (N+1) - d \tag{48}$$

Relation (48) can also be used to measure the *Figure of Suitability* for a stationary lift cabin.

c) If the lift cabin is assigned to move away from the caller, NC sets *FS* to a quantity that is independent of the length between the lift and the caller.

NC assigns the call to the lift with highest *Figure of Suitability*; if more than one elevator has the same *FS*, the algorithm assigns the call to the nearest vehicle. Figure 12 shows the flowchart of a traditional Nearest Car policty.

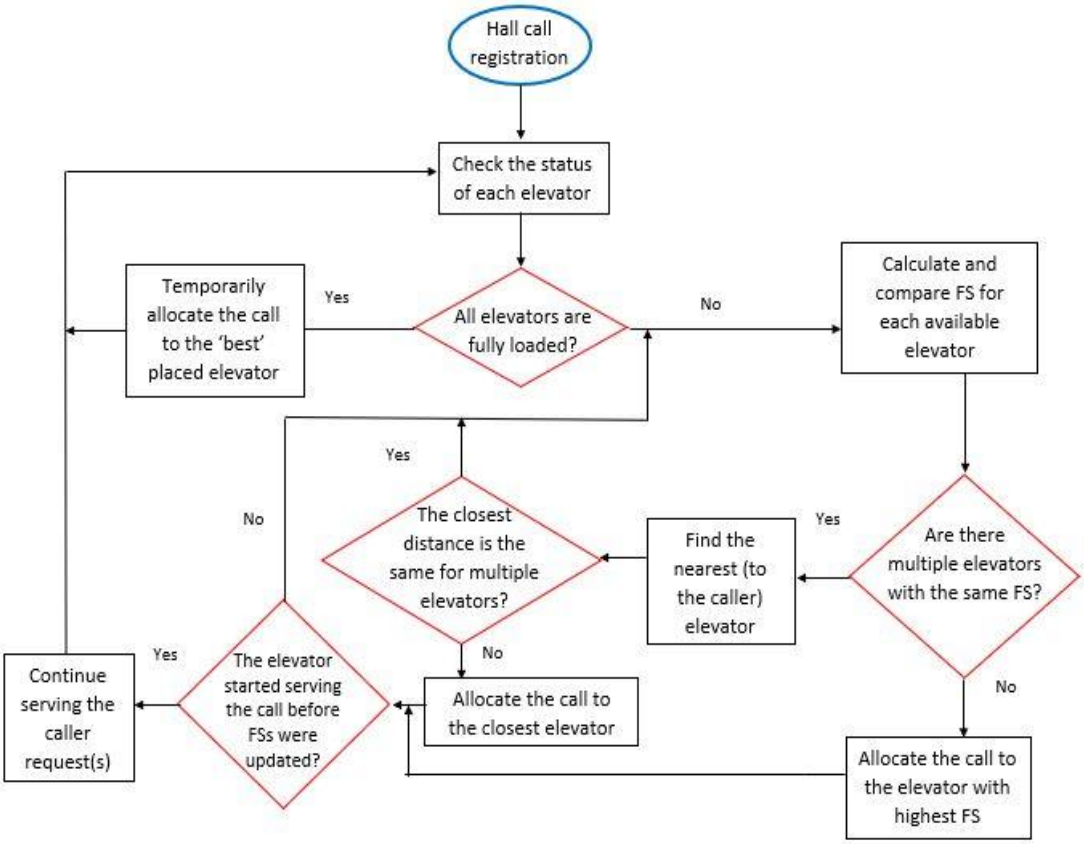


Figure 12. Flow diagram of standard NC policy

The key drawback of classical NC algorithms, according to reference [63], is their failure to tackle down-peak and up-peak traffic conditions adequately. As a result, this policy can be employed in buildings with few floors where the lift cabin delivery is not critical and the traffic intensity is not so high.

3.2.2. Proposed Elevator Group Control using modified Nearest Car Policy

Unlike standard EGCs that use a standard Nearest Car control policy, the elevator group control algorithm presented in this work utilizes security cameras to determine the number of passengers standing in hallways and riding lift cabins.

Based on the total count of people in line for a lift and the total area of an open space within the lift cabin, the algorithm assigns the call to one or several lift cabins. Every second, EGC refreshes the status of each lift cabin as well as each level. To put it another way, every second, EGC decides the positions and usable space in each lift cabin and total count of travelers on each level. The following list of actions complements the flow diagram depicted in **Figure 13**

1. After a call is allocated, EGC counts the number of travelers queuing in front of the elevator doors and the level at which the call was registered.

2. EGC monitor every lift cabin's status (position, free space, trip direction).
3. During this step, EGC checks the availability of lift cabins.

   A) If all lift cabins are full and there is no available space inside them, EGC monitors every lift cabin's status one more time every second.

   B) If there is free space in the lift cabins, EGC ranks them according to their FSs and sends commands starting from the lift cabin with highest FS value. In case there are several lifts with the same FS, the orders will be sent to the closest lift cabin.

4. The control system assigns calls to elevators in the order mentioned in **3B** before all passengers on the registered floor are seated in elevator cars.

5. Every second, the control system would cycle over the entire elevator allocation process, searching for a better option, before all travelers have boarded the elevator cars.

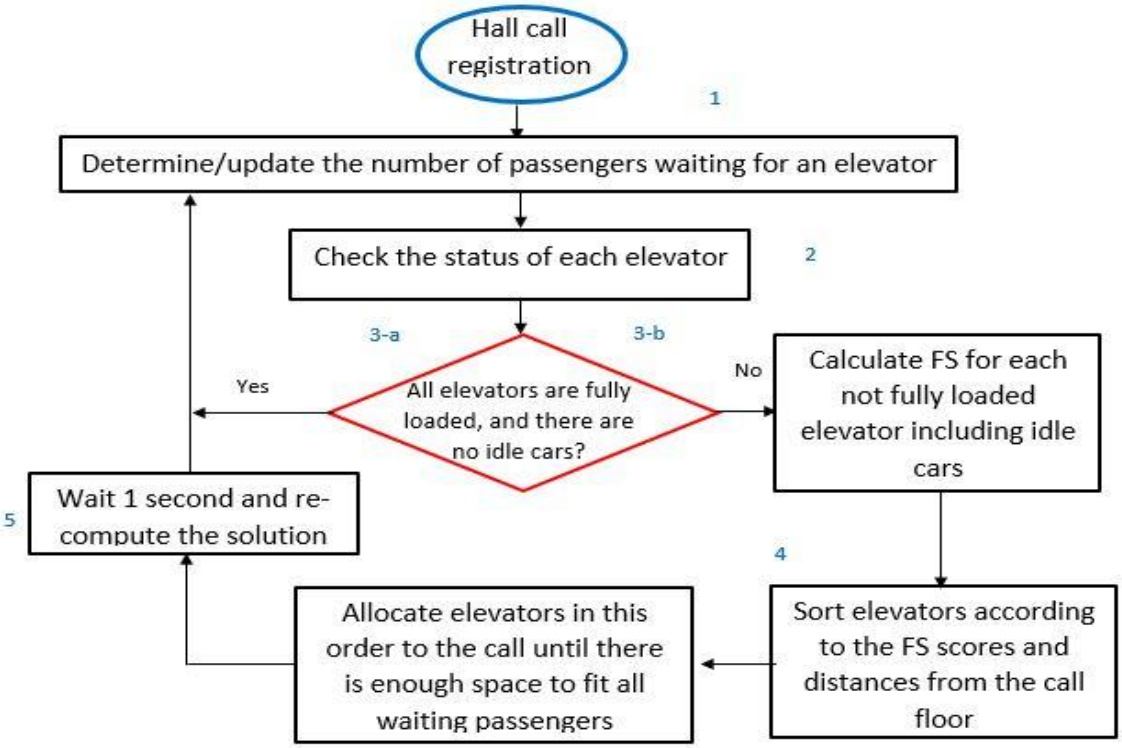


Figure 13. Flow diagram of EGC using modified Nearest Car elevator control policy

3.2.3. Proposed EGC using modified NC control algorithm and BN subroutine

The Bayes' law, which is at the heart of the suggested algorithm, calculates the likelihood of an occurrence based on prior knowledge of circumstances with a known relationship to the event. The cornerstone of Bayesian Inference is Bayes' rule, which is

built on top of conditional probability (see section 3.1.8. for more information about Bayes' theorem).

Bayesian network (BN), a guided acyclic graphical model in which random variables are expressed by nodes and causal relationships between nodes are represented by arcs, is an integral part of Bayesian inference.

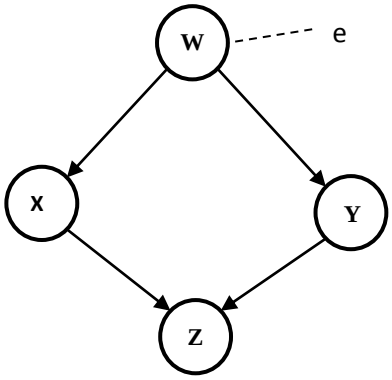


Figure 14. Simplified BN example

The hierarchical or family-like arrangement of BNs is implied by the unidirectional relationship between the nodes. Figure 14 depicts the kinship relationship between nodes, with *W* and *Y* representing the parent and child nodes, respectively. A child node may affect its parent, but not the other way around. Ancestor nodes are nodes that are higher in the hierarchy than a node of interest, while descendant nodes are nodes that are lower in the hierarchy. Finally, a root node is a node that has no parents, and a sink node is a node that has no children.

The potential to depict the joint probability functions in a compact and consistent manner is the strength of graphical representation of probabilistic models [64]. References [64], [65] and [66] have a more detailed overview of BN architectures and their elements.

The Conditional Probability Distribution (CPD) at each node is defined by the model parameters, which are an essential part of Bayesian inference. The conditional relationship between the nodes can be expressed in terms of Conditional Probability Tables (CPT) in the case of discrete random variables. The design of CPT is carried out in the following manner [30]:

- The conditional probability of a random variable with respect to the values of the parent nodes must be represented by each row.
- Each row's total must be equal to one.
- There must be one row for each of the root nodes.

BN-based models' computational complexity is determined by their composition, number of nodes, and number of states per variable. According to several studies [67], [68] using BNs for probabilistic inference is *an NP-hard* problem. For instance, consider the nodes *X* and *Y* depicted in Figure 14. Assuming that both X and Y are dichotomous random variables, the resulting CPT would have 2^2 potential states.

In order to go forward, it would be helpful to add certain general principles related to BNs. A variable is conditionally independent of other variables given its neighbors, according to the Local Markov property [69]. The following is how the Local Markov property can be applied to BNs:

$$X_v \perp X_{ND(v)} \mid X_{PA(v)} \tag{49}$$

where X_v is a random variable represented by a BN node, $X_{ND(v)}$ is a non-descendant node and $X_{PA(v)}$ is a parent node.

Consider a simple BN presented in Figure 14, where *X* is conditionally independent of non-descendant *(W|Y)*, this yields:

$$p(X \mid W,Y) = p(X \mid W) \tag{50}$$

The chain rule given by the following equation is used to decompose a joint distribution of variables in BN:

$$\begin{aligned} p(X_1,...,X_n) = p(X_n \mid X_1,...,X_{n-1}) \times \\ p(X_{n-1} \mid X_1,...,X_{n-2}) \times \\ p(X_2 \mid X_1) P(X_1) \end{aligned} \tag{51}$$

Equation (51) can then be used to derive a general form of the chain rule for BN.

$$p(X_1,...,X_n) = \prod_{i=1}^{N} p(X_i \mid PA(X_i)) \tag{52}$$

Although the algorithm outlined in section 3.2.2 uses data from surveillance cameras, this data is called deterministic. To put it another way, the algorithm believes that the count of individuals given by the video/image processing system is absolutely accurate. In practice, however, video/image processing software cannot necessarily calculate correct count of travelers queuing in a corridor or inside the lift cabins. There's still the risk of a flaw due to human actions or things that the machine identifies as people (bags, boxes, domestic animals etc).

Using BNs to reflect the number of passengers inside elevator cars and people waiting for an elevator in the hallways is one way to deal with this confusion. The BN subroutine is defined in general terms below.

First of all, assume there are *N* effective number of people who are in line for a lift on a certain floor, where *N* is greater than *3*, then N_1 available lifts that should be sent to these floors where:

$$N_1 = \frac{N}{E_C} \tag{53}$$

where E_c is the capacity of a lift cabin.

The variable *N2* is assigned to this floor if there are no idle lifts. These lifts are chosen from the lifts that travel to the level where the call was registered.

The BN subroutine is represented by the pseudocode below, where N2 is the total count of loops in the script.

*If: (passenger capacity of a lift – effective number of passengers inside the closest lift **i** moving in the same direction) > N*

Then: send the lift and proceed to the final line

*If: (passenger capacity of a lift - effective number of passengers inside the closest lift **j** moving in the same direction) < N*

Then: send the lift and proceed to the next line

*If: (capacity of a lift - effective number of passengers inside the second closest lift j+1 moving in the same direction) < N - (passenger capacity of a lift - effective number of passengers inside the previous closest lift **j** moving in the same direction)*

Then: proceed to the next line.

…

Count loops=N2

End.

The effective number of passengers who are waiting for a lift on the N^{th} level is given by the Bayesian Network depicted in Figure 15. The **Effective Number (EN)** of people riding the lift cabin is measured by the same Bayesian Network which has different count of states.

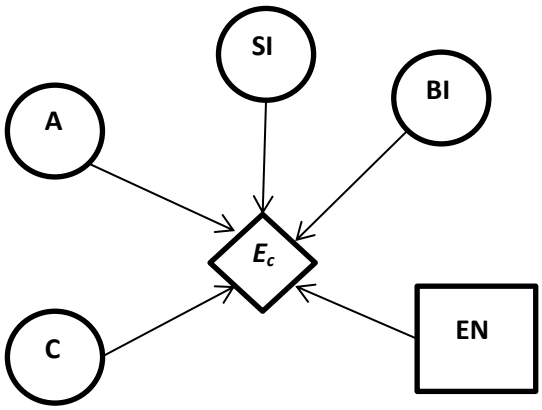


Figure 15. Bayesian Network for effective number of passengers

The BNs are made up of the nodes listed below, with the maximum capacity of the elevator car being 10 passengers and the assumption that the lift hall area can accommodate 20 adults, children, and items (passenger belongings).

- **Adults (A)**
  - 21 states from 0 to 20 for cameras on every building level
  - 11 states ranging 0 to 10 for cameras inside a lift cabin
- **Children (C)**
  - for each floor's cameras - 21 states ranging from 0 to 20
  - 11 states ranging from 0 to 10 for cameras inside a lift cabin
- **Small Items (SI)**
  - for each floor's cameras - 21 states ranging from 0 to 20
  - 11 states ranging from 0 to 10 for cameras inside a lift cabin
- **Big Items (BI)**
  - 21 states, ranging from 0 to 20, for each floor's cameras
  - 11 states ranging from 0 to 10 for cameras inside elevator cars

Children and small items should be weighed at 0.5 units, while adults and large items should be weighed at 1 unit.

For each node, the theoretical camera video/image processing software calculates a probability based on the possible states. For example, for one specific floor, the video/image processing system may produce the results as presented below:

9 adult passengers with certainty 70 percent, 8 adult passengers with probability of 20 percent, 10 adult passengers with probability of 10 percent. Furthermore, one large piece has probability of 65 percent and a case without any large piece has probability of 35 percent. The scenario without any child passengers has probability of 100 percent, two small pieces have a probability of 80 percent and a scenario with a single small piece has a probability of 20 percent, and so on.

A choice for *EN* value is suggested by the decision node. This is a real number that ranges from 0 to the sum of the maximum weights in each of the categories presented above. As a result, in this scenario, a maximum of twenty adult passengers are permitted, as well as a maximum of twenty kids carrying a maximum of twenty large or small belongings (kids and small items with weight 0.5 and adult passengers and large items with weight of 1).

As a result, it's an integer with 0.5 steps. A decision with a value of 12 indicates that the biggest expected utility "observes" twelve adult passengers, or ten adult passengers and four kids, or four small items, or any other combination. It's worth noting that the term "effective number of people/passengers" refers to people who are traveling with items.

The nodes' prior probabilities are filled in automatically, with the same quantities for all states. The following formulae are used to calculate utility values:

$$U = anX1 + knX0.5 + bnX1 + snX0.5. \tag{54}$$

where the state numbers of an adult, kid, big and small pieces are *an*, *kn*, *bn* and *sn* respectively.

Equation (55) gives the expected utility $U(A_i, H_j)$ for the decision node:

$$EU(A_i) = \sum_{a=1}^{3} \sum_{j=1}^{N} (A_i, H_j^a)\, p(H_j^a) \tag{55}$$

where A_i is the mutually exclusive variable and $i = 1,\ldots, n$, represents requests for actions together with three variables H_a with possible states H_j where $j = 2,\ldots, m$ representing hypothesis influencing the decision. Another key characteristic of this approach is that the request for actions are unrelated to *P(H)*.

Every second, the Bayesian Network is refreshed with information provided by the image/video processing system. As an instance of new information provided by the

image/video processing system, assume there are currently ten adult passengers with a certainty of 58 percent, three child passengers with a certainty of 33 percent, and one piece with a certainty of 84 percent, among other things. Figure 16 shows the Flow diagram of EGC using modified Nearest Car control policy and Bayesian Network-based subroutine.

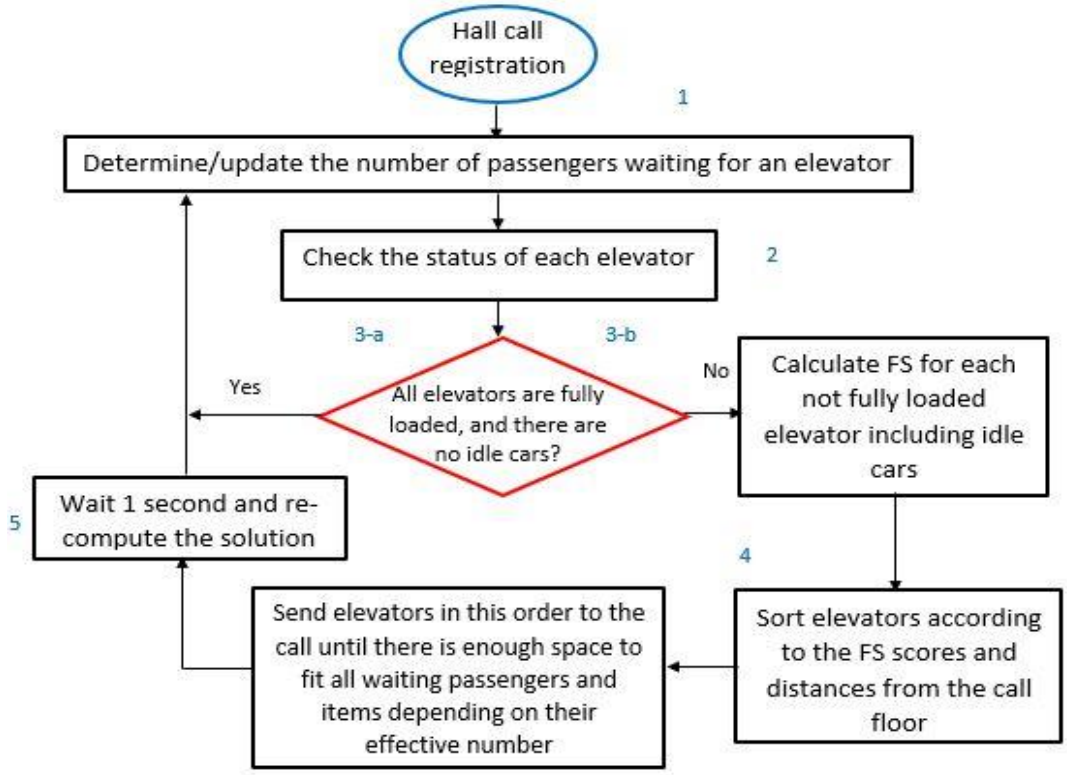


Figure 16. Flow diagram of EGC using modified Nearest Car control policy and Bayesian Network-based subroutine

If it is decided by experts or historical evidence that children have a higher priority than items (for example, because of fair/polite attitude adult passengers traveling with kids reach the lift cabins first relative to the adult passengers traveling with belongings), so the Bayesian Network developer will extend the usefulness of such kind of situations. Two children and one adult weigh more than two small items and one adult or one large object and one adult. As a result, in the case of one adult and one child, the number of successful passengers would be higher than in the case of one adult and one small thing. This technique was not used in this research, but it is discussed here to illustrate the value of utility.

The traditional Nearest Car control policy, the modified Nearest Car control policy assuming image/video processing software providing deterministic count of passengers (does not differentiate between child and adult passenger or whether there are any

belongings with them), and the modified NC algorithm with BN that produces an effective number of waiting "passengers/units" that is determined from a probabilistic approach will all be contrasted in the next chapter.

To start, the camera assigns probabilities to four different types of travelers (adult passengers, child passengers, small and large belongings). Secondly, the effective number (preferred state for decision) is the one that has the highest predicted utility, which takes into account each state's probability value as well as the utility priority weight set by the BN's creator.

When comparing the performance of elevator group control policies based on the Nearest Car, modified Nearest Car, and modified Nearest Car with BN, the latter control policy could outperform the other two. To validate this hypothesis, each EGC algorithm will need to be quantitatively validated and their output validated on a common set of scenarios.

3.2.3.1. Variable Elimination Technique

In reality, using BNs can be challenging since we frequently have to work with a huge number of random variables, each with several states. Using the whole joint distribution and summing up all latent variables is a simple way to do inference in BNs [70]. This task, however, can be very time consuming for large BNs, since the complete joint probability table for n binary variables would have 2^n entries [31]. In order to reduce the computational burden when performing inference, a basic but efficient technique known as **Variable Elimination (VA)** can be used.

The following is a simplified case of estimating a subset of queried variables X given evidence E and latent variables Y. The ratio of the combined probability distributions of X and E to the marginal probability distribution of E is the conditional probability of X given proof E:

$$p(X \mid E = e) = \frac{p(X, E = e)}{p(E = e)} \tag{56}$$

The numerator of equation (56) must be calculated by marginalizing all latent variables $Y_1, \ldots, Y_n$:

$$p(X = x_i, E = e) = \sum_{Y_1} \ldots \sum_{Y_n} p(Y_1, \ldots, Y_n, X = x_i, E = e) \tag{57}$$

at this point factors serving as the multi-dimensional tables that are used to avoid duplicate calculations are introduced. The joint probability of all variables can be expressed in terms of factors i.e., $f(X, E_1, \ldots, E_k, Y_1, \ldots, Y_n)$. The joint probability of X and E can be calculated by assigning $E_1=e_1, \ldots, E_k=e_k$ and marginalizing out the latent variables $Y_1, \ldots, Y_n$ one by one as follows:

$$p(X, E_1 = e_1, \ldots, E_k = e_k) = \sum_{Y_1} \ldots \sum_{Y_n} f(X, E_1, \ldots, E_k, Y_1, \ldots, Y_n)_{E_1=e_1, \ldots, E_k=e_k} \tag{58}$$

Next, the chain rule for BNs given by equation (52) can be used to express the joint factors as a product of factors, as seen below:

$$\begin{aligned} p(X_i \mid PA(X_i)) &= f(X_i, PA(X_i)) \\ &= f_i\, p(X, E_1 = e_1, \ldots, E_k = e_k) \\ &= \sum_{Y_n} \ldots \sum_{Y_1} f(X, E_1, \ldots, E_k, Y_1, \ldots, Y_n)_{E_1=e_1, \ldots, E_k=e_k} \\ &= \sum_{Y_n} \ldots \sum_{Y_1} \prod_{i=1}^{N} (f_i)_{E_1=e_1, \ldots, E_k=e_k} \end{aligned} \tag{59}$$

As a result, inference in BNs ultimately comes down to adding the properties of the last term of the equation (58). The terms that do not include the latent variables must be factored out in order to calculate the last term of equation (58) effectively.

3.2.3.2. Expectation-Maximization Algorithm

The use of BNs in practice reveals that we often have to work with data that is incomplete. Data can be lost due to technological problems with the data collection process, or data could also be based on the values of the observed variables [65]. When the likelihood of missing data is independent of observed values, the data is referred to as missing at random completely (MARC), but when the absence of the data is based on observed values, it is referred to as missing at random (MAR). Incomplete data sets may cause parameter estimates to be dramatically skewed, resulting in a highly inaccurate probabilistic model. Implementing data generation algorithms can help to solve the issue of

missing data. The Expectation-Maximization Algorithm is used to produce arbitrarily missing data in this analysis.

Given a BN model structure with variables $X_1, \ldots, X_n$ we introduce θ_{ijk} - the parameter corresponding to the conditional probability of X_i in state k, at j^{th} configuration of its parent nodes i.e., $p(X_i=k|PA(X_i)=j)$. According to this notation, for a data set $D = \{d_1, \ldots, d_m\}$, the likelihood estimate θ'_{ijk} can be found as follows [65]:

1. Let $\theta^0 = \{\theta_{ijk}\}$, where $1 \le i \le n$, $1 \le k \le |SP(X_i)| - 1$, and $1 \le j \le |SP(PA(X_i))|$ are the arbitrary initial estimates of the parameters, and $SP(X_i)$ is the state space of X_i.
2. Set t := 0;
3. E-step: For each $1 \le i \le n$ calculate the expected counts:

$$\begin{aligned} &\mathbb{E}_{\theta^t}[N(X_i, PA(X_i)) \mid D] \\ &= \sum_{d \in D} P(X_i, PA(X_i) \mid d, \theta^t) \end{aligned} \tag{60}$$

where N denotes the total number of counts. Provided the observed portion of the data and the current values of the parameters, this step calculates the conditional expectation of the complete-data log likelihood.

4. M-step: Determine a new probability calculation for all θ_{ijk} using the predicted counts.

$$\theta'_{ijk} = \frac{\mathbb{E}_{\theta^t}[N(X_i = k, PA(X_i) = j) \mid D]}{\sum_{h=1}^{|SP(X_i)|} \mathbb{E}_{\theta^t}[N(X_i = h, PA(X_i) = j) \mid D]} \tag{61}$$

Set $\theta^{t+1} := \theta'$ and $t := t+1$.

Assuming that the data is complete, this step simply entails performing a maximum likelihood estimate of θ.

Steps 3 and 4 can be repeated before convergence or other stopping conditions are reached.

Chapter 4 – Results

4.1. Electric Grid Systems

4.1.1. Evaluation Methodology

The proposed model was validated using the IEEE Two-Area RTS (Figure 17), which is made up of two One-Area RTSs linked by three interconnection lines [71]. The standard Two-Area RTS was changed for the purpose of this analysis by replacing the hydroelectric unit at bus 122 with a 600 MW wind farm and a 200 MW solar farm. Thermal unit operational restrictions and cost function coefficients were taken from [72]. Each load node in this analysis is presumed to be a DRP with a 30 percent potential load reduction, a 10 USD/MW capacity charge, and a 20 USD/MWh marginal cost. The operating horizon for the grid system was set to 24 hours, and the VOLL was set to four thousand USD/MWh.

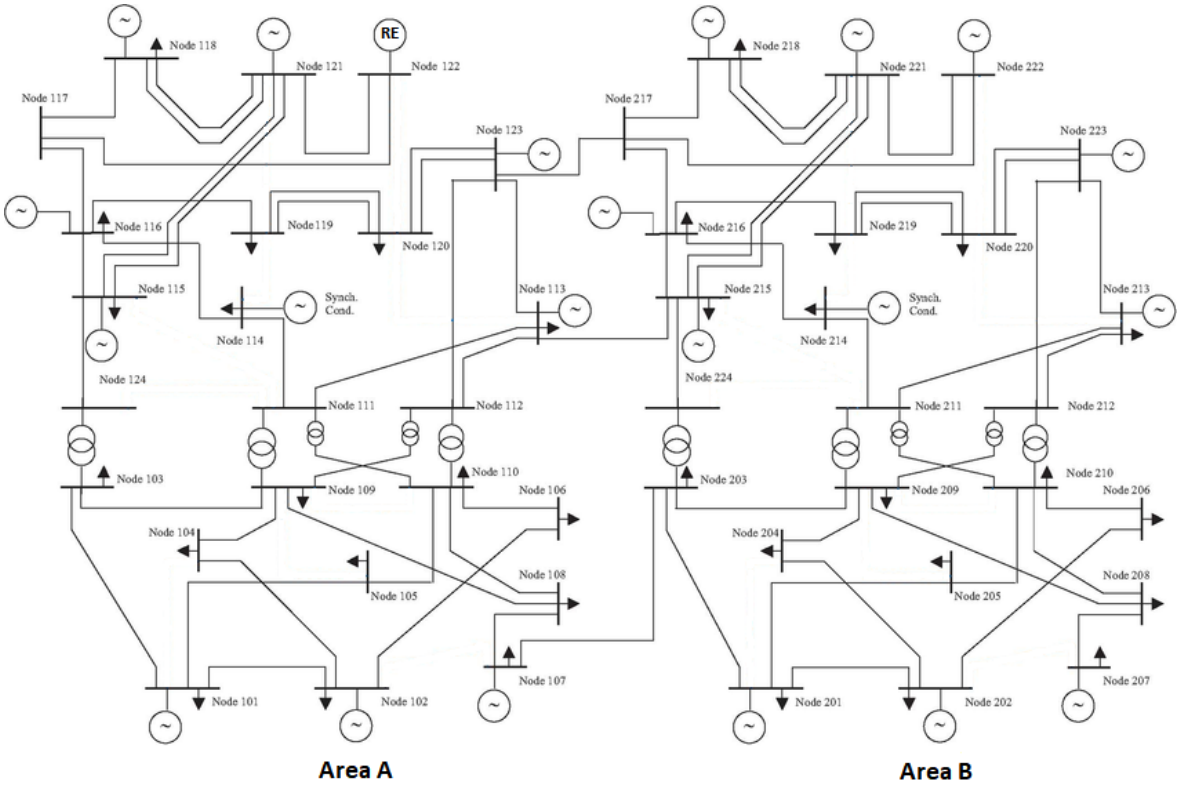


Figure 17 . IEEE Two-Area Reliability Test System

Area A is selected as the assisted network in this study, while Area B is the assisting grid system. It should be noted that both Area A and Area B are identical to each other in terms of generating capacity (type of generating units, installed capacity and buses) with total installed capacity in each area equal to 3 105 MW.

According to the proposed market arrangement, inter-zonal power will participate in the ancillary service market if the assisting system's capacity requirements are completely met.

The model was created in MATLAB R2018a [73], with IBM ILOG CPLEX Optimization Studio 12.7.1 [74] doing MILP optimization. BayesiaLab (Version: 9.1 PE-L 00000, Manufacturer: Bayesia S.A.S., Changé, France) [75] was used to apply the BN-based reserve adjustment algorithm. In order to reduce the problem's computational complexity the grid system's states that have probability lower than 10^5 were neglected. This threshold corresponds to the point when the inclusion of the grid system's states with lower probabilities does not increase the overall accuracy of the model, but significantly increases computing time.

The proposed model was tested on three separate network configurations to see how the inclusion of inter-zonal reserves and DRP improves stability and lowers running costs of the test system. The descriptions of the network configurations are presented below:

Network Configuration A: Area A is in island mode (i.e. operating without any assistance from neighboring Area B system), with no demand response;

Network Configuration B: Area A and Area B are connected and Area B is acting as an assisting system while Area A is acting as an assisted system. Demand response is disabled in this network configuration;

Network Configuration C: Area A is connected to Area B, and as in previous network configuration Area B provides assistance to Area A. Demand response is active in this network configuration.

The main metrics that were evaluated and compared in this analysis are reserve requirement, Expected Energy Not Supplied, and the cost of spinning reserve capacity.

4.1.2. Spinning Reserve Requirements

The spinning reserve schedules determined for various network configurations are shown in Figure 18. Throughout the operational horizon, the system configurations A and C resulted in the lowest and highest spinning reserve allocation, respectively. It is important to mention that the amounts of available generating capacity were held constant for all three scenarios (i.e. 3 105 MW in each Area).

It's worth noting that the scale and diversity of the spinning reserve capacity play a big role in reserve allocation optimization. In this test case, network configuration C, for example, has the largest and most diverse pool of spinning reserves as compared to configurations A and B. In comparison to configurations A and B, network configuration C

clearly has more inexpensive units available for spinning reserve allocation. As previously said, spinning reserve is allocated in such a way that the overall cost of spinning reserve provision and the socioeconomic cost of load curtailment are kept to a minimum. Because of the abundance of low-cost reserve power, it is possible to increase spinning reserve requirements without raising the overall cost of spinning reserve provision.

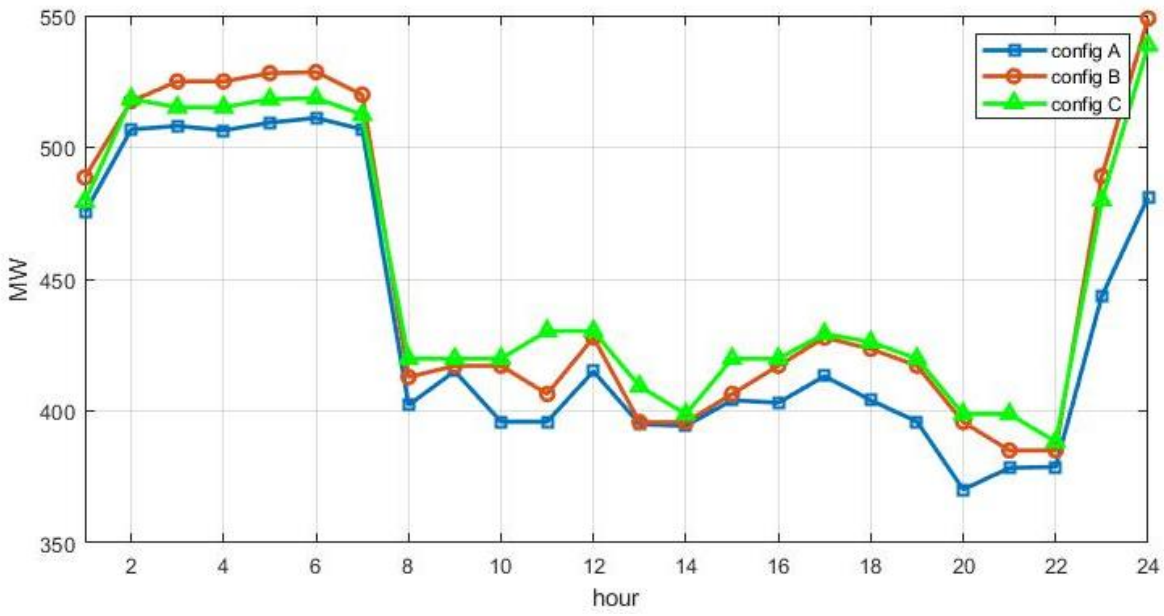


Figure 18. Spinning reserve allocation in various network configurations

Following that, we are interested in estimating how DR marginal costs impact spinning reserve distribution. This part of the study is based on network configuration C, which has DR marginal costs of 10 USD/MWh, 20 USD/MWh, and 30 USD/MWh, with the base case being 20 USD/MWh. The 10 USD/MWh price, according to the findings, is low enough to compete with traditional thermal units during off-peak hours. The lowest marginal cost of thermal units was about 9 dollars per megawatt hour.

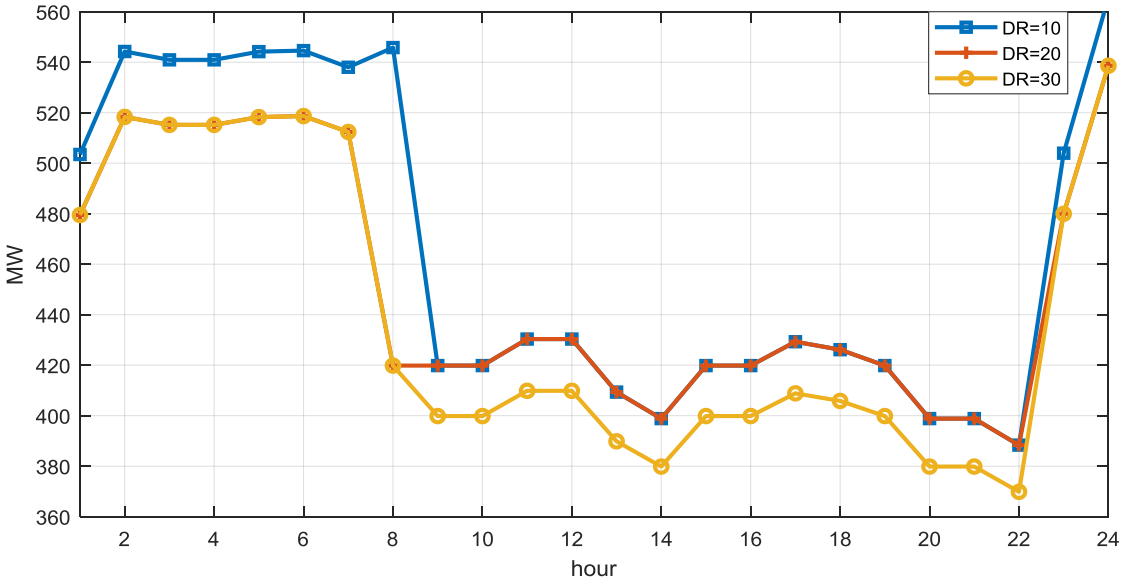


Figure 19. Spinning reserve allocation in various DR marginal cost scenarios

4.1.3. Reliability Indices

We can conclude from equations (17), (18) and (33) that the level of EENS is directly proportional to the sum of allocated spinning reserves. Table 5 shows the cumulative amount of Expected Energy Not Supplied, which was determined by integrating EENS over the operational horizon. The simulation results showed that the network configuration C had the lowest EENS. When comparing Figure 18 and Figure 20, it is clear that EENS is directly dependent on spinning reserve allocation. For network configurations B and C, a high risk of capacity outage during peak hours raises EENS, while EENS is marginal during off-peak hours.

We can infer that there is insufficient affordable inter-zonal reserve power based on the results obtained for network configuration A, since EENS is relatively high even during off-peak times.

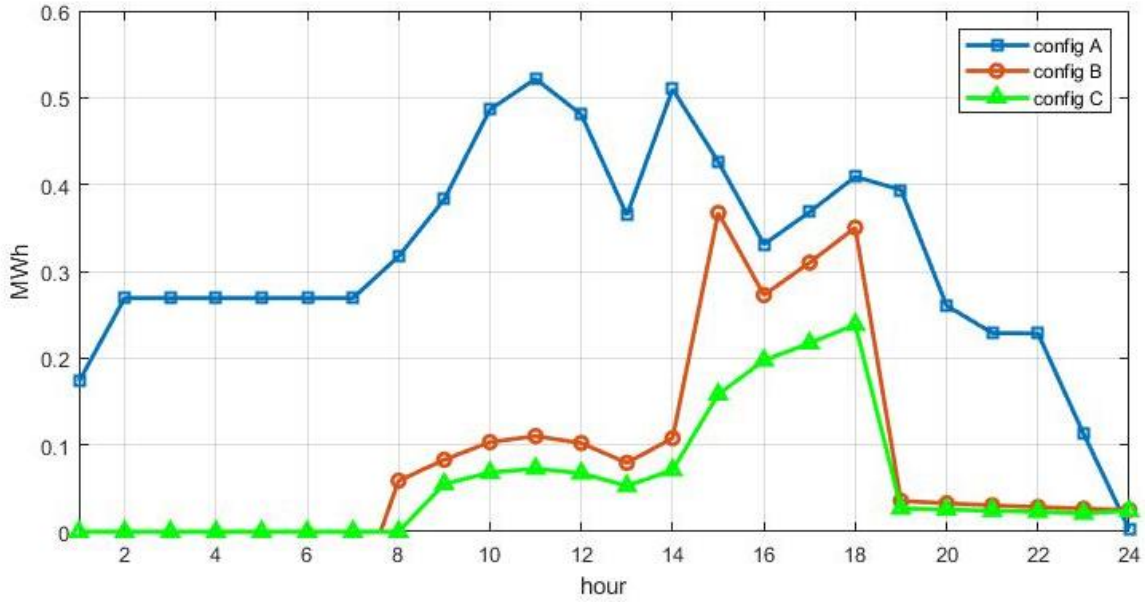


Figure 20. EENS under different network configurations.

Table 5 - Reserve, EENS and total system running costs determined for various network configurations.

	Network Configuration A	**Network Configuration B**	**Network Configuration C**
Reserve (MW)	10 410	10 802	10 825
EENS (MWh)	9.05	1.66	1.5
Total Costs (USD)	1 203 800	969 740	958 790

Figure 20 and Figure 21 show a clear relation between EENS and spinning reserve distribution – changes in the sum of available reserves have a significant impact on EENS. As seen in Figure 21, a low DR marginal cost decreases EENS dramatically.

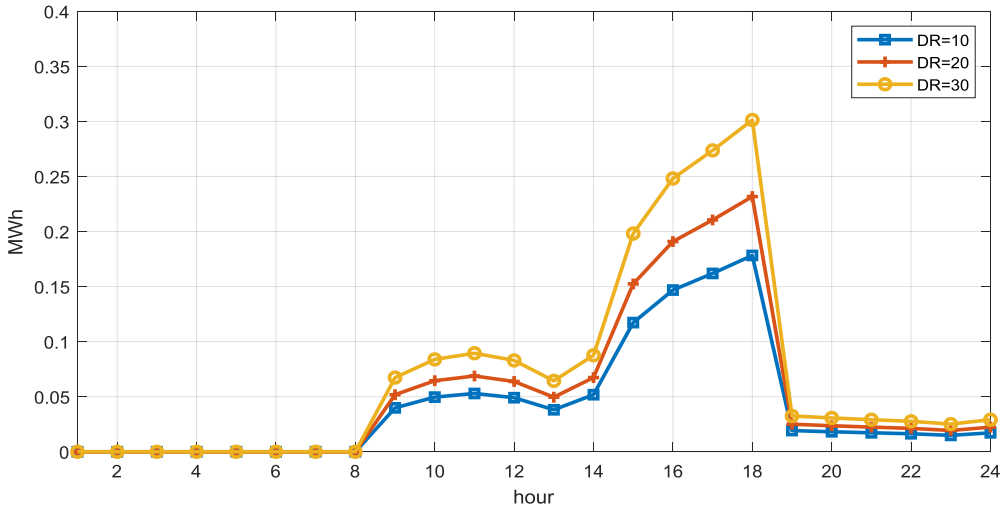


Figure 21. EENS calculated for various DR marginal costs.

4.1.4. Economic Evaluation

The overall cost of spinning reserve is made up of the running costs of allocated spinning reserves as well as the cost of load curtailment due to a power outage, as calculated by equation (62).

$$CR_{total} = \sum_{t=1}^{24} (CR^t + CE^t \cdot VOLL) \tag{62}$$

where CR^t is the cost of the spinning reserves deployment during time interval ***t***. According to the findings shown in Figure 11, network configurations A and C had the highest and lowest cost of reserves, respectively. By comparing Figure 18 and Figure 20, we can infer that the cost of reserve for network configurations B and C at off-peak hours is primarily driven by the operational cost of reserve capacity, as *EENS* is nearly negligible in these two situations.

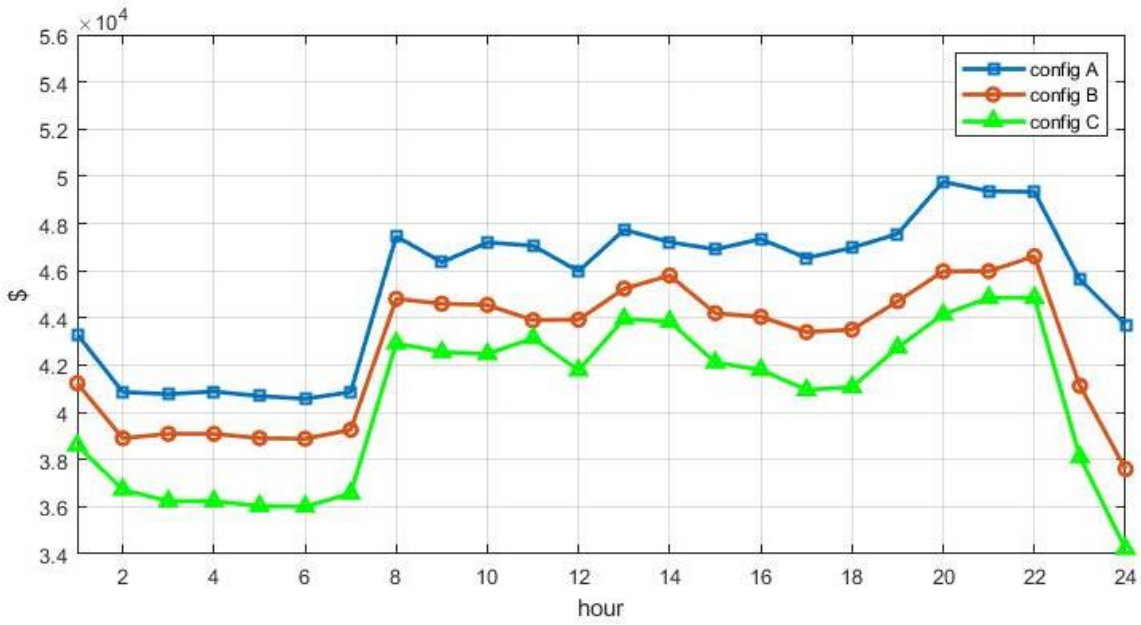


Figure 22**.** Total cost of reserves calculated for various network configurations

Figure 23 shows total reserve costs at DR marginal costs of 10, 20 and 30 USD/MWh. Table 5 shows the net costs of allocated spinning reserves in the base case, as well as the total costs of allocated spinning reserves over the operational horizon. The system with the lowest net cost of allocated spinning reserve has a DR marginal cost of 10 USD/MWh. During off-peak hours, the systems with marginal costs of 20 to 30 USD/MWh are at the same amount, although the latter is much lower during peak hours.

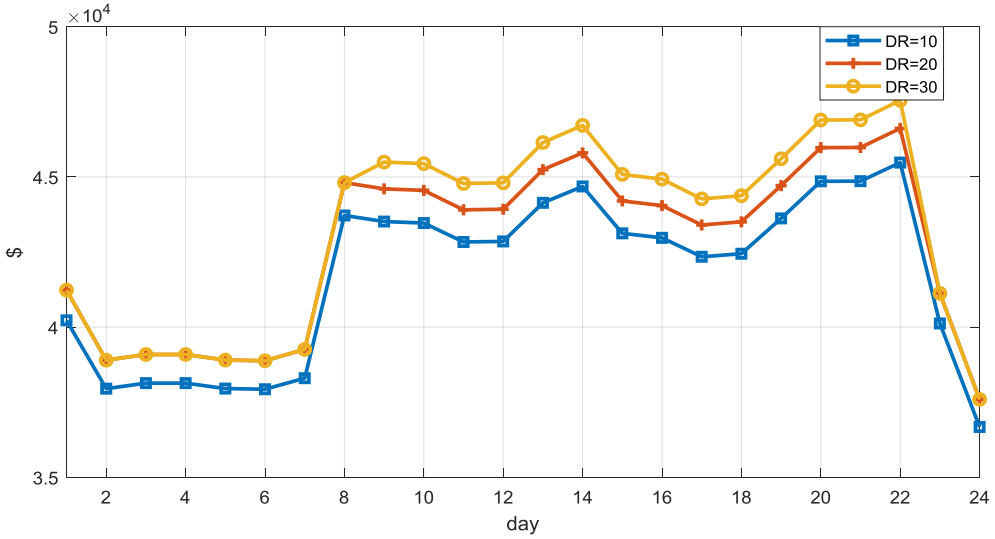


Figure 23. Total cost of reserves calculated for various DR marginal costs.

Evaluation of Reserve Allocation Adjustment Algorithm was conducted by comparing CR_{total} which was quantified for one calendar year (365 days), Figure 24 presents the results.

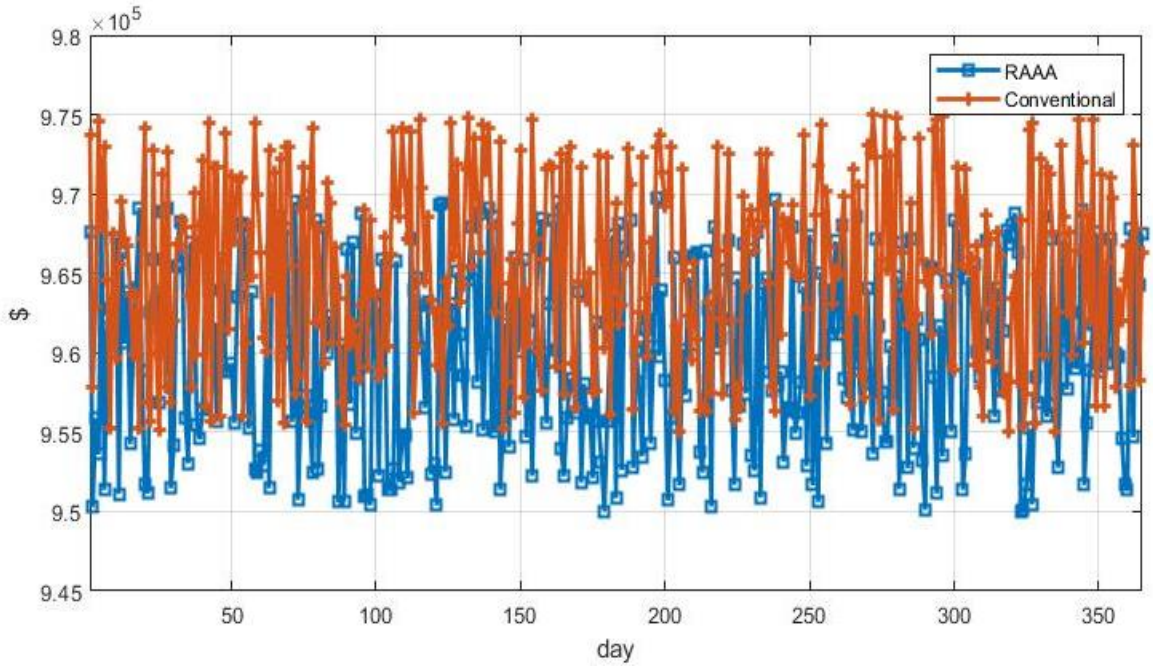


Figure 24. Total cost of allocated spinning reserves quantified using RAAA and traditional approaches

According to simulation data, implementing RAAA resulted in an average reduction of 1,12% in reserve costs. The RAAA changes resulted in 56 percent reserve schedules with a lower average cost of reserve as compared to the traditional methodology. It's also worth noting that almost 30% of the 365 simulations generated identical results.

This is due to the fact that a sizable number of cases result in a spinning reserve schedule that remains unchanged.

4.1.5. Comparison of Wind Forecasting Methods

First and foremost, it should be noted that the aim of this section is not to equate the precision of univariate and bivariate wind prediction methods, but rather to show how the grid system adequacy differs if both models are applied separately. Reference [55] is recommended for readers interested in evaluating these models in terms of precision.

Figures Figure 25 and Figure 26 show the reserve criteria and EENS of various network configurations calculated using univariate and bivariate wind forecasting methodologies, respectively.

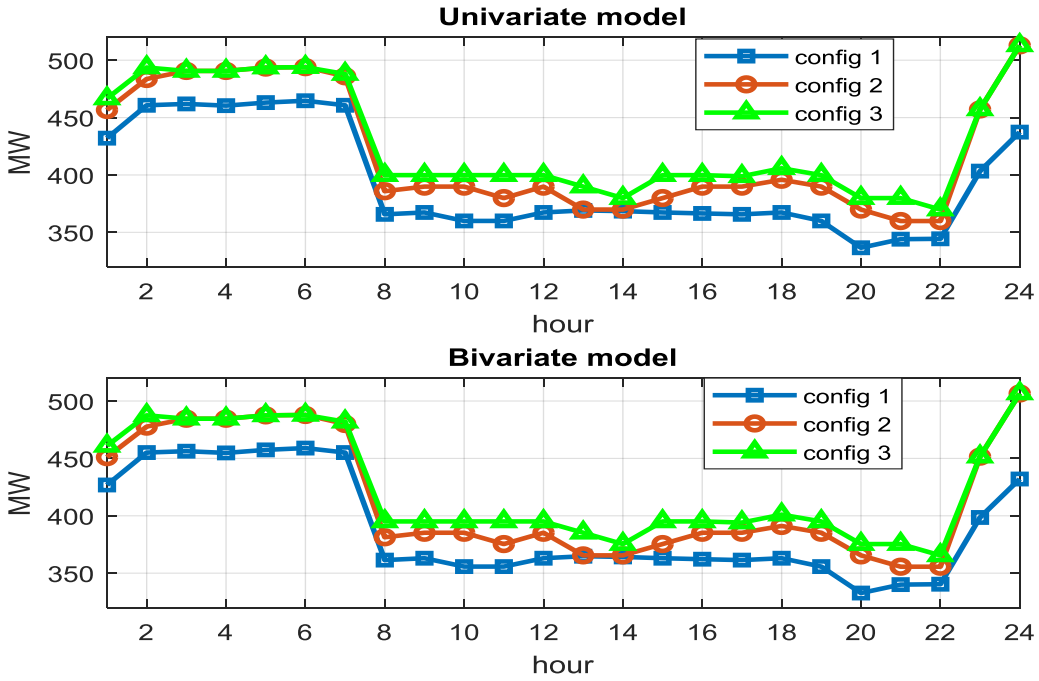


Figure 25. Spinning reserve allocation for various network configurations calculated using univariate and bivariate wind forecasting methods

As can be seen, the reserve requirements obtained by using the bivariate wind forecasting method are slightly lower than those obtained by using the univariate method. In particular, the average reduction in reserve requirements was 0.27 percent. The drop in wind prediction error and the comparatively high share of wind power output in the test method are the key reasons for this decrease (5.84 percent in terms of generated energy). Similar results were observed in EENS (see Figure 26).

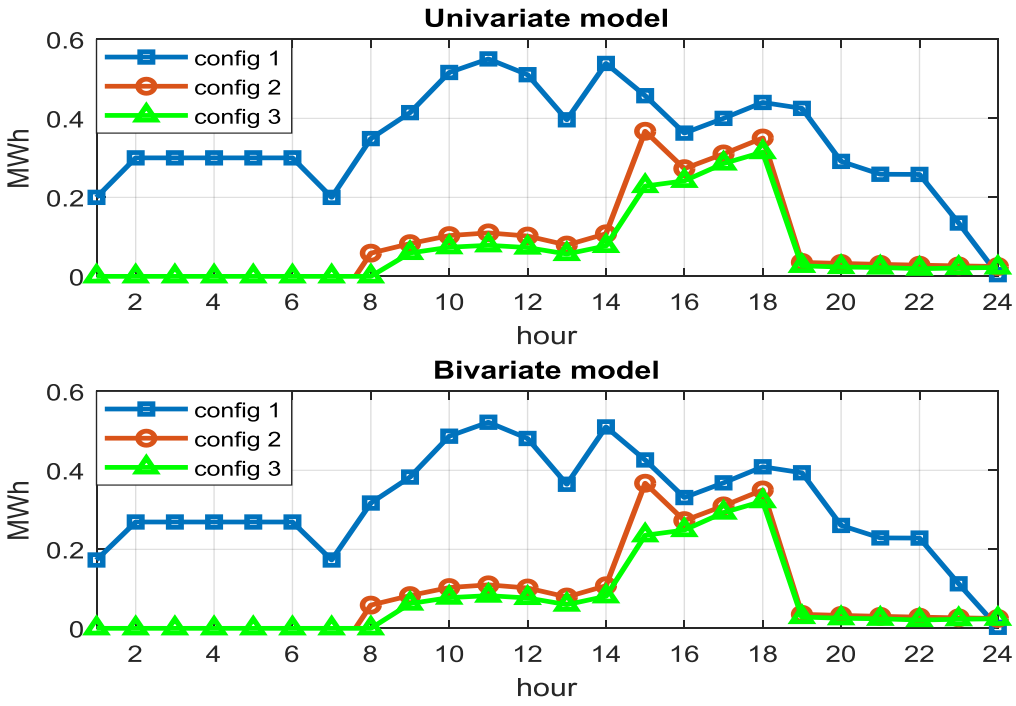


Figure 26. EENS for various network configurations calculated using univariate and bivariate wind forecasting methods

4.1.6. Computational Complexity

The simulations in the preceding parts were conducted on a desktop PC with an Intel Core i5-7200 processor running at 2.50 GHz and 4.00 GB of RAM with Windows 10 operating system. There were 53 units and 48 buses in the evaluation scheme. The model takes an average of 19 minutes to run, with the majority of the time spent on unit commitment and spinning reserve allocation optimization. The simulation calculation time can be cut in half by lowering the COPT resolution or ignoring the low-probability grid states. It is important to remember, though, that this reduction would have a detrimental impact on the model's accuracy.

Similar experiments were performed on a series of test systems of 2, 3, and 5 times as many generators and buses as in the initial two-area RTS to assess the proposed model's performance on a larger scale. The findings reveal that the time it takes to run the simulations grows exponentially, with an estimated computation time of nearly 80 minutes for a system consisting of 265 generators and 240 buses. This level of efficiency is obviously unsustainable in a true grid system with a much more complicated topology and a greater number of generators and buses. Nonetheless, by applying the model in a higher-order programming language and running it on a high-performance machine, the computational time can be conveniently reduced to an appropriate range.

4.2. Conventional Passenger Elevators

4.2.1. Evaluation Methodology

This section delves into a case study of a 10-story office complex with four elevators. The key aim of this analysis was to measure and compare the performance metrics for the EGC algorithms presented in Section 3.2 for the group of conventional elevators.

The elevator group performance metrics are given by the following indicators.

Passenger Average Travel Time (ATT) is the amount of time a passenger spends in an elevator vehicle, estimated in seconds, from the time they board the lift before they walk off the destination level.

Passenger Average Journey Time (AJT) is the time taken by a passenger from the time he or she presses the call button, or joins other people who have already pressed the call button, before an elevator door opens at the destination level, measured in seconds.

Passenger Average Waiting Time (AWT) is the amount of time a passenger spends waiting for an elevator to open the doors at the boarding level, estimated in seconds, from the time they push the call button or join other people who have already pressed the call button.

4.2.1.1 Building Dataset

As it was already mentioned, the case study simulations were conducted on a 10-story building containing 4 elevator cars. It is important to point out that the building is assumed to be a regular office building with working hours from 9.00 to 18.00 and lunch time from 13.00 to 14.00.

Additional variables comprising the building dataset are as follows:

Inter-floor Distance (H_f) – is the distance between adjacent floors expressed in meters. In this test case H_f is equal to 3 m.

Express Jump (*EJ*) – is an extra distance between the main terminal floor (lobby) and the second floor. In this test case *EJ* is equal to 1.

4.2.1.2 Elevator System Dataset

The elevator system considered in this analysis is an elevator group system consisting of 4 elevator cars each having up and down call buttons on the outside and 10 floor buttons inside the elevator car. All elevator cars work under the collective control strategy.

Additional variables comprising the elevator dataset are as follows:

Elevator Car Capacity (E_c) – is the maximum number of people (items) that a single elevator car can carry. In this test case E_c is equal to 10 adult people (large items).

Elevator Rated Speed (V_e) – is an average velocity of an elevator car expressed in m/s. In this test case V_e is constant and equals to 1 m/s.

Inter-floor Flight Time (t_{if}) – is the time, expressed in seconds, required for an elevator car to cover the distance between two adjacent floors. In this test case t_{if} is equal to 3 seconds.

Door Opening Time (t_{do}) – is the time, expressed in seconds, required to open the elevator doors from the moment of complete stop on the destination floor until the moment when the door is fully open. In this test case t_{do} is equal to 3 seconds.

Door Closing Time (t_{dc}) – is the time, expressed in seconds, required to close the elevator doors from the moment of starting the door closure until the moment when the elevator door is fully closed. In this test case t_{dc} is equal to 3 seconds.

4.2.1.3 Passenger Traffic Dataset

For the purpose of this study it was assumed that the building is populated not only by the office workers, but also by the random people traveling with children and bags or other belongings. It is also important to point out that the traffic dataset cannot be estimated with absolute accuracy, due to an unpredictable nature of human behavior. The variables comprising the passenger traffic dataset are as follows:

Number of Passengers Boarding from a Floor (P_b) – is the number of passengers boarding from a specific floor whose destination floors could be the main terminal or any other floor in the building. P_b is highly dependent on the number of people residing on a specific floor.

Number of Passengers Alighting at a Floor (P_a) – is the number of passengers alighting from an elevator car on a specific floor. P_a is dependent on the capacity of an elevator car.

Traffic Mode *(TM)* – the traffic mode indicates the movement direction of the passengers. A unidirectional traffic mode occurs during morning up-peaks or evening down-peaks when people come and leave their work places. The multidirectional mode usually occurs during the work day and lunchtime. The multidirectional mode represents a random inter-floor movement of the people without any clear pattern.

Passenger Transfer Time (*PTT*) – is the time required for passengers to enter and leave an elevator car.

Passenger Actions *(PA*) – passenger misbehavior including but not limited to door holding, excessive operation of pushbuttons and so on.

The last two variables are highly unpredictable and influence the randomness and variability of the passenger traffic.

4.2.2. Performance under Different Traffic Conditions

To fully understand whether the performance of the proposed elevator control algorithm is superior to the existing methods, the test simulations should be conducted on different traffic patterns. As a result, three sets of scenarios were used to measure elevator efficiency parameters, each reflecting the following passenger traffic trends:

The term **Up-peak Traffic** refers to a traffic pattern in which most people are moving upward. During the up-peak traffic condition, the passengers move from the main terminal to the other floors. This trend can be seen in office buildings in the mornings as people arrive at work, and to a lesser degree towards the end of the lunch break;

Down-peak Traffic is a form of traffic that happens as most vehicles are moving downward. Passengers appear to transfer from the upper floors to the main terminal during off-peak traffic. The down-peak happens towards the end of the working day, as people leave the workplace, and to a lesser degree at the start of the lunch break, in contrast to the up-peak trend;

If there is no discernible pattern of passenger flow, the traffic pattern is known as **Random Inter-floor** traffic. The daily movement of people in the building on a working day causes a random inter-floor traffic pattern.

It was also assumed that 30% of the population are girls, and 50% of adults have luggage, half of which are small bags and the other half big objects.

Python (Version 3.7, Manufacturer: Python Software Foundation, Wilmington, Delaware, United States) [76], was used to simulate the standard EGC algorithms (NC and updated NC), whereas the proposed EGC algorithm was implemented in Python and Bayesialab (Version: 9.1 PE-L 00000, Manufacturer: Bayesia S.A.S., Changé, France) [75].

4.2.2.1 Up-peak Traffic

This section provides the simulation results obtained from modeling the up-peak traffic patterns. Figure 27 represents the dependence of the average travel time on the traffic intensity during an up-peak passenger traffic pattern with the traffic size equal to 100, 200 and 300 people respectively.

According to the results, in cases with traffic size equal to 200 and 300 people, the proposed MNCBN performs in the same way as MNC when the traffic intensity is very high. Both algorithms show worsening in the average travel time with decreasing traffic

intensity until it reaches 0.1 sec/passenger. Both, MNCBN and MNC start showing some progress with traffic intensity decreasing from 0.1 sec/passenger. Interestingly, both algorithms overpassed by the conventional NC when the traffic intensity is between 0.01 and 1.8 sec/passenger. The average travel time of the proposed MNCBN is lowest when the traffic intensity reaches the value of 1.8 sec/passenger.

However, when the traffic size is below 100 people, the conventional NC algorithm shows significant superiority when the traffic intensity is very high.

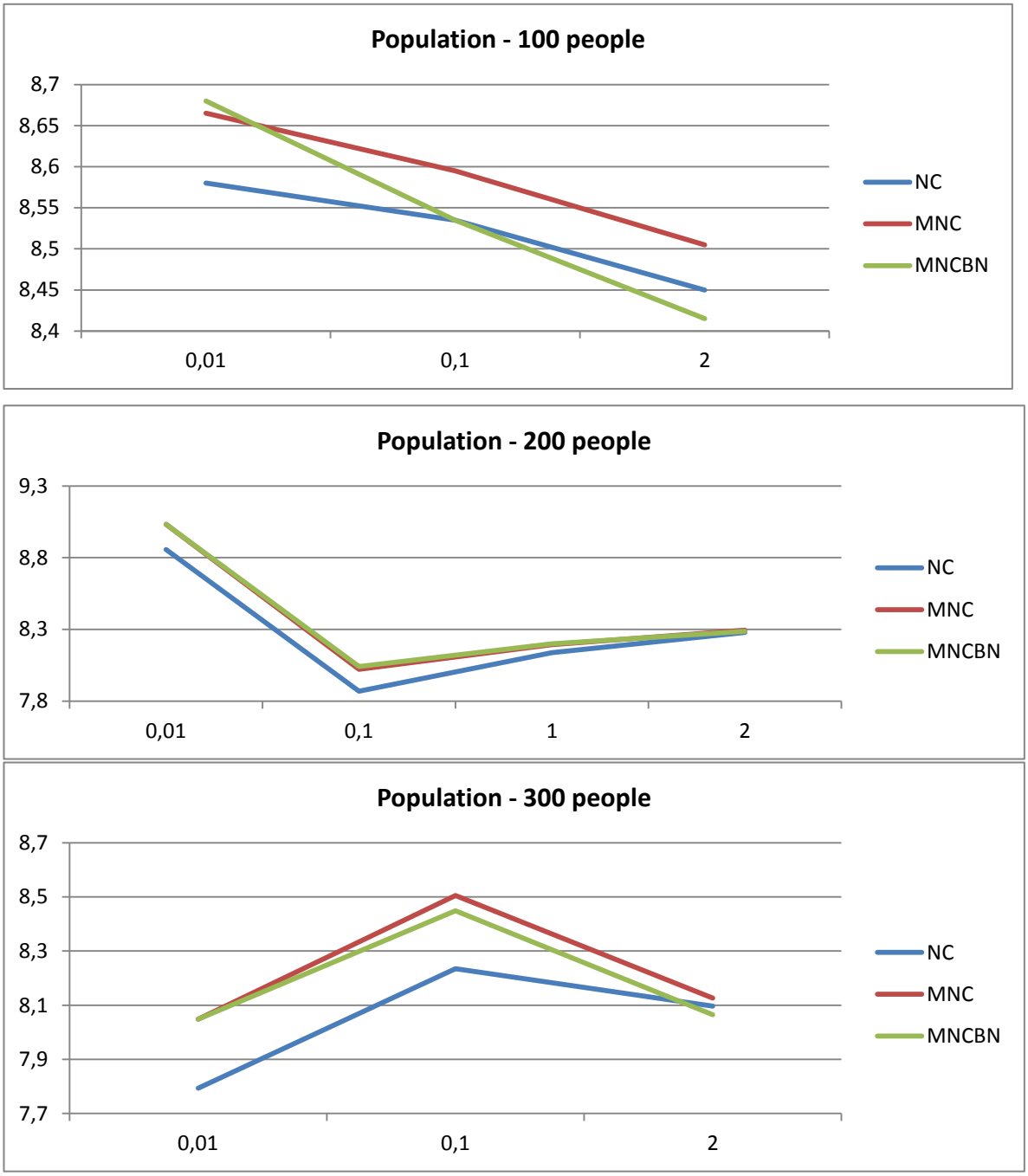


Figure 27. Correlation of ATT and passenger traffic intensity during an up-peak passenger traffic pattern.

Figure 28 represents the dependence of the average journey time on the traffic intensity during an up-peak passenger traffic pattern with the traffic size equal to 100, 200 and 300 people respectively.

According to the simulation results, the proposed MNCBN and MNC show completely the same dependence of the average journey time on the traffic intensity regardless of the traffic size. Both algorithms show reduction of the average journey time with decreasing traffic intensity. Significant reduction in the average journey time is observed when the traffic intensity is decreased from 0.1 sec/passenger, to 2 sec/passenger. The conventional NC algorithm results in almost twice higher values of the average journey time in scenarios with high traffic intensity (0.01 and 0.1 sec/passenger). However, scenarios with low traffic intensity (2 sec/passenger) result in similar performance of all three algorithms.

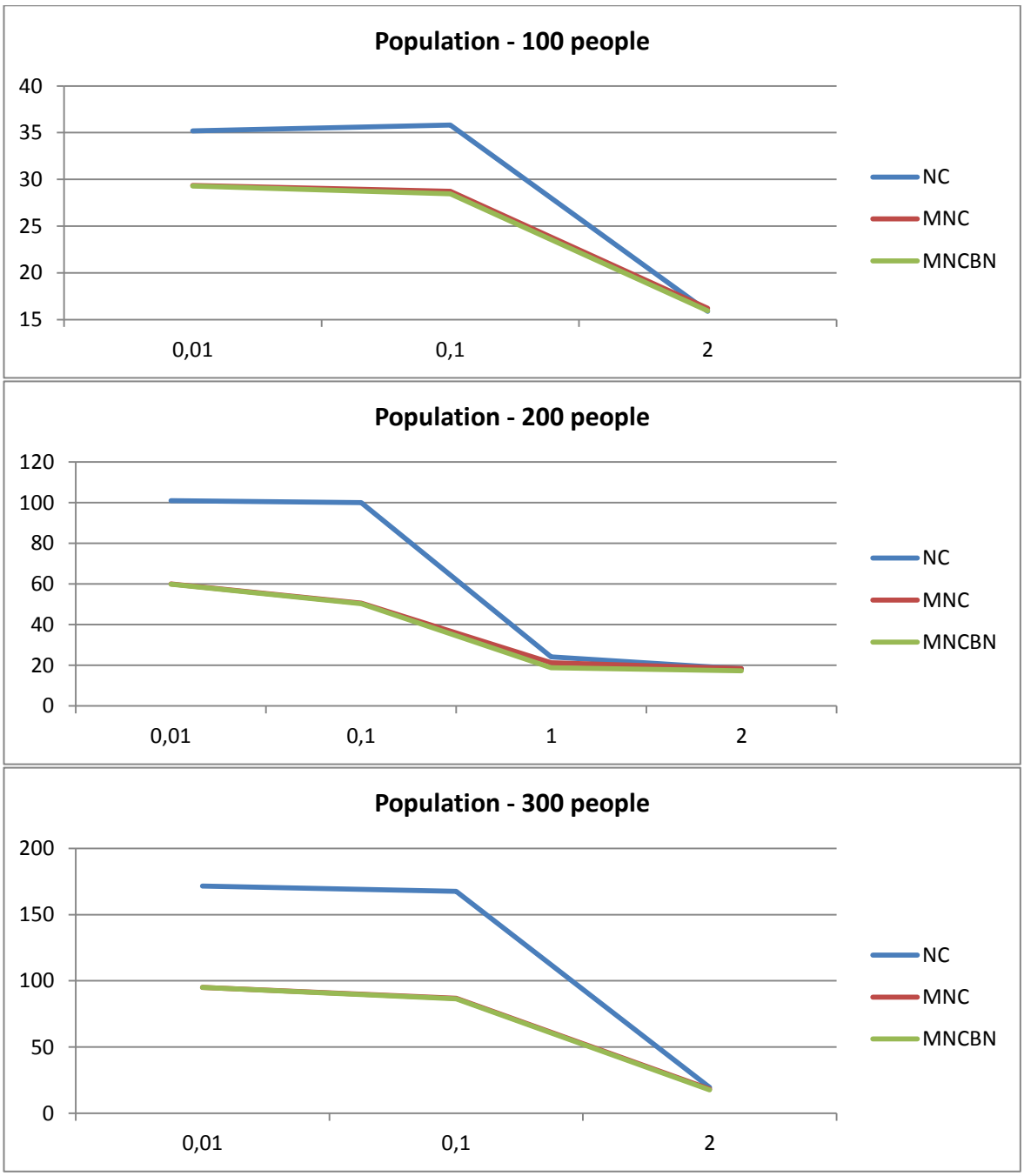


Figure 28**.** Correlation of AJT and passenger traffic intensity during an up-peak passenger traffic pattern.

Figure 29 represents the dependence of the average waiting time on the traffic intensity during an up-peak passenger traffic pattern with the traffic size equal to 100, 200 and 300 people respectively.

Similarly as in case with the average journey time, the proposed MNCBN and MNC show very similar dependence of the average waiting time on the traffic intensity regardless of the traffic size. Both algorithms show reduction of the average waiting time with decreasing traffic intensity. Significant reduction in the average waiting time is observed when the traffic intensity is decreased from 0.1 sec/passenger, to 2 sec/passenger.

The application of the conventional NC algorithm results in almost twice higher values of the average waiting time in scenarios with high traffic intensity (0.01 and 0.1 sec/passenger). However, scenarios with low traffic intensity (2 sec/passenger) result in similar performance of all three algorithms.

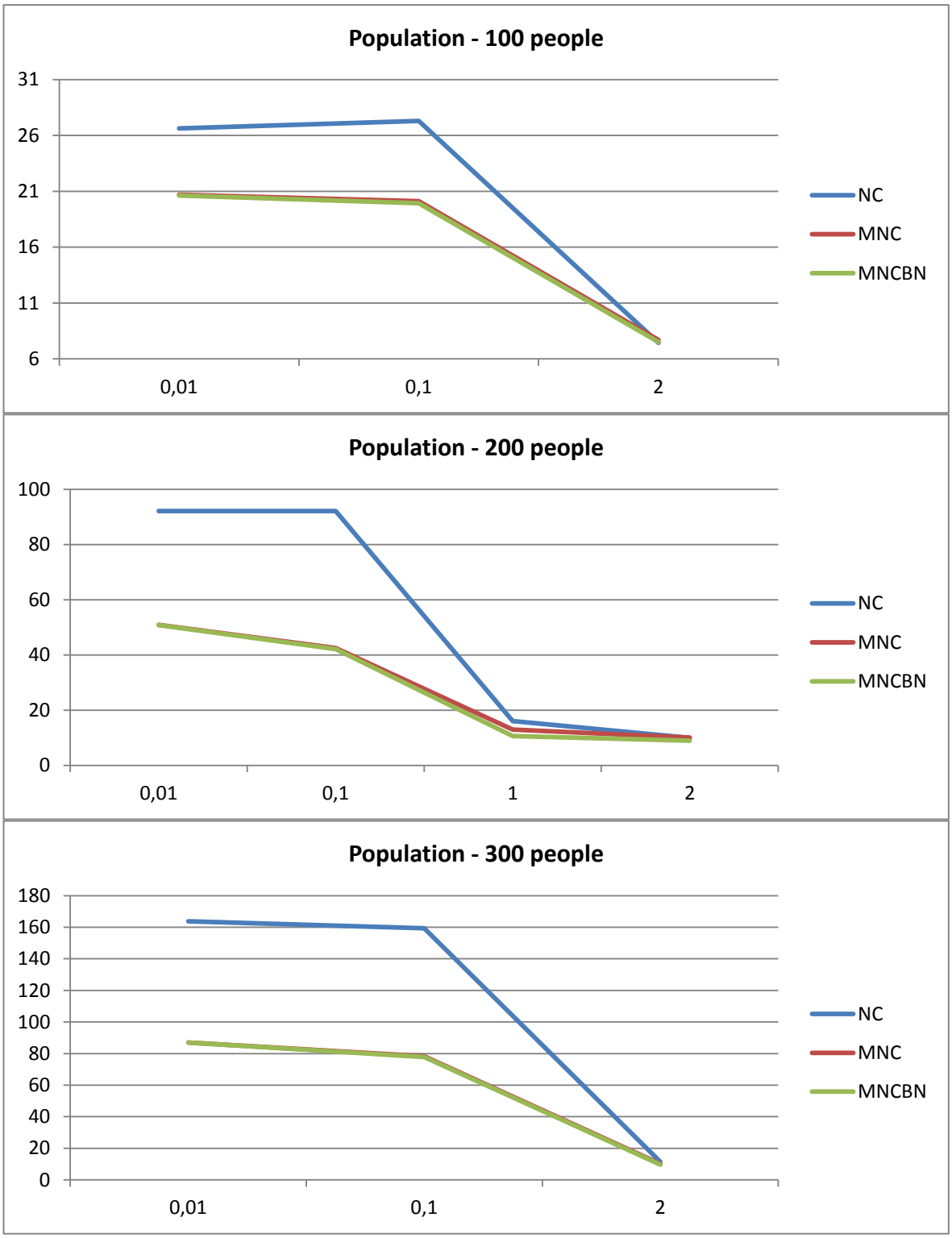


Figure 29**.** Correlation of AWT and passenger traffic intensity during an up-peak passenger traffic pattern.

4.2.2.2 Down-peak Traffic

This section provides the simulation results obtained from modeling the down-peak traffic patterns. Figure 30 represents the dependence of the average travel time on the traffic intensity during a down-peak passenger traffic pattern with the traffic size equal to 100, 200 and 300 people respectively.

According to the results, in cases with traffic size equal to 200 and 300 people the proposed MNCBN shows almost the same performance as MNC when the traffic intensity is very high. Both algorithms show slight worsening in the average travel time with decreasing traffic intensity. When the traffic intensity reaches the value of 2 sec/passenger, MNC start showing some slight improvement in its performance compared to MNCBN. The performance of the conventional NC algorithm is significantly worse than that of MNC and MNCBN. The average travel time increase is between 21-23% for the traffic intensities ranging between 0.01 and 0.1 sec/passenger. This difference is almost doubled when the traffic intensity decreased up to 2 sec/passenger.

However, when the traffic size is below 100 people, the conventional NC algorithm shows significant superiority when the traffic intensity is very high

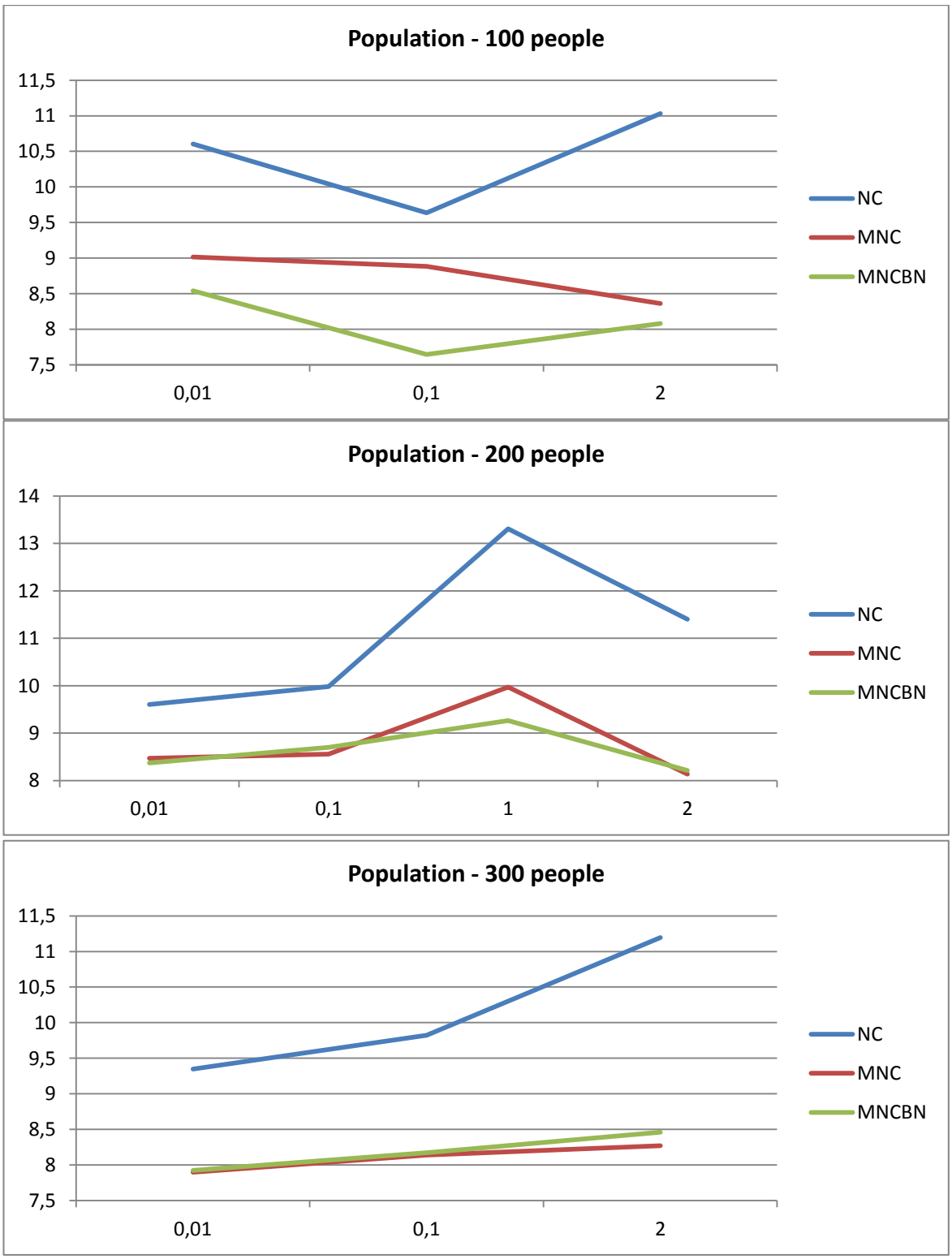
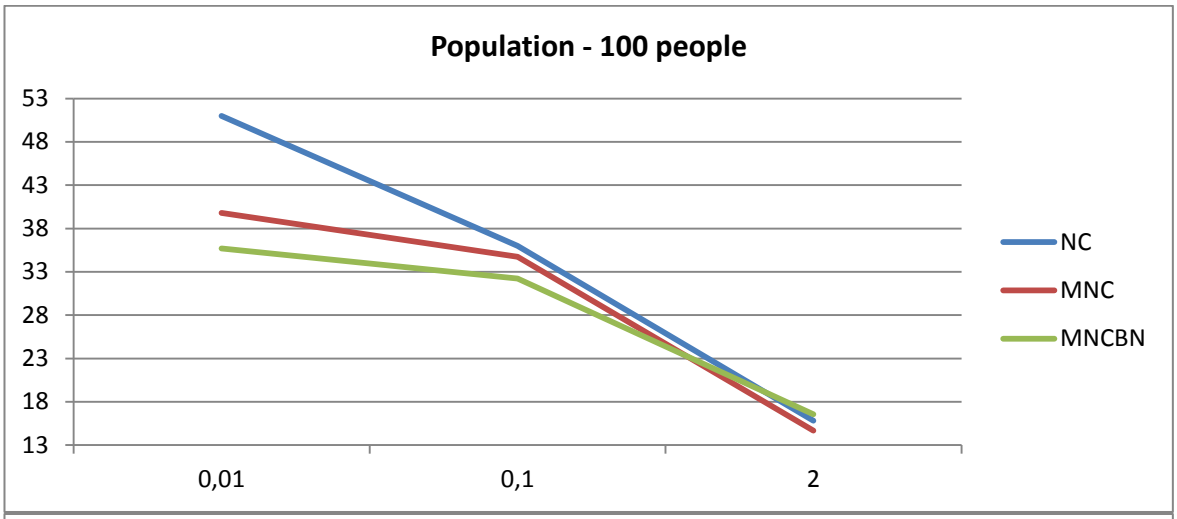


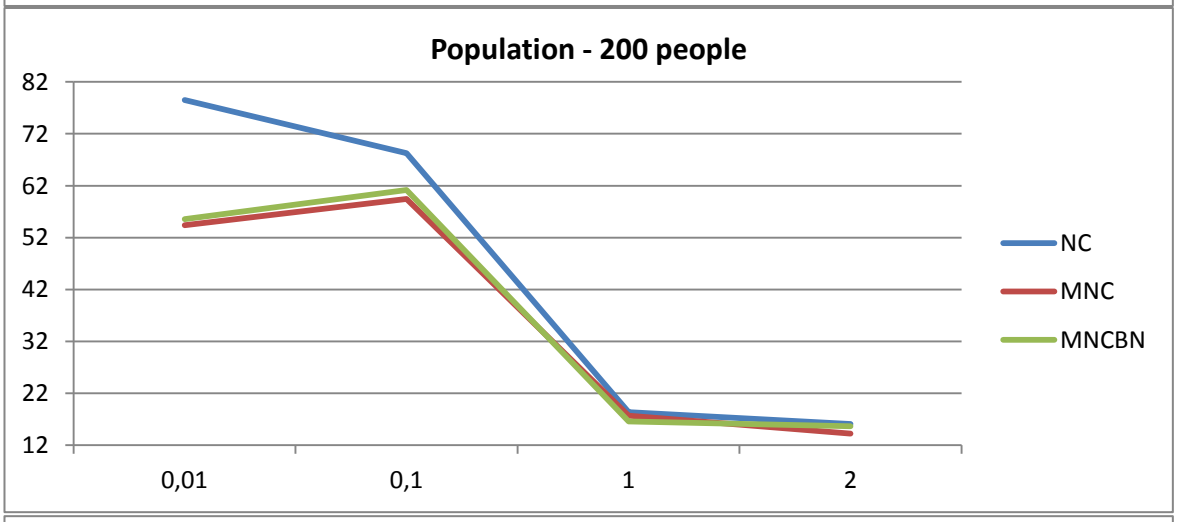
Figure 30**.** Correlation of ATT and passenger traffic intensity during a down-peak passenger traffic pattern.

Figure 31 represents the dependence of the average journey time on the traffic intensity during a down-peak passenger traffic pattern with the traffic size equal to 100, 200 and 300 people respectively.

According to the simulation results, the proposed MNCBN and MNC show almost the same dependence of the average journey time on the traffic intensity within the traffic

intensity range equal to 0.01 and 0.1 sec/passenger. Both algorithms show reduction of the average journey time with decreasing traffic intensity. Significant reduction in the average journey time is observed when the traffic intensity is decreased from 0.1 sec/passenger, to 2 sec/passenger. The conventional NC algorithm results in values that are 20-25% higher of the average journey time of MNCBN and MNC in scenarios with high traffic intensity (0.01 and 0.1 sec/passenger). However, scenarios with low traffic intensity (2 sec/passenger) result in similar performance of all three algorithms.





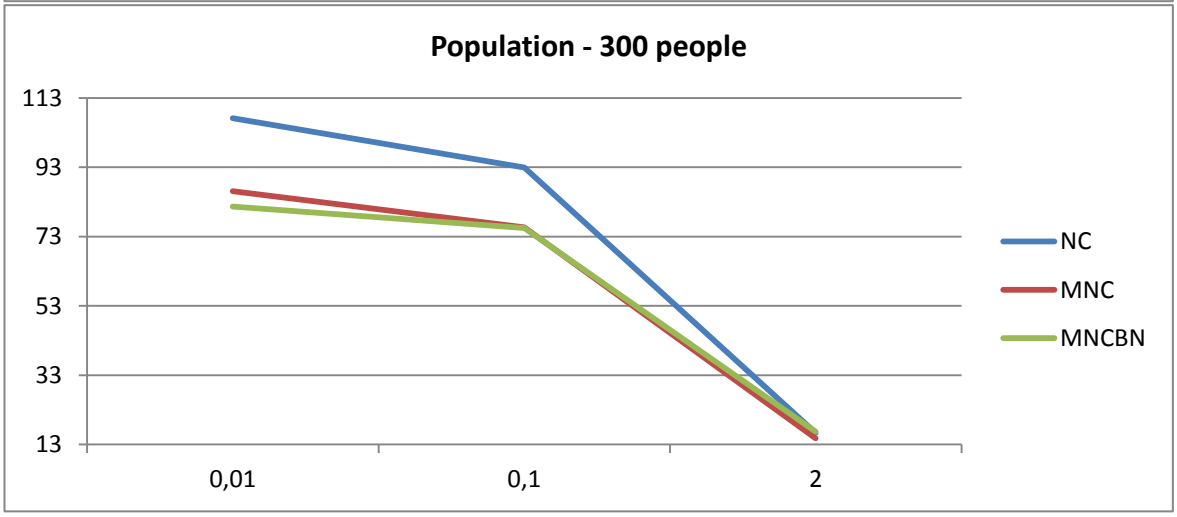


Figure 31**.** Correlation of AJT and passenger traffic intensity during a down-peak passenger traffic pattern.

Figure 32 represents the dependence of the average waiting time on the traffic intensity during a down-peak passenger traffic pattern with the traffic size equal to 100, 200 and 300 people respectively.

Similarly as in case with the average journey time, the proposed MNCBN and MNC show almost the same dependence of the average waiting time on the traffic intensity. Both algorithms show reduction of the average waiting time with decreasing traffic intensity. Significant reduction in the average waiting time is observed when the traffic intensity is decreased from 0.1 sec/passenger, to 2 sec/passenger. The application of the conventional NC algorithm results in values that are 20-25% higher of the average journey time of MNCBN and MNC with high traffic intensity (0.01 and 0.1 sec/passenger). However, scenarios with low traffic intensity (2 sec/passenger) result in similar performance of all three algorithms.

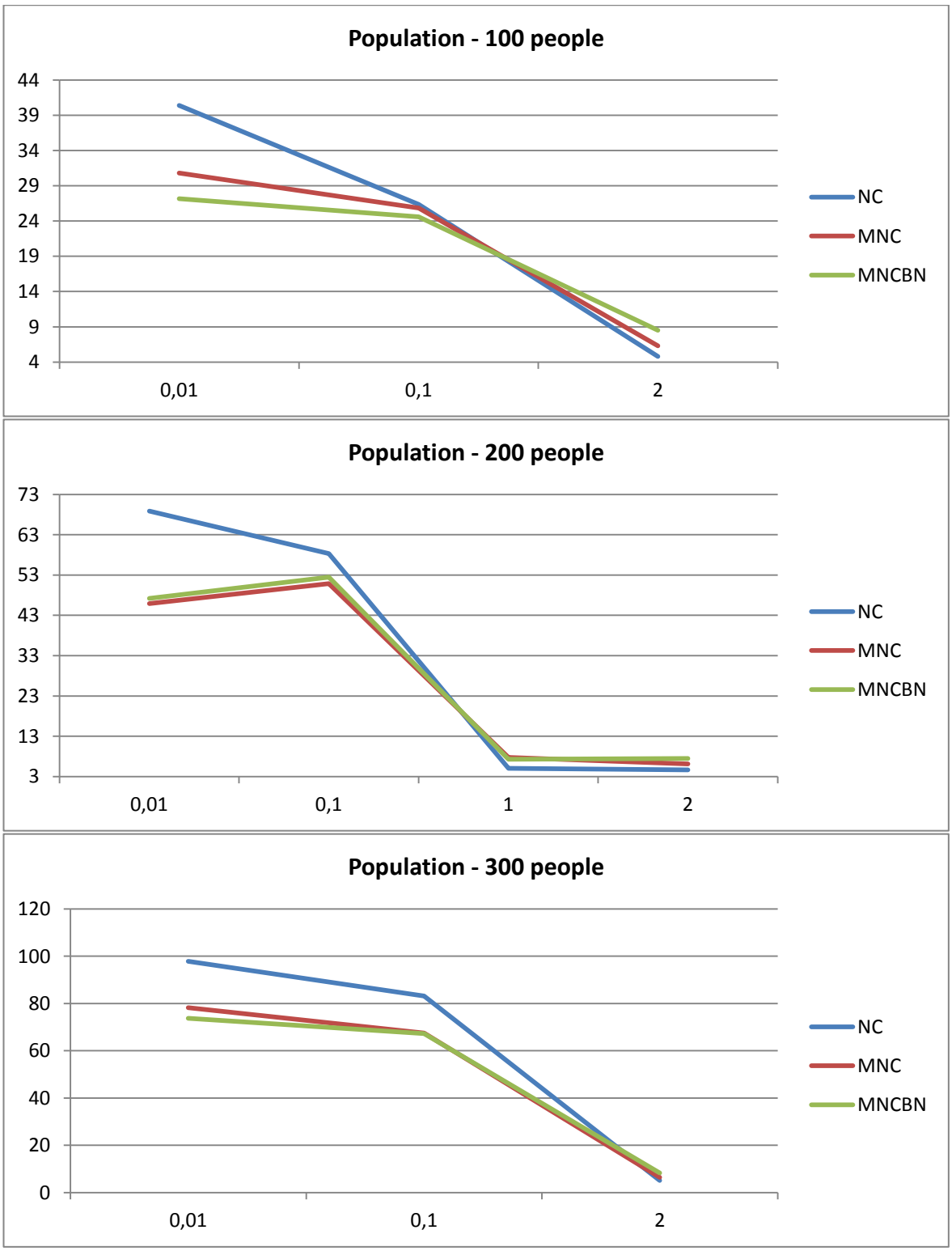


Figure 32**.** Correlation of AWT and passenger traffic intensity during a down-peak passenger traffic pattern.

4.2.2.3 Inter-floor Traffic

This section provides the simulation results obtained from modeling the inter-floor traffic patterns. Figure 33 represents the dependence of the average travel time on the

traffic intensity during an inter-floor passenger traffic pattern with the traffic size equal to 100, 200 and 300 people respectively.

According to the results, the proposed MNCBN shows almost the same performance as MNC. Both algorithms show slight worsening in the average travel time with decreasing traffic intensity. When the traffic intensity reaches the value of 0.1 sec/passenger, MNCBN start showing some slight improvement in its performance compared to compared to the other two algorithms. The performance of the conventional NC algorithm is significantly worse than that of MNC and MNCBN. The average travel time increase is between 52-58% for the traffic intensities ranging between 0.01 and 0.1 sec/passenger. This difference is reduced by more than 50% when the traffic intensity decreased up to 2 sec/passenger.

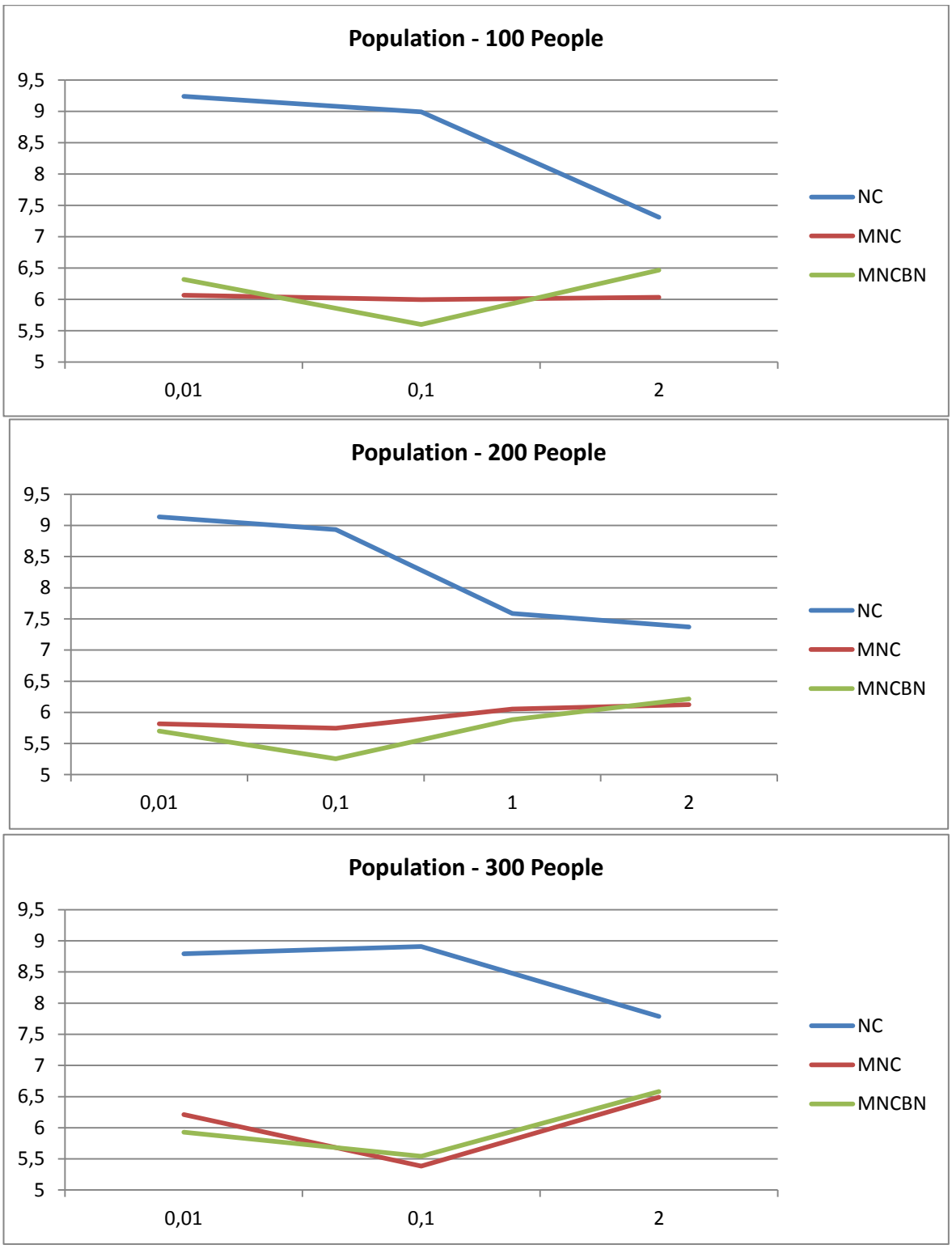


Figure 33. Correlation of ATT and passenger traffic intensity during an inter-floor passenger traffic pattern.

Figure 34 represents the dependence of the average journey time on the traffic intensity during the inter-floor passenger traffic pattern with the traffic size equal to 100, 200 and 300 people respectively.

According to the simulation results, the proposed MNCBN and MNC show almost the same dependence of the average journey time on the traffic intensity within the traffic

intensity range equal to 0.01 and 0.1 sec/passenger. Both algorithms show reduction of the average journey time with decreasing traffic intensity. Significant reduction in the average journey time is observed when the traffic intensity is decreased from 0.1 sec/passenger, to 2 sec/passenger. As oppose to the results obtained for the up-peak and down-peak simulations, the inter-floor traffic pattern does not show significant difference between the conventional NC and the proposed MNCBN. In this particular case, the conventional NC algorithm results in values that are at most 20% higher of the average journey time of MNCBN and MNC in scenarios with high traffic intensity (0.01 and 0.1 sec/passenger). Scenarios with low traffic intensity (2 sec/passenger) result in almost similar performance of all three algorithms.

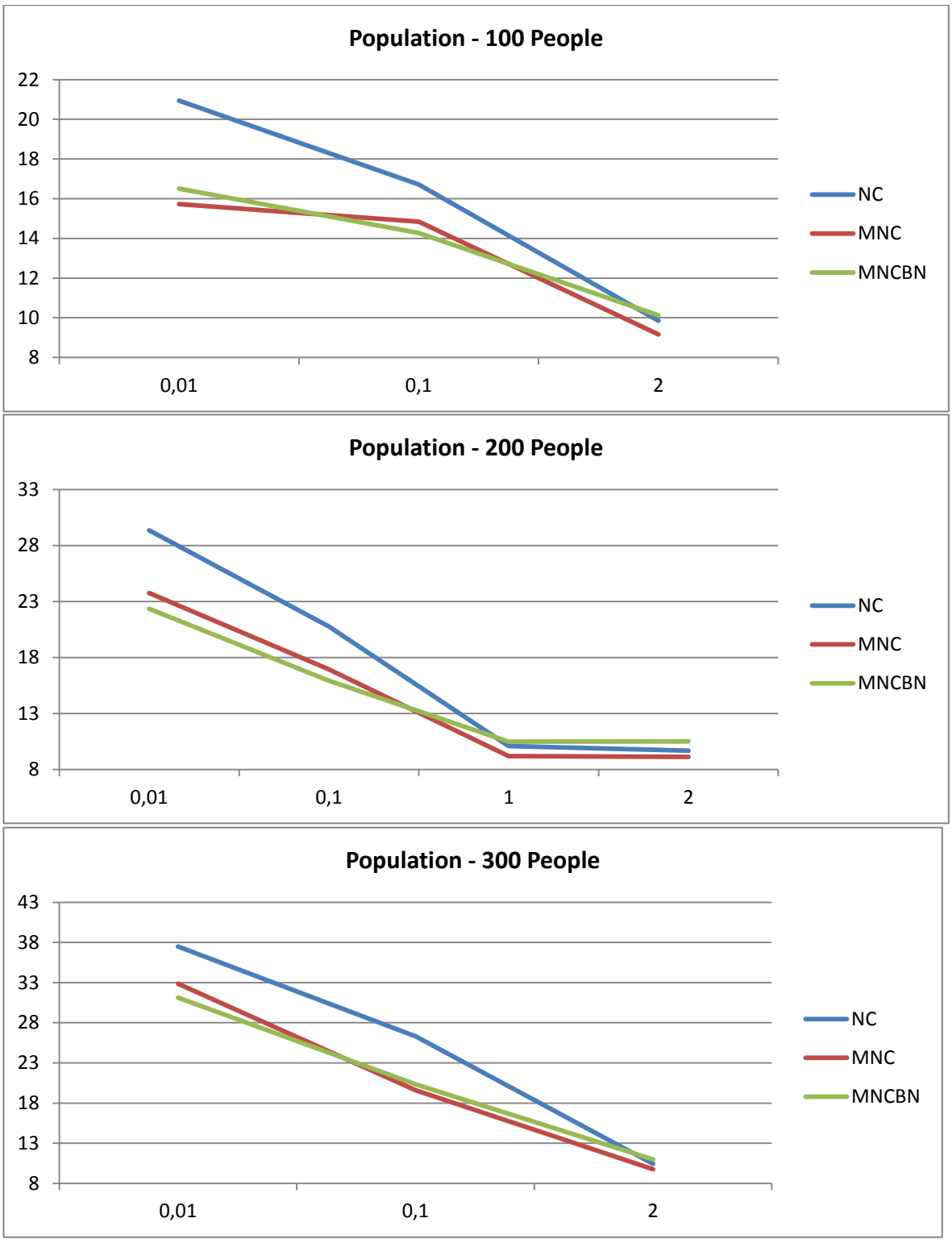
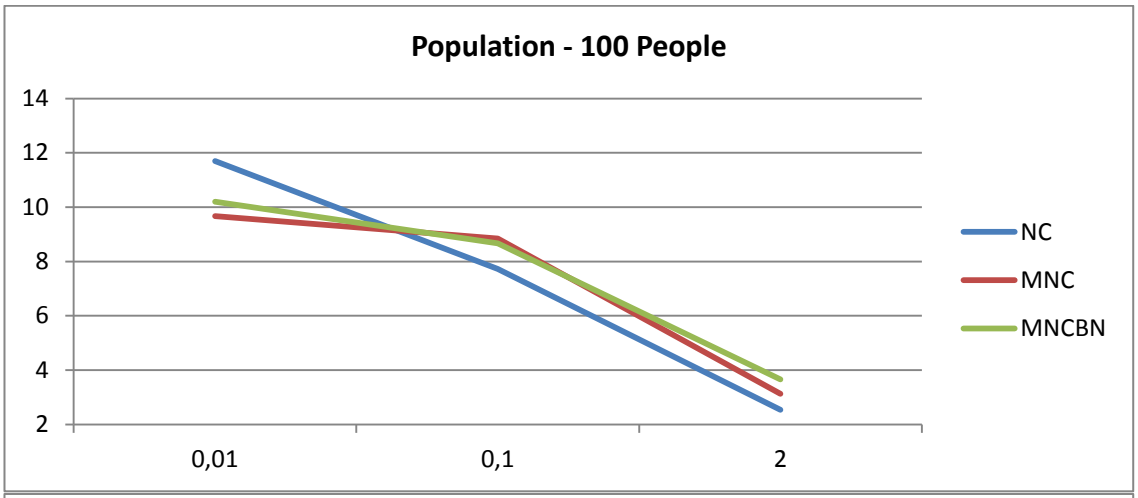


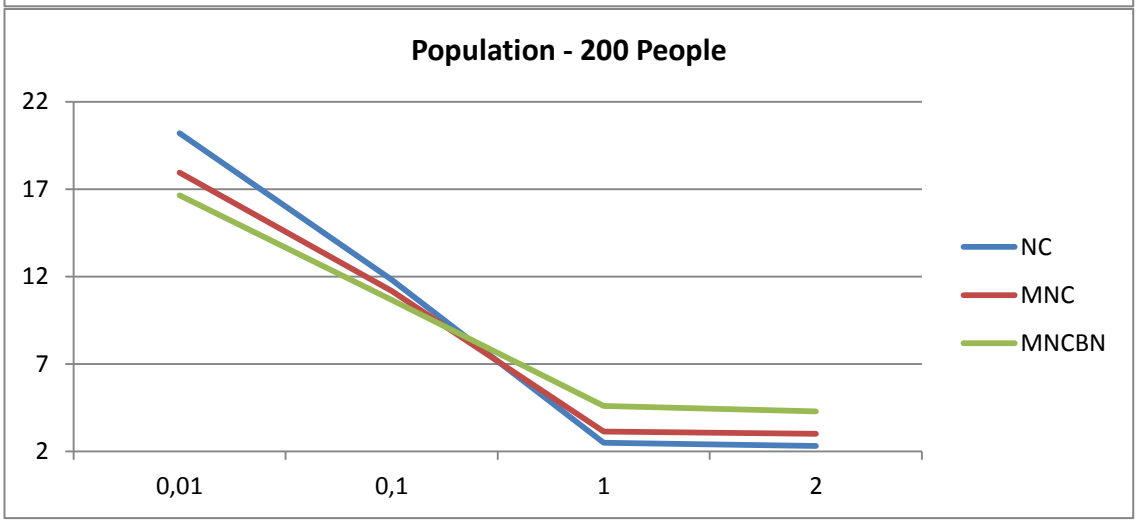
Figure 34**.** Correlation of AJT and passenger traffic intensity during an inter-floor passenger traffic pattern.

Figure 35 represents the dependence of the average waiting time on the traffic intensity during an inter-floor passenger traffic pattern with the traffic size equal to 100, 200 and 300 people.

Similarly as in case with the average journey time, the proposed MNCBN and MNC show almost the same dependence of the average waiting time on the traffic

intensity. Both algorithms show reduction of the average waiting time with decreasing traffic intensity. However, the reduction in the average waiting time is not as significant as in case with the average journey time when the traffic intensity is decreased from 0.1 sec/passenger, to 2 sec/passenger. Moreover, the application of the conventional NC algorithm results in values that are very close to those of MNC and MNCBN. The scenarios with population less than 200 people result in the average waiting time values of NC lower than those of MNC and MNCBN.





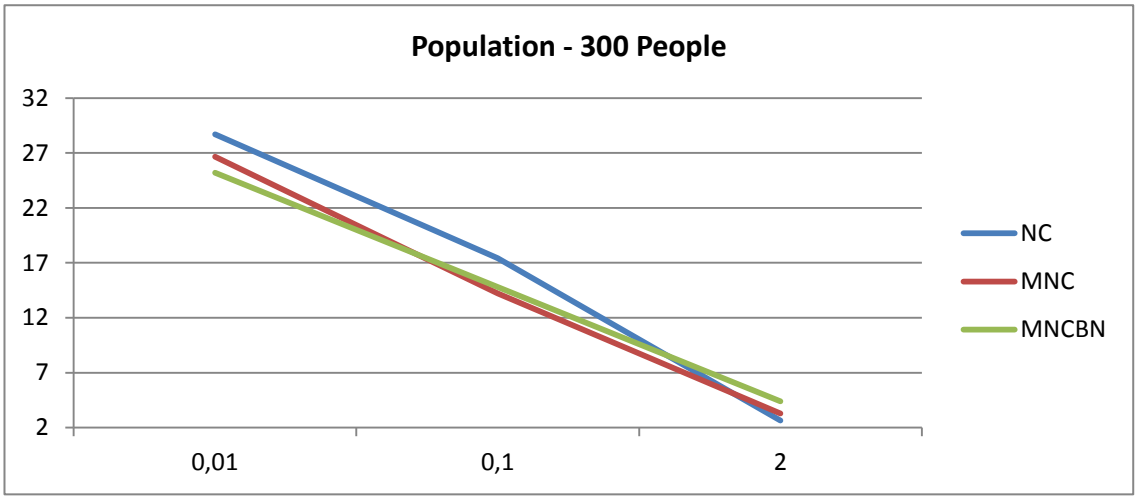


Figure 35**.** Correlation of AWT and passenger traffic intensity during an inter-floor passenger traffic pattern.

4.2.3. Correlation with Traffic Intensity and Traffic Size

The EGC algorithms were put to the test in situations of differing traffic sizes and intensities. The number of passengers in the traffic ranged from 100 to 300. The average time between passenger arrivals, or traffic intensity, ranged from 0.01 to 2 seconds. The simulation results are presented in Table 6.

The simulation results for the up-peak traffic pattern are shown in Table 6. In cases with high traffic intensity, NC showed the best results in terms of ATT, as seen in the table. In situations with a short passenger arrival interval, the modified nearest car EGC algorithm (MNC) and the modified nearest car EGC algorithm with BN (MNCBN) do best in terms of AJT and AWT. When the traffic volume was very poor, both algorithms tended to behave similarly.

Table 6. The up-peak traffic pattern's simulation findings

Traffic Size	Interval	NC [1]			MNC [2]			MNCBN [3]		
		ATT	AJT	AWT	ATT	AJT	AWT	ATT	AJT	AWT
100	0.01	8.58	35.20	26.62	8.66	29.36	20.70	8.68	29.30	20.62
200	0.01	8.86	100.94	92.08	9.03	59.99	50.96	9.03	59.87	50.84
300	0.01	7.79	171.52	163.73	8.05	94.98	86.93	8.05	94.98	86.93
100	0.1	8.54	35.83	27.30	8.59	28.72	20.13	8.53	28.47	19.94
200	0.1	7.87	100.02	92.15	8.02	50.49	42.47	8.04	50.20	42.16
300	0.1	8.24	167.53	159.29	8.50	86.88	78.38	8.45	86.33	77.88
200	1	8.14	24.18	16.05	8.20	21.19	12.99	8.20	18.84	10.64
100	2	8.45	15.89	7.44	8.51	16.21	7.71	8.42	15.93	7.52
200	2	8.28	18.27	9.99	8.30	18.40	10.11	8.29	17.24	8.95
300	2	8.10	19.56	11.47	8.13	18.28	10.16	8.07	17.81	9.74

[1] Nearest car elevator group control (EGC) algorithm. [2] Modified nearest car EGC algorithm. [3] Modified nearest car EGC algorithm with BN.

The findings for the down-peak traffic trend as seen in Table 7. In terms of any elevator efficiency index, the NC algorithm's results under down-peak conditions were the lowest. In terms of AJT and AWT, the MNC and MNCBN recorded identical results. Similarly to the previous example, with increasing passenger arrival intervals, both algorithms appeared to yield similar effects.

Table 7. Findings of the simulation for the down-peak traffic pattern

Traffic Size	Interval	NC [1]			MNC [2]			MNCBN [3]		
		ATT	AJT	AWT	ATT	AJT	AWT	ATT	AJT	AWT
100	0.01	10.61	51.00	40.39	9.01	39.80	30.78	8.54	35.72	27.18
200	0.01	9.61	78.49	68.89	8.47	54.38	45.91	8.37	55.58	47.21
300	0.01	9.35	107.15	97.80	7.90	86.09	78.19	7.92	81.68	73.76

100	0.1	9.64	36.00	26.36	8.89	34.72	25.84	7.65	32.24	24.60
200	0.1	9.99	68.28	58.29	8.56	59.44	50.88	8.70	61.18	52.48
300	0.1	9.82	92.94	83.12	8.14	75.63	67.49	8.17	75.45	67.27
200	1	13.31	18.35	5.04	9.97	17.69	7.72	9.27	16.56	7.29
100	2	11.03	15.84	4.81	8.36	14.67	6.31	8.08	16.58	8.50
200	2	11.40	16.06	4.66	8.14	14.24	6.10	8.21	15.67	7.46
300	2	11.20	16.34	5.15	8.27	14.77	6.50	8.46	16.74	8.28

[1] Nearest car EGC algorithm. [2] Modified nearest car EGC algorithm. [3] Modified nearest car EGC algorithm with BN.

The simulation results for the random inter-floor traffic pattern as shown in Table 8. Except in the case of a traffic size of 200 individuals, traditional NC recorded the worst results. MNC and MNCBN showed comparable situations for ATT and AJT in the rest of the scenarios. In situations of 100 and 300 participants, however, MNCBN outperformed NC and MNC in terms of AWT.

Table 8**.** Random inter-floor traffic pattern simulation results

Traffic Size	**Interval**	**NC [1]**			**MNC [2]**			**MNCBN [3]**		
		ATT	**AJT**	**AWT**	**ATT**	**AJT**	**AWT**	**ATT**	**AJT**	**AWT**
100	0.01	9.24	20.94	11.70	6.07	15.73	9.67	6.32	16.51	10.20
200	0.01	9.14	29.34	20.20	5.82	23.75	17.94	5.70	22.35	16.65
300	0.01	8.79	37.49	28.70	6.21	32.87	26.66	5.93	31.11	25.18
100	0.1	8.99	16.72	7.73	6.00	14.84	8.85	5.60	14.27	8.67
200	0.1	8.93	20.75	11.82	5.75	16.92	11.18	5.26	15.92	10.67
300	0.1	8.91	26.32	17.41	5.38	19.60	14.22	5.54	20.35	14.80
200	1	7.59	10.08	2.49	6.06	9.19	3.13	5.88	10.49	4.60
100	2	7.31	9.85	2.54	6.03	9.16	3.13	6.47	10.13	3.66
200	2	7.37	9.67	2.30	6.13	9.13	3.00	6.22	10.50	4.29
300	2	7.79	10.44	2.65	6.49	9.77	3.28	6.58	10.97	4.39

[1] Nearest car EGC algorithm. [2] Modified nearest car EGC algorithm. [3] Modified nearest car EGC algorithm with BN.

As previously mentioned, up-peak and down-peak traffic trends exist twice a day in office buildings, with a combined length of 2–3 hours a day. We based more on testing the efficiency of EGC algorithms in this traffic environment since the spontaneous inter-floor traffic trend prevails during the day.

As can be seen in Figure 36, MNCBN has the highest performance in terms of total travel time with traffic sizes ranging from 10 to 300 passengers and an average passenger arrival time of 0.1. MNC and NC, on the other hand, showed very comparable performances, with MNC slightly outperforming NC.

Figure 36. Correlation of ATT and traffic size

Figure 37 and Figure 38 illustrate the relationship between average journey time and average waiting time and traffic size. Similarly to the previous example, the proposed MNCBN's AJT and AWT were smaller than those of the other two algorithms. The curves of the MNC algorithm, on the other hand, were much steeper than the curves of NC and MNCBN in this case. Both MNCBN and NC provided comparable results, with the MNCBN algorithm marginally outperforming the NC algorithm.

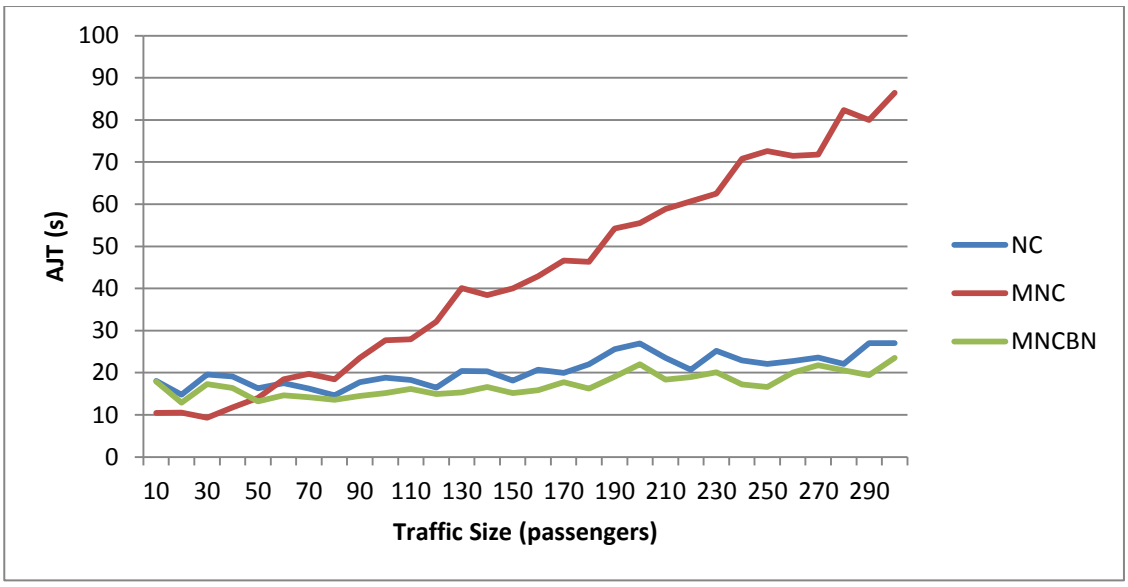


Figure 37. Correlation of AJT and traffic size

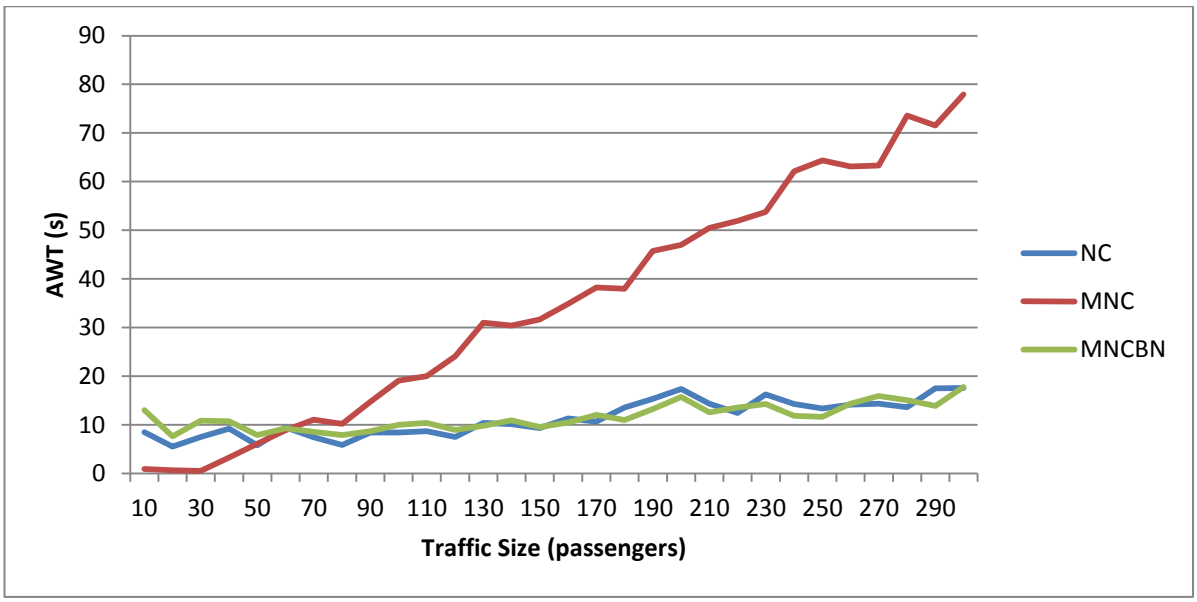


Figure 38. Correlation of AWT and traffic size.

Figure 39 represents the relationship between average travel time and traffic intensity. As can be seen in the table, both MNC and MNCBN behaved equally, while the ATT of the traditional NC was higher. However, the curves appear to approach each other as traffic volume increases, suggesting that MNC and MNCBN outperformed in situations when inter-floor traffic was particularly heavy.

Figure 39. Correlation of ATT and traffic intensity.

Figure 40 and Figure 41 illustrate the relationship between average journey time and average waiting time and traffic volume, respectively. It's worth noting that these findings were collected under the same conditions as before. Surprisingly, the proposed MNCBN algorithm's average journey and waiting times tended to increase as traffic intensity increased, while the AJT and AWT of NC and MNC tended to decrease,

suggesting that the proposed MNCBN algorithm's efficiency decreased as inter-floor traffic intensity increased. One of the reasons for underperformance of the MNCBN in scenarios with very high traffic intensity is that in these scenarios the passengers call an elevator on every floor of the building (or almost on every floor, depending on how intense the passenger traffic is). In such scenario, NC, and to some extent MNC, require each elevator car to stop on every floor, whereas the proposed MNCBN tends to skip the floors with a few people. The number of passengers waiting for an elevator on these floors grows rapidly, which mostly negatively affects the average waiting time.

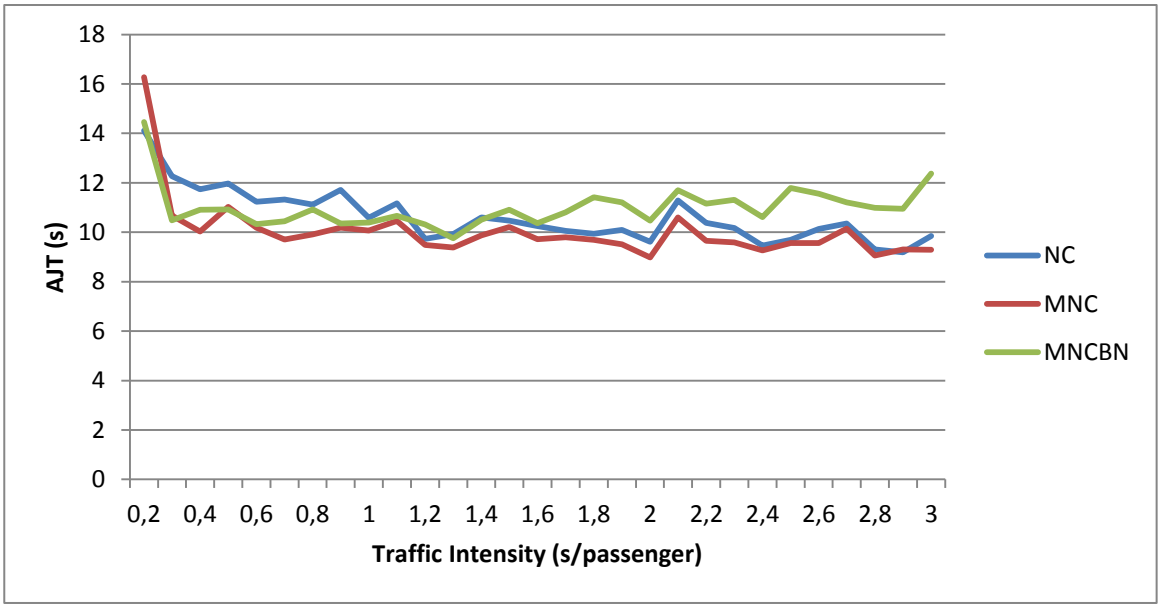


Figure 40. Correlation of AJT and traffic intensity.

According to the findings shown in Figure 41, NC performed better in terms of average waiting time during high inter-floor traffic intensity.

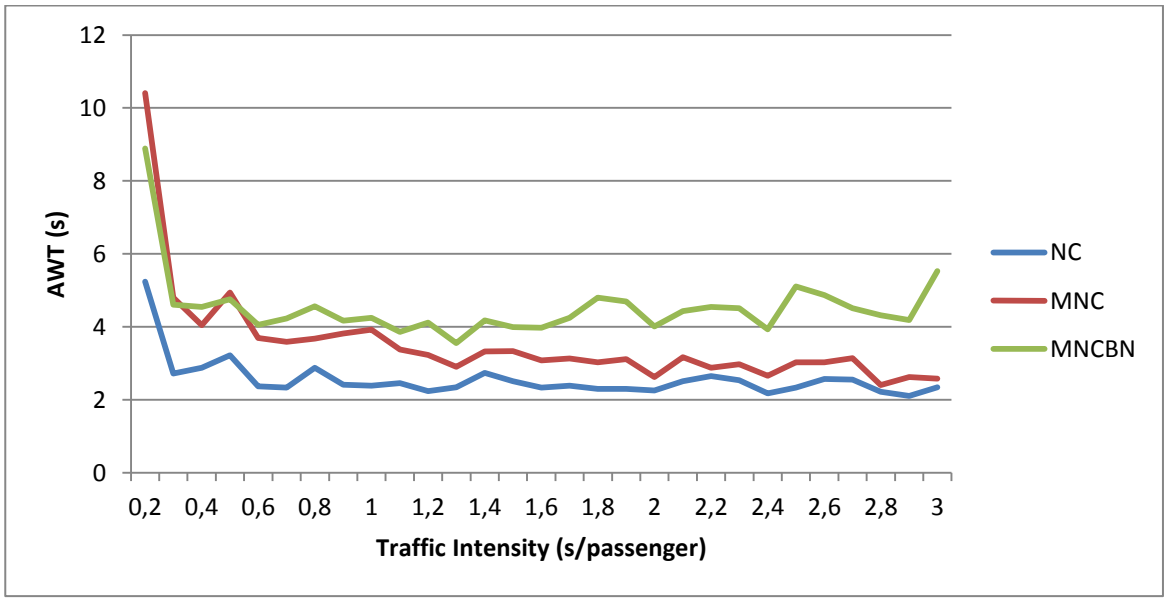


Figure 41. Correlation of AWT and traffic intensity.

4.2.4. Correlation of the Performance Metrics with Building Height

The height of a building is a crucial element in the efficiency of an EGC algorithm. The traditional NC algorithm, for example, is ideal for mid-rise buildings (7–10 floors), according to [63]. The efficiency of the MNC and MNCBN EGC algorithms is compared to the performance of the conventional NC EGC algorithm in this segment for buildings with 11–30 floors and 5 elevators. The case in which a total of 300 persons arrived with a 0.1 second delay was also considered, as was the case in the previous segment. Figure 42 demonstrates how ATT is influenced by the height of the house.

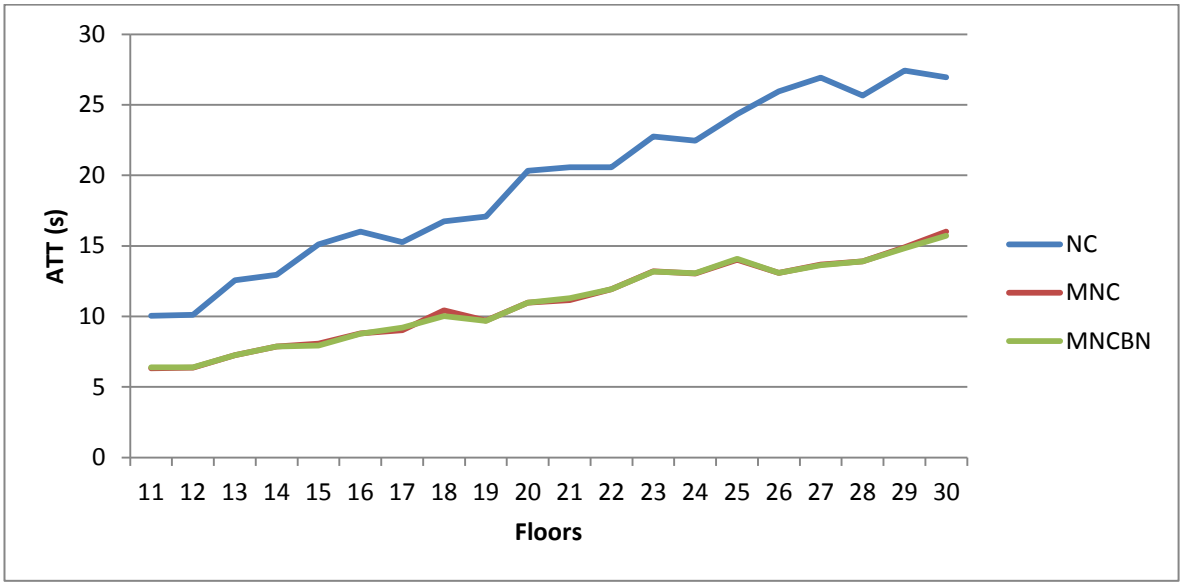


Figure 42. Correlation of ATT and building height for the random inter-floor traffic pattern

The ATT of all three algorithms grew as the height of the building increased. However, as opposed to MNC and MNCBN, the traditional NC curve has a higher gradient. The same can be seen in the AJT and AWT tests, which are seen in Figure 43 and Figure 44, respectively.

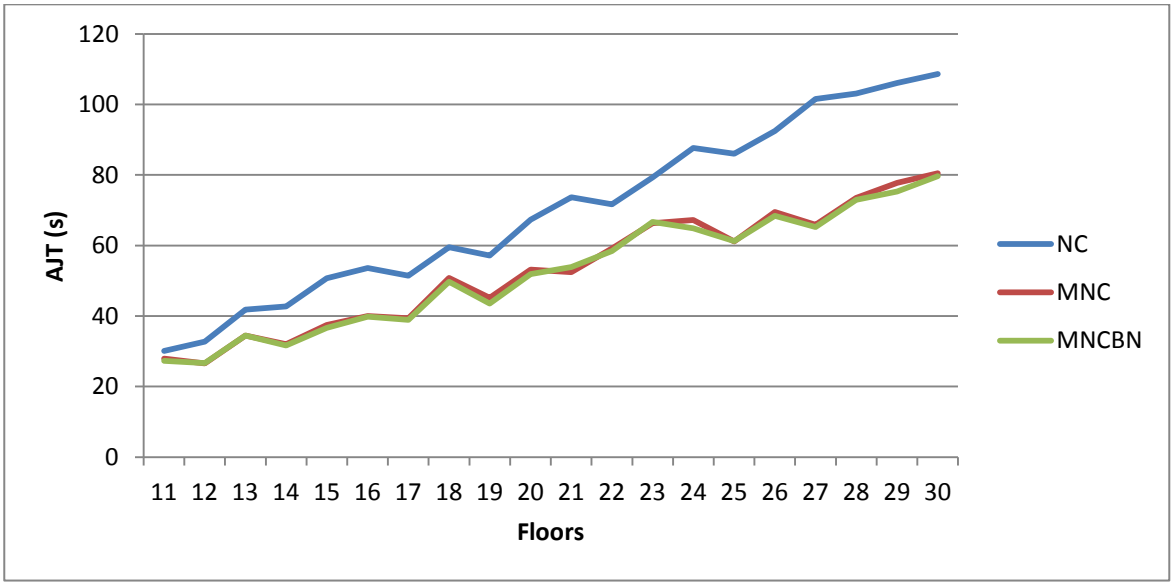


Figure 43. Correlation of AJT and building height for the random inter-floor traffic pattern

Similarly to ATT, we discovered that AJT and AWT increased as the number of building floors increased, but the MNC and MNCBN curves are steeper in this situation. As a result, it's fair to conclude that the traditional NC algorithm is not optimal for a high-rise structure. In reality, implementing building clustering is a popular method for improving the efficiency of an NC algorithm in high-rise buildings.

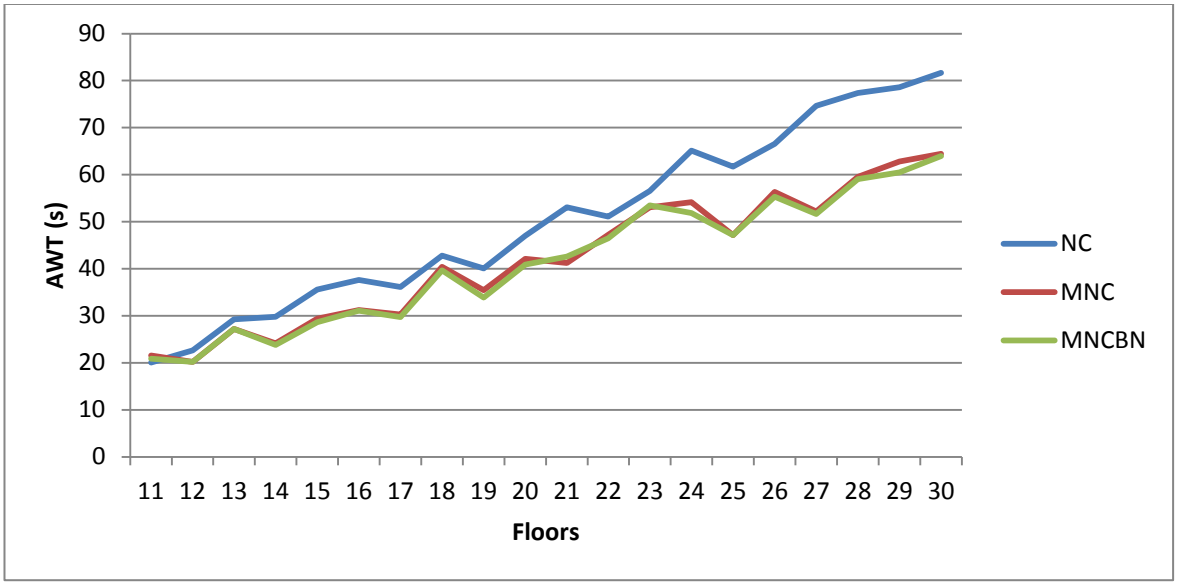


Figure 44. Correlation of AWT and building height for the random inter-floor traffic pattern

Chapter 5 – Discussion

5.1. Electric Grid Systems

The first section of this paper explored a probabilistic approach combined with the BN-based Reserve Allocation Adjustment Algorithm for estimating spinning reserves in integrated systems including renewable resources (primarily wind and solar power) and demand response.

First of all, it should be noted the proposed model was tested on a power grid system assuming that there are free and liberal energy and ancillary services markets in place as oppose to a vertically integrated market structure. The market stricter and its rules determine how the market participants interact with each other, and it plays a key role in the overall power grid operation.

The proposed model consists of several submodels, and each submodel is responsible for modeling a specific part of the grid system, such as conventional power generating units, renewable power generators, demand respond providers and so on. In this model, contingencies such as spontaneous outages of traditional generators, as well as load and renewable prediction errors, were taken into account when calculating spinning reserves. A two-state Markov method was applied to each independent power generating unit to model the random generator outages. The modeling of the whole power generating system was done by means of the Capacity Outage Probability Table.

The wind and solar power were considered as a negative load, so the net electricity demand could be simplified to the difference between electric load and total renewable power. As it was mentioned previously, in terms of renewable generation, this study focuses only on wind and solar power. The standard and Beta distributions were used to model load and solar generation forecast errors, respectively. The Farlie-Gumbel-Morgenstern bivariate probability density function was used to model wind production, which takes into account not only wind speed predictions but also wind direction predictions. Finally, the net demand was discretized using seven interval approximation technique.

Intra-zonal generators and demand response can be used to provide spinning reserve power under the market framework proposed in this study. The equivalent assisting unit method was used to model the interaction of interconnected grid networks. The main idea behind the equivalent assisting unit approach is to consider the interconnected power grid as an assisting power generating unit which can be at multiple states. Further, this multi-state unit can be incorporated into COPT of the system of interest.

The evaluation of the risks associated with each system state was done by determining the amount of undelivered electricity due to possible generator failure or significant reduction in renewable generation. This amount of unserved energy is represented by the index called the Expected Energy not Supplied which is also used to calculate possible socioeconomic loss caused by unexpected outage or significantly low renewable generation.

The stochastic unit commitment was modeled as a two-stage mixed-integer linear problem, with the first stage evaluating energy generation scenarios and the second stage evaluating the risk of load shedding. The maximum spinning reserve was found by minimizing both the spinning reserve running expense and the socioeconomic cost of load shedding at the same time.

Finally, the BN-based RAAA fine-tunes the reserve allocation process by adjusting intra-hour reserve allocation based on the information on actual realization of net demand and unit commitment, as well as other parameters such as day and hour type.

An updated version of the standard IEEE Two-Area RTS was used to test the proposed model. The aim of the case study was to see how participation in the ancillary service market of the inter-zonal conventional power generation units and demand response providers would impact the adequacy and economic efficiency of the Area A grid system. Furthermore, the case study evaluated the impact of the BN-based RAAA and the bivariate wind forecasting method on the overall reliability of the test system.

The system design with inter-zonal capacity and DR has shown the best results in terms of stability and system costs among the three considered network configurations. Because of their failure to cope with low-cost traditional units, based on the given cost of DR, the existence of DRPs did not impact reserve requirements during off-peak hours.

DR, on the other hand, had a favorable impact on system stability at peak hours by reducing the loss of load expectation. Further research revealed that lowering the DR marginal cost improved DRPs capability to compete with traditional units, resulting in DR reserves being added to the off-peak reserve schedule.

The integration of interconnected power generation capacity and DRPs into the ancillary service market reduced EENS and the net cost of spinning reserve provision, according to the findings presented in section 4.1. Of course, the value of spinning reserve generated by these entities influenced the extent of this reduction, but the fact that DRPs are more efficient (in a technological sense) in terms of providing up-spinning reserve than traditional units is another aspect that positively influenced overall network efficiency.

This model has shown that diversity in spinning reserve power has a positive impact on not only the efficiency of power grids, but also on overall social welfare.

Finally, the study revealed that by reducing the volatility associated with wind forecasts, the bivariate wind forecasting method decreased total reserve requirements and EENS in all three configurations.

Similarly, the proposed approach's state-of-the-art features, such as the bivariate wind prediction model and the BN-based RAAA, increased grid stability and lowered total reserve allocation cost. The use of bivariate PDF resulted in more reliable wind energy output predictions, lowering the total volatility associated with net demand scenarios. The use of the BN-based RAAA resulted in a cost-effective distribution of spinning reserves, which was achieved by taking into account the actual realization of net demand as well as the scheduling period's temporal characteristics. The adoption of these novelties resulted in average Expected Energy not Supplied and overall cost of reserves reductions of 2.66 percent and 1.12 percent, respectively.

5.2. Conventional Passenger Elevators

To begin, it's crucial to understand that successfully implementing a novel elevator control method, such as the algorithm suggested in this study, in reality would necessitate rethinking current elevator control practices. This knowledge is important for fine-tuning the elevator control algorithm. In order to generate effective control decisions, for an elevator group control system, it is important to perform tuning of the control system based on the set of fuzzy rules and data provided by the image acquisition and processing system.

A probabilistic EGC algorithm based on the closest car elevator dispatch strategy was introduced in this study. The information collected from the imaginary security cameras installed in the elevator halls and inside the elevator cars is used by the proposed algorithm for effective decision-making. The number of people standing in hallways or riding elevators, as well as their associated odds, are used by the elevator control system. This data is changed every second and is used to ensure that elevators are dispatched as quickly as possible.

The estimation of the figure of suitability and the determination of the closest elevator car are used to optimize elevator dispatching. The suggested algorithm dispatches elevator cars to gather the largest number of people from the most crowded floors. Furthermore, the algorithm considers each person's height and whether or not they are holding large-size personal belongings. The effective number of passengers is measured

using BNs, and this value is then used to maximize elevator dispatch. The proposed algorithm's target is to cut down on average travel, journey, and waiting times.

Initially, with the help of BayesiaLab tools [75], a BN model was created on a system consisting of a single elevator car to implement the proposed algorithm. In order to refresh the network with proof data, randomly selected scenarios were analyzed. The algorithm's decisions were then analyzed, and the probability distributions of BN variables were modified to improve decision-making. The algorithm matched the golden decisions 94 percent of the time after a few tweaks.

The algorithm was then put into operation on a 10-story office complex with four elevator cars. The results revealed that in contexts of limited traffic sizes, the proposed algorithm works the best (less than 200 people). In up-peak traffic conditions with high traffic volume and a large number of users, the proposed algorithm primarily underperforms. This could be explained by the fact that highly intense, large traffic significantly reduce the number of elevator dispatching scenarios. In other words, during highly intense traffic, the elevators' actions are limited to stopping on every floor to pick up or let out the passengers.

The proposed algorithm's best output was seen in situations with random inter-floor conditions. The average transit time for situations of varying traffic size and volume increased by 39.94 percent and 19.53 percent, respectively, according to the findings.

Comparison of the inter-floor traffic pattern findings obtained using the relevant samples Friedman's two-way analysis of variance by ranks revealed that among all the algorithms, ATT and AJT are distinct, while AWT is not.

The proposed algorithm has the following benefits:

- A graphical data processing model that is both clear and straightforward;
- Data with a high degree of complexity can be included and thoroughly investigated;
- Decision-making techniques can be tweaked to fit the preferences of the consumer;
- Since the algorithm's decision-making rules are not hard-coded, they can be tweaked or changed;
- Modifying the topology of the model or reassigning conditional probability of different variables would suffice to implement new elevator control laws.

The proposed algorithm has a number of drawbacks that are listed below:

- A sensitivity analysis is needed to identify factors that have a significant impact on final decisions;
- Implementers of the algorithm must be familiar not only with elevator control and dispatching, but also BNs and probabilistic inference in general.

The following are key aspects of the proposed algorithm's implementation:

- Fuzzy rule development was done in collaboration with field experts;
- Fuzzy rules were converted to numerical values using vector defuzzification and a three-stage algorithm tuning;
- The number of nodes that impact the utility node is held to a minimum.

Chapter 6 – Conclusion

The derivation of the conclusion for this PhD thesis requires the recapitulation of the research gap that I tried to cover by conducting this study. In this study I tried to find an answer for the following research question:

> ***Conventional operation and control strategies employed in power grids and passenger elevator systems can be improved, in terms of reliability and effectiveness, by implementation of probabilistic and machine learning algorithms based on the Bayesian inference.***

The results of this study showed that, indeed, the implementation of probabilistic methods based on the Bayesian inference can improve the reliability and effectiveness of the power grids and the passenger elevator systems. The following two sections present the conclusions relevant to improving the reliability and effectiveness of conventional power grids and passenger elevator systems.

6.1. Electric Grid Systems

In the modern world, society tends to favor ecologically friendly ways of power production. Considering the above, the conventional operation and control practices show their ineffectiveness and lead to the long-term problems that a power system would not be able to solve in a short time. For instance, the very recent blackout in the State of Texas, USA showed that the availability of system interconnection plays significant role in keeping the power grid stable. Other important measures that could be taken in order to improve the reliability of a power grid are presented below.

First of all, in most cases, introducing cutting-edge technology like smart-grid would necessitate the modernization of existing grid networks as well as the modification of current business structures. The introduction of a demand response scheme, on the other hand, would enable customers to compete as service providers in the electricity sector. In this situation, electric energy customers would operate as up-spinning backup suppliers, allowing security criteria to be relaxed and grid stability to improve.

Second, when we talk about developing countries like Kazakhstan, the installed capacity of intermittent renewable energy sources is negligible, but grid operators do not put a high enough priority on renewable forecasting accuracy. The accuracy of load and renewable forecasting frameworks affects stochasticity in power grids; hence, device operators can reduce the risks associated with load and renewable power by providing extremely reliable forecasting frameworks.

Finally, the proper integration of renewable energy into the power grid structure necessitates a rethinking of traditional organizational methods, including a strong focus on grid resilience. Increased transmission capability, energy storage, and maneuverable generation are the clearest ways to boost grid stability. These initiatives, though, necessitate major expenditures which require a long time to introduce. Instead, concentrating on less capital-intensive initiatives including reshaping operating processes will be more successful. For example, as this study showed, improved grid versatility can be accomplished by substituting probabilistic or hybrid security criteria for the traditional deterministic (N-1) reliability criterion.

6.2. Conventional Passenger Elevators

Conventional elevator control methods also show their ineffectiveness when we start introducing uncertainty, a big portion of uncertainty is usually caused by the passenger traffic. Most of the conventional elevator control algorithms simply do not take into account things like how many people are waiting for an elevator, their travel directions and whether or not they have bags or other belongings with them.

Most of the conventional vertical transportation systems were designed decades ago when the computer technology was not very well developed. At that time, limited computing capacity of the processers did not allow to implement sophisticated algorithms, such as the one presented in this study. Currently, the abundance of the video surveillance cameras in the buildings along with cheap and powerful computers allows the application of sophisticated algorithms that could improve the performance of passenger elevators. In fact, in my humble opinion, the implementation of such algorithms will be ubiquitous in the future.

Appendix A

Spinning Reserve Estimation Model

```
clear all
close all
Test_System;
RTS_24_bus_System;
no_units_int = 24;


B_int=[ 0.000017 0.000012 0.000007 -0.000001 -0.000005 -0.000002  0.000017 0.000012
0.000007 -0.000001 -0.000005 -0.000002
  0.000012 0.000014 0.000009 0.000001 -0.000006 -0.000001 0.000012 0.000014
0.000009 0.000001 -0.000006 -0.000001
  0.000007 0.000009 0.000031 0.000000 -0.000010 -0.000006 0.000007 0.000009
0.000031 0.000000 -0.000010 -0.000006
  -0.000001 0.000001 0.000000 0.000024 -0.000006 -0.000008 -0.000001 0.000001
0.000000 0.000024 -0.000006 -0.000008
  -0.000005 -0.000006 -0.000010 -0.000006 0.000129 -0.000002 -0.000005 -0.000006 -
0.000010 -0.000006 0.000129 -0.000002
  -0.000002 -0.000001 -0.000006 -0.000008 -0.000002 0.000150 -0.000002 -0.000001 -
0.000006 -0.000008 -0.000002 0.000150
  0.000017 0.000012 0.000007 -0.000001 -0.000005 -0.000002  0.000017 0.000012
0.000007 -0.000001 -0.000005 -0.000002
  0.000012 0.000014 0.000009 0.000001 -0.000006 -0.000001 0.000012 0.000014
0.000009 0.000001 -0.000006 -0.000001
  0.000007 0.000009 0.000031 0.000000 -0.000010 -0.000006 0.000007 0.000009
0.000031 0.000000 -0.000010 -0.000006
  -0.000001 0.000001 0.000000 0.000024 -0.000006 -0.000008 -0.000001 0.000001
0.000000 0.000024 -0.000006 -0.000008
  -0.000005 -0.000006 -0.000010 -0.000006 0.000129 -0.000002 -0.000005 -0.000006 -
0.000010 -0.000006 0.000129 -0.000002
 -0.000002 -0.000001 -0.000006 -0.000008 -0.000002 0.000150 -0.000002 -0.000001 -
0.000006 -0.000008 -0.000002 0.000150];


for k = 1:length(L)


P_res_9{k} = P_max_rts(9) - (P_rts{k}(9)+ Reserve_rts{k}(9));
P_res_10{k} = P_max_rts(10) - P_rts{k}(10)+ Reserve_rts{k}(10);
P_res_11{k} = P_max_rts(11) - P_rts{k}(11)+ Reserve_rts{k}(11);
P_res_int_rts{k} = P_res_9{k} + P_res_10{k} + P_res_11{k};
A_int{k} = (A_rts(9,1)*P_res_9{k}./P_res_int_rts{k} +
A_rts(10,1)*P_res_10{k}./P_res_int_rts{k} +
A_rts(11,1)*P_res_11{k}./P_res_int_rts{k});
alpha_int{k} = (alpha_rts(9,1)*P_res_9{k}./P_res_int_rts{k} +
alpha_rts(10,1)*P_res_10{k}./P_res_int_rts{k} +
alpha_rts(11,1)*P_res_11{k}./P_res_int_rts{k});
beta_int{k} = (beta_rts(9,1)*P_res_9{k}./P_res_int_rts{k} +
beta_rts(10,1)*P_res_10{k}./P_res_int_rts{k} +
beta_rts(11,1)*P_res_11{k}./P_res_int_rts{k});
```

```
gamma_int{k} = (gamma_rts(9,1)*P_res_9{k}./P_res_int_rts{k} +
gamma_rts(10,1)*P_res_10{k}./P_res_int_rts{k} +
gamma_rts(11,1)*P_res_11{k}./P_res_int_rts{k});
P_min_int = [P_min; 100];
P_res_int{k} = [P_res{k}; P_res_int_rts{k}];
end
for ii=1:length(L)
P_max_int{ii} = [P_max; P_res_int_rts{ii}];
A_int{ii} = [A; A_int{ii}];
alpha_int{ii} = [alpha; alpha_int{k}];
beta_int{ii} = [beta; beta_int{k}];
gamma_int{ii} = [gamma; gamma_int{k}];
COPT_int{ii} = Generator_COPT(no_units_int, P_max_int{ii}, A_int{ii});
COPT_int{ii} = COPT_int{ii}(COPT_int{ii}(:,3)>0.00000000000001,:); %Remove states
with low probability
CIS_int{ii} = COPT_int{ii}(:,1)';
CIS_int{ii} = CIS_int{ii}';
CIS_int_mtx = cell2mat(CIS_int);
for l = 1:length(CIS_int{ii}(:))
[ Reserve_int{ii,l}(:), CR_int{ii,l}(:) ] = LIUC( CIS_int{ii}(l), no_units_int,
gamma_int{ii}, beta_int{ii}, alpha_int{ii}, P_res_int{ii}, P_min_int, B_int );
Reserve_int{ii,l} = sum(Reserve_int{ii,l}); %Reserves in MW
CR_int{ii,l} = sum(CR_int{ii,l}); %Cost of Reserve
Diff{ii,l} = DA{ii}-CIS_int{ii}(l);
CE_ti_int{ii,l} = Diff{ii,l}.*intProb(:);
CE_int{ii,l} = sum(CE_ti_int{ii,l}(:));
N = CE_int{ii,l};
N(N < 0) = 0;
CE_int{ii,l} = N;
EENS_int{ii,l} = CE_int{ii,l}.*COPT_int{ii}(l,3);%Expected Energy Not Served in MW
end
EENS_int_mtx = cell2mat(EENS_int);
ECLS_int = EENS_int_mtx.*VOLL;
EENS_int_tot = sum(EENS_int_mtx,2); %Total EENS
ECLS_int_tot = EENS_int_tot'.*VOLL; %Expected Cost of Load Shedding in $


    end

%       P_res_mtx_rts = cell2mat(P_res_rts);
%       P_res_tot_rts = sum(P_res_mtx_rts);
 CR_int_mtx = cell2mat(CR_int);


C_res_ecls_int = ECLS_int + CR_int_mtx; %Total cost (Cost of Reserve and
Socioeconomic costs)
C_res_ecls_min_int = min(C_res_ecls_int, [], 2); %Find the minimum value
C_res_ecls_int_tot = sum(C_res_ecls_min_int);
temp = bsxfun(@eq,C_res_ecls_int, C_res_ecls_min_int);
temp = bsxfun(@times, temp, 1:size(C_res_ecls_int,2));
index_int = max(temp, [],2);
index_cell_int = num2cell(index_int);
```

```
 for m = 1:length(L)
    Res_opt_int{m} = Reserve_int{m,index_cell_int{m}};
    CR_pa_int{m} = CR_int{m,index_cell{m}};
    EENS_pa_int{m} = EENS{m,index_cell{m}};

 end

 Reserve_opt_int = cell2mat(Res_opt_int); %create a matrix representing amount of
reserve scheduled by probabilistic model
 CR_pa_int_tot = sum(cell2mat(CR_pa_int));
 EENS_pa_int_tot = sum(cell2mat(EENS_pa_int));


figure(1)
   plot(Reserve_opt_int)
   hold on
   plot (Reserve_nc)
   axis([1 24 0 800])
%     title('Reserved Capacity - Interconnected Test System')        % Add title and axis
labels
   xlabel('hour')
   ylabel('MW')
   legend('Isolated', 'Interconnected') % Add a legend

% clear close all
%===INPUT DATA FOR RTS 24 BUS
SYSTEM=====================================
format short g
sigma_int = 7; %Number of intervals
intProb = [0.006; 0.061; 0.242; 0.382; 0.242; 0.061; 0.006]; %seven-interval probability
%===Generator
Constraints================================================
P_max_rts = [152; 152; 350; 591; 60; 155; 155; 400; 400; 300; 310; 350]; % Maximum
generator capacities
P_min_rts = [10; 10; 10; 20; 0; 10; 10; 10; 20; 10; 10; 10]; % Minimum generator
capacities
P_min_res_rts = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
A_rts = [0.97; 0.90; 0.95; 0.96; 0.94; 0.93; 0.97; 0.90; 0.95; 0.96; 0.94; 0.93]; % Generator
outages
no_units_rts = numel(P_max_rts); %Number of generating units


%===Transmission Line Loss
Coefficients=============================================
B_int=[ 0.000017 0.000012 0.000007 -0.000001 -0.000005 -0.000002  0.000017 0.000012
0.000007 -0.000001 -0.000005 -0.000002
  0.000012 0.000014 0.000009 0.000001 -0.000006 -0.000001 0.000012 0.000014
0.000009 0.000001 -0.000006 -0.000001
  0.000007 0.000009 0.000031 0.000000 -0.000010 -0.000006 0.000007 0.000009
0.000031 0.000000 -0.000010 -0.000006
  -0.000001 0.000001 0.000000 0.000024 -0.000006 -0.000008 -0.000001 0.000001
0.000000 0.000024 -0.000006 -0.000008
```

```
  -0.000005 -0.000006 -0.000010 -0.000006 0.000129 -0.000002 -0.000005 -0.000006 -
0.000010 -0.000006 0.000129 -0.000002
  -0.000002 -0.000001 -0.000006 -0.000008 -0.000002 0.000150 -0.000002 -0.000001 -
0.000006 -0.000008 -0.000002 0.000150
  0.000017 0.000012 0.000007 -0.000001 -0.000005 -0.000002  0.000017 0.000012
0.000007 -0.000001 -0.000005 -0.000002
  0.000012 0.000014 0.000009 0.000001 -0.000006 -0.000001 0.000012 0.000014
0.000009 0.000001 -0.000006 -0.000001
  0.000007 0.000009 0.000031 0.000000 -0.000010 -0.000006 0.000007 0.000009
0.000031 0.000000 -0.000010 -0.000006
  -0.000001 0.000001 0.000000 0.000024 -0.000006 -0.000008 -0.000001 0.000001
0.000000 0.000024 -0.000006 -0.000008
  -0.000005 -0.000006 -0.000010 -0.000006 0.000129 -0.000002 -0.000005 -0.000006 -
0.000010 -0.000006 0.000129 -0.000002
 -0.000002 -0.000001 -0.000006 -0.000008 -0.000002 0.000150 -0.000002 -0.000001 -
0.000006 -0.000008 -0.000002 0.000150];

%===Generator Cost Function
Variables===================================
alpha_rts = [350; 350; 250; 400; 200; 300; 300; 400; 450; 300; 400; 400]; % cost function
coefficients
beta_rts = [10.2; 10.2; 7.5; 15.7; 5.8; 8.9; 8.9; 15.2; 15.5; 10.7; 11.8; 12.9];
gamma_rts = [0.06; 0.07; 0.011; 0.01; 0.008; 0.006; 0.06; 0.07; 0.015; 0.013; 0.01; 0.006];


%===Renewables=======================================================
==========
WP_max_rts = 400; %Installed capacity of wind power plants
W_err_max = 0.15; %Maximum wind forecast error


%===Load=============================================================
==========
L_rts = [1775.835 1669.815 1590.3 1563.795 1563.795 1590.3 1961.37 2279.43 2517.975
2544.48 2544.48 2517.975 2517.975 2517.975 2464.965 2464.965 2623.995 2650.5
2650.5 2544.48 2411.955 1899.915 1934.865 1669.815];
L_err_max = 0.1; %Maximum load forecast error
VOLL = 4000; % Value of Lost Load in $/MW


%===N-1 Security
Constraints=================================================
NC = 0.1; %N-1 criterion
Criterion_nc_rts = max(P_max_rts);


% clc, clear all,close all
% load('inputs.mat')
% opf=fopen('Results.doc','w+');
%===Lagrangian Iteration Unit
Committment=================================
P_rts = cell(length(L_rts),1);
C_rts = cell(length(L_rts),1);
```

```
for h = 1:length(L_rts)
[ P_rts{h}, C_rts{h} ] = LIUC( L_rts(h), no_units_rts, gamma_rts, beta_rts, alpha_rts,
P_max_rts, P_min_rts, B_rts);
totalCost_rts(h)=sum(C_rts{h});


Tariff_rts{h} = C_rts{h,:}.\P_rts{h}';


end


P_gen_rts=cell2mat(P_rts');


%===Risk Assessment
Model==================================================
COPT_rts = Generator_COPT(no_units_rts,P_max_rts,A_rts);
COPT_rts = COPT_rts(COPT_rts(:,3)>0.00000000001,:); %Remove states with low
probability
WP_min_rts = 0; %Minimum wind power output
W_rts = ((WP_max_rts - WP_min_rts).*rand(length(L_rts),1)/3 + WP_min_rts)'; %Wind
forecasting error
LRV_rts = arrayfun(@(x) linspace(-x,x,sigma_int),L_rts,'uni',false); % B is a cell array of
vectors
WRV_rts = arrayfun(@(x) linspace(-x,x,sigma_int),L_rts,'uni',false);
DRV_rts = arrayfun(@(x) linspace(-x,x,sigma_int),L_rts,'uni',false);
TSFE_rts = cell(length(L_rts),1); %Total System Forecast Error
LRV_rts = cell(length(L_rts),1); %Create a LFE range cell


for n = 1 : length(L_rts)
[LRV_rts{n}, LPDF{n}] = Normdist(0, L_rts(n)*0.025, -L_rts(n).*L_err_max,
L_rts(n).*L_err_max, sigma_int); %Create PDF of Load Forecast Error
[WRV_rts{n}, WPDF{n}] = Normdist(0, W_rts(n)*0.001, -W_rts(n).*W_err_max,
W_rts(n).*W_err_max, sigma_int); %Create PDF of Wind Forecast Error
D_rts(n) = L_rts(n) - W_rts(n); %Forecasted demand taking into account renewable power
sigma_d_rts(n) = sqrt((L_rts(n)*0.025)^2 + (W_rts(n)*0.001)^2); %Standard deviation of
demand error distribution
[DRV_rts{n}, DPDF_rts{n}] = Normdist(0, sigma_d_rts(n), (-L_rts(n)*L_err_max -
W_rts(n)*W_err_max), (L_rts(n)*L_err_max + W_rts(n).*W_err_max), sigma_int);
%Create PDF of Net Demand Error
DA_rts{n} = D_rts(n) + DRV_rts{n}; %Actual demand
R_rts{n} = sum(P_max_rts) - DA_rts{n}; %Maximum possiable reserve
DA_exp_rts(n) = mean(DA_rts{n});


% CIS = COPT(:,1)';
CIS_rts = flip(COPT_rts(:,1))'; %Capacity in service
for j = 1:length(CIS_rts)
CE_ti_rts{n,j} = (DA_rts{n}-CIS_rts(j)).*intProb(:);
```

```
CE_rts{n,j} = sum(CE_ti_rts{n,j}(:));
N = CE_rts{n,j};
N(N < 0) = 0;
CE_rts{n,j} = N;
CE_ti_nc_rts{n} = (DA_rts{n}-max(P_max_rts)).*intProb;
CE_nc_rts{n} = sum(CE_ti_nc_rts{n}(:));
NN = CE_nc_rts{n};
NN(NN < 0) = 0;
CE_nc_rts{n} = NN;
CE_nc_mtx_rts = cell2mat(CE_nc_rts);
EENS_nc_rts = CE_nc_mtx_rts.*2.87381606400001e-11;%Expected Energy Not Served
in MW
% CE{n,j} = DA_exp(n)-CIS(j); %Curtailed Energy
% M = CE{n,j};
% M(M < 0) = 0;
% CE{n,j} = M;
EENS_rts{n,j} = CE_rts{n,j}*(COPT_rts(j,3));%Expected Energy Not Served in MW
end
EENS_mtx_rts = cell2mat(EENS_rts);
ECLS_rts = EENS_mtx_rts.*VOLL;
EENS_tot_rts = sum(EENS_mtx_rts,2);
ECLS_tot_rts = EENS_tot_rts'.*VOLL; %Expected Cost of Load Shedding in $
end


for k = 1:length(L_rts)


P_res_rts{k} = P_max_rts - P_rts{k};
P_res_mtx_rts = cell2mat(P_res_rts);
P_res_tot_rts = sum(P_res_mtx_rts);
no_res(k) = sum(P_res_mtx_rts(:,k)~=0);
no_res_c = num2cell(no_res);

        for l = 1:length(CIS_rts)
           [ Reserve_rts{k,l}, CR_rts{k,l} ] = LIUC( CIS_rts(l), no_units_rts, gamma_rts,
beta_rts, alpha_rts, P_res_rts{k}, P_min_rts, B_rts );


            Reserve_rts_tot{k,l} = sum(Reserve_rts{k,l}); %Reserves in MW
            CR_rts{k,l} = sum(CR_rts{k,l}); %Cost of Reserve
        end

        CR_mtx_rts = cell2mat(CR_rts);
%           Reserve_nc = cell(size(L));
%           CR_nc = cell(size(L));
        [ Reserve_nc_rts{k}, CR_nc_rts{k} ] = LIUC( Criterion_nc_rts, no_units_rts,
gamma_rts, beta_rts, alpha_rts, P_res_rts{k}, P_min_res_rts, B_rts );
          Reserve_nc_rts{k} = sum(Reserve_nc_rts{k});

          CR_nc_rts{k} = sum(CR_nc_rts{k});

   end
```

```
        C_res_ecls_rts = ECLS_rts + CR_mtx_rts; %Total cost (Cost of Reserve and
Socioeconomic costs)
        C_res_ecls_min_rts = min(C_res_ecls_rts, [], 2); %Find the minimum value
        C_res_ecls_rts_tot = sum(C_res_ecls_min_rts);
        temp = bsxfun(@eq,C_res_ecls_rts, C_res_ecls_min_rts);
        temp = bsxfun(@times, temp, 1:size(C_res_ecls_rts,2));
        index_rts = max(temp, [],2);
        index_cell_rts = num2cell(index_rts);

 for m = 1:length(L_rts)
   Res_opt_rts{m} = Reserve_rts_tot{m,index_cell_rts{m}};
   CR_pa_rts{m} = CR_rts{m,index_cell_rts{m}};
   Reserve_pa_rts{m} = Reserve_rts_tot{m,index_cell_rts{m}};
   EENS_pa_rts{m} = EENS_rts{m,index_cell_rts{m}};
 end

CR_nc_tot_rts = sum(cell2mat(CR_nc_rts));
CR_pa_tot_rts = sum(cell2mat(CR_pa_rts));
Reserve_pamtx_rts = cell2mat(Reserve_pa_rts); %create a matrix representing amount of
reserve scheduled by probabilistic model
Reserve_ncmtx_rts = cell2mat(Reserve_nc_rts);
Reserve_nc_tot_rts = sum(Reserve_ncmtx_rts);
Reserve_pa_tot_rts = sum(Reserve_pamtx_rts);


function [ X, f ] = Normdist( mu, sigma, min_x, max_x, n )
%This function calculates the probability distribution of given variable
X = zeros(n,1);
f = zeros(n,1);
x = min_x;
dx = (max_x - min_x)/(n - 1);
for k = 1:n
   X(k) = x;
   f(k) = 1/(sqrt(2*pi)*sigma)*exp(-(x-mu)^2/(2*sigma^2));
   x = x+dx;
end
end

function COPT = Generator_COPT(G,PR,A)
format short g
X=ff2n(G);
InitiationMatrix=[zeros(1,2^G);zeros(1,2^G);ones(1,2^G);zeros(1,2^G)]; %initiation of
COPT matrix
GeneratorCOPTMatrixTemp=InitiationMatrix';
for j=1:2^G
for i=1:G
if (X(j,i)==0)
GeneratorCOPTMatrixTemp(j,1)=GeneratorCOPTMatrixTemp(j,1)+PR(i,1); %create
availability column
GeneratorCOPTMatrixTemp(j,3)=GeneratorCOPTMatrixTemp(j,3)*A(i,1); %create the
state probability column
```

```
else
GeneratorCOPTMatrixTemp(j,2)=GeneratorCOPTMatrixTemp(j,2)+PR(i,1); %create
unavailability column
GeneratorCOPTMatrixTemp(j,3)=GeneratorCOPTMatrixTemp(j,3)*(1-A(i,1)); %create
cumulative probability column
end
end
end
TemporaryMatrix=GeneratorCOPTMatrixTemp; %temporary matrix for COPT values
for m=1:(2^G)
for n=1:(2^G)
if(GeneratorCOPTMatrixTemp(m,1)==GeneratorCOPTMatrixTemp(n,1)&& m~=n &&
n>m)
GeneratorCOPTMatrixTemp(m,3)=GeneratorCOPTMatrixTemp(m,3)+GeneratorCOPTM
atrixTemp(n,3);
else end
end
end
for m=1:2^G
for n=1:2^G
if(GeneratorCOPTMatrixTemp(m,1)==GeneratorCOPTMatrixTemp(n,1) && m<n &&
m~=n && GeneratorCOPTMatrixTemp(m,1)~=0)
GeneratorCOPTMatrixTemp(n,:)=zeros;
else end
end
end
for m=1:1:((2^G)-1)
   for n=1:1:((2^G)-1)
   if (GeneratorCOPTMatrixTemp(n,1)<GeneratorCOPTMatrixTemp((n+1),1))
      temp1=GeneratorCOPTMatrixTemp(n,1);
      temp2=GeneratorCOPTMatrixTemp(n,2);
      temp3=GeneratorCOPTMatrixTemp(n,3);
      GeneratorCOPTMatrixTemp(n,1)=GeneratorCOPTMatrixTemp((n+1),1);
      GeneratorCOPTMatrixTemp(n,2)=GeneratorCOPTMatrixTemp((n+1),2);
      GeneratorCOPTMatrixTemp(n,3)=GeneratorCOPTMatrixTemp((n+1),3);
      GeneratorCOPTMatrixTemp((n+1),1)=temp1;
      GeneratorCOPTMatrixTemp((n+1),2)=temp2;
      GeneratorCOPTMatrixTemp((n+1),3)=temp3;
   end
   end
end
GeneratorCOPTMatrix=GeneratorCOPTMatrixTemp;
GeneratorCOPTMatrix(~any(GeneratorCOPTMatrixTemp,2),:)=[];
GeneratorCOPTMatrix;
c=length(GeneratorCOPTMatrix(:,1));
suma=0;
for i=c:-1:1
   suma=suma+GeneratorCOPTMatrix(i,3);
   GeneratorCOPTMatrix(i,4)=suma;
end
COPT = GeneratorCOPTMatrix;
COPT = COPT(COPT(:,3)>0.001,:);
```

```
end


function [ Pg, F ] = LIUC( Pd, no_units, a, b, c, Pmax, Pmin, B )
itermax=1000;
epsilon=0.1;
alpha=a; %a, b, c - alpha, beta and gamma coeficients of fuel cost function
clc
Pg=zeros(no_units,1);
lambda=7;
del_lambda=0.010;
tic;deltaP=10;iter=0;
EPd=Pd/no_units; %Pd - load


while abs(deltaP)>epsilon && iter< itermax
 iter=iter+1;
 for i=1:no_units
 sigma=B(i,:)*Pg-B(i,i)*Pg(i); %B - transmission loss coefficients
  Pg(i)=(1-(b(i)/lambda)-(2*sigma))/(alpha(i)/lambda+2*B(i,i)); %Pg - Committed
generator capacity
 if Pg(i)<Pmin(i)  %Pmax and Pmin - maximum and minimum generator outputs
 Pg(i)=Pmin(i);
 end
 if Pg(i)>Pmax(i)
 Pg(i)=Pmax(i);
 end
 end

  P_loss=Pg'*B*Pg;
 Pt=sum(Pg);
 deltaP=Pt-Pd-P_loss;
 error(iter)=deltaP;
 if deltaP>0
 lambda=lambda-del_lambda;
 end
 if deltaP<0
 lambda=lambda+del_lambda;
 end
end


Ft=0.0; %F - Cost of committed generator
for i=1:no_units
 F(i)=c(i)+b(i)*Pg(i)+a(i)*Pg(i)*Pg(i);


Ft=Ft+F(i);
end

end
```

Appendix B

The Python code of the proposed elevator croup control algorithm

```
import simpy
from collections import deque, defaultdict
from itertools import accumulate
from operator import itemgetter
from enum import Enum
import random
from glob import glob
import pandas as pd

Direction = Enum('Direction', 'IDLE DOWN UP') # possible elevator direction states


class Elevator:
    def __init__(self, capacity = 10, index = None, building = None, env = None):
        """ Each elevator has its
            building it belongs to,
            capacity,
            queue of floors to be visited next,
            list of passengers in the elevator,
            current direction (up, down or idle),
            whether it accepts up passengers,
            whether it accepts down passengers,
            current height in the shaft
        """
        self.index = index
        self.building = building
```

```
        self.capacity = capacity
        self.queue = deque()
        self.passengers = []
        self.direction = Direction.IDLE
        self.accept_up = False
        self.accept_down = False
        self.height = 0

        self.env = env

        # Only one process for traversal at a time HACK
        self.traverse_resource = simpy.Resource(env, capacity=1)

        # Events
        self.e_queue_updated = env.event()
        self.e_floor_arrived = env.event()
        self.e_floor_almost_arrived = env.event() # Useful for managing accept_up and
accept_down
        self.e_idle = env.event() # triggers on init
        self.e_passengers_loaded = env.event()

        # Adding default event callbacks
        env.process(self.__queue_updated())
        env.process(self.__floor_arrived())
        env.process(self.__idle())

    def passengers_weight(self):
        """Calculates weight of all passengers currently in the elevator"""
        weight = 0
```

```
    for passenger in self.passengers:
        weight += passenger.get_weight()
    return weight

def go(self, floor, now = False):
    """ Add a floor to the queue
        if now == True, adds floor to the beginning of the queue

    """
    # Check if it is a valid  floor
    if not floor in range(self.building.floors_n):
        return

    if now:
        self.queue.appendleft(floor)
    else:
        self.queue.append(floor)

    self.e_queue_updated.succeed()
    self.e_queue_updated = self.env.event()

    self.env.process(self.queue_traverse())

def queue_traverse(self):
    """
    """
    request = self.traverse_resource.request()
    yield request
```

```
        self.__movebreak = False # for breaking from loops
        while len(self.queue) > 0:
#           print(self.queue)
            # Get front element from queue
            target_floor = self.queue[0]
            # _move to the height of that element
            target_height = self.building.heights[target_floor]
            if target_height > self.height:
                self.direction = Direction.UP
            else:
                self.direction = Direction.DOWN
            yield self.env.process(self.__move(target_height))

            if self.height == target_height: # Arrived at floor
                self.e_floor_arrived.succeed((target_floor, self.index))
                self.e_floor_arrived = self.env.event()

            # if queue is updated break
            if self.__movebreak:
                break
            self.queue.popleft()
#           print("popped")
        # Done
        if not self.__movebreak:
            self.direction = Direction.IDLE
            # Invoke onIdle
            self.e_idle.succeed()
#           self.e_idle = self.env.event() # Done in callback
```

```
        self.traverse_resource.release(request)


    def __move(self, target):
        """Move the elevator car to target height"""


        dt = 0.5
        dh = 1


        # Useless move check,
        # should still trigger almost arrived
        if self.height == target:
            self.e_floor_almost_arrived.succeed((self, target))
            self.e_floor_almost_arrived = self.env.event()
            yield self.env.timeout(0.0000001) # Miniscule but still in the future, so that callbacks work in time


        while self.height != target:
            if self.direction == Direction.UP:
                if self.height + dh < target:
                    self.height += dh
                    if not self.height + dh < target:
                        # Almost arrived
                        self.e_floor_almost_arrived.succeed((self, target))
                        self.e_floor_almost_arrived = self.env.event()
                else:
                    self.height = target
            elif self.direction == Direction.DOWN:
                if self.height - dh > target:
```

```
                self.height -= dh
                if not self.height - dh > target:
                    # Almost arrived
                    self.e_floor_almost_arrived.succeed((self, target))
                    self.e_floor_almost_arrived = self.env.event()
            else:
                self.height = target
        yield self.env.timeout(dt)
        # check for break
        if self.__movebreak:
            break

    # -----------------------
    # Default event callbacks
    # -----------------------

    def __queue_updated(self):
        while True:
            yield self.e_queue_updated
            self.__movebreak = True
            self.env.process(self.queue_traverse())

    def __floor_arrived(self):
        while True:
            floor, index_ele = yield self.e_floor_arrived
            print("Elevator {} came to floor {} at {}".format(index_ele,floor + 1,
self.env.now))
#             print("Queue: {}, Floor: {}".format(self.queue,
building.floors[floor].passengers))
```

```
        # Unload passengers
        for passenger in self.passengers:
            if passenger.dest_floor == floor:
                passenger.arrive_time = self.env.now
                self.building.delivered.append(passenger)
                print("Passenger {} arrived at floor {}".format(passenger.index, floor + 1))
        self.passengers = [p for p in self.passengers if not p.dest_floor == floor]

        # Load passengers

        if self.accept_down: # Get down passengers
            while len(self.building.floors[floor].passengers['down']) > 0 and self.capacity - self.passengers_weight() >= self.building.floors[floor].passengers['down'][0].get_weight():
                passenger = self.building.floors[floor].passengers['down'].popleft()
                passenger.load_time = self.env.now
                self.passengers.append(passenger)
                print("Loaded passenger, their weight: {}, total passengers weight: {}".format(passenger.get_weight(), self.passengers_weight()))
        if len(self.building.floors[floor].passengers['down']) == 0:
            self.building.floors[floor].buttons['down'] = False

        if self.accept_up: # Get up passengers
            while len(self.building.floors[floor].passengers['up']) > 0 and self.capacity - self.passengers_weight() >= self.building.floors[floor].passengers['up'][0].get_weight():
                passenger = self.building.floors[floor].passengers['up'].popleft()
                passenger.load_time = self.env.now
                self.passengers.append(passenger)
                print("Loaded passenger, their weight: {}, total passengers weight: {}".format(passenger.get_weight(), self.passengers_weight()))
```

```
            if len(self.building.floors[floor].passengers['up']) == 0:
                self.building.floors[floor].buttons['up'] = False

            self.e_passengers_loaded.succeed(floor)
            self.e_passengers_loaded = self.env.event()

    def __idle(self):
        self.e_idle.succeed()
#         self.e_idle = self.env.event()
        print("{} started __idle at {}".format(self.index, self.env.now))
        while True:
            yield self.e_idle
            self.e_idle = self.env.event()
            print("Elevator {} Idle".format(self.index))

class Floor:
    def __init__(self, index=None, height=None, env=None):
        """ Each floor has two buttons (up, down)
                each button can be either pressed(True) or not(False),
            passengers waiting for the elevator
                that want to go either up or down,
            height of the floor
        """
        self.index = index

        self.buttons = {'up': False, 'down' : False}
        self.passengers = {'up' : deque(), 'down' : deque()}
        self.height = height
```

```
        self.env = env

        # Events
        self.e_button_pressed = env.event()

        # Event callbacks
        env.process(self.__button_pressed())

    def __button_pressed(self):
        while True:
            floor, direction = yield self.e_button_pressed
            print("The {} button is pressed on floor {}!".format(direction, floor + 1))
            # code

    def press_button(self, passenger):
        """
        """
        d = '' # Direction
        if passenger.entry_floor > passenger.dest_floor:
            d = 'down'
        else:
            d = 'up'
        self.passengers[d].append(passenger)
        self.buttons[d] = True

        # trigger event - button pressed
        self.e_button_pressed.succeed((passenger.entry_floor, d))
```

```
        self.e_button_pressed = self.env.event()


class Passenger:
    """ Passengers have their building,
        entry floor,
        destination floor
    """
    def __init__(self, index=None, building=None, entry_floor=None, dest_floor=None,
env=None, is_child=False, item_weight=0):
        self.index = index
        self.building = building
        self.entry_floor = entry_floor
        self.dest_floor = dest_floor

        self.env = env

        # Timing:
        self.begin_time = env.now
        self.load_time = None
        self.arrive_time = None

        # Is child?
        self.is_child = is_child

        # Weight of the item passenger is holding
        self.item_weight = item_weight

    def register(self):
        """Place the passenger at their entry floor and call the elevator"""
```

```
        self.building.floors[self.entry_floor].press_button(self)


    def get_weight(self):
        if self.is_child:
            return 0.5 + self.item_weight
        else:
            return 1 + self.item_weight


class Building:
    def __init__(self, floors=4, elevators=2, heights = None, env=None):
        """ Buildings have floors and elevators,
            floors have heights: [3,5,4] means first floor of height 3, second of 5, etc.
        """
        # Floors:
        DEFAULT_FLOOR_HEIGHT = 3
        self.floors_n = floors
        if heights is None:
            heights = [DEFAULT_FLOOR_HEIGHT for x in range(floors)]
        heights.insert(0,0)
        self.heights = list(accumulate(heights))
        self.floors = [Floor(index=floor_i, height=heights[floor_i], env = env) for floor_i in
range(floors)]


        # Elevators:
        self.elevators_n = elevators
        self.elevators = [Elevator(index = elevator_i, building = self, env = env) for elevator_i
in range(elevators)]


        # Data (for metric calculation):
```

```
        self.delivered = []


    def draw(self):
        # Find nearest floors for all elevators
        positions = []
        for elevator_n in range(self.elevators_n):
            distances = [abs(x - self.elevators[elevator_n].height) for x in self.heights]
            position = min(enumerate(distances), key=itemgetter(1))[0]
            positions.append(position)


        for floor_n in reversed(range(self.floors_n)):
            print("{}| up: {}, down: {}".format(floor_n+1,
len(self.floors[floor_n].passengers['up']), len(self.floors[floor_n].passengers['down'])),
end='')
            print("  ", end='')
            for elevator_n in range(self.elevators_n):
                icon = '*'
                if positions[elevator_n] == floor_n: # Elevator at this floor
                    if self.elevators[elevator_n].direction == Direction.UP:
                        icon = '↑'
                    elif self.elevators[elevator_n].direction == Direction.DOWN:
                        icon = '↓'
                    else:
                        icon = 'x'
                    icon += str(len(self.elevators[elevator_n].passengers))
                else:
                    icon = '  '
                print('|{}|'.format(icon), end='')
            print()
```

```
def to_dict(entry):
    return {
        'begin_time': entry.begin_time,
        'load_time': entry.load_time,
        'arrive_time': entry.arrive_time,
        'tt': entry.arrive_time - entry.load_time,
        'jt': entry.arrive_time - entry.begin_time,
        'wt': entry.load_time - entry.begin_time,
    }


def getFS(call, elevator, heights):
    """Returns FS for a call-elevator pair
     N is the height of the highest floor, this was done to support
        variable floor heights.

        Calls are of a format (floor, direction)

        Heights - just put building.heights in there.
    """
    d = abs(heights[call[0]] - elevator.height)
    N = heights[–2]

    if elevator.direction == Direction.IDLE:
        return N + 1 - d
    elif elevator.direction == Direction.DOWN:
```

```
        if building.heights[call[0]] > elevator.height: # Call is above
            return 1
        elif call[1] == 'down': # Direction same as elevator
            return N + 2 - d
        elif call[1] == 'up': # Direction opposite to elevator
            return N + 1 - d
    elif elevator.direction == Direction.UP:
        if building.heights[call[0]] < elevator.height: # Call is above
            return 1
        elif call[1] == 'up': # Direction same as elevator
            return N + 2 - d
        elif call[1] == 'down': # Direction opposite to elevator
            return N + 1 - d
import random

def generate_scenario(env, passenger_number=200, filename="scenario", spawn_rate=0.1,
floor_n=10, traffic="random"):
    """
        New passengers are created roughly each spawn_rate seconds, with three types of
traffic,
        "random", "up", or "down". Down - everyone is trying to get to floor 0, Up - everyone
is
        arriving to floor 0 and going up.
    """
    generated_data = [] # List of dicts


    for i in range(passenger_number):
        # Is it a child? p = 0.3
        is_child = random.random() < 0.3
```

```
        # How much weight do they hold? p = 0.5
        weight = 0
        if random.random() < 0.5:
            if random.random() < 0.5:
                weight = 0.5 # small weight
            else:
                weight = 1 # big weight
        yield env.timeout(random.expovariate(1 / spawn_rate))
        time = env.now
        floors = set(range(floor_n))
        if traffic == "random":
            entry_floor = random.choice(tuple(floors))
            floors.remove(entry_floor)
            dest_floor = random.choice(tuple(floors))
            generated_data.append({'time': time,'entry_floor': entry_floor, 'dest_floor': dest_floor, 'is_child': is_child, 'holds_weight': weight})
        else:
            floors.remove(0)
            if traffic == "up":
                generated_data.append({'time': time,'entry_floor': 0, 'dest_floor': random.choice(tuple(floors)), 'is_child': is_child, 'holds_weight': weight})
            elif traffic == "down":
                generated_data.append({'time': time,'entry_floor': random.choice(tuple(floors)), 'dest_floor': 0, 'is_child': is_child, 'holds_weight': weight})
    pd.DataFrame(generated_data).to_csv(filename, index=False)


for traffic in ['random', 'up', 'down']:
    for spawn_rate in [0.01, 0.1, 2]:
        for p_number in [100, 200, 300]:
```

```
        env = simpy.Environment()

        env.process(generate_scenario(env, passenger_number=p_number,
spawn_rate=spawn_rate, traffic=traffic,

                                      filename='scenarios_final/{}floor_t{}_sr{}_p{}.csv'.format(10,
traffic, spawn_rate, p_number)))

        env.run()


def passenger_arrivals(env, arrivals, building):

    for n, t in enumerate(arrivals['time']):

        if n == 0:

            yield(env.timeout(t))

        else:

            yield(env.timeout(t - arrivals['time'][n-1]))

        # Add passengers here!

        entry_floor = arrivals['entry_floor'][n]

        dest_floor = arrivals['dest_floor'][n]

        print("Passenger {} arrived at {}s going from {} to {}".format(n, env.now,
entry_floor+1, dest_floor+1))

        Passenger(index=n, building=building, entry_floor=entry_floor,
dest_floor=dest_floor, env=env).register()


def passenger_arrivals_BN(env, arrivals, building):

    for n, t in enumerate(arrivals['time']):

        if n == 0:

            yield(env.timeout(t))

        else:

            yield(env.timeout(t - arrivals['time'][n-1]))

        # Add passengers here!

        entry_floor = arrivals['entry_floor'][n]

        dest_floor = arrivals['dest_floor'][n]
```

```
        is_child = arrivals['is_child'][n]
        holds_weight = arrivals['holds_weight'][n]
        Passenger(index=n, building=building, entry_floor=entry_floor,
dest_floor=dest_floor, env=env,
                  is_child=is_child, item_weight=holds_weight).register()
        print("Passenger {} arrived at {}s going from {} to {}".format(n, env.now,
entry_floor+1, dest_floor+1))


def get_buttonpress(env, building):
    global calls
    global val
    while True:
        button_events = [floor.e_button_pressed for floor in building.floors]
        a = AnyOf(env, button_events)
        val = yield a
        for e in list(val):
            calls.add(e.value)
            target_floor = e.value[0]
def end_calls(env, building):
    global calls
    global calls_assigned
    while True:
        passengers_loaded = [elevator.e_passengers_loaded for elevator in building.elevators]
        a = AnyOf(env, passengers_loaded)
        val = yield a
        answered = set() # set of answered calls
        for e in list(val):
            floor = e.value
            # Check if this floor's calls are satisfied
```

```
            for call in calls:
                if call[0] == floor:
                    direction = call[1]
                    if len(building.floors[floor].passengers[direction]) == 0:
                        print("Call {} answered".format(call))
                        answered.add(call)
                        if call in calls_assigned:
                            del calls_assigned[call]
        calls = calls - answered
def assign_elevators(env, building):
    global calls
    global calls_assigned
    global ele_assigned
    while True:
        for call in calls: # indexes 0 is floor: (0, 1, ...), 1 is direction: ('up', 'down')
            # Select not fully loaded elevators
            available_elevators = []
            for elevator in building.elevators:
                if elevator.capacity > len(elevator.passengers):
                    available_elevators.append(elevator)

            # If all are loaded, calculate FS for all
            if len(available_elevators) == 0:
                available_elevators = building.elevators

            FS = 1
            selected_car = available_elevators[0]
            selected_d = 10000 # big number
```

```
            for elevator in available_elevators:
                newFS = getFS(call, elevator, building.heights)
                d = abs(building.heights[call[0]] - elevator.height)
                if newFS > FS or (newFS == FS and d < selected_d): # Pick highest FS or nearest if same FS
                    selected_car = elevator
                    FS = newFS
                    selected_d = d

            # Save the elevator for the call!
#               print("For call {}, best elevator is {} with FS = {}".format(call, selected_car.index, FS))
            calls_assigned[call] = selected_car

        # Change elevator queues to reflect assigned calls
        ele_assigned = {} # Reverse of calls_assigned
        for key, value in calls_assigned.items():
            ele_assigned.setdefault(value, list()).append(key)

        for elevator in building.elevators:
            # Determine the floors we need to go to
            need_to_go = set()
            # Combination of calls we need to answer
            if elevator in ele_assigned:
                for req in ele_assigned[elevator]:
                    need_to_go.add(req[0])
            # And floors pressed by passengers
            need_to_go.update([x.dest_floor for x in elevator.passengers])
```

```
    # Divide sorted list
    # Based on our current direction and position
    need_to_go = sorted(list(need_to_go))

    distances = [x - elevator.height for x in building.heights]

    lower = []
    higher = []
    if len(need_to_go) > 0:
        elevator.queue = deque() # Rebuild queue
    for floor in need_to_go:
        if distances[floor] > 0: # Higher than elevator
            higher.append(floor)
        elif distances[floor] == 0:
            elevator.go(floor)
        else:
            lower.append(floor)

    lower.reverse()
    if elevator.direction == Direction.UP:
        need_to_go = [*higher, *lower]
    else:
        need_to_go = [*lower, *higher]
    for floor in need_to_go:
        elevator.go(floor)

yield env.timeout(1) # Reassign elevators every 1 second,
```

```
for scenario_index in range(30):
    scenario_name = glob("scenarios_final/10f**")[scenario_index]
    df = pd.read_csv(scenario_name)

    env = simpy.Environment()
    building = Building(floors=10, elevators=5, env=env)

    env.process(passenger_arrivals(env, df, building))

    env.process(get_buttonpress(env, building))

#     env.process(direction_manage(env, building))
    # env.process(passenger_arrivals(env))
    val = ''
    calls = set() # List of all active calls
    calls_assigned = {} # call: elevator
    ele_assigned = {} # elevator: call

    env.process(end_calls(env,building))
    env.process(assign_elevators(env,building))

    # TODO: Add logic to this:
    for elevator in building.elevators:
        elevator.accept_up = True
        elevator.accept_down = True

    env.run(until = 3000)
```

```
    metrics = pd.DataFrame.from_records([to_dict(passenger) for passenger in
building.delivered])
    att = metrics['tt'].mean()
    ajt = metrics['jt'].mean()
    awt = metrics['wt'].mean()
    results.append({
        'scenario_name': scenario_name,
        'algo': 'Nearest Car',
        'att': att,
        'ajt': ajt,
        'awt': awt,
    })


def assign_elevators(env, building):
    global calls
    global calls_assigned
    global ele_assigned
    N = building.floors_n
    while True:
        for call in calls: # indexes 0 is floor: (0, 1, ...), 1 is direction: ('up', 'down')
            # Selected_cars priorities:
            # 1st idle cars, 2nd cars that are moving towards the landing floor


            # Get total number of passengers on the landing floor:
            passenger_number =
len(building.floors[call[0]].passengers['up'])+len(building.floors[call[0]].passengers['down'
])
#             passenger_number = 0 # Switch between old and new version is done here:
#             for u_p in building.floors[call[0]].passengers['up']:
```

```
#                passenger_number += u_p.get_weight()
#            for d_p in building.floors[call[0]].passengers['down']:
#                passenger_number += d_p.get_weight()

        selected_cars = []

        elevator_fs_d = [(elevator, getFS(call, elevator, building.heights),
                          abs(building.heights[call[0]] - elevator.height)) for elevator in
building.elevators]
        elevator_fs_d.sort(key = lambda x: (x[1] * -1, x[2])) # Sort first by FS first, by d
second

#            print("For call {}, best elevator is {} with FS = {}".format(call,
elevator_fs_d[0][0].index, elevator_fs_d[0][1]))

        for car, _, _ in elevator_fs_d:
            if car.capacity - car.passengers_weight() < 1: # Skip full elevators
                print("SKIPPED, cap: {}, cur: {}".format(car.capacity, len(car.passengers)))
                continue
            if passenger_number < 1:
                break
            selected_cars.append(car)
            passenger_number -= car.capacity - len(car.passengers) # Also switch here

        calls_assigned[call] = selected_cars

#        print(calls_assigned)
      ele_assigned = defaultdict(list)
```

```
for key, values in calls_assigned.items():
    for value in values:
        ele_assigned[value].append(key)

for elevator in building.elevators:
    # Determine the floors we need to go to
    need_to_go = set()
    # Combination of calls we need to answer
    if elevator in ele_assigned:
        for req in ele_assigned[elevator]:
            need_to_go.add(req[0])
    # And floors pressed by passengers
    need_to_go.update([x.dest_floor for x in elevator.passengers])

    # Divide sorted list
    # Based on our current direction and position
    need_to_go = sorted(list(need_to_go))

    distances = [x - elevator.height for x in building.heights]

    lower = []
    higher = []
    if len(need_to_go) > 0:
        elevator.queue = deque() # Rebuild queue
    for floor in need_to_go:
        if distances[floor] > 0: # Higher than elevator
            higher.append(floor)
        elif distances[floor] == 0:
```

```
                elevator.go(floor)
            else:
                lower.append(floor)

        lower.reverse()
        if elevator.direction == Direction.UP:
            need_to_go = [*higher, *lower]
        else:
            need_to_go = [*lower, *higher]
        for floor in need_to_go:
            elevator.go(floor)

    yield env.timeout(1) # Reassign elevators every 1 second, TODO: should probably change this


for scenario_index in range(30):
    scenario_name = glob("scenarios_final/10f**")[scenario_index]
    df = pd.read_csv(scenario_name)

    env = simpy.Environment()
    building = Building(floors=10, elevators=5, env=env)
    env.process(direction_manage(env, building)) # In the upper cell
#     env.process(passenger_arrivals(env, df, building)) # In the upper cell!
    env.process(passenger_arrivals_BN(env, df, building)) # BN version that takes being a child and cargo weight in account
    # env.process(passenger_arrivals(env))
    val = ''
    calls = set() # List of all active calls
    calls_assigned = {} # call: elevators
```

```
    ele_assigned = {} # ele: call


    env.process(get_buttonpress(env, building))

    env.process(end_calls(env,building))

    env.process(assign_elevators(env,building))

    # TODO: Add logic to this:

    for elevator in building.elevators:

        elevator.accept_up = True

        elevator.accept_down = True

    env.run(until = 3000)

    metrics = pd.DataFrame.from_records([to_dict(passenger) for passenger in
building.delivered])

    att = metrics['tt'].mean()

    ajt = metrics['jt'].mean()

    awt = metrics['wt'].mean()

    results.append({

        'scenario_name': scenario_name,

        'algo': 'Many Nearest Cars',

        'att': att,

        'ajt': ajt,

        'awt': awt,

    })
```

Bibliography

[1] A. Whiteman, J. Esparrago and S. Elsayed, "Renewable Energy Statistics," The International Renewable Energy Agency, Abu Dhabi, 2018.

[2] IEA, "International Energy Agency," International Energy Agency, 14 November 2017. [Online]. Available: https://www.iea.org/weo2017/. [Accessed 15 April 2018].

[3] J. M. Morales, A. J. Conejo, H. Madsen, P. Pinson and M. Zugno, Integrating renewables in electricity markets: operational problems, vol. 205, New York: Springer, 2014.

[4] R. Billinton and N. R. Allan, Reliability Evaluation of Power Systems, New York: Plenum, 1996.

[5] United Nations Organization, "World Urbanization Prospects: The 2018 Revision (Key Facts)," UN, New York, 2019.

[6] S. F. Smith, G. J. Barlow, X.-F. Xie and Z. B. Rubinstein, "SURTRAC: Scalable Urban Traffic Control," in *TRB 2013 Annual Meeting*, 2013.

[7] Rapid Flow Technologies LLC, "RAPID FLOW BLOG," Rapid Flow Technologies LLC, 30 August 2018. [Online]. Available: https://www.rapidflowtech.com/blog/surtrac-deployment-at-urban-grid-networks-in-pittsburgh-neighborhoods. [Accessed 5 August 2020].

[8] F. Bouffard and F. Galiana, "An electricity market with a probabilistic spinning reserve criterion," vol. 19, no. 1, p. 300–307, 2004.

[9] D. Chattopadhyay and R. Baldick, "Unit commitment with probabilistic reserve," in *IEEE Power Engineering Society Winter Meeting*, New York, NY, USA, 2002.

[10] M. Ortega-Vazquez and D. Kirschen, "Estimating the spinning reserve requirements in systems with significant wind power generation penetration," *IEEE Trans. Power Syst*, vol. 24, no. 1, pp. 114-124, Feb.2009.

[11] G. Liu and K. Tomsovic, "Quantifying Spinning Reserve in Systems With Significant Wind Power Penetration," *IEEE Trans. Power Syst*, vol. 27, no. 4, p. 2385–2393, Nov. 2012.

[12] Y. Bapin, M. Mussard and M. Bagheri, "Estimation of Operating Reserve Capacity in Interconnected Systems with Variable Renewable Energy Using Probabilistic Approach," in *18th Conference on Probabilistic Methods Applied to Power Systems*, Boise, ID, USA, 2018.

[13] R. B. Burke and M. I. Henderson, "Incorporating Demand Response In Operating Reserve In New England," in *IEEE Power Engineering Society General Meeting*, Piscataway, NJ, USA, 2005.

[14] E. Shayesteh, A. Yousefi, F. Daneshvar and M. Parsa Moghaddam, "An Approach for Improving Spinning Reserve Capacity by Means of Optimal Utilization of DR Program," in *2nd IEEE International Conference on Power and Energy*, Johor Bahrau, Malasia, 2008.

[15] M. Parvania and M. Fotuhi-Firuzabad, "Demand response scheduling by stochastic SCUC,"

5] *IEEE Trans. Smart Grid,* vol. 1, no. 1, p. 89–98, Jun 2010.

[1 M. A. Khorsand, H. Heydari and A. Zakariazadeh, "Interruptible Load Participation as
6] Operating Reserve in Joint Energy and Spinning Reserve markets Using Stochastic Security Analysis," in *2010 The 2nd International Conference on Computer and Automation Engineering*, Singapore, Feb 2010.

[1 E. Karangelos and F. Bouffard, "Towards full integration of demand-side resourcesin joint
7] forward energy/reserve electricity markets," *IEEE Trans. Power Syst*, vol. 27, no. 1, p. 280–289, 2012.

[1 F. Bouffard, F. D. Galiana and A. J. Conejo, "Market-clearing with stochastic security—Part I:
8] Formulation," *IEEE Trans. Power Syst*, vol. 20, no. 4, p. 1818–1826, Nov. 2005.

[1 F. Bouffard, Galiana, D. F and A. J. Conejo, "Market-clearing with stochastic security—Part II:
9] Case studies," *IEEE Trans. Power Syst*, vol. 20, no. 4, p. 1827–1835, Nov. 2005.

[2 G. Liu and K. Tomsovic, "A Full Demand Response Model in Co-optimized Energy and
0] Reserve Market," *Electric Power Systems Research,* vol. 111, pp. 62-70, 2014.

[2 J. Olamaei, M. Khademi and A. Arjomand, "Model to Incorporate Demand Response Programs
1] in Security Constrained Unit Commitment," in *IEOM 2015 - 5th International Conference on Industrial Engineering and Operations Management, Proceeding*, Dubai, UAE, 2015.

[2 H. Darvish, A. Darvishi and H. Hejazi, "Integration of Demand Side Management in Security
2] Constrained Energy and Reserve Market," in *2015 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference*, 2015.

[2 H. Abdi, E. Dehnavi and F. Mohammadi, "Dynamic Economic Dispatch Problem Integrated
3] With Demand Response (DEDDR) Considering Non-Linear Responsive Load Models," *IEEE Trans. Smart Grid,* vol. 7, no. 6, pp. 2586-2595, Nov 2016.

[2 V. Babu, V. Madhusudan and V. Ganesh, "TSS based Composite Power System Reliability
4] Evaluation Considering Demand Response," *International Journal of Latest Technology in Engineering, Management & Applied Science,* vol. 6, no. 6, pp. 150-154, 2017.

[2 A. M. L. D. Silva, J. F. D. C. Castro and R. Billinton, "Probabilistic Assessment of Spinning
5] Reserve via Cross-Entropy Method Considering Renewable Sources and Transmission Restrictions," *IEEE Transactions on Power Systems*, vol. 33, no. 4, p. 4574–4582, 2018.

[2 E. Shayesteh, M. Amelin and L. Soder, "[24] E. ShArea equivalents for spinning reserve
6] determination in interconnected power systems," *Energy,* vol. 88, pp. 907–916,, 2015.

[2 S. S. Reddy, P. R. Bijwe and A. R. Abhyankar, "Joint Energy and Spinning Reserve Market
7] Clearing Incorporating Wind Power and Load Forecast Uncertainties," *IEEE Systems Journal*, vol. 9, no. 1, p. 152–164, 2015.

[2 N. Padmanabhan, M. Ahmed and K. Bhattacharya, "Simultaneous Procurement of Demand
8] Response Provisions in Energy and Spinning Reserve Markets," *[26] N. Padmanabhan, M. Ahmed, and K. Bhattacharya, "Simultaneous Procurement of Demand Response Provisions in E*

IEEE Transactions on Power Systems, vol. 33, no. 5, p. 4667–4682, 2018.

[29] Y. Bapin and V. Zarikas, "Smart building's elevator with intelligent control algorithm based on Bayesian networks," . *International Journal of Advanced Computer Science and Applications,* vol. 10, no. 2, pp. 16-24, 2019.

[30] A. Amrin, V. Zarikas and C. Spitas, "Reliability analysis and functional design using Bayesian networks generated automatically by an “Idea Algebra” framework," *Reliability Engineering and System Safety,* vol. 180, pp. 211-225, 2018.

[31] V. Zarikas, E. Papageorgiou, D. Pernebayeva and N. Tursynbek, "Medical decision support tool from a fuzzy-rules driven Bayesian network.," in *10th International Conference on Agents and Artificial Intelligence*, Funchal, Portug, 2018.

[32] D. C. Yu, T. Nguyen and P. Haddawy, "Bayesian Network Model for Reliability Assessment of Power Systems," *IEEE Transactions of Power Systems,* vol. 14, no. 2, pp. 426-432, 1999.

[33] Z. Yongli, H. Limin, Z. Liguo and W. Yan, "Bayesian Network Based Time-sequence Simulation for Power System Reliability Assessment," in *Seventh Mexican International Conference on Artificial Intelligence* , Atizapan de Zaragoza, Mexico, 2008.

[34] A. Ebrahimi and T. A. Daemi, "Novel Method for Constructing the Bayesian Network for Detailed Reliability Assessment of Power Systems," in *International Conference on Electric Power and Energy Conversion Systems*, Sharjah, United Arab Emirates, 2009.

[35] A. Shofield, T. Stonham and P. Mehta, "Automated people counting to aid lift control," *Automation in Construction,* vol. 6, pp. 437-445, 1997.

[36] H. Shao, L. Li, P. Xiao and M. K. Leung, "ELEVIEF: An Active Elevator Video Surveillance System," in *Workshop on Human Motion*, Austin, Texas, USA, 2000.

[37] Y. Lee, T. Song, H. Kim, D. K. Han and H. Ko, "Hostile Intent and Behaviour Detection in Elevators," in *4th International Conference on Imaging for Crime Detection and Prevention 2011 (ICDP 2011)*, London, UK, 2011.

[38] J. Zou and Q. Zhao, "Occupancy Detection in Elevator Car by Fusing Analysis of Dual Videos," in *13th IEEE Conference on Automation Science and Engineering (CASE)*, Xi'an, China, 2017.

[39] Z. Sun, B. Xu, D. Wu, M. Lu and J. Cong, "A real-time video surveillance and state detection approach for elevator cabs," in *International Conference on Control, Automation and Information Sciences (ICCAIS)*, Chengdu, China, 2019.

[40] N. Ding, P. B. Luh, T. Chen and H. Zhang, "Optimization of Elevator Evacuation Considering Potential Over-Crowding," in *11th World Congress on Intelligent Control and Automation*, Shenyang, China, 2014.

[41] F. Mohamudally, C. S. Inn, L. S. Yeong and C. W. Chong, "Estimating Free Space in an Elevator’s Confined Environment," in *TENCON, IEEE Region 10 International Conference*, Macao, China, 2015.

[42] J.-H. Kim and B.-R. Moon, "Adaptive Elevator Group Control with Cameras," *TRANSACTIONS*

2] *ON INDUSTRIAL ELECTRONICS,* vol. 48, no. 2, pp. 377-382, 2001.

[4 Z. Zhang, Y. Zheng, H. Xu and H. Li, "A novel elevator group control algorithm based on
3] binocular-cameras corridor passenger detection and tracking," *Multimedia Tools and Applications,* vol. 74, p. 1761–1775, 2015.

[4 S.-Y. Chou, A. B. D., A. Dewabharata and F. E. Zulvia, "Improving Elevator Dynamic Control
4] Policies Based on Energy and Demand Visibility," in *International Conference on Intelligent Green Building and Smart Grid (IGBSG)*, Yi-Lan, Taiwan, 2018.

[4 A. Kyritsis, D. Voglitsis, N. Papanikolaou, S. Tselepis, C. Christodoulou, I. Gonos and S.
5] Kalogirou, "Evolution of PV systems in Greece and review of applicable solutions for higher penetration levels," *Renewable Energy,* vol. 2017, p. 487–499.

[4 T. Tsiftsis, N. Papanikolaou, M. Loupis and V. Zarikas, "On the Application of Cooperative
6] Communications in Renewable Energy Sources for Maximizing the Reliability of Power Distribution Networks," *Journal of Green Engineering,* vol. 3, p. 403–420, 2013.

[4 M. Mazidi, A. Zakariazadeh, S. Jadid and P. Siano, " Integrated scheduling of renewable
7] generation and demand response programs in a microgrid," *Energy Conversion and Management,* vol. 86, p. 1118–1127, 2014.

[4 Y. Li, Z. Yang, G. Li, D. Zhao and W. Tian, "Optimal Scheduling of an Isolated Microgrid With
8] Battery Storage Considering Load and Renewable Generation Uncertainties," *IEEE Trans.Indust.Elec,* vol. 66, p. 1565–1575, 2019.

[4 Y. Li, B. Feng, G. Li, J. Qi, D. Zhao and Y. Mu, "Optimal distributed generation planning in
9] active distribution networks considering integration of energy storage," *Appl. Energy,* vol. 210, p. 1073–1081, 2018.

[5 J. Wu, B. Zhang, H. Li, Z. Li, Y. Chen and X. Miao, "Statistical distribution for wind power
0] forecast error and its application to determine optimal size of energy storage system," *Int. J. Electr. Power Energy Syst.* , vol. 55, p. 100–107, 2014.

[5 J. Usaola, "Probabilistic load flow with correlated wind power injections.," *Electr.Power Syst.*
1] *Res.* , vol. 80, p. 528–536, 2010.

[5 K. Balasubramaniam, P. Saraf, R. Hadidi and E. Makram, "Energy management system for
2] enhanced resiliency of microgrids during islanded operation," *Electr. Power Syst.Res.,* vol. 137, p. 133–141, 2016.

[5 K. Brunnix and E. Delarue, " A statistical description of the error on wind power forecasts for
3] probabilistic reserve sizing," *IEEE Trans. Sustain. Energy.,* vol. 5, p. 995–1002, 2014.

[5 A. A. Teyabeen, F. R. Akkari and A. E. Jwaid, "Power Curve Modelling for Wind Turbines," in
4] *19th International Conference on Computer Modelling & Simulation*, Cambridge, United Kingdom, 2017.

[5 T. H. Soukissian and F. E. Karathanasi, "On the selection of bivariate parametric models for
5] wind data," *Applied Energy.* , vol. 188, p. 280–304, 2017.

[5 D. Voglitsis, N. P. Papanikolaou, C. A. Christodoulou, D. K. Baros and I. F. Gonos, "
6] Sensitivity Analysis for the Power Quality Indices of Standalone PV Systems," *IEEE Access*, vol. 5, p. 25913–25922, 2017.

[5 N. P. Papanikolaou, T. E. C. and K. A., "Analytical model for PV — Distributed generators,
7] suitable for power systems studies," in *13th European Conference on Power Electronics and Applications*, Barcelona, Spain, 2009.

[5 C. W. Watchorn, "The Determination and Allocation of the Capacity Benefits Resulting from
8] Interconnecting Two or More Generating Systems," *Transactions of the American Institute of Electrical Engineers*, vol. 69, no. 2, p. 1180–1186, 1950.

[5 M. Parvania, M. Fotuhi-Firuzabad and M. Shahidehpour, "Demand Response Participation in
9] Wholesale Energy Markets," in *IEEE Power and Energy Society General Meeting*, San Diego, California, USA, 2012.

[6 Y. Bapin and V. Zarikas, "Probabilistic Method for Estimation of Spinning Reserves in Multi-
0] Connected Power Systems with Bayesian Network-Based Rescheduling Algorithm," in *11th International Conference on Agents and Artificial Intelligence*, Prague, Czech Republic, 2019.

[6 M. Craciun, S. Kilyeni, C. Barbulescu and R. G. Meszaros, "Bayesian Networks Applications in
1] Power System Engineering. A Review.," *Journal of Sustainable Energy.*, vol. 8, p. 99–105, 2017.

[6 R. Allan, R. Billinton and N. Abdel-Gawad, "The IEEE Reliability Test System - Extensions to
2] and Evaluation of the Generating System," *IEEE Trans. Power. Syst.* , vol. 6, p. 1–7, 1986.

[6 G. Barney and L. Al-Sharif, Elevator Traffic Handbook Theory and Practice, New York:
3] Routledge, 2016.

[6 J. Pearl, Casuality: models, reasoning, and inference, 2005: Cambridge University Press,
4] Cambridge.

[6 F. V. J. a. T. D. Nielsen, Bayesian networks and decision graphs. : , 2010., New York, NY:
5] Springer, 2010.

[6 T. K. a. J. M. Noble, Bayesian networks: an introduction, Chichester: Wiley, 2009.
6]

[6 G. F. Cooper, "The computational complexity of probabilistic inference using bayesian belief
7] networks," *Artificial Intelligence*, vol. 42, no. 2-3, p. 393–405, 1990.

[6 S. L. Lauritzen, Graphical models, Oxford: Clarendon Press, 2004.
8]

[6 C. P. d. Campos, "New complexity results for MAP in Bayesian networks," in *International
9] Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, 2011.

[7 M. Richards, Introduction to Artificial Intelligence, 2008.
0]

[7 C. Grigg, P. Wong, P. Albrecht, R. Allan, M. Bhavaraju, R. Billinton, Q. Chen, C. Fong, S.
1] Haddad, S. Kuruganty, W. U. K. Mukerji, D. Patton, N. Rau, D. Reppen, A. Schneider, M. Shaliidehpour and C. C. Singh, "The IEEE Reliability Test System-1996. A report prepared by the Reliability Test System Task Force of the Application of Probability Methods Subcommittee," *IEEE Trans. Power Syst.,* vol. 14, pp. 1010-1020, 1999.

[7 W. C. and S. M. Shahidehpour, "Effects of ramp-rate limits on unit commitment and economic-
2] dispatch," *IEEE Trans. Power Syst.*, vol. 8, p. 1341–1350, 1993.

[7 "MATLAB – Matlab & Simulink," MathWorks, [Online]. Available:
3] http://www.mathworks.com/access/helpdesk/help/techdoc/. [Accessed 16 July 2019].

[7 "CPLEX Optimizer," IBM, [Online]. Available: https://www.ibm.com/products/ilog-cplex-
4] optimizationstudio. [Accessed 16 July 2019].

[7 "Bayesialab 8 – Bayesian Networks for Research, Analytics, and Reasoning.," Bayesia,
5] [Online]. Available: https://www.bayesia.com/introduction?__hstc=221168007.d6327a006492fd03c47179bdedf277c4.1549810675300.1549810675300.1565871411196.2&__hssc=221168007.17.156587141.

[7 "Python 3.7 – Python Releases for Windows," Python Software Foundation, [Online]. Available:
6] https://www.python.org/downloads/windows/. [Accessed 16 August 2020].

[7 L. Grigsby, Power system, CRC Press, 2013.
7]

[7 G. Provan, "A Bayesian Network Framework for Stochastic Discrete-Event Control," in
8] *American Control Conference*, Minneapolis, USA, 2006.

[7 C. Yuhu, W. Xuesong and Z. Yiyang, "A Bayesian Reinforcement Learning Algorithm Based
9] on Abstract States for Elevator Group Scheduling Systems," *Chinese Journal of Electronics*, vol. 19, no. 3, pp. 394-398, 2010.