SECURITY-CONSTRAINED UNIT COMMITMENT PLANNING USING PARTICLE SWARM OPTIMIZATION

ROBERT COLLETT

Security-Constrained Unit Commitment Planning Using Particle Swarm Optimization

By

© Robert Collett, B.Eng.

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

October, 2006

St. John's Newfoundland

Your file Votre référence
ISBN: 978-0-494-30456-3
Our file Notre référence
ISBN: 978-0-494-30456-3

Abstract

This investigation explores the development of day-ahead security-constrained unit commitment (SCUC) plans for electrical power systems. Such plans involve the coordination of power system generators in response to variations in loading conditions over a twenty-four hour period. This plan must minimize costs associated with fuel consumption and system losses while satisfying all operational constraints.

This investigation proposes a novel hybrid approach involving the use of particle swarm optimization (PSO) for SCUC planning. As this biologically-inspired methodology is both robust and is based on an advanced strategy for exploring large search-spaces, it is well suited for this highly-constrained power systems problem.

A proposed methodology is implemented in software and a series of test cases are used to assess its functionality. The results of the simulations indicate that the software produces generation schemes that meet all system constraints and that have lower operating costs than those produced with linear programming methodologies. The hybrid PSO solution may therefore be seen as an effective tool for SCUC planning.

Acknowledgements

Special thanks to Dr. John Quaicoe for his assistance with this project and for providing so much guidance throughout my graduate and undergraduate studies. Thank you to Dr. Benjamin Jeyasurya for his many insights and for his assistance in finding technical resources. I would also like to thank the Faculty of Engineering and Applied Science of Memorial University of Newfoundland and also the Power Systems Engineering Research Center (PSERC) at Cornell University for their Matpower software, without which this investigation would not have been possible. Thank you as well to NSERC for graciously providing funding of this project.

Table of Contents

List of Tables

List of Figures

List of Appendices

Chapter 1

Introduction

With the tremendous competition of the modern deregulated electric utility industry, generation, transmission, and distribution companies are forced to minimize operating costs wherever possible. Companies involved in the transmission and distribution of electricity must carefully balance capital and maintenance budgets while relying on the application of new technologies and optimization methodologies to improve the efficiency of their transmission networks.

In this chapter, a brief overview of the types of analyses used by electric generating companies is introduced. The particle swarm optimization approach proposed in this work is presented, along with a brief summary of the contributions of this investigation. An outline of this thesis is also provided at the end of the chapter.

1.1 Unit Commitment in a Day-Ahead Market

In many highly-interconnected electric markets such as those found in Norway, Denmark, Finland, and Sweden [1] and also in California [2] companies coordinate the sale of electricity based on day-ahead planning. In such arrangements, companies present data relating to electrical supply and demand for the following day. These figures are then used in negotiation of electrical energy prices as sales and purchases are made.

While such arrangements depend on accurate day-ahead load forecasts, companies with the capacity for generating electricity (GENCOs) are under additional pressure to juggle generation schemes to ensure that each Watt is produced in an optimal manner. If

a GENCO is to survive in such a market, it must coordinate the production of electricity to minimize fuel costs, system losses, and the amount of power that must be purchased from neighbouring utilities.

Generator scheduling, also known as unit commitment (UC) or economic dispatch (ED) planning is a type of analysis used by electric generating companies to accomplish this task [3]. With consideration given to aspects such as fuel, environmental, and maintenance costs, GENCOs in a day-ahead market must successfully coordinate the operation of their electric generators for successive twenty-four periods.

1.2 Security-Constrained Unit Commitment

The desire to minimize generator production costs often comes in direct conflict with the need to ensure the secure operation of a power system. Despite market-driven pressures, system security remains the most important aspect of power systems engineering. In an attempt to reach a compromise between these two opposing objectives, security-constrained unit commitment (SCUC) has emerged.

It should be noted that in this investigation, the term "security" will focus primarily on the operation of a power system while ensuring that no operational constraints are violated. This definition is in contrast to the standard definition of power system security which involves the capability of an ac network to remain in a normal operating state following all likely contingencies [4]. In this investigation, such contingencies will not be considered.

As discussed in Chapter 4, SCUC involves a tremendously complex optimization problem as power systems are non-linear, time-varying entities. In addition, the coordination of electric generators involves a mixed-integer problem [5].

1.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a powerful approach for the optimization of non-linear problems that was first proposed by Kennedy and Eberhart [6] in 1995. Like the genetic algorithm before it, this methodology is biologically-inspired and represents a *heuristic* approach that, while not guaranteeing an optimal solution, attempts to find a near-optimal solution in a reduced computational time.

PSO has been shown to be a computationally efficient approach that may be implemented in as few as two lines of code [6]. When compared to other evolutionary algorithms, PSO has been shown as often being faster (by at least one order of magnitude) while demonstrating a resistance to becoming trapped in local minima [7]. Theoretical explanation of PSO methodologies are provided in Chapter 3.

1.4 SCUC Using Particle Swarms

This thesis proposes a hybrid strategy for dealing with SCUC planning by combining conventional power flow optimization techniques with particle swarms. In this strategy, linear programming power flow optimization techniques are used to identify generators within a system that are operating inefficiently. The particle swarms may then

be used to determine which of these units should be deactivated to ensure that the system is running in an optimal or near-optimal manner.

The effectiveness of this swarm-based strategy was evaluated as it was applied in two day-ahead SCUC case studies involving theoretical power systems. The results of these studies indicate that this approach has the ability to reduce operating costs while adhering to system constraints.

1.5 Contributions to Research

As a result of this research, the following contributions were made by the author:

1. The completion of an investigation relating to the effectiveness of a hybrid particle swarm optimization tool for reducing electric utility operating costs by improving the efficiency of generator scheduling;

2. The analysis of security-constrained unit commitment problems by investigating aspects such as SCUC constraints, costs, and proposed methodologies;

3. The publication of technical papers [8, 9] relating to application of particle swarm methodologies in SCUC applications;

4. The development of a software application with the following capabilities:
   a. Capacity to develop a unit commitment scheme given day-ahead loading data for a power system;

b. Capability to coordinate generators such that all system constraints relating to system power flows, generator operation, and computational time and storage are met;

c. Capacity to reduce operating costs associated with fuel consumption, system losses, and other parameters based on forecasted load data;

1.6 Thesis Summary

The outline of this report is as follows: Chapter 2 involves a summary of research relating to conventional techniques used for the analysis of UC and SCUC problems. Chapter 3 contains a discussion of background information relating to particle swarm optimization and includes a review of relevant literature. Chapter 4 involves a description of the SCUC planning problem along with all associated constraints and operational costs. A software implementation of the proposed methodology for dealing with the SCUC problem is introduced in Chapter 5. Chapter 6 involves a review of two SCUC case studies along with an assessment of simulated results. Conclusions and recommendations for future SCUC research are presented in Chapter 7.

Chapter 2

Related Work

This chapter provides an overview of literature relating to the formulation of the UC/SCUC problem and to the conventional methodologies that are employed to develop effective solutions. It should be noted that literature relating to the theory and applications of PSO methodologies are discussed in the following chapter.

2.1 UC and SCUC Planning

The literature described in this section relates to UC and SCUC planning in electric power systems. This research provides general background information relating to these optimization problems and also to the various conventional methodologies that may be used to develop acceptable solutions.

With respect to the formulation of UC problems, consideration must be given to system constraints and operating costs. Both of these aspects are introduced in what has become a classic power systems textbook by Wood and Wollenberg [3], a reference that presents both the mathematical and power systems theory behind UC planning.

The work of Cohen et al. [10] assists in the definition of aspects such as system spinning reserve requirements and the limitations of individual units. In this research, definitions for concepts such as unit minimum up and down times as well as emission limits are explained. Wang et al. [11] assist in supplementing the above discussion though the consideration of unit ramp limits and their inherent effect on the UC problem.

In other research, particularly in that of Rajan et al. [12], it may be noted that alternative models for generator operating limits are considered. This investigation involves the modelling of generator minimum on and off times as an exponential function as opposed to the discrete model described by Cohen.

As an important aspect of the UC problem involves the minimization of fuel costs, several researchers have explored the modelling of generator fuel consumption as well as applicable constraints. Vemuri et al. [13] provide a thorough description of these constraints, while Chiang [14] explores the relationship between generator fuel consumption and power output. Chiang's work is also notable as it describes the inaccuracies in popular cost function models as a result of aspects such as valve-points, a concept further discussed in the research of Park et al. [15].

Economic dispatch problems are further complicated when consideration is given to power flow constraints. These constraints are discussed in the research of Ma et al. [16] where aspects such as transmission line loading and bus voltage regulation are considered.

A wide variety of conventional approaches have been explored in the research to assist with UC planning problems. The methodologies include exhaustive enumeration, priority listing, dynamic programming, and linear programming.

Exhaustive enumeration has been explored in the works of Hara et al. [17] and Kerr et al. [18]. These investigations involve an assessment of all possible unit combinations over the short-term timeframe. Each of these works discusses that although this approach guarantees an optimal solution to the UC planning problem, it is not practical for larger systems due to time constraints.

Priority listing techniques are described in the works of Lee et al. [19]. These works describe how such an approach dramatically reduces computation time as generator outputs are controlled solely based on the thermal rate of each unit. The researchers all give consideration to fuel consumption constraints [20] and restrictions relating to power flow transactions between neighbouring systems [21].

Many studies have also been performed relating to the application of dynamic and linear programming methodologies to UC problems. With respect to dynamic programming, the primary focus of this research is directed at reducing the relatively large amount of computational time required when using this approach. Ouyang et al. [22] attempt to modify traditional dynamic programming techniques by using a variable *optimization window* that reduces the overall number of UC combinations. The work of Chang et al. [23] investigates linear programming and how this approach is better able to deal with power system constraints. The overall conclusions that may be drawn from these investigations indicate that even when the computational time required for the dynamic programming strategy is reduced, this methodology is often unable to generate a solution that meets all system constraints within a specified time limit. As Chang describes, although linear programming is a faster approach and is better able to deal with system constraints, this methodology is inconsistent in its ability to converge on an acceptable solution.

A variety of alternative conventional methods have also been thoroughly explored in UC research. These methods are presented in the work of Padhy [5] in his extensive survey of literature in this field. In addition to the techniques described above, Padhy provides an overview of approaches that include Lagrangian relaxation, branch and

bound methods, interior point optimization, Tabu searches, simulated annealing, expert systems, fuzzy systems, and artificial neural networks. While detailed explanations of these advanced methodologies are beyond the scope of this investigation, it is important to recognize that a variety of strategies are being developed for dealing with SCUC optimization problems.

Chapter 3

Particle Swarm Optimization

Particle swarm optimization (PSO) was first proposed by Kennedy and Eberhart [6] in 1995 as a new technique for the optimization of nonlinear problems. Like other types of evolutionary computation before it, this *heuristic* approach is a form of evolutionary computation and is biologically-inspired. To date, variations of PSO have been developed for dealing with the optimization of continuous functions and also with binary decision making [24]. As indicated in the literature, each of these variations has been shown to be both simple in its implementation and computationally inexpensive.

This chapter involves an introduction to particle swarms and heuristic methodologies. Concepts relating to evolutionary computation are discussed and the advantages of such strategies are presented. An overview of the mathematical, sociological, and computer engineering aspects of particle swarms are also examined. The chapter concludes with a review of literature that pertains to the development of the theory and applications of particle swarm optimization. It should be noted that particular emphasis is placed on the application of particle swarms and other evolutionary methodologies to UC and SCUC problems.

3.1 Heuristic Methodologies

One of the primary attributes of particle swarm optimization is that it is a heuristic methodology. By definition, a heuristic technique is an approach that does not guarantee an optimal solution, but hopes to identify a very good or *near-optimal* solution in a

reduced computational time [25]. This strategy is typically employed in optimization problems that may not be solved using conventional algorithms due to time constraints or due to the degree of complexity of the problem.

When faced with such a problem, a heuristic strategy is used to provide a shortcut where only a portion of the overall set of solutions is evaluated. Rather than using a brute-force algorithm, a heuristic approach limits its search to only a subset of solutions where the optimum is likely to be found. This subset is chosen based on a strategy that has the capacity to use the information from each evaluated solution to help direct the search towards the optimum. Such a strategy must be robust and must be able to efficiently manoeuvre through the set of possible solutions without getting caught in obstacles such as local minima. Since heuristic approaches are able to dynamically respond and adapt to information collected at each step in the search for the optimum solution, they are defined as a type of artificial intelligence.

3.2 Evolutionary Computation

One of the fundamental paradigms of heuristics involves the use of evolutionary computation to assist in the identification of an optimal solution. These strategies have developed over the past fifty years and consist of a number of sub-branches that include both the genetic algorithm and evolutionary programming. While each of these fields is unique, both attempt to evolve an optimal solution based on the concept of *survival of the fittest.*

The genetic algorithm was first proposed by John Holland [26] in the early 1960's in an attempt to develop an adaptive system that could dynamically respond to its

environment. In his work, he proposed the use of a population of individuals where each member of the population served as a particular solution to a problem. Specifically this population was seen as being analogous to a set of chromosomes – the biological building blocks of DNA. Just as a chromosome is characterized as having a set of genes, each member of Holland's population was encoded with a variety of traits that defined each particular solution.

The strategy proposed by Holland was to evaluate the quality of each solution and to effectively quantify the fitness of each member of the population. Once this process is complete, probabilistic methods may be used to select members of the population that will be able to transfer their genetic information to the next generation through reproduction.

As in the biological world, only the fittest members of a species would survive and the ideal situation would therefore involve the transmission of healthy genetic information to future generations. At the same time, negative characteristics would be eliminated from the gene pool. The members of the subsequent generation would therefore consist of combinations of the best genetic material from their predecessors. As a result, the overall fitness of the population would improve. In terms of an optimization problem, this process is iterated until the population converges on an optimal or near-optimal solution.

While the genetic algorithm focuses primarily on genetic crossover to evolve an acceptable solution to an optimization problem, evolutionary programming relies on an alternative biological process: mutation. This process was proposed by Fogel in 1994 [25] and involves randomly varying the individual genetics of the population members.

As a result of this process, new genetic combinations may be found and member fitnesses may be improved.

As with the genetic algorithm, future generations are created based on probabilistic methods where the fitness of each member is taken into consideration. It should be noted, however, that evolutionary programming does not employ genetic recombination.

Although these methodologies differ in their implementation, both processes may be characterized by the following steps:

1. Initialize a random population of potential solutions.
2. Calculate the fitness of each member of the population.
3. Use probabilistic methods to select members of the population for reproduction. For example, a weighting system may be employed to increase the likelihood that the fittest members of a population will have their characteristics transferred to future generations
4. Perform evolutionary operations (crossover, mutation).
5. Iterate the above procedure.

3.3 The Advantages of Evolutionary Computation

The evolutionary computation strategies described above have several advantages over traditional optimization paradigms. These advantages include the ability to avoid getting stuck in local minima and a lack of dependence on auxiliary calculations [27].

With respect to ability to avoid local minima, an evolutionary computational strategy represents a powerful alternative as it employs a *population of search operators*. This approach is beneficial in that it allows for the implementation of a searching strategy that is based on the *collective knowledge* of the population. In addition, the use of operations such as crossover and mutation allows for the operators to move freely through the *hyperspace of potential solutions* to regions where the optimal solution may likely be found [7].

Evolutionary computation strategies are also beneficial in that they do not depend on auxiliary calculations. While many hill-climbing paradigms rely on functional derivatives to direct the search for an optimal solution, evolutionary searches rely solely on the calculation of fitness values for each member of its population. These fitnesses are often directly related to the output of the function that is being optimized and therefore present a more efficient alternative.

3.4 The Particle Swarm Methodology

Based on the advantages described above, Kennedy and Eberhart began the development of a new evolutionary strategy that was based on the biological and sociological behaviour of animals such as those demonstrated by flocking birds or schooling fish. While such animals demonstrate the ability to effectively communicate and cooperate as a group to reach common objectives, the particle swarm methodology hopes to imitate this behaviour to help direct its population toward an optimal solution.

As described above, particle swarm optimization is accomplished using a population of search agents (known as particles) that are randomly scattered to a set of

positions in the hyperspace of potential solutions. For a problem consisting of n variables, the hyperspace would have n dimensions and the ith particle would be assigned the position vector:

$$X_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in}) \tag{3.1}$$

In (3.1), X_i is a vector of continuous variables that represents a particular solution to the *cost function* that is to be optimized. By evaluating the cost functions for each member of the population, the particles begin to explore what is known as the *fitness landscape* of the optimization problem [25].

Unlike other evolutionary strategies, particle swarm optimization does not rely on reproductive operations such as crossover or mutation to explore the n-dimensional hyperspace. Rather, the population of particles adopts the behaviour of a group of animals such as a flock of birds searching for food in a field. In this process, each particle is assigned a velocity in the hope of directing it toward the regions where the highest fitnesses have been found. The velocity vector of the ith particle is specified as:

$$V_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{in}) \tag{3.2}$$

Once each particle has been assigned a velocity, the new position of each particle may be calculated as:

$$x_{in} = x_{in} + v_{in} \tag{3.3}$$

This process is iterated in the hopes that the particles will begin to swarm in the vicinity of the optimal solutions to the problem.

According to the model proposed by Kennedy and Eberhart, the velocities that are to be assigned to each particle are based on a series of mathematical equations that represent sociological phenomena. As each particle must intelligently determine which direction it should travel, it must rely on a combination of its own experiences and on the collective knowledge of the group to determine the best course of action.

With respect to particle learning based in individual experiences, Kennedy and Eberhart suggest that an intelligent organism that is performing a search will likely be drawn to a location where it has had success in the past. For example, migrating birds often return to the same nesting areas after travelling for extended periods of time. With respect to a mathematical optimization problem, a component of the velocity of the ith particle will therefore direct it toward P_{in}, the position within the n-dimensional search-space where it has attained its highest fitness to date. From a computer programming perspective, the coordinates of P_{in} must be recorded for each member of the swarm.

The velocity assigned to each particle also depends on the collective knowledge of the swarm population. From a sociological sense, this refers to an individual's desire to imitate the behaviour of elite members of a community. In reference to particle swarm optimization, each member of the swarm community will be attracted to P_{gn}, the location of the highest fitness to date. This rule would require a computer program to store the coordinates of P_{gn} and update as required.

The final particle velocity component prescribed by Kennedy and Eberhart consists of an inertia factor that influences a particle to continue moving in the same

direction as the previous iteration. This additional parameter permits the swarm to explore the area in the vicinity of a previously-attained maximum value. To accomplish this strategy, a computer program would be required to store a velocity vector for each particle.

These three components may be seen in the formula proposed by Kennedy and Eberhart:

$$v_{in} = w * v_{in} + c_1 * rand_1 * (P_{in} - x_{in}) + c_2 * rand_2 * (P_{gn} - x_{in}) \tag{3.4}$$

It may be noted that each of these components is weighted by acceleration factors, represented by w, c_1, and c_2, respectively. In addition, to ensure that all particle velocities have a certain degree of randomness, $rand_1$ and $rand_2$ represent random numbers between zero and one. This use of random numbers is employed as it ensures that the movement of each particle is different from the other members of the population. This is particularly useful in later iterations of the optimization process when several of the particles have likely converged on a particular region of the search-space. In such a case, the randomness decreases the probability that the particles will follow the same path and therefore ensures that a larger portion of the search-space is explored.

As stated above, these equations provide solutions consisting of continuous variables. In some optimization problems, however, solutions may be binary in nature and the ith particle of the swarm will be defined by a vector having components $x_{i1},\ldots, x_{in}$ that are ones and zeros. When considering such a problem, solutions may be defined as bitstrings of length n and the search-space becomes an n-dimensional hypercube as illustrated in Figure 3.1.

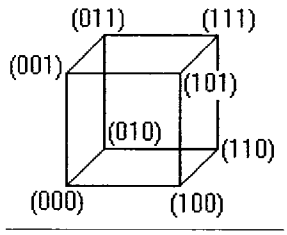

Figure 3.1 Three-Dimensional Hypercube

Kennedy and Eberhart [24] described binary particle swarm optimization (BPSO) as being similar to ordinary PSO. As in the continuous case, the population of particles occupies a set of positions within the *n*-dimensional search-space. As well, each particle is assigned a velocity that is calculated based on the relative fitnesses of the population. The primary difference with BPSO, however, is that these velocities are used to determine if the positional vector elements of each particle should be set to one or zero. This binary decision-making process is accomplished by thresholding the velocities to the range [0.0 1.0] using the sigmoid function:

$$s(v_{id}) = \frac{1}{1+\exp(-v_{id})} \tag{3.5}$$

The resulting number is then compared against ρ_{id}, a vector of random numbers drawn from a uniform distribution between 0.0 and 1.0. Kennedy and Eberhart then prescribed that the following formula be applied:

$$\text{if } \rho_{id} < s(v_{id}) \text{ then } x_{id} = 1; \text{ else } x_{id} = 0 \tag{3.6}$$

It should be noted that in both the continuous and binary versions of PSO, a maximum velocity, V_{max}, may be set to control the behaviour of the swarm [28]. Such a parameter assists in reducing erratic particle movement in the continuous case, while ensuring that each particle has the ability to change state in the BPSO case. In this BPSO case, the capping of large velocities prevents the saturation of the sigmoid function. For example, by limiting V_{max} to a magnitude of 4, the sigmoid function cannot produce a value lower than 0.017 or higher than 0.982. As a result, there is at least a ~2% chance that the particle will change its binary state. The particle therefore cannot become "trapped", regardless of the system parameters.

As indicated by the above equations, PSO involves a relatively simple methodology that may be implemented in just a few lines of code. As well, the principal memory requirement for the swarm involves the storage and manipulation of two n-dimensional arrays (one for position, one for velocity).

3.5 The Development of Particle Swarm Optimization (PSO)

The literature presented in this section relates to particle swarm optimization and its evolution as a strategy for the optimization of complex non-linear problems. In addition to the comprehensive studies that have been published relating to particle swarm techniques, this section also involves a review of investigations pertaining to the analysis and selection of swarm parameters, the capacity of particle swarms to deal with system constraints, and the comparison of particle swarms to other evolutionary techniques. Other topics discussed in this section relate to particle swarm optimization strategies and applications that have been investigated in recent research.

As described in Chapter 1, particle swarm optimization was first proposed by Kennedy and Eberhart in 1995 [6]. Since its initial discovery, a series of comprehensive works have been published to explore relevant mathematical and sociological phenomena associated with this technique. The first such study was also performed by Kennedy and Eberhart in 2000 with the publication of Swarm Intelligence [27]. The work represents one of the most recognized studies of the methodology and is widely referenced in the literature. More recent comprehensive studies have been published by Englebrecht in 2005 with Fundamentals of Computational Swarm Intelligence [29] and by Clerc in 2006 with Particle Swarm Optimization [25]. Each of these publications investigates particle swarm optimization from a multidisciplinary standpoint by reviewing how the subject relates to the fields of computer science, mathematics, sociology, and engineering.

Over the past decade a series of studies have been performed in an attempt to assess the behaviour of particle swarms and to improve their performance. Many of these investigations are concerned with the selection of swarm parameters and how they may be designed to improve efficiency and convergence time. One such study is that of Eberhart and Shi [28] where a study was performed to address the stability of particle swarms. In this research, particle speed limitations were established in an attempt to control the movement of the swarm as it searched for an optimal solution. Other such investigations include those of Zheng et al. [30] in their analysis of the application of time-varying acceleration factors to improve the control of swarm movement.

Other relevant studies associated with particle swarm optimization involve constraint handling. As many engineering and mathematical problems are highly constrained, studies such as those performed by Coath et al. [31] investigate alternatives

for ensuring that an acceptable solution is found in an efficient manner. This study involves the application of *penalty factors* and the analysis of how swarm convergence may be affected. In addition, a comparison is made between this strategy and the alternative concept of a *feasible solution method* where solutions are rejected if they do not meet all system constraints. Other studies involving the ability of particle swarms to perform in a constrained optimization environment include the work of Michalewicz et al. [32]. In this study, a *stepping* procedure is proposed where the optimization process does not attempt to meet all system constraints at once. Rather, this procedure tries to meet them one at a time in a *divide and conquer* strategy.

Other research in the field of particle swarm optimization relates to the comparison of this methodology to other types of evolutionary computation. In the work of Kennedy and Spears [33], the rate of convergence of the swarms is compared to that of the genetic algorithm. This investigation notes that, for a series of non-linear mathematical optimization problems, the particle swarms are almost universally faster than the genetic algorithm and are less affected by dimensionality. A similar study was performed by Habib and Al-Kazemi [34]. This study investigated how particle swarms are better able to optimize mathematical distance functions than other evolutionary methodologies.

A significant amount of research has also been performed to investigate ways to improve the searching strategies of particle swarms. In the work of Clerc [35], a series of experiments are performed to examine how to deal with a *no hope swarm* that is unable to find a suitable solution. In this investigation, Clerc explores the notion of how such a

swarm may be *reset* after a predefined amount of time if its fitness does not meet a minimum set of criteria.

Particle swarm strategies are also examined by Eberhart and Shi [36]. In this study, the ability of swarms to solve dynamic optimization problems is examined. As a proposed solution in this investigation, portions of the particle swarm population are re-randomized in an attempt to respond to the time-varying aspects of the system that is to be optimized.

Coello and Lechuga [37] have also investigated particle swarm behaviour in an attempt to develop strategies for dealing with multi-objective optimization problems. In this research, the fitnesses of particles within a swarm are ranked based on the concept of *Pareto dominance*. In such a strategy, all system objectives are prioritized and may therefore be used to dictate the behaviour of the swarms.

Another development in the field of particle swarms was proposed by Kennedy and Eberhart [24] in their exploration of a strategy by which the swarms may be used in the optimization of discrete problems. In this paper, it is described that many optimization problems are binary in nature and it is shown that swarms may be employed to determine an optimal set of system states.

An important portion of ongoing research relating to particle swarms involves the application of particle swarms to solve mathematical, computer science, and engineering problems. With specific interest in the area of electrical engineering, a number of investigations have been preformed to explore the applications of particle swarm to such areas as the development of neural networks, state of charge estimation in battery packs, and the optimization of antenna systems.

With respect to neural network development, Hu et al. [38] have explored how particle swarms may be used as an alternative to back-propagation methodologies. In this investigation, the swarms are used to help determine how to optimally interconnect neural networks and to optimize network learning. The research indicates that the swarms may be particularly useful for systems containing non-differentiable components.

Neural networks are applied in a practical electrical engineering problem in the work of Peng et al. [39]. In this study, a series of neural networks are defined using particle swarm optimization to estimate the state of charge in battery pack system. These neural networks provide an alternative to a complex mathematical model that is particularly useful as the problem involves a hybrid vehicle system that is defined by a number of non-differentiable parameters.

Particle swarm optimization is also used as tool for the optimization of antenna systems as found in the work of Boeringer et al. [40]. This study further demonstrated the capability of particle swarms in dealing with the optimization of non-linear systems and discusses the potential of this methodology for dealing with a wide variety of electromagnetics problems.

Based on the studies described above, particle swarm optimization may be seen as a relatively new optimization tool that is particularly beneficial for dealing with systems that are either non-linear or non-differentiable in nature. To date, particle swarms have been used in a wide range of electrical engineering problems and it is expected that further applications will be discovered as new swarm strategies are developed. While it has been shown that swarms often perform better than other evolutionary strategies, it is

expected that the popularity of this methodology will continue to grow as research is performed to help further improve its efficiency.

3.6 The Application of Evolutionary Techniques to UC and SCUC Planning

With the growing popularity of evolutionary methodologies such as the genetic algorithm and particle swarm optimization, several researchers have investigated the potential application of these techniques to unit commitment problems. The following studies relate specifically to investigations where power systems optimization is performed using these evolutionary approaches.

Several studies have been performed involving the use of the genetic algorithm to assist in economic dispatch problems. In the research of Rajan et al. [41] and Yang et al. [42] investigations have indicated that genetic techniques generally have higher convergence rates than methodologies involving dynamic programming or even more advanced techniques such as Lagrangian relaxation. It was also found that the genetic algorithm produces a higher quality result.

Despite consistent convergence, however; Kazarlis et al. [43] discusses how the genetic approach requires a significant amount of computational time for effective power systems optimization. In addition to this shortcoming, the work of Mantawy et al. [44] discussed another drawback to this approach involving a relatively high memory requirement.

In response to these issues and due to the strength of particle swarm techniques in dealing with non-linear optimization problems, several researchers have begun to explore

the application of swarm methodologies in power systems analysis. As described in the work of Chang and Lu [45], distribution networks may be optimized by using particle swarm techniques to coordinate system feeders. In this study, feeder optimization allows for load aggregation where improved system load factors lead to reduced operating costs.

The economic analysis of power systems is also studied in the research of Xiaohui et al. [46], which focuses primarily on the relationships of neighbouring systems in a modern, deregulated electric utility market. In this study, particle swarms are used to maximize profitability through the optimization of the sale and purchase of electricity.

Studies specifically involving the optimization of economic dispatch problems through the use of particle swarm include the work of Gaing [47]. This study compares the performance of particle swarms to the genetic algorithm with respect to unit commitment analysis and finds that the swarms have a higher convergence rate than the alternative evolutionary strategies. It is still found, however, that the speed of convergence is unacceptably long for a day-ahead planning situation.

More recent studies include the work of Ting et al. [48] where a binary particle swarm approach is used in an attempt to optimize a power system generation scheme while adhering to operational constraints. While this study indicates that the proposed solution is capable of producing acceptable results, it is noted that the use of a separate particle swarm for each hour of the operating period presents a variety of difficulties. Such a strategy requires a large computational time and may not be suitable for a day-ahead planning situation. In addition, the use of multiple swarms results in a lack of correlation between successive hours and may lead to large variations in the prescribed active outputs of system generators. As a result, such a strategy may cause violations of

the operational constraints for system generators and may therefore cause a unit's lifespan to be reduced.

A variety of hybrid approaches have been proposed in recent research in an attempt to improve the ability of particle swarms to handle unit commitment constraints. Such studies include the work of Sriyanyong and Song [49] where particle swarms are combined with Lagrangian relaxation techniques and that of Victoire and Jeyakumar [50], where Tabu search methodologies are employed. The results of each of these investigations indicate that such hybrid techniques may be used to improve the overall performance of the swarm.

Other recent developments relating to the application of particle swarms involve the use of multi-objective optimization schemes where attempts are made to reduce not only operating costs, but also to help reduce the emission of greenhouse gases. The study of Al-Rashidi and El-Hawary [51] presents a strategy whereby the Pareto dominance approach [37] described above may be employed to meet all operating constraints.

In summary, evolutionary strategies such as the genetic algorithm and particle swarm optimization have been employed in a number of studies to assist with power systems optimization. The results of these studies indicate that the strategies are able to eliminate the shortcomings of conventional methodologies such as high computational times and memory requirements. Recent trends in the literature also indicate that alternatives such as hybrid approaches are able to further improve the computational efficiencies. Based on these findings, a hybrid particle swarm optimization approach would likely provide an acceptable means of responding to the challenges of day-ahead SCUC planning.

Chapter 4

Security Constrained Unit Commitment (SCUC) Planning

Security-constrained unit commitment (SCUC) planning involves the coordination of a power system's generating units in an attempt to minimize operating costs while adhering to a set of operational constraints. As discussed below, this optimization is exceedingly difficult due to its classification as a time-varying, non-linear, mixed integer problem. This chapter provides an overview of these challenges and presents the numerous constraints that further complicate the optimization process. SCUC operational costs are also discussed at the end of this chapter.

4.1 SCUC Problem Identification

The classical unit commitment problem consists of an attempt to respond to a time-varying load by modifying a system's generating scheme. The time-varying nature of this problem requires that a power system not only be optimized in one particular state. Rather, all system constraints must be met and costs must be minimized for the overall operating period. This may be particularly problematic, however, as the desire to reduce instantaneous costs may conflict with the optimization of the overall period.

Another challenge presented in SCUC planning involves the non-linear nature of power systems optimization. In classical unit commitment problems [3], a simplified model of a system is presented and the non-linearities brought about by system load flows are avoided. In these analyses, the system is modeled as a single bus that is connected to all generating units and to all loads. In such a system, many parameters are

neglected and a simple priority list methodology may be employed. Unfortunately, this technique pays no attention to load flow constraints and is therefore unacceptable for SCUC planning.

When a system is to be optimized and load flow constraints are considered, well established calculus-based techniques exist for optimizing generator schemes. For larger systems, these techniques are implemented in computer programs such as the Matpower 3.0.0 toolbox for Matlab [52]. These programs are capable of calculating each generator's optimal active power output within seconds. Such programs are unsuccessful, however, in that they are unable to deal with the mixed-integer aspect of the optimization problem.

Problems arise with calculus-based algorithms in that they can only examine a system in one particular mode of operation. More simply, these techniques are unable to turn a generator on or off to identify an optimal solution. Rather, they will only minimize the output of inefficient generators. This limitation may be particularly problematic in systems where generator outputs have non-zero minima and the system is under light loading conditions. In such cases, several (if not all) generator outputs may be set to minimum, non-zero values. As a result, planners must somehow choose which minimized units to deactivate. It should be noted that this procedure becomes exceeding difficult in larger systems as n minimized generators would result in 2^n possible on/off combinations.

4.2 SCUC Constraints

While the challenges described in the previous section demonstrate the degree of difficulty of an SCUC optimization problem, a significant number of constraints must also be considered when developing an acceptable solution [2, 11, 13, 16, 20, 48, 49].

These constraints relate to system power flow limitations, generator operation limitations, and the computational time and storage requirements of the optimization algorithm.

4.2.1 Power Flow Constraints

As stated in the introduction, the first priority of a power systems engineer is reliability. All SCUC solution must therefore be defined such that no power flow constraints are breached during the specified operating period. These power flow constraints include:

- bus voltage magnitudes
- bus voltage angles
- generator active power outputs
- generator reactive power outputs
- apparent power flow in system branches

The voltage magnitude at each bus must be held within maximum and minimum values. For example, typical bus voltage specifications require that voltages stay within a range of [0.95-1.05] per-unit.

Due consideration must also be given to bus voltage angles as a generator may lose synchronism if a bus voltage angle exceeds 90^0. Such a situation would lead to system instability and is therefore not permitted.

Generator active and reactive power limits represent other important power flow constraints. As a generator excitation is varied, the reactive power produced by a unit may vary from a negative to a positive value. Reactive power limits therefore typically have a negative minimum and a positive maximum (e.g. from -40 to 40 MVar). With

respect to active power output, certain generators may not be permitted to operate below a specified value for efficiency reasons. As a result, active power limits typically have a positive minimum, as well as a positive maximum. It should be noted that generator motoring is not permitted in this investigation.

The final operating constraint for a system involves the flow of apparent power in the various system branches. These values are limited by the maximum MVA limit of each line and must be calculated at both the sending and receiving ends.

4.2.2 Generator Operating Constraints

In addition to the standard power flow criteria, unit commitment problems must also consider maintenance-related issues involving the start-up and shutdown of each generator [2]. For example, if a generating unit is frequently switched on and off, its lifespan may be dramatically reduced. Many generators therefore have limits dictating their minimum on/off times.

Other maintenance-related issues include generator ramping limits [11]. It is often undesirable or perhaps beyond a unit's capability to dramatically increase or decrease active power output in a short period of time. In response to this, ramp limits must be enforced.

It should be noted that this investigation involves the analysis of systems containing only thermal units. As a result, constraints associated with generators that are driven by hydroelectric or other means have not been explored. While system elements such as "must run" units are beyond the scope of this investigation, they may be considered in future research based on the content of this thesis as described in Chapter 7.

4.2.3 Computational Time & Storage Constraints

As described in the introduction, system generating schemes are usually based on the day-ahead market. It is therefore reasonable to assume that generating schemes must often be planned in less than twenty-four hours. As part of the planning process, time should also be allotted to ensure that a generating scheme is properly verified so that it may be approved and implemented.

With respect to computational power and storage space, SCUC planners may set constraints based on available hardware and software.

4.3 SCUC Costs

As stated above, the SCUC problem involves finding an optimal (or near-optimal) solution that minimizes operating costs while adhering to all power flow and maintenance constraints. The calculation of operating costs requires the consideration of:

- fuel consumption costs
- start-up costs
- shutdown costs
- system losses

Fuel consumption costs are a combination of a unit's heat rate and fuel costs and use a polynomial expression to convert the amount of active power produced into dollars [43]. Typically this relationship is represented by a three-term expression:

$$hourly\ fuel\ \cos t(\$) = a + bP + cP^2 \tag{4.1}$$

P in the above equation represents the active power [MW] produced by the generator, while the coefficients a, b, and c have units of [$/h], [$/MWh], and [$/MW^{2}h], respectively.

Start-up costs represent the estimated cost of fuel consumed during the start-up of the generator. A typical start-up cost found in this investigation is approximately $100 for a 100W unit. Similarly, shutdown costs consider the cost of fuel consumed when a unit is taken offline. Typically, shutdown costs are deemed to be negligible.

System losses involve the costs associated with power that does not reach the customer as it is consumed by system transmission lines. Losses are calculated by identifying the amount of active power consumed by the resistive component of the system transmission and multiplying the resulting MWh value by the specified electrical rate.

Chapter 5

SCUC Software Implementation

For the purposes of this investigation, a software package was developed to assist in day-ahead SCUC planning and the creation of a generating scheme that would meet all system constraints while improving operating efficiency. This chapter involves a discussion of the six modules that include: the Input Module, the Minimum Generators Module, the Particle Swarm Module, the Pathfinder Module, the Load Flow Verification Module, and the Output Module. Code for all software modules may be found in Appendices A, B, and C while a flow control diagram illustrating their interaction is provided in Figure 5.1.

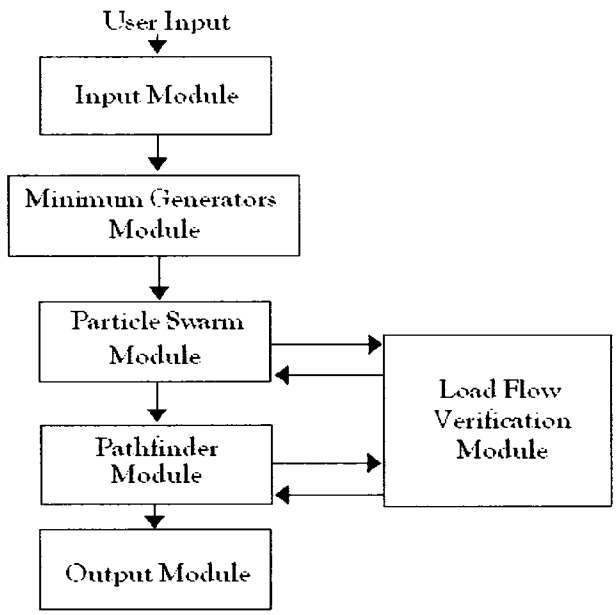

Figure 5.1 Software Flow Control Diagram for the SCUC Software

5.1 Overview of SCUC Modules

The optimization process begins with the Input Module which allows for the specification of all system parameters and operational constraints. Once all system variables have been defined, power flows within the system may be optimized using calculus-based linear programming techniques in the Minimum Generators Module. These techniques may be used to identify system generators that have minimal active power outputs at various stages of the operating period.

Upon completion of this step, the hour with the highest number of minimized generators is assessed by The Particle Swarm Module. In this module, the objective is to determine the optimal generator on/off combinations for that particular hour using the binary particle-swarm methodology described in Chapter 3.

The next step of the SCUC planning process involves determining the overall generation sequence for the operating period. For this stage, the Pathfinder Module is employed to use the combinations defined by the Particle Swarm Module to extrapolate a generation scheme for all remaining hours. For a sequence to be acceptable, however, it must meet the system operating constraints defined in Chapter 4.

During the Particle Swarm and Pathfinder stages, the Load-Flow Verification Module is used to ensure the reliable operation of the power system by ensuring that all power flow constraints are met.

Upon completion of the above stages, control is passed to the Output module which is responsible for providing all required unit commitment and operational data to the user.

It should be noted that all SCUC software was developed using a series of Matlab m-files. With respect to software design, global variables were used for a many of the main data storage matrices to minimize virtual memory requirements. While this practice may reduce the efficiency of the program, it was chosen due to simplicity of implementation. Improvements to this aspect of the software design are discussed in Chapter 7 which includes recommendations for future work.

5.2 The Input Module

The primary objective of the Input Module is to allow for SCUC software to have the versatility to develop economical day-ahead solutions for a wide variety of power systems. The software was developed to allow a user to input system-specific information so that load flows may be optimized. The various features of this module allow the user to include both technical parameters and financial data associated with the operation of thermal units and system losses.

5.2.1 System Case Files

As stated above, one of the primary components of the Input Module is the capacity to store load flow data for the power system that is to be optimized. A user may input such data through the creation of a *case file* – a Matlab m-file that contains data relating to system buses, branches, generators, areas, and generator costs. Each of these parameters may be input in matrix form where each row refers to a specific entity within

the power system and each column relates to the characteristics of that entity. Each of these matrices is defined in Appendix D.

It should be noted that once a system case file has been prepared containing all necessary information in the proper format, the user may instruct the SCUC software as to the name of the desired file so that the required parameters may be obtained.

5.2.2 Hourly Load Data and Other Parameters

In addition to entering parameters into system case files, users may also use the Input Module to provide the software with day-ahead loading data that may be used for SCUC planning. For improved simplicity, the Input Module has the capacity to extract this data from a Microsoft Excel spreadsheet. This task is performed automatically once the user has input the name of the desired file and of the specific spreadsheet containing the system loading data.

Other information that may be entered as part of the Input Module includes energy costs, which are measured in dollars per megawatt-hour. These values assist in the calculation of system losses.

5.2.3 Particle Swarm and Optimization Parameters

The parameters associated with the particle swarm optimization methodology are also provided in the Input Module. These parameters allow for a user to customize the optimization strategy to best suit a particular power system planning problem. Based on computational time limits and the size of the power system in question, the user can

adjust variables including the swarm population size, the number of iterations the swarm is allowed to search, and the overall software time limit at which point the software must provide the day-ahead power system generation scheme.

Other variables associated with the particle swarms include other parameters discussed in Chapter 3, such as acceleration coefficients, w, c_1, and c_2, and the maximum particle speed, *Vmax*. As discussed, these variables help to control the movement of the particles within the swarm [25, 29]. By fine tuning these parameters through experimental trials, a systems planner may modify the movement of the particles to ensure that the optimization process functions as efficiently as possible.

A final variable that may be defined in the input module is the Boolean parameter *allOn*. This variable is typically set to false when the software is to perform its optimization process under normal operating conditions. Under certain circumstances, it may be desirable to deactivate the particle swarm optimization algorithm and to simply use a calculus-based linear programming solver to optimize the system and to leave all system generators operational. In such a case, the *allOn* variable may be switched to TRUE. This will result in the execution of a much simpler optimization algorithm that will require much less computational time. Because all generators are active for the entire operating period, this algorithm is unlikely to produce the most cost effective solution.

5.3 The Minimum Generators Module

Once all system parameters have been entered and processed by the Input Module, the first calculations performed by the SCUC are to use the calculus-based linear programming software to optimize the power system for each hour of the operation

period with all system generators in operation. As discussed in Chapter 4, the solver will assess which generators are inefficient and should therefore have minimal active power outputs. For these calculations, this module employs the system information from the case file and scales the loading data for each hour according to the values specified in the Microsoft Excel source file. It is important to note that all system generators are activated for this analysis.

Once the process is complete, the Minimum Generators Module tallies the number of minimized generators for each hour of the operating period. The hour with the highest number of minimum generators is identified and output from the module as the variable *lightHour*. As this hour has the highest number of possible generator on/off combinations, it is this hour that will be the focus of the particle swarm optimization and the third module of the SCUC software.

As an example of this procedure, the eight-unit power system below may be considered. The system was simulated for each hour of the operating period with all units activated and was analyzed by the Minimum Generators Module. Based on these analyses, generators having a minimal and non-minimal active power outputs were identified as follows:

(In the matrices below: 1 = non-minimal output, 0 = minimal output)

Hour 1:	[1 0 0 1 0 0 0 1]	Minimum Generators: 5	
Hour 2:	[1 0 0 1 1 0 0 1]	Minimum Generators: 4	
Hour 3:	[1 0 0 1 1 0 0 1]	Minimum Generators: 4	(5.3)
Hour 4:	[1 0 1 1 1 0 0 1]	Minimum Generators: 3	

Hour 5: [1 0 1 1 1 1 0 1] Minimum Generators: 2

⋮

In the example above, *Hour 1* has the highest number of minimum generators and would therefore be labelled as *lightHour*. As this hour has five minimized units, there exist $2^5 = 32$ possible generator on/off combinations. This information is required for the analyses that are performed by the other modules of the SCUC software.

5.4 The Particle Swarm Module

The objective of the Particle Swarm Module is to determine inexpensive generator on/off combinations for the hour identified as *lightHour* by the Minimum Generators Module. This is accomplished through the application of the optimization strategy discussed in Chapter 3 where binary particle swarms [24] are used to identify which generators should remain functional while a calculus-based linear programming solver is used to specify the active power of these units and to calculate operational costs.

5.4.1 Particle Swarm Tables

While particle swarm optimization processes have the objective of finding the ideal or least-expensive solution, it is important to note that a large number of generator on/off combinations will be evaluated during the swarm's search. As well, as defined in Chapter 4, identifying optimal solutions for each individual hour may not necessarily lead to an optimal generation schedule for the overall period [48]. It should therefore be noted

that the Particle Swarm Module does not have the objective of finding a single optimal combination, but rather a set of combinations that could potentially become strong candidates for the optimal generation scheme of the overall twenty-four hour operating period.

With this strategy in mind, the Particle Swarm Module is designed to maintain a series of tables that are used for the storage of the large number of results that are calculated during the optimization. As the various operational costs are calculated for each combination of generator on/off settings, these values are stored in ascending order by total cost. When the Pathfinder Module (described below) attempts to find an optimal scheme for the overall operating period, it can therefore begin its search with the least expensive combinations for the first hour.

The maximum number of generator combinations that may be evaluated during the optimization process will equal the product of the swarm population size, *popSize*, and the number of iterations allowed for the search, *iterCount*. This product may also be known as *maxCombos*, a variable used in the SCUC program. It should be noted that if *lightHour* contains a large number of minimized generators, *maxCombos* will likely be less than the total number of possible generator on/off combinations. System engineers must therefore take this into consideration when selecting the variables defined in the Input Module

The following tables are created by the Particle Swarm Module as data is recorded during the optimization process:

1. *settings* – This table is a one-dimensional array that contains binary on/off settings for all power system generators. The array can hold a number of

elements equal to the product of *maxCombos* times the number of power system generators. For example, if a system containing n generators were to be optimized by a particle swarm with a population size of p that was allowed to search for i iterations, the *settings* matrix would have a length of $n \times p \times i$ elements. The array below indicates how generator setting would be stored if a system with n generators were to be optimized by a particle swarm where $maxCombos = m$:

$$[combo_1 gen_1 \ \ldots \ combo_1 gen_n \ \ldots \ combo_m gen_1 \ \ldots \ combo_m gen_n] \tag{5.4}$$

2. *fuelCosts* – This table is also a one-dimensional array that contains the fuel costs (in dollars) of all system generators. These values are calculated based on thermal rate polynomial equations defined in (4.1). This table has the same dimensions as the *settings* matrix and data is stored in the same format.

3. *pfCosts* – This table includes all power flow costs that are calculated during the particle swarm optimization. (Fuel costs are not included in these calculations). Each element of this array represents the sum of all costs associated with the start-up and shutdown of system generators, costs associated with system losses and costs that result from *penalty factors* – monetary values that are charged as a result of the violation of an operational constraint [31]. *Penalty factors* are further described below. Each element of

the *pfCosts* matrix represents the sum of costs for the entire power system. This matrix therefore has a length equal to the value of *maxCombos*.

4. *comboCosts* – The elements of this table represent the total operational cost (in dollars) of the generator on/off combinations that are assessed during the particle swarm optimization process. The generator fuel costs from the *fuelCosts* table are summed with the corresponding costs from the *pfCosts* table. Like the *pfCosts* table, *comboCosts* has an array length equal to the value of *maxCombos*.

5. *comboIndices* – This table serves as an index for the other tables of the Particle Swarm module. It contains a set of reference numbers that have been assigned to each generator on/off combination evaluated during the particle swarm optimization process. Reference numbers are assigned to specific combination based on the conversion of the base-two bitstring representing the set of active and inactive generators (where active generators are represented by ones and inactive generators are represented by zeros) to a base-ten number. For example, if the eight-generator system described above were assigned the following combination by the particle swarm optimization process:

Generator 7: Active
Generator 6: Inactive
Generator 5: Inactive
Generator 4: Inactive
Generator 3: Active
Generator 2: Inactive

Generator 1: Inactive
Generator 0: Active

this combination may be represented by the binary string 10001001. If this binary number is converted to a base ten number, 137 would be the reference number assigned to the combination as:

$$2^7 + 2^3 + 2^0 = 137 \tag{5.5}$$

To reduce computational time, when the particle swarm is presented with a generator combination, it will calculate the reference number for that combination and scan the *comboIndices* table to verify that it has not already been evaluated. This process prevents the execution of duplicate load flows. This process also guarantees that the total number of load flows performed will be less than the value of *maxCombos*.

As stated above, the elements of each table are sorted based on calculated cost values. This indicates that as less expensive solutions are identified, more expensive solutions are shifted to make room. This process also ensures that corresponding values in the various tables are stored in the same order. For example, the first reference number in the *comboIndices* table has power flow costs equal to the value in the first cell of the *pfCosts* table and a total cost equal to the value of the first cell of the *comboCosts* table. If this example system were specified as having n generators, the first n elements of the *settings* and *fuelCosts* arrays would also be associated with this combination. Equivalently, the values in the *settings* and *fuelCosts* tables associated with the second

reference number in the *comboIndices* table would be located in the elements ranging from $\{ n+1 : 2n \}$.

It should be noted that this process has a linear order of complexity and could potentially require that the operating cost of each solution be compared again all other solutions. This design was chosen, however, as the developed process would be much more efficient than the complex calculus-based analyses required when unnecessary linear optimizations are performed.

5.4.2 The Particle Swarm Optimization Process

The optimization process defined by the Particle Swarm Module is executed as defined in Chapter 3. The process begins by initializing a random population of generator on/off combinations for *lightHour*. As this optimization procedure is only concerned with whether generators with minimized active power outputs should be activated or deactivated, all other (non-minimized) generators will be left in operation.

Each member of the population of generator combinations is then assigned a reference number and a validity check is performed to ensure that the system capacity remaining after all specified generators have been disabled is greater than the system load. Combinations failing to comply with this requirement are immediately discarded to prevent unnecessary load flow calculations.

All acceptable, non-duplicate generator combinations are then simulated by the software as copies of the original power system case files (as defined in Section 5.3.1) are made and the generator matrices are modified to deactivate the specified units. This modified case file may then be passed back to the calculus-based linear programming

optimizer where the active power outputs of all remaining generators are determined. At this time, system operating costs are recalculated and the Load Flow Verification Module (described below) is used to ensure that no system constraints are violated. Once this process is complete, all results are stored in the appropriate Particle Swarm Table as described above. It should be noted that if no acceptable generator on/off combinations can be found without violating system constraints, a Boolean variable known as *criticalFlag* is activated and the software returns an error to the user.

Once an iteration of the optimization process is complete and all combinations of the swarm population have been evaluated, calculations are performed to determine particle velocities [6]. These calculations follow the binary particle swarm strategy described in Chapter 3 where probabilistic methodologies are used to determine if specific generators should be activated or deactivated [24]. This process is then repeated until the specified number of iterations is reached. At this point, control is passed to the Pathfinder Module.

In the eight-unit power system example defined above, the matrices below represent the best generator on/off combinations identified by the Particle Swarm Module and their respective costs:

(For the Matrices Below: 1 = unit is active, 0 = unit is inactive)

lightHour Combination 1: [1 0 0 1 0 0 0 1] Hourly Cost = $11,500

lightHour Combination 2: [1 0 0 1 1 0 0 1] Hourly Cost = $11,800 (5.6)

lightHour Combination 3: [1 0 1 1 0 0 1 1] Hourly Cost = $12,500

⋮

5.5 The Pathfinder Module

The overall objective of the Pathfinder Module is to use the tables created by the Particle Swarm Module to determine an optimal generator schedule for the overall twenty-four hour operating period. The algorithm proposed for the Pathfinder module involves taking each proposed generator combination for *lightHour* and applying the same (or similar) settings to all other hours of the operating period. The combinations that were identified for *lightHour* may therefore be seen as a base from which the rest of the generator schedule is extrapolated.

The implemented design was chosen due to the need to enforce generator operating constraints (particularly generator ramping limits) when developing an acceptable SCUC solution. This objective may be achieved by finding a set of optimal generator settings for a particular hour and making as few changes as possible when compensating for load variations as time progresses. This may be contrasted with a design that employs particle swarm optimization for every hour. In such a design, the lack of connectivity between each hourly solution could result large variations in generator outputs over the operating period and may therefore decrease the likelihood of finding an acceptable solution.

These extrapolations are performed by taking each combination that was identified for *lightHour* in the Particle Swarm Module and using the logical 'OR' function with the matrices created by the Minimum Generators Module. As described above, these Minimum Generator Matrices indicate which generators were identified as having a minimum active power outputs at each hour of the operating period by the calculus-based solver.

This procedure is illustrated below in the continuation of the eight-unit example. As described above, the least expensive on/off combination identified by the Particle Swarm Module for *lightHour* was defined as:

lightHour Combination 1: [1 0 0 1 0 0 0 1] Hourly Cost = $11,500 (5.7)

This combination is therefore applied to all other hours of the operating period using the bitwise logical 'OR' operator with the Minimum Generator Matrices:

Generator On/Off Settings:

Hour1: [1 0 0 1 0 0 0 1] OR [1 0 0 1 0 0 0 1] = [1 0 0 1 0 0 0 1]

Hour2: [1 0 0 1 1 0 0 1] OR [1 0 0 1 0 0 0 1] = [1 0 0 1 1 0 0 1]

Hour3: [1 0 0 1 1 0 0 1] OR [1 0 0 1 0 0 0 1] = [1 0 0 1 1 0 0 1] (5.8)

Hour4: [1 0 1 1 1 0 0 1] OR [1 0 0 1 0 0 0 1] = [1 0 1 1 1 0 0 1]

Hour5: [1 0 1 1 1 1 0 1] OR [1 0 0 1 0 0 0 1] = [1 0 1 1 1 1 0 1]

⋮

The new on/off combinations are then fed back to the calculus-based solver to determine the active power outputs of all operational system units. The resulting operating costs of the prescribed settings were determined to be:

Hour 1 Operating Cost:	$11,500
Hour 2 Operating Cost:	$12,500
Hour 3 Operating Cost:	$13,600

(5.9)

Hour 4 Operating Cost: $14,300

Hour 5 Operating Cost: $15,500

⋮

Once all operating costs have been calculated for the overall SCUC generator schedule, the process is repeated using a different combination for *lightHour*. This process therefore involves cycling through as many combinations as possible until either the computational time limit for the SCUC program expires or all *lightHour* possibilities have been exhausted.

To ensure power system reliability, all hourly generator combinations are analyzed by the Load Flow Verification Module (described below). If it is found that a particular set of generator on/off settings results in the violation of power system constraints, the overall generator schedule is rejected and the software moves on to the next *lightHour* combination on which a new schedule may be developed. In the unlikely event of a case where all SCUC solutions are in violation of system constraints, a *criticalFlag* variable is activated and the software notifies the user of the error.

Another important attribute of the Pathfinder Module is the capacity to ensure that generator ramping limits are enforced for the operating period [11]. This is accomplished as the maximum and minimum active power output limits for all units are recalculated for each hour based on the settings for the previous hour. As an example of this functionality, a generator that is rated to produce an active power output in the range of 30-200MW and that has a ramping limit of 50MW may be considered. If the prescribed active power output for Hour 1 is 130MW, the Pathfinder Module will modify the allowable operating

range of the unit for Hour 2 to ensure that the output stays within the range of 80-180MW. It should also be noted that the Pathfinder Module has a *looping* feature to ensure continuity between the end of one operating period and the beginning of the next.

5.5.1 Review of the Pathfinder Methodology

The proposed Pathfinder procedure will result in a generator schedule where unit settings for consecutive hours will be very similar due to the incorporation of attributes such as ramping limits. This is desirable as frequent switching on and off of a system generator may result in a reduced lifespan for the unit. As described in Chapter 2, previous investigations [48] have found that if particle swarms were to be performed for each hour of the operating period, the generator schedule would be highly discontinuous and could potentially violate minimum on/off time constraints of system units.

Disadvantages of the proposed methodology are such that the search for the optimal generator schedule for the operating period will involve a very limited exploration of the overall search-space. This is due to the fact that the optimization strategy does not attempt to optimize each hour of the operating period.

5.5.2 Pathfinder Module Record Keeping

As part of its procedure, the Pathfinder Module creates a table know as *UCpath* – a data structure that contains all of the pertinent information for the twenty-four hour SCUC generator schedule. This table contains the set of all operational costs as well as the prescribed generator active power output settings for all units. The table is designed

such that each row contains information for a specific hour of the operating period. In a day-ahead SCUC planning exercise, the *UCpath* table would therefore contain twenty-four rows.

The operational data contained within the *UCpath* table are stored in a series of fields that are defined as follows:

Field 1 – *comboNumber* – This field contains the reference numbers of the generator on/off combination employed for each hour of the operating period.

Field 2 – *runningTotal* – This field represents the total operational cost of the system.

Field 3 – *comboCost* – This field represents the hourly overall operating costs for the system

Field 4 – *comboFuelCost* – This field represents the hourly fuel costs for the system

Field 5 – *comboTimeCost* – This field represents the hourly generator costs for the system (excluding fuel costs). Generator start-up and shutdown costs are considered along with penalty factors incurred due to violations of minimum on/off time contraints and ramping limits.

Field 6 – *comboPFCost* – This field represents the hourly power flow costs for the system. Penalty factors incurred due to power flow violations and expenses due to system losses are considered for this total.

Field 7 – *comboSimilarity* – This field serves as means of tracking the similarity of the generator on/off combinations for consecutive hours of the

operating period. This figure represents the sum of the units that have been switched on or off during each hour.

In addition to these columns, the *UCpath* table also contains information relating to the prescribed active power settings for each generator of the power system. These values represent the optimal outputs calculated by the calculus-based solver as defined above. The overall number of columns of the *UCpath* table will therefore equal to seven (the number of fields) plus the total number of generator units.

As defined above, all generator schedules created by the Pathfinder Module are extrapolated based on the set of generator on/off combinations for *lightHour*. The *UCpath* table will therefore be filled with a unique generator schedule based on each *lightHour* combination. This schedule will then be erased as the process is repeated. Before being erased, the overall operating cost of each schedule is evaluated and compared to that of the best SCUC solution to date. If the cost of a particular solution is found to be lower, the entire *UCpath* table is copied to a new table known as *bestPath*. The *bestPath* table therefore contains all prescribed generator settings and operational costs for the best possible SCUC solution identified during the Pathfinder procedure. At the end of this procedure, this table is passed to the Output Module where all required information may be provided to the user.

5.6 The Load Flow Verification Module

The Load Flow Verification Module works in conjunction with both the Particle Swarm and the Pathfinder modules and is used to ensure that all generator on/off combinations do not violate any of the power system constraints defined in Section 4.2.1. This software is located in a separate Matlab m-file and a separate function known as *lfCheck* that may be called as required. If any load flow violations are detected by this module, a penalty factor fee is added to the calculated power system operating cost consisting of fuel costs, etc. As all power flow restrictions must be met to ensure the reliable operation of the power system, these penalty factors are significantly large and will cause the SCUC software to immediately reject the proposed generating scheme. This strategy may therefore be seen as a combination of both penalty factor and feasible solution methodologies [31].

5.7 The Output Module

At the completion of the procedure performed by the Pathfinder Module, the *bestPath* table containing all generator settings and operational cost data is forwarded to the Output Module for the final stages of the SCUC software. The Output Module performs a series of tasks including verification of an acceptable SCUC solution, reordering of the generator scheduling, and the plotting of output graphs.

The first task performed by the Output Module involves verification of the *criticalFlag* variable to ensure that an acceptable SCUC solution has been found. If no

such solution was found, the user is notified of the error and the program immediately terminates.

The second objective of the Output Module is to reorganize the data of the *bestPath* table such that it is in the proper chronological order. As stated above, the first rows of the *bestPath* and *UCpath* tables contain data for the hour known as *lightHour*. It should be noted, however, that *lightHour* may not be the first hour of the operating period. The Output Module must therefore reconfigure this data as it is copied to a new table known as *orderedPath*. As part of this process, the *runningTotal* field of the *bestPath* table must be recalculated starting from the first hour of the operating period.

Once this process is complete, the Output Module outputs a series of graphs that may be analyzed by power systems engineers so that the final proposed SCUC solution may be approved for implementation. Two such graphs are printed and these include (*a*) a graph that displays the active output of all generating units for the overall operating period and (*b*) a graph that illustrates the hourly operating costs of the power system. Graph (*b*) contains a series of plots associated with fuel costs, generator operational costs, and system costs associated with losses.

Once all required graphs have been generated, the SCUC program terminates its operation.

Chapter 6

SCUC Case Studies and Results

For the evaluation of the proposed SCUC planning methodology, a set of case studies were performed involving simulated power systems and day-ahead planning scenarios. For each system, the SCUC software was required to generate a twenty-four hour generation scheme that would meet all system constraints at an operating cost lower than the results obtained using only the calculus-based linear programming methodology [23] employed by existing Matpower software [52].

This chapter provides an overview of the power system case studies and the specifications that were defined for the software simulations. The benchmark simulations involving the linear programming methodology are presented along with the resulting generation schemes. This chapter also discusses the simulations that were performed using the particle swarm software. The case studies are discussed with reference to the software modules presented in Chapter 5 and the simulation results are presented and compared to those of the benchmark system.

6.1 Power System Case Studies and Simulation Specifications

The simulated power systems chosen for the case studies were based on a number of networks from an IEEE test case archive [53]. These systems include a 57-bus system with 7 thermal units, and a 118-bus test system with 54 thermal units. All operating parameters for the systems are defined in Appendices E and F, respectively, where Matlab case files for both systems are provided. For each of these power systems,

simulated load factors were varied uniformly over the operating period as illustrated by Figure 6.1. This curve is based on actual loading data from CAISO, The California Independent System Operator [54]. This corporation is responsible for overseeing the day-ahead energy market in California and ensuring the reliable operation of the power system.

Simulations for both test cases were performed using MATLAB 6.5 and the Matpower 3.0.0 toolbox. Applications were executed on laptop computer with an Intel Celeron 2.2GHz processor. The Windows XP operating platform was employed with 3.5GB of available virtual memory.

With respect to the ability of the software to meet specified time constraints, the simulations were performed to represented day-ahead planning situations. As a result, a computational time limit of sixteen hours was specified. (Such a timeframe would provide adequate opportunities for the verification of the generated results and the implementation of the generation scheme within the corporate energy management system). It should be noted, however, that analyses relating to the computational complexity and the time requirements associated with the various processes of the implemented software are beyond the scope of this investigation. Although the software has been developed using a modular format, the modules have been defined for the purposes of illustrating the various methodologies that have been employed for the SCUC planning process. The computing resources required for each module are therefore not considered as part of the performance evaluation of the software. Such considerations are described in Chapter 7 as part of the recommendations for future work.

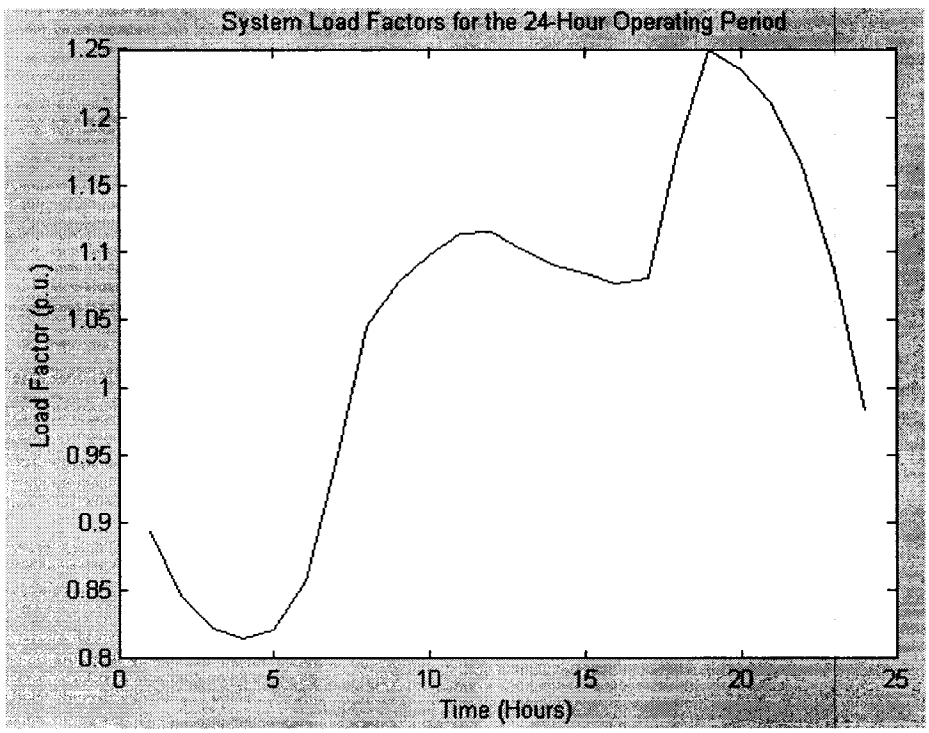

Figure 6.1 System Load Factors for the 24-Hour Operating Period

6.2 System Optimization Using Only Calculus-Based Solvers

Both power system test cases were first optimized for the specified operating period using existing calculus-based linear programming software. In this portion of the investigation, the *runopf* function Matpower 3.0 toolbox was used to optimize the power system for each hour of the operating period. As this software does not have the capacity to deactivate system generators as a potential means of reducing costs, all generators were kept in operation for the entire twenty-four hours. Benefits for this strategy include the fact that computational time is minimized as no time is spent optimizing the

activation and deactivation of system generators. In addition, planners need not be concerned with generator constraints relating to minimum on/off times.

6.2.1 Linear Programming Optimization of the 57-Bus System

The Matpower software was used to optimize the active power outputs of the seven generators of the 57-bus power system. The results of this optimization are specified in Figure 6.2 and Table 6.1 which illustrate the prescribed generator settings for each hour of the operating period.

As a result of this optimization, the operating costs for the system over the twenty-hour hour period are illustrated in Figure 6.3 and Table 6.2. These figures illustrate costs due to generator fuel consumption and system losses. It may be noted that as all generators remain active for the entire operating periods, no costs associated with generator start-up are incurred. Based on these values, the total estimated operating costs for the power system is approximated as $275,300.

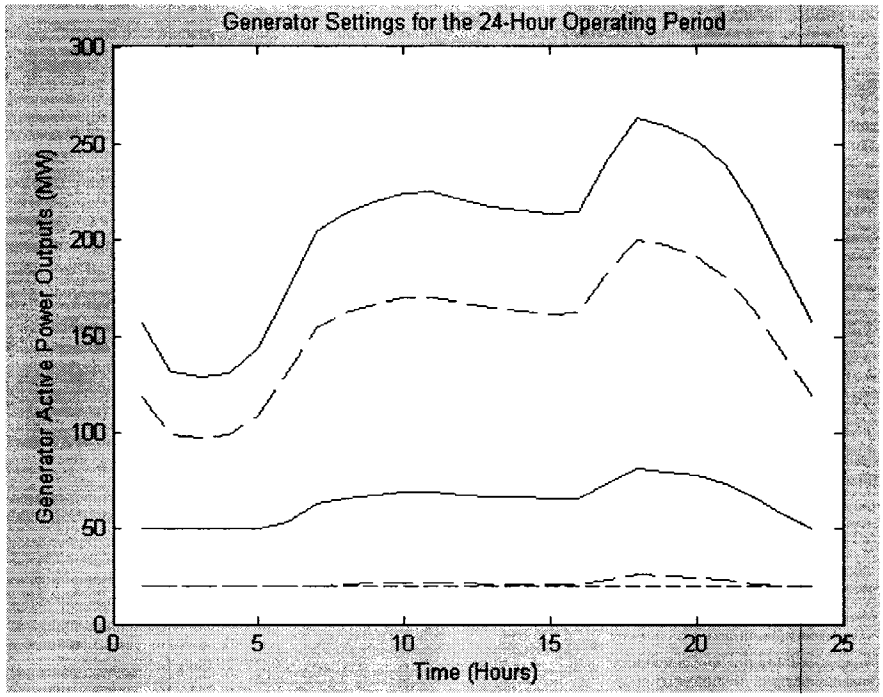

Figure 6.2 LP-Optimized Generator Settings for the 57-Bus Power System
(Three generators are operating at minimal values)

Table 6.1 LP-Optimized Scheme for the 57-Bus Power System (MW)

Hour>>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Gen 1	50	50	50	50	50	53	62	65	67	69	69	68	67	66	65	66	74	81	80	77	73	66	57	50
Gen 2	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
Gen 3	20	20	20	20	20	20	20	21	21	22	22	21	21	21	21	21	23	25	25	24	23	21	20	20
Gen 4	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
Gen 5	157	131	128	131	144	173	204	214	220	224	225	221	217	216	213	215	243	264	260	252	239	216	185	157
Gen 6	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
Gen 7	118	99	97	99	108	131	154	162	166	169	170	167	164	163	161	162	184	200	197	191	181	164	140	118

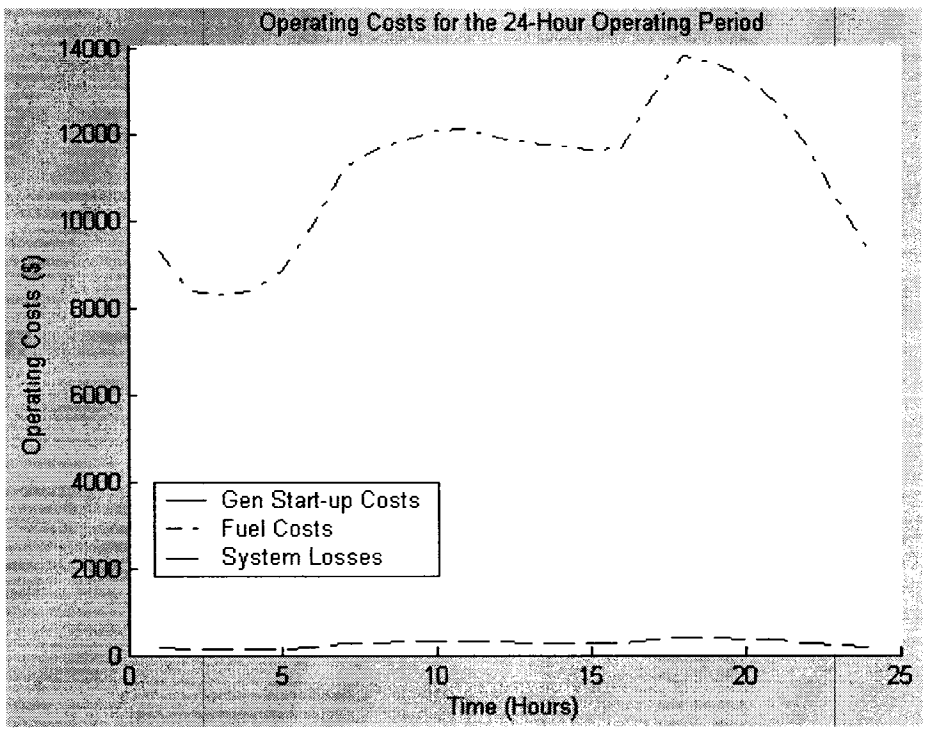

Figure 6.3 LP-Optimized Operating Costs for the 57-Bus Power System

Table 6.2 LP-Optimized Operating Costs for the 57-Bus Power System

Hour	Fuel Cost	Gen Start-up Cost	System Losses	Total
1	$ 9,314.60	$ -	$ 177.59	$ 9,492.10
2	$ 8,423.10	$ -	$ 146.60	$ 8,569.70
3	$ 8,325.30	$ -	$ 143.64	$ 8,468.90
4	$ 8,400.00	$ -	$ 145.89	$ 8,545.90
5	$ 8,855.40	$ -	$ 160.70	$ 9,016.10
6	$ 9,963.10	$ -	$ 202.99	$ 10,166.00
7	$ 11,236.00	$ -	$ 258.14	$ 11,495.00
8	$ 11,644.00	$ -	$ 277.34	$ 11,922.00
9	$ 11,904.00	$ -	$ 289.79	$ 12,194.00
10	$ 12,095.00	$ -	$ 299.12	$ 12,394.00
11	$ 12,120.00	$ -	$ 300.34	$ 12,420.00
12	$ 11,945.00	$ -	$ 291.75	$ 12,236.00
13	$ 11,805.00	$ -	$ 284.98	$ 12,090.00
14	$ 11,733.00	$ -	$ 281.55	$ 12,014.00
15	$ 11,626.00	$ -	$ 276.50	$ 11,903.00
16	$ 11,684.00	$ -	$ 279.20	$ 11,963.00
17	$ 12,916.00	$ -	$ 341.29	$ 13,257.00
18	$ 13,826.00	$ -	$ 391.80	$ 14,218.00
19	$ 13,642.00	$ -	$ 381.24	$ 14,023.00
20	$ 13,314.00	$ -	$ 362.91	$ 13,677.00
21	$ 12,736.00	$ -	$ 331.75	$ 13,067.00
22	$ 11,762.00	$ -	$ 282.93	$ 12,045.00
23	$ 10,448.00	$ -	$ 222.81	$ 10,671.00
24	$ 9,314.60	$ -	$ 177.59	$ 9,492.10
Total	**$ 269,032.10**	$ -	**$ 6,308.44**	**$ 275,339.80**

6.2.2 Linear Programming Optimization of the 118-Bus System

The Matpower linear programming software was also used to optimize the active power outputs of the fifty-four generators of the 118-bus power system. The results of this optimization are specified in Figure 6.4 and Table 6.3 which illustrate the prescribed generator settings for all hours of the operating period.

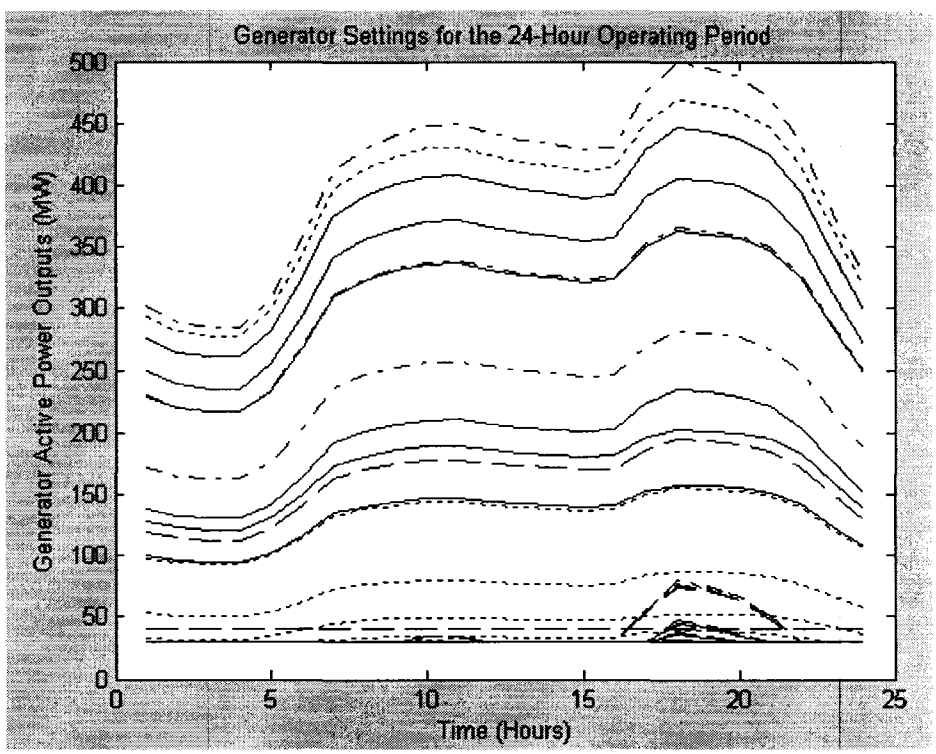

Figure 6.4 LP-Optimized Generator Settings for the 118-Bus Power System (Twenty-eight generators are operating at minimal values)

Table 6.3 LP-Optimized Scheme for the 118-Bus Power System (MW)

Hour>>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Gen 1	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	45	42	34	30	30	30	30
Gen 2	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 3	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 4	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 5	250	239	235	235	254	295	341	355	365	371	372	366	361	359	355	357	391	406	404	400	387	360	312	271
Gen 6	53	50	50	50	54	63	73	76	78	80	80	78	77	77	76	76	84	88	87	86	83	77	66	57
Gen 7	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	38	34	30	30	30	30	30
Gen 8	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 9	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	36	33	30	30	30	30	30
Gen 10	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 11	119	114	112	112	122	141	163	170	174	177	178	175	173	172	170	171	187	195	194	191	185	172	149	130
Gen 12	172	165	162	162	176	203	235	246	252	257	257	253	250	248	245	247	270	281	280	276	267	249	216	187
Gen 13	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 14	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 15	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 16	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 17	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	31	30	30	30	30	30	30
Gen 18	30	30	30	30	30	30	30	30	30	32	32	30	30	30	30	30	56	77	74	67	51	30	30	30
Gen 19	30	30	30	30	30	30	30	30	30	34	35	30	30	30	30	30	55	74	71	64	51	30	30	30
Gen 20	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 21	128	123	121	121	131	151	174	181	186	189	189	187	184	183	181	182	196	203	201	199	195	183	159	139
Gen 22	33	32	31	31	34	39	45	47	48	49	49	49	48	48	47	47	51	52	52	51	50	48	41	36
Gen 23	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	53	76	72	64	48	30	30	30
Gen 24	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	55	79	75	66	49	30	30	30
Gen 25	99	95	94	94	101	117	135	140	144	146	147	145	143	142	140	141	152	158	157	155	151	142	124	108
Gen 26	98	94	92	92	100	115	132	138	141	143	144	142	140	139	137	138	150	155	154	153	148	139	121	106
Gen 27	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 28	230	220	217	217	234	270	311	324	332	338	338	333	329	327	323	325	353	366	364	359	350	327	285	249
Gen 29	229	220	217	217	234	269	309	322	330	336	336	332	327	325	322	323	351	363	361	357	347	326	284	248
Gen 30	294	281	277	277	299	344	396	413	423	430	431	425	419	416	412	414	451	468	465	460	447	418	364	318
Gen 31	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 32	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 33	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 34	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	32	30	30	30	30	30	30
Gen 35	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	48	44	35	30	30	30	30
Gen 36	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 37	276	265	261	261	281	325	375	391	401	407	408	402	397	394	390	392	429	447	444	438	424	395	344	299
Gen 38	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 39	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 40	302	289	285	285	308	357	412	430	441	448	449	442	436	433	429	431	476	499	495	488	470	435	378	328
Gen 41	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 42	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 43	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 44	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 45	139	132	130	130	141	165	193	201	207	211	211	208	205	203	201	202	224	234	233	229	221	204	176	151
Gen 46	30	30	30	30	30	30	31	33	34	34	35	34	33	33	33	33	37	38	38	38	36	33	30	30
Gen 47	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 48	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 49	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	36	33	30	30	30	30	30
Gen 50	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 51	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
Gen 52	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	43	40	35	30	30	30	30
Gen 53	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 54	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30

As a result of this optimization, the operating costs for the system over the twenty-four hour period are illustrated in Figure 6.5 and Table 6.4. As with Test Case 1, these figures illustrate costs due to generator fuel consumption and system losses. Again, as all generators remain active for the entire operating periods, the Test Case does not involve costs associated with generator start-up. Based on calculated values, the total estimated operating costs for the power system is approximated as $2.813M.

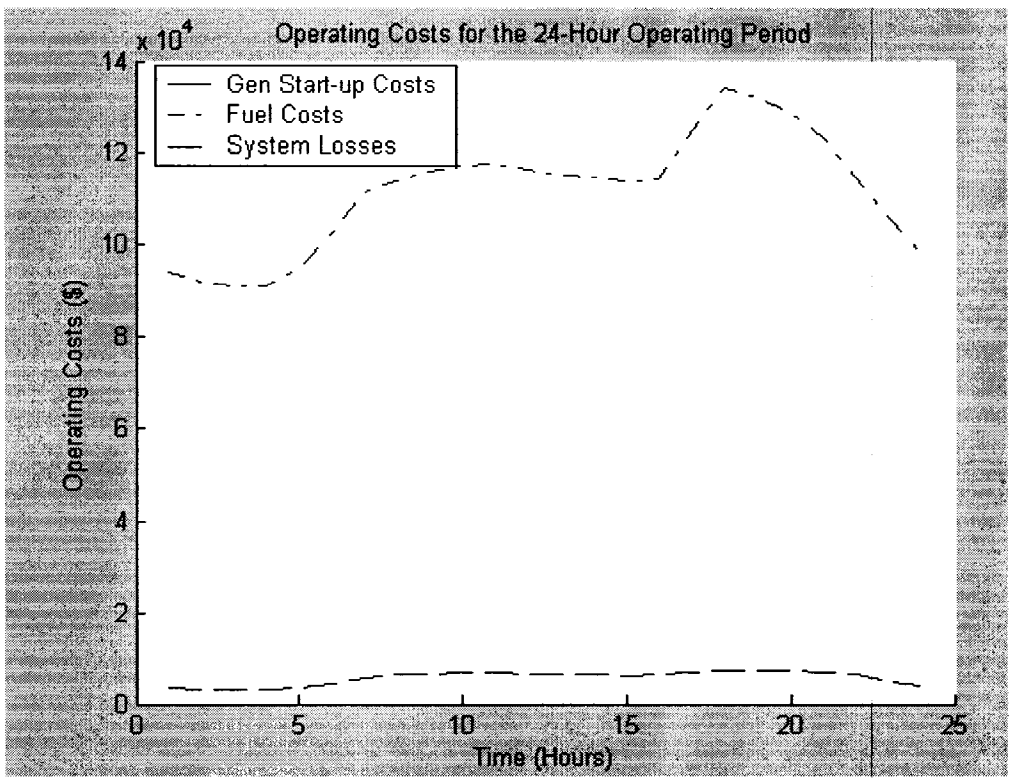

Figure 6.5 LP-Optimized Operating Costs for the 118-Bus Power System

Table 6.4 LP-Optimized Operating Costs for the 118-Bus Power System

Hour	Fuel Cost	Gen Start-up Cost	System Losses	Total
1	$ 91,161.00	$ -	$ 3,025.90	$ 94,187.00
2	$ 94,826.00	$ -	$ 3,431.20	$ 98,257.00
3	$ 102,500.00	$ -	$ 4,393.00	$ 106,890.00
4	$ 111,360.00	$ -	$ 5,695.00	$ 117,050.00
5	$ 114,200.00	$ -	$ 6,155.60	$ 120,360.00
6	$ 116,020.00	$ -	$ 6,460.70	$ 122,480.00
7	$ 117,450.00	$ -	$ 6,641.00	$ 124,100.00
8	$ 117,660.00	$ -	$ 6,656.90	$ 124,310.00
9	$ 116,300.00	$ -	$ 6,508.60	$ 122,810.00
10	$ 115,320.00	$ -	$ 6,342.60	$ 121,660.00
11	$ 114,820.00	$ -	$ 6,258.50	$ 121,080.00
12	$ 114,080.00	$ -	$ 6,134.90	$ 120,210.00
13	$ 114,480.00	$ -	$ 6,201.00	$ 120,680.00
14	$ 124,880.00	$ -	$ 6,956.00	$ 131,840.00
15	$ 134,140.00	$ -	$ 7,142.70	$ 141,280.00
16	$ 132,130.00	$ -	$ 7,130.10	$ 139,260.00
17	$ 128,710.00	$ -	$ 7,084.90	$ 135,800.00
18	$ 123,240.00	$ -	$ 6,880.00	$ 130,120.00
19	$ 115,020.00	$ -	$ 6,292.50	$ 121,320.00
20	$ 105,870.00	$ -	$ 4,864.50	$ 110,740.00
21	$ 98,005.00	$ -	$ 3,810.60	$ 101,820.00
22	$ 93,905.00	$ -	$ 3,326.20	$ 97,231.00
23	$ 91,837.00	$ -	$ 3,098.80	$ 94,936.00
24	$ 91,161.00	$ -	$ 3,025.90	$ 94,187.00
Total	**$ 2,679,075.00**	$ -	**$ 133,517.10**	**$ 2,812,608.00**

6.3 Test Case Optimization Using SCUC Software

Simulated day-ahead planning for both power systems was performed using the implemented SCUC software described in Chapter 5. The sections below contain descriptions of the operation of the various software modules.

6.3.1 Test Case Optimization: The Input Module

As described above, system case files containing all power flow parameters were input in the specified formats along with the twenty-four hour loading data. Energy costs

were also input to allow for the calculation of system losses. This value was chosen based on Newfoundland Power's electrical rate of 8.458 ¢/kWh [55].

For both power systems, particle swarm parameters were provided for the Input Module according to the recommendations of Kennedy and Eberhart [27]. While the size of the particle swarm population and the permitted number of search iterations varied for each power system, values of 0.1, 0.5, and 0.5 were chosen for the inertia weight, w, and the acceleration coefficients, c_1 and c_2, respectively. These values allow for improved stability in the movement of the swarm and would therefore be more likely to produce a higher-quality result. As well, the maximum particle speed, *Vmax*, was set to a value of four [28].

6.3.2 Test Case Optimization: The Minimum Generators Module

As described above, the minimum generators module is responsible for identifying power system generators that have been minimized by the calculus-based linear programming solver. The results of this procedure for the 57-bus and 118-bus systems are illustrated in Table 6.5 and Table 6.6, respectively. As indicated in these tables, the maximum number of minimum generators in the 57-bus system was observed during the first five hours of operation. During this period, the outputs of five system generators were minimized. For the 118-bus system, the number of minimum generators was maximized in Hour 4 where 41 of the 54 generators are minimized. Based on these findings, the *lightHour* variable may be defined as Hour 1 for the first test case (selected arbitrarily from the first five hours) and as Hour 4 for the second.

Table 6.5 Minimum and Non-Minimum Generators for the 57-Bus Power System

(Zeros indicate minimum generators, while ones indicate non-minimum generators)

Hour>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Gen 1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Gen 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MinGens	5	5	5	5	5	5	4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4

Table 6.6 Minimum and Non-Minimum Generators for the 118-Bus Power System

(Zeros indicate minimum generators, while ones indicate non-minimum generators)

Hour>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Gen 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
Gen 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 6	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
Gen 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
Gen 10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Gen 18	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	1	1	0	0
Gen 19	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	1	1	0	0
Gen 20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
Gen 24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
Gen 25	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 26	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 28	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 29	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Gen 35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
Gen 36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 37	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 40	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 45	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gen 46	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Gen 47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
Gen 50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
Gen 53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinGens	40	40	40	41	40	40	40	39	39	39	37	37	39	39	39	39	39	35	27	29	32	35	39	40

6.3.3 Test Case Optimization: The Particle Swarm Module

Once *lightHour* was identified for both test cases, particle swarm optimizations were performed to determine the least expensive generator on/off combinations for the specified hours.

The optimization procedure was greatly simplified for the first test case as the five minimum generators produced a relatively small search-space of $2^5 = 32$ possible generator on/off combinations. Based on this result, a particle swarm with a population of twenty was selected and permitted to search for twenty iterations. Although these selections may seem very large for such a small search-space, the functionality built in to the Particle Swarm Module prevents the evaluation of duplicate load flows. As a result, the swarm was able to quickly exhaust all thirty-two possibilities without unnecessary calculations. Therefore, for small cases such as these, the software is capable of indirectly performing exhaustive enumeration.

The results of this procedure are illustrated in Table 6.7 which lists the ten best generator combinations identified by the Particle Swarm Module.

Table 6.7 Particle Swarm Results for the 57-Bus Power System

Combo>	1	2	3	4	5	6	7	8	9	10
Gen 1	61	0	0	54	55	55	57	58	58	0
Gen 2	0	0	0	20	0	0	20	0	0	20
Gen 3	0	22	0	20	20	20	0	0	0	20
Gen 4	0	0	0	0	0	20	0	0	20	0
Gen 5	198	219	232	179	178	177	189	187	187	209
Gen 6	0	0	0	0	20	0	0	20	0	0
Gen 7	147	165	175	133	133	133	140	140	141	157
Cost	$8,325	$8,353	$8,436	$8,497	$8,712	$8,714	$8,718	$8,734	$8,741	$8,742

The benefits of the particle swarm optimization are much more evident in the second test case where there is an extremely large search-space of $2^{41} = 2.2 \times 10^{12}$ possible generator on/off combinations. While it would be impossible to exhaust all possibilities as in the previous test-case, the software must rely on the capability of the SCUC software to limit its search to areas of the hypercube where an optimal solution is more likely to be found.

A population size of twenty was once again chosen. This specification was made as research has indicated that even for extremely large search-spaces, search efficiency does not dramatically improve with a larger swarm population [25]. Instead, the swarms are to be given a longer time in which to perform their search. As a result, a limit of 100 iterations was selected. Based on these specifications, the Particle Swarm Module is permitted to search a maximum of two thousand of the over two trillion possibilities.

The progress of the swarm is shown in Figure 6.6 which illustrates how the overall operating price for the hour was reduced as new and more efficient generator combinations were identified. As shown, the swarm gradually reduced the hourly operating price for *lightHour* from approximately $83,500 to under $81,000.

The ten best generator combinations and their respective operating costs for the second test case are illustrated in Table 6.8. It should be noted that 1982 generator on/off combinations were assessed for the second test case and that all of these settings were passed along to the Pathfinder Module. It was found that 18 combinations were either duplicates or were rejected due to violation of system constraints.

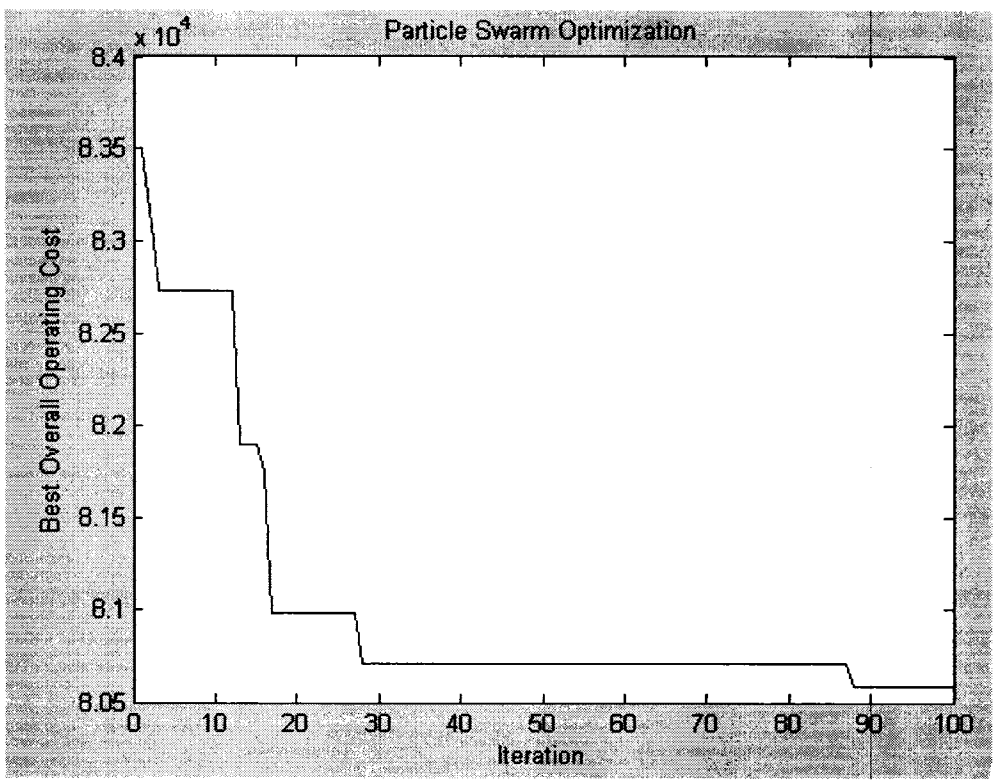

Figure 6.6 Progress of the Particle Swarm for the 118-Bus System

Table 6.8 Particle Swarm Results for the 118-Bus Power System

Combo>	1	2	3	4	5	6	7	8	9	10
Gen 1	0	0	0	0	0	0	30	30	0	0
Gen 2	0	0	0	0	0	30	0	0	0	30
Gen 3	30	0	0	0	0	30	30	0	30	0
Gen 4	158	165	168	162	162	164	161	158	158	155
Gen 5	228	237	243	235	234	237	232	227	228	225
Gen 6	0	0	0	0	0	0	0	0	0	30
Gen 7	30	0	0	30	30	0	0	30	30	30
Gen 8	0	30	0	30	0	0	30	0	0	0
Gen 9	30	0	0	30	0	30	0	0	0	0
Gen 10	30	30	0	0	30	0	30	30	30	0
Gen 11	0	0	0	0	0	30	0	0	30	0
Gen 12	0	0	0	0	30	0	0	0	0	0
Gen 13	30	30	0	0	0	0	0	0	0	30
Gen 14	159	166	167	165	165	166	167	162	160	161
Gen 15	41	43	42	42	42	43	43	42	41	41
Gen 16	30	0	30	0	0	0	0	0	30	30
Gen 17	0	0	0	0	0	0	0	0	0	0
Gen 18	122	128	127	126	127	127	128	124	123	124
Gen 19	121	127	126	125	126	126	127	123	122	123
Gen 20	0	30	30	0	0	0	0	0	30	0
Gen 21	288	302	302	296	298	299	300	292	290	292
Gen 22	286	300	300	294	296	297	299	290	288	290
Gen 23	371	390	389	379	383	384	387	375	374	374
Gen 24	0	0	0	0	0	0	0	0	0	30
Gen 25	0	0	0	0	0	0	0	0	0	30
Gen 26	0	0	0	30	0	30	0	0	0	0
Gen 27	0	30	0	0	30	0	0	0	0	30
Gen 28	30	0	0	0	0	0	0	30	0	0
Gen 29	30	0	0	30	30	0	0	0	0	0
Gen 30	350	372	366	356	363	363	366	352	353	359
Gen 31	0	0	30	0	30	0	0	0	0	0
Gen 32	30	0	30	30	0	30	30	0	30	0
Gen 33	405	440	411	405	420	408	409	399	405	425
Gen 34	0	0	0	0	0	30	0	0	0	0
Gen 35	0	0	0	0	0	0	30	30	0	0
Gen 36	0	0	0	0	0	0	30	0	0	0
Gen 37	0	0	0	0	0	30	0	30	0	0
Gen 38	189	208	195	188	199	195	199	183	188	199
Gen 39	31	0	33	0	34	33	0	30	31	0
Gen 40	0	0	0	0	30	0	0	0	0	0
Gen 41	30	0	30	30	0	0	0	0	0	0
Gen 42	0	0	30	30	0	30	0	0	30	0
Gen 43	0	0	0	30	0	0	0	30	30	30
Gen 44	40	0	0	40	0	0	40	40	0	0
Gen 45	0	30	0	0	0	0	0	30	30	30
Gen 46	30	0	0	0	30	0	30	30	30	0
Gen 47	0	0	0	30	0	0	0	0	0	0
Gen 48	0	0	0	0	0	0	30	0	0	0
Gen 49	30	30	0	0	0	0	0	30	0	0
Gen 50	0	0	0	0	0	30	30	0	0	0
Gen 51	30	0	0	30	0	0	0	0	30	0
Gen 52	339	347	341	336	343	334	324	328	326	335
Gen 53	72	73	72	71	73	71	68	70	69	71
Gen 54	0	30	0	0	0	0	0	0	30	0
Cost	$80,590	$80,708	$80,935	$80,976	$81,086	$81,292	$81,477	$81,624	$81,635	$81,686

6.3.4 Test Case Optimization: The Pathfinder and Output Modules

For both test cases, the Pathfinder Module was used to extrapolate an optimized generation scheme for the overall operating period. This task is accomplished by cycling through as many combinations as possible of those identified by the Particle Swarm Module.

For the 57-bus power system of Test Case 1, the generation scheme illustrated in Figure 6.7 and Table 6.9 was generated by the Output Module. This proposed scheme had an estimated operating cost of $247,500 for the twenty-four hour period. This value represents a savings of 11.7% over the costs of the linear programming generator schedule, provided above in Table 6.2. Operating costs for the SCUC-optimized solution are provided in Figure 6.8 and Table 6.10. As the generation scheme met all operating constraints and demonstrated an improved efficiency, this may be seen as an acceptable SCUC solution.

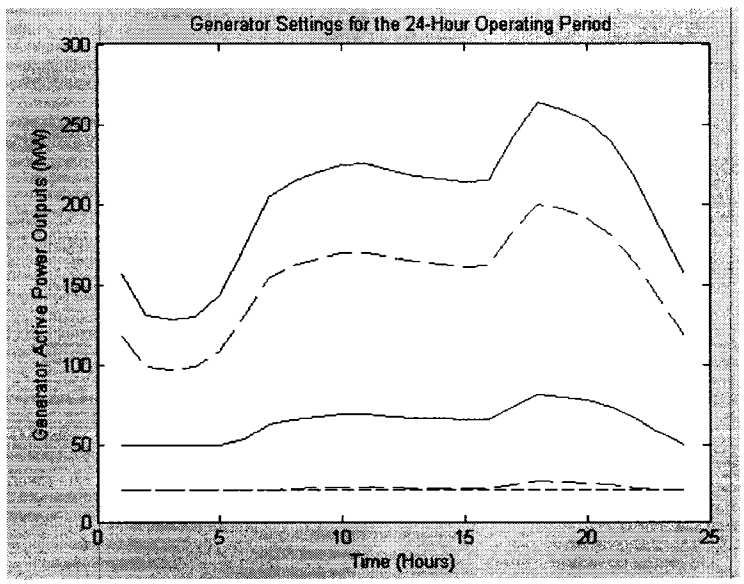

Figure 6.7 SCUC-Optimized Generation Scheme for the 57-Bus Power System

Table 6.9 SCUC-Optimized Scheme for the 57-Bus Power System (MW)

Hour>>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Gen 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 3	20	20	20	20	20	21	25	26	27	27	27	27	26	26	26	26	29	32	31	31	29	26	23	20
Gen 4	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
Gen 5	196	171	168	170	183	213	248	258	265	271	271	267	263	261	258	259	293	317	312	303	288	262	226	196
Gen 6	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
Gen 7	150	131	129	130	140	164	190	199	204	208	208	205	202	200	198	199	225	244	240	233	221	201	174	150

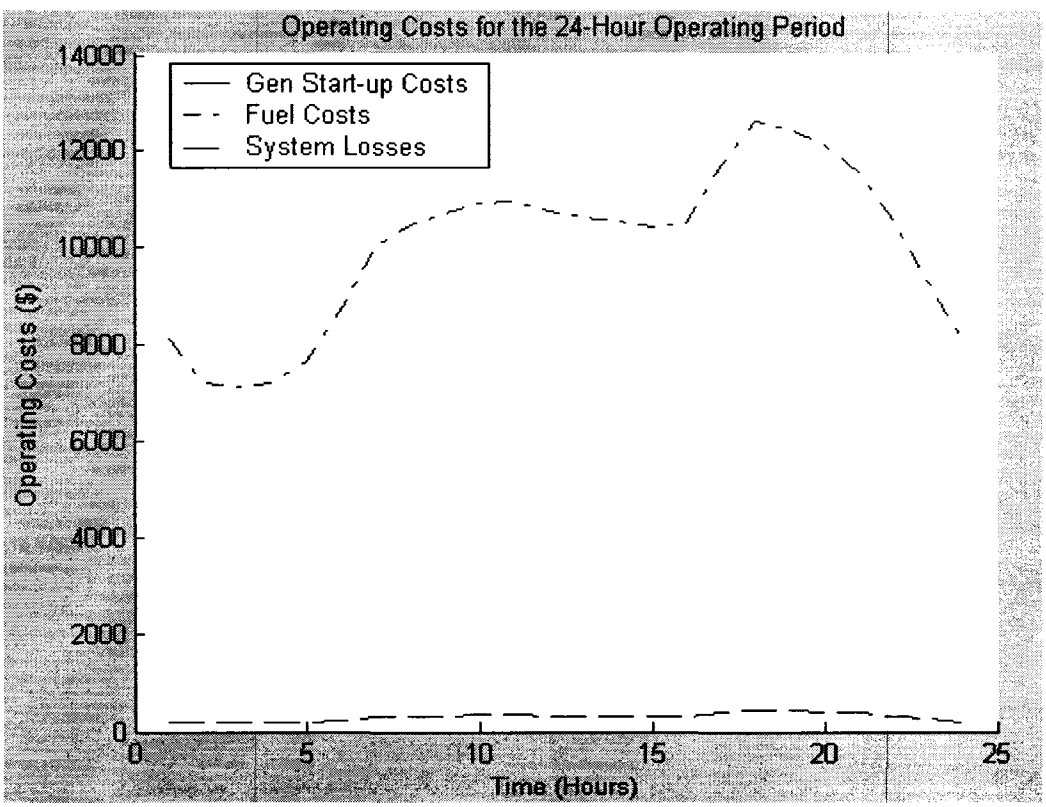

Figure 6.8 SCUC-Optimized Operating Costs for 57-Bus Power System

Table 6.10 SCUC-Optimized Operating Costs for 57-Bus Power System

Hour	Fuel Cost	Gen Start-up Cost	System Losses	Total
1	$ 8,122.00	$ -	$ 203.11	$ 8,325.10
2	$ 7,229.20	$ -	$ 168.04	$ 7,397.30
3	$ 7,131.20	$ -	$ 164.44	$ 7,295.60
4	$ 7,206.10	$ -	$ 167.19	$ 7,373.20
5	$ 7,662.20	$ -	$ 184.49	$ 7,846.70
6	$ 8,771.30	$ -	$ 231.33	$ 9,002.60
7	$ 10,046.00	$ -	$ 290.65	$ 10,337.00
8	$ 10,454.00	$ -	$ 311.47	$ 10,766.00
9	$ 10,715.00	$ -	$ 325.19	$ 11,040.00
10	$ 10,906.00	$ -	$ 335.45	$ 11,241.00
11	$ 10,930.00	$ -	$ 336.80	$ 11,267.00
12	$ 10,755.00	$ -	$ 327.34	$ 11,083.00
13	$ 10,615.00	$ -	$ 319.88	$ 10,935.00
14	$ 10,543.00	$ -	$ 316.11	$ 10,859.00
15	$ 10,436.00	$ -	$ 310.54	$ 10,747.00
16	$ 10,494.00	$ -	$ 313.52	$ 10,807.00
17	$ 11,728.00	$ -	$ 381.75	$ 12,109.00
18	$ 12,639.00	$ -	$ 437.01	$ 13,076.00
19	$ 12,454.00	$ -	$ 425.47	$ 12,880.00
20	$ 12,127.00	$ -	$ 405.44	$ 12,532.00
21	$ 11,547.00	$ -	$ 371.29	$ 11,918.00
22	$ 10,572.00	$ -	$ 317.63	$ 10,890.00
23	$ 9,256.80	$ -	$ 253.08	$ 9,509.90
24	$ 8,122.00	$ -	$ 203.11	$ 8,325.10
Total	**$ 240,461.80**	**$ -**	**$ 7,100.33**	**$ 247,562.50**

For the 118-bus power system of Test Case 2, the SCUC software produced the generation scheme illustrated in Figure 6.9 and Table 6.11. This proposed scheme had an estimated operating cost of $2.588M for the twenty-four hour period. This represents a savings of 8.0% over the costs of the linear programming solution, illustrated above in Table 6.4. Operating costs for the SCUC-optimized solution are provided in Figure 6.10 and Table 6.12. Again, this generation scheme meets all operating constraints and improves the efficiency of the system.

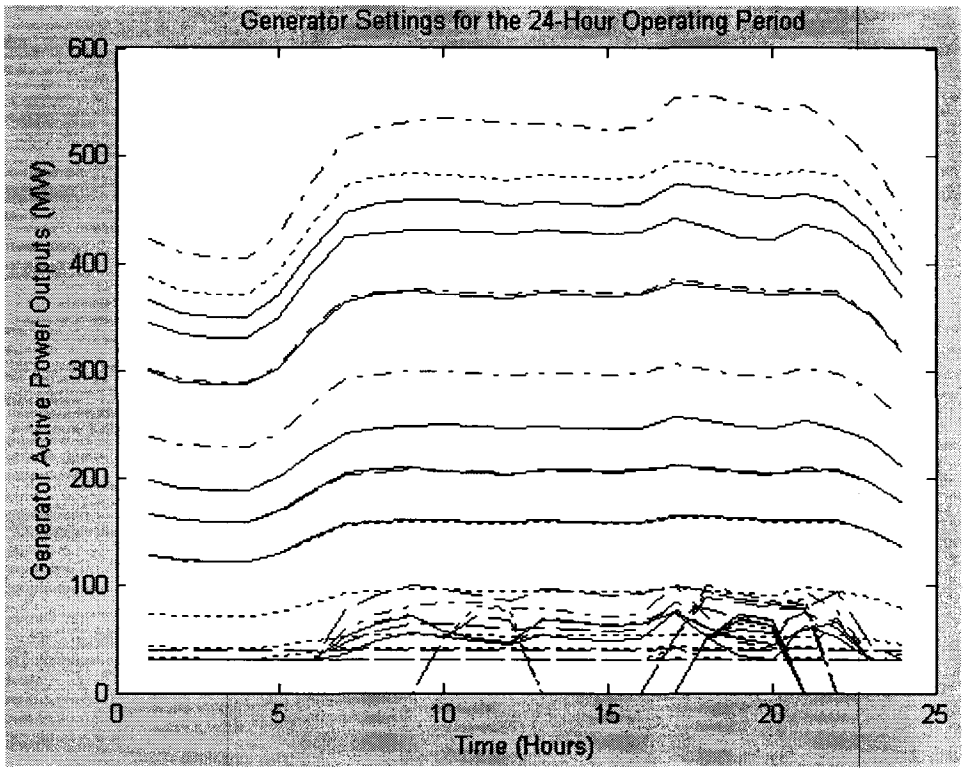

Figure 6.9 SCUC-Optimized Generation Scheme for 118-Bus Power System

Table 6.11 SCUC-Optimized Scheme for the 118-Bus Power System (MW)

Hour>>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Gen 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	72	67	0	0	0	0
Gen 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 5	346	335	331	331	350	391	424	428	431	431	429	427	430	429	427	428	441	433	423	421	436	429	408	367
Gen 6	74	71	70	70	75	84	92	94	95	95	94	94	95	94	94	94	98	94	91	91	97	95	88	79
Gen 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	68	63	0	0	0	0
Gen 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	67	63	0	0	0	0
Gen 10	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	42	33	30	30	33	30	30	30
Gen 11	165	160	158	158	167	187	202	205	207	207	206	205	206	206	205	205	212	209	205	203	209	206	195	175
Gen 12	238	231	228	228	242	270	292	296	299	299	297	296	298	297	296	297	306	301	295	293	302	298	281	253
Gen 13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 14	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 16	30	30	30	30	30	30	52	65	73	57	49	45	70	68	64	66	78	60	47	42	60	69	30	30
Gen 17	30	30	30	30	30	30	52	64	70	57	50	46	68	66	63	65	75	60	49	45	60	67	30	30
Gen 18	0	0	0	0	0	0	0	0	0	50	79	76	0	0	0	0	50	94	87	83	87	0	0	0
Gen 19	0	0	0	0	0	0	0	0	0	50	71	68	0	0	0	0	50	88	82	78	77	0	0	0
Gen 20	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 21	166	161	159	159	168	189	204	207	209	206	204	203	208	208	207	207	211	209	207	205	205	208	196	177
Gen 22	43	41	41	41	43	48	52	53	53	53	53	52	53	53	53	53	54	53	53	52	52	53	50	45
Gen 23	30	30	30	30	30	30	76	90	99	95	91	85	96	93	90	92	100	94	88	81	79	94	45	30
Gen 24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	99	92	85	82	0	0	0
Gen 25	128	124	122	122	130	145	156	159	161	160	159	158	160	160	159	159	164	163	162	160	160	160	151	136
Gen 26	127	123	121	121	129	144	155	157	159	158	158	157	158	158	157	158	162	162	160	159	159	158	149	135
Gen 27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 28	301	291	288	288	305	341	367	373	377	375	373	371	375	374	373	374	385	382	377	374	377	375	354	320
Gen 29	299	289	286	286	303	338	364	370	374	372	369	367	373	372	370	371	381	378	374	371	373	372	351	318
Gen 30	387	375	371	371	392	437	471	479	484	482	480	477	482	481	479	480	495	492	485	481	486	481	454	411
Gen 31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	64	58	0	0	0	0
Gen 35	30	30	30	30	30	30	46	57	64	64	63	58	61	59	57	58	83	79	69	63	74	60	30	30
Gen 36	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	33	31	30	30	30	30	30	30
Gen 37	365	354	350	350	370	414	447	454	459	459	457	454	457	456	454	455	472	471	465	460	465	457	430	388
Gen 38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 39	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Gen 40	422	409	405	405	428	476	515	525	531	534	533	529	529	527	525	526	553	557	548	542	546	528	495	448
Gen 41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 45	197	191	189	189	200	224	242	246	249	249	249	247	248	247	246	246	258	254	249	247	254	247	234	210
Gen 46	33	32	31	31	33	38	41	41	42	42	42	41	41	41	41	41	43	42	41	40	43	41	39	35
Gen 47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 48	30	30	30	30	30	30	59	72	80	84	83	78	77	75	72	74	100	71	50	45	95	76	33	30
Gen 49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	59	55	0	0	0	0
Gen 50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gen 51	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	42	40	40	40	41	40	40	40
Gen 52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	70	67	0	0	0	0
Gen 53	30	30	30	30	30	30	35	48	56	53	49	44	53	51	48	49	74	53	35	30	63	52	30	30
Gen 54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

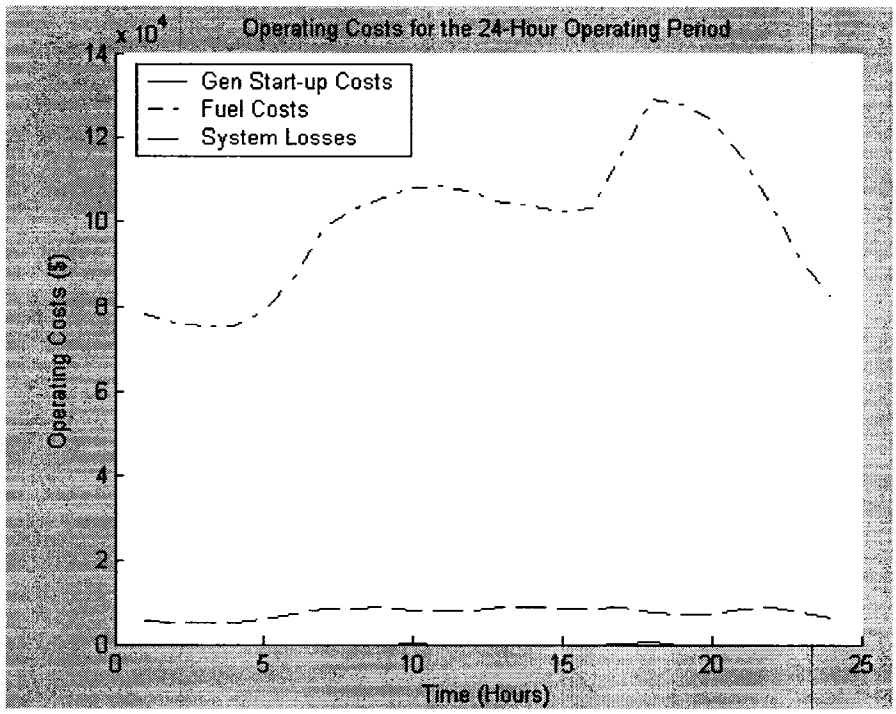

Figure 6.10 SCUC-Optimized Operating Costs for 118-Bus Power System

Table 6.12 SCUC-Optimized Operating Costs for 118-Bus Power System

Hour	Fuel Cost	Gen Start-up Cost	System Losses	Total
1	$ 75,475.00	$ -	$ 5,115.30	$ 80,590.00
2	$ 79,192.00	$ -	$ 5,735.90	$ 84,928.00
3	$ 86,981.00	$ -	$ 7,170.60	$ 94,151.00
4	$ 98,605.00	$ -	$ 8,274.70	$ 106,880.00
5	$ 102,910.00	$ -	$ 8,534.70	$ 111,440.00
6	$ 105,630.00	$ -	$ 8,721.60	$ 114,350.00
7	$ 108,120.00	$ 200.00	$ 8,184.30	$ 116,500.00
8	$ 108,740.00	$ -	$ 8,038.80	$ 116,780.00
9	$ 106,880.00	$ -	$ 7,947.20	$ 114,820.00
10	$ 104,590.00	$ -	$ 8,648.80	$ 113,240.00
11	$ 103,840.00	$ -	$ 8,597.30	$ 112,430.00
12	$ 102,720.00	$ -	$ 8,521.30	$ 111,240.00
13	$ 103,320.00	$ -	$ 8,562.30	$ 111,880.00
14	$ 117,030.00	$ 300.00	$ 8,673.90	$ 126,000.00
15	$ 129,350.00	$ 600.00	$ 7,413.90	$ 137,360.00
16	$ 127,990.00	$ -	$ 7,064.10	$ 135,060.00
17	$ 124,270.00	$ -	$ 6,991.20	$ 131,260.00
18	$ 115,820.00	$ -	$ 8,237.90	$ 124,060.00
19	$ 104,140.00	$ -	$ 8,618.10	$ 112,760.00
20	$ 90,740.00	$ -	$ 7,750.90	$ 98,491.00
21	$ 82,417.00	$ -	$ 6,307.00	$ 88,724.00
22	$ 78,258.00	$ -	$ 5,576.20	$ 83,834.00
23	$ 76,161.00	$ -	$ 5,226.80	$ 81,388.00
24	$ 75,475.00	$ -	$ 5,115.30	$ 80,590.00
Total	**$ 2,408,654.00**	**$ 1,100.00**	**$ 179,028.10**	**$ 2,588,756.00**

6.4 Analysis of Results

Based on the operational results provided above, the implemented hybrid particle swarm approach produced SCUC solutions with reduced operating costs compared to those generated by conventional linear programming techniques. One of the main reasons for this phenomenon is the fact that the linear programming methodology did not have the capacity to deactivate inefficient generators. Without this capability, many generators were forced to produce minimal active power outputs, resulting in a higher operating cost.

In response to this shortcoming, a potential solution could involve the relaxation of generator operating constraints and allow the units to produce an output lower than the specified minimal value. For example, without a specified minimum output voltage, many inefficient generators could potentially be minimized to produce zero MW and therefore be deactivated. When performing SCUC planning, however, this practice is unacceptable as it would introduce conditions where generators could potentially be prescribed to produce a finite active power output greater than zero but lower than the specified minimum. Such a generation scheme would eventually lead to increased maintenance costs as the affected units would be operating outside of their specified range of operation. For the purposes of this investigation, the enforcement of minimum generator active power output constraints was therefore deemed to be necessary.

Chapter 7

Conclusions and Recommendations for Future Work

An investigation was performed involving the application of particle swarm optimization techniques to SCUC planning. These heuristic techniques are advanced in their capacity to explore large search-spaces and may be used to find near-optimal solutions in a reduced operating time.

The use of particle swarms is recommended as conventional techniques are unable to adequately perform SCUC planning as the process involves a time-varying, non-linear, mixed-integer problem. This problem is also further complicated as it is highly constrained. Limitations associated with generating units and power flow constraints must be considered while an efficient methodology must be used to meet computational time and data storage requirements.

7.1 Summary

The methodology proposed for this investigation involves a combination of calculus-based linear programming optimization techniques with particle swarms. The SCUC planning methodology was implemented in a software package consisting of six modules including an Input Module, a Minimum Generator Module, a Particle Swarm Module, a Pathfinder Module, a Load-Flow Verification Module and an Output Module.

The SCUC software was applied to two theoretical test systems requiring day-ahead generation strategies. The results that were generated met all system constraints and provided a reduced operating cost. It should be noted, however; that the results

generated for both test cases are not guaranteed to be the optimal solutions as only a fraction of the overall search-space is explored. In addition, the exploration of the search-space is further reduced due to the fact that a heuristic particle swarm technique is only used to optimize one hour of the operating period. In this design, generator scheduling for the remaining hours is based on extrapolation.

One of the primary objectives of this investigation, however, is to provide a near-optimal solution in a simulated day-ahead environment. Based on the result of this investigation, the proposed methodology produced more cost-efficient results than those produced by the linear programming method and may be seen as a viable means of power system optimization given the specified time constraints.

7.2 Future Work

As the investigation undertaken in this thesis is exploratory in nature, it is designed to identify and highlight challenges that need to be pursued in order to advance the effectiveness of the proposed technique. The positive results of this study may therefore be seen as a foundation for individuals who wish to perform further research in the area of security constrained unit commitment using particle swarms.

Based on the content of this thesis, opportunities for future work include: the verification of the reliability of the methodologies described in this report, the comparison of the performances of alternative heuristic techniques such as the genetic algorithm, the analysis of software efficiency and the modification of the parameters/strategies used in this report, and the expansion of the functionality of the developed software.

7.2.1 Analysis of the Reliability of the Proposed Methodology

While the methodology described in this report was evaluated using two test cases, future research may be conducted to assess the performance of the software in response to variations in power system operating conditions. Although this investigation has verified that the software is capable of generating solutions for systems of varying size, it would be beneficial to assess the capability of the software when faced with variations in aspects such as loading schemes, power system constraints, and generator operating constraints.

With respect to loading schemes, the test case data for this investigation were based on actual data for a system operating under normal conditions. By modifying this data to reflect a more volatile system, it would be possible to ensure that the software can develop SCUC solutions for cases with more extreme variations in system loading.

Power system and generator constraints could also be modified to simulate more extreme circumstances and to test the capacity of the software to function under tighter operating constraints. Research may therefore be performed to examine practical systems and to determine acceptable limits for the operating conditions and constraints of such networks.

It may also be noted that if constraints are particularly rigid, situations may arise when no security-constrained unit commitment solution exists and the proposed software program may therefore be incapable of recommending an acceptable generation scheme. Future research may therefore be used to investigate real-life systems and utility operating strategies for dealing with such cases. If all constraints cannot be met, the system constraints must be prioritized and quantified by associating penalty factors with

constraints that cannot be met. For example, if a generator must be turned on before the expiration of a *minimum off time*, a penalty factor could be determined based on such aspects as an anticipated increase in maintenance costs.

Recommendations for future work relating to the reliability of the proposed software should also consider the use of actual system data with respect to generator ramping limits. For example, the hypothetical data used in the test cases may not accurately reflect the reality of system generators where large variations in active power outputs are unacceptable. The use of generator data from manufacturers and utility companies could therefore be used to ensure that the proposed SCUC software is able to produce acceptable solutions given realistic constraints.

7.2.2 Performance Comparison of Alternative Heuristic Techniques

The methodology described in this investigation involves the use of particle swarms to handle the binary aspect of the mixed-integer security-constrained unit commitment problem. It is recommended, however, that the performance of alternative heuristic techniques such as the genetic algorithm and other types of evolutionary programming be explored. A comparison of such techniques would ensure that the methodology employed by the software is capable of converging on an optimal or near-optimal solution as quickly as possible.

In addition to entirely heuristic techniques, future researchers may also be interested in investigating how the proposed methodology compares to hybrid algorithms involving Lagrangian relaxation [49] or Tabu searches [50]. Such a comparison could

explore which techniques are better equipped for dealing with SCUC problems in terms of computational time, reliability, and strength of solution.

7.2.3 Software Analysis and the Modification of Parameters

As described in Chapter 6, the analysis of computational resource requirements and the complexity of the SCUC software has not been considered as part of this investigation. Such analysis, however, would serve as a means of evaluating the efficiency of the software package and potentially improving the quality of the generated solutions. It is therefore recommended that various Matlab functions including "tic" and "toc" be used to monitor the efficiency of each module. The results of these assessments may then be used in an attempt to reduce required computational times. Once these computational times have been recorded, consideration may be given to the optimization of the parameters and strategies that have been specified for the particle swarms.

As described in Chapter 5, design considerations associated with particle swarm optimization include selection of the number of particles that will formulate the swarm and the number of generations over which the swarm will be allowed to search. Other parameters include the acceleration factors that place the relative weightings on the swarm sociological factors that were described in Chapter 3. Each of the above parameters was tuned on the researched recommendations [27, 29]. However, time-varying values may potentially be used to improve the efficiency of the swarm [30]. Other papers even suggest a variety of strategies where time-varying concepts may be used so that a swarm may dynamically respond to an optimization problem [36]. It is therefore recommended that investigations be performed to assess if these strategies may

be used as part of SCUC planning to improve the quality of results and computational time.

Other approaches for improving the efficiency of the SCUC software would involve the application of alternative software engineering techniques to reduce the complexity of the algorithms employed in the various modules. For example, as opposed to applying linear searches when sorting the various tables of the Particle Swarm Module, binary search techniques may be used. In addition, the operating efficiency of Matlab may be improved by eliminating the use of global variables as described in Chapter 5.

7.2.4 Expanding Software Functionality

This investigation has focused primarily on the optimization of power systems containing only thermal units. It is therefore recommended that the functionality of the developed software be expanded to include alternative energy sources such as hydroelectric and wind power.

While this modification would allow for the SCUC planning for a wider variety of power systems, this expanded functionality could be used to address environmental aspects such as the reduction of the emission of greenhouse gases. By incorporating government regulations relating to the Kyoto Protocol, limits on emissions may be enforced. In addition, proposed systems where electric utilities would receive credits for displacing greenhouse gases may be included in operating cost calculations.

As the production of electricity by alternative means would involve new sets of constraints, research should be performed to identify financial, maintenance, and environmental considerations that would need to be included in the implemented

software. For example, when considering hydroelectric units, environmental constraints involving minimum flow rates and must run units often need to be enforced to prevent damage to rivers and stream ecosystems. When wind power units are considered, power quality issues may involve the calculation of the system's overall wind penetration. This figure would need to be limited to prevent the propagation of harmonics.

7.3 Summary of Recommendations

The recommendations for future work based on this investigation and the contents of this thesis may be summarized as follows:

1. Assess the reliability of the implemented software in response to variations in system loading schemes and operational constraints. Investigate the failure-handling alternatives such as the incorporation of a set of penalty factors that are calculated based on unit maintenance requirements.

2. Investigate the performance of alternative methodologies for the optimization of the binary component of the SCUC mixed-integer problem. Techniques such as the genetic algorithm may be implemented and compared to the particle swarm-based software. Other hybrid methodologies incorporating techniques such as Lagrangian relaxation may also be considered.

3. Perform analyses to assess the computational time requirements of the various modules of the SCUC software. Parameters associated with the particle

swarms may then be optimized. The number of particles, iterations, and the values of acceleration coefficients associated with the movement of the swarms should be considered. Formal programming methods may also be employed to improve software efficiency.

4. Expand the functionality of the SCUC software so that systems containing generating units that produce energy from alternative energy sources may be considered. Constraints associated with these generators may be researched and implemented in the software. Financial and environmental considerations relating to the Kyoto protocol may also be included.

References

[1] Fu, Y.; Shahidehpour, S.M.; Li, Z.; Security-constrained unit commitment with AC constraints; IEEE Transactions on Power Systems, Volume 20, Issue 2, May 2005, Pages:1001 – 1013.

[2] Shahidehpour, S.M.; Marwali, M.; Maintenance Scheduling in Restructured Power Systems; 2000, Springer, Pages 14-16.

[3] Wood, A.J.; Wollenberg, B.F.; Power Generation, Operation, and Control, 2nd Edition, 1996, John Wiley & Sons, NY.

[4] Grisby, L.; The Electric Power Engineering Handbook, 2001, CRC Press and IEEE Press, Boca Raton/Piscataway. Pages: 11-121.

[5] Padhy, N.P.; Unit commitment-a bibliographical survey; IEEE Transactions on Power Systems, Volume 19, Issue 2, May 2004, Pages: 1196 – 1205.

[6] Kennedy, J.; Eberhart, R.; Particle swarm optimization; IEEE International Conference on Neural Networks, Volume 4, 27 Nov.-1 Dec. 1995, Pages:1942 - 1948 vol.4.

[7] Eberhart, R.; Kennedy, J.; A new optimizer using particle swarm theory; Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 4-6 Oct. 1995, Pages:39 – 43.

[8] Collett, R.; Quaicoe, J.; Power Systems Optimization Using Particle Swarms; IEEE Newfoundland Electrical and Computer Engineering Conference 2005, NECEC 2005, Volume 1, 8 November 2005.

[9] Collett, R.; Quaicoe, J.; Security-Constrained Unit Commitment Using Particle Swarms; IEEE Canadian Conference on Electrical and Computer Engineering 2006, CCECE 2006, Volume 1, 7-10 May 2006, Pages:1332 - 1336 vol.1.

[10] Cohen, A.I.; Sherkat, V.R.; Optimization-based methods for operations scheduling; Proceedings of the IEEE, Volume 75, Issue 12, Dec. 1987 Pages: 1574 – 1591.

[11] Wang, C.; Shahidehpour, S.M.; Effects of ramp-rate limits on unit commitment and economic dispatch; IEEE Transactions on Power Systems, Volume 8, Issue 3, Aug. 1993, Pages: 1341 – 1350.

[12] Rajan, C.C.A.; Mohan, M.R.; An evolutionary programming-based tabu search method for solving the unit commitment problem; IEEE Transactions on Power Systems, Volume 19, Issue 1, Feb. 2004, Pages: 577 – 585.

[13] Vemuri, S.; Lemonidis, L.; Fuel constrained unit commitment; IEEE Transactions on Power Systems, Volume 7, Issue 1, Feb. 1992, Pages: 410 – 415.

[14] Chiang C.L.; Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels; IEEE Transactions on Power Systems, Volume 20, Issue 4, Nov. 2005, Pages: 1690 – 1699.

[15] Jong-Bae Park; Ki-Song Lee; Joong-Rin Shin; Lee, K.Y.; A particle swarm optimization for economic dispatch with nonsmooth cost functions; IEEE Transactions on Power Systems, Volume 20, Issue 1, Feb. 2005, Pages: 34 – 42.

[16] Ma, H.; Shahidehpour, S.M.; Unit commitment with transmission security and voltage constraints; IEEE Transactions on Power Systems, Volume 14, Issue 2, May 1999, Pages: 757 – 764.

[17] Hara, K.; Kimura, M.; Honda, N.; A method for planning economic unit commitment and maintenance of thermal power systems; IEEE Transactions on Power Application Systems, Volume 85, May 1966, Pages: 427-436.

[18] Kerr, R.H.; Scheidt, J.L.; Fontana, A.J.; Wiley, J.K.; Unit Commitment; IEEE Transactions on Power Application Systems, Volume 85, May 1966, Pages: 417-421.

[19] Lee, F.N.; Short-term thermal unit commitment-a new method; IEEE Transactions on Power Systems, Volume 3, Issue 2, May 1988, Pages: 421 - 428.

[20] Lee, F.N.; A fuel-constrained unit commitment method; IEEE Transactions on Power Systems, Volume 4, Issue 3, Aug. 1989, Pages: 1208 – 1218.

[21] Lee, F.N.; Feng, Q.; Multi-area unit commitment; IEEE Transactions on Power Systems, Volume 7, Issue 2, May 1992, Pages: 591 – 599.

[22] Ouyang, Z.; Shahidehpour, S.M.; An intelligent dynamic programming for unit commitment application; IEEE Transactions on Power Systems, Volume 6, Issue 3, Aug. 1991, Pages: 1203 – 1209.

[23] Chang, G.W.; Tsai, Y.D.; Lai, C.Y.; Chung, J.S.; A practical mixed integer linear programming based approach for unit commitment; 2004 IEEE Power Engineering Society General Meeting, 6-10 June 2004, Pages:221 - 225 Vol.1.

[24] J. Kennedy, R. Eberhart; A discrete binary version of the particle swarm algorithm; Proceedings of the 1997 International Conference on Systems, Man, and Cybernetics, Volume 5, 12-15 Oct. 1997, Pages:4104 - 4108 vol.5.

[25] M. Clerc; Particle Swarm Optimization; 2006, ISTE.

[26] Holland, J.; Adaptation in natural and artificial systems; The University of Michigan Press, Ann Arbor, Michigan, 1975.

[27] Kennedy, J.; Eberhart, R.; Swarm Intelligence; 2000, Morgan Kaufman, Pages 293-295.

[28] Eberhart, R.C.; Shi, Y.; Comparing inertia weights and constriction factors in particle swarm optimization; Evolutionary Computation, Volume 1, 16-19 July 2000, Pages:84 – 88.

[29] A. P. Engelbrecht; Fundamentals of Computational Swarm Intelligence; 2005, John Wiley & Sons.

[30] Zheng, Y.L.; Ma, L.H.; Zhang, L.Y.; Qian, J.X.; On the convergence analysis and parameter selection in particle swarm optimization; 2003 International Conference on Machine Learning and Cybernetics, Volume 3, 2-5 Nov. 2003, Pages:1802 - 1807 Vol.3.

[31] Coath, G.; Halgamuge, S.K.; A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems; The 2003 Congress on Evolutionary Computation, CEC '03, Volume 4, 8-12 Dec. 2003, Pages:2419 - 2425 Vol.4.

[32] Michalewicz, Z.; Deb, K.; Schmidt, M.; Stidsen, T.J.; Towards understanding constraint-handling methods in evolutionary algorithms; Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99, Volume 1, 6-9 July 1999.

[33] Kennedy, J.; Spears, W.M.; Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator; Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, 4-9 May 1998, Pages:78 – 83.

[34] Habib, S.J.; Al-Kazemi, B.S.; Comparative study between the internal behavior of GA and PSO through problem-specific distance functions; The 2005 IEEE Congress on Evolutionary Computation, Volume 3, 2-5 Sept. 2005, Pages:2190 - 2195 Vol. 3.

[35] Clerc, M.; The swarm and the queen: towards a deterministic and adaptive particle swarm optimization; Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99, Volume 3, 6-9 July 1999.

[36] Eberhart, R.C.; Yuhui Shi; Tracking and optimizing dynamic systems with particle swarms; Proceedings of the 2001 Congress on Evolutionary Computation, Volume 1, 27-30 May 2001, Pages:94 - 100 vol. 1.

[37] Coello Coello, C.A.; Lechuga, M.S.; MOPSO: a proposal for multiple objective particle swarm optimization; Proceedings of the 2002 Congress on Evolutionary Computation, Volume 2, 12-17 May 2002, Pages:1051 – 1056.

[38] Xiaohui Hu; Yuhui Shi; Eberhart, R.; Recent advances in particle swarm; 2004 Congress on Evolutionary Computation, CEC2004, Volume 1, 19-23 June 2004, Pages:90 - 97 Vol.1.

[39] Jinchun Peng; Yaobin Chen; Eberhart, R.; Battery pack state of charge estimator design using computational intelligence approaches; The Fifteenth Annual Battery Conference on Applications and Advances, 11-14 Jan. 2000, Pages:173 – 177.

[40] Boeringer, D.W.; Werner, D.H.; Particle swarm optimization versus genetic algorithms for phased array synthesis; IEEE Transactions on Antennas and Propagation, Volume 52, Issue 3, March 2004, Pages:771 – 779.

[41] Rajan, C.C.A.; Mohan, M.R.; Manivannan, K.; Improved genetic algorithm solution to unit commitment problem; IEEE/PES 2002 Asia Pacific Transmission and Distribution Conference and Exhibition, Volume 1, 6-10 Oct. 2002, Pages:255 - 260 vol.1.

[42] Yang, H.T.; Yang, P.C.; Huang, C.L.; Applications of the genetic algorithm to the unit commitment problem in power generation industry; Proceedings of 1995 IEEE International Conference on Fuzzy Systems, Volume 1, March 1995, Pages: 267 – 274.

[43] Kazarlis, S.A.; Bakirtzis, A.G.; Petridis, V.; A genetic algorithm solution to the unit commitment problem; IEEE Transactions on Power Systems, Volume 11, Issue 1, Feb. 1996, Pages:83 – 92.

[44] Mantawy, A.H.; Abdel-Magid, Y.L.; Selim, S.Z.; A new genetic algorithm approach for unit commitment; Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA 97, 2-4 Sept. 1997, Pages:215 – 220.

[45] Chang, R.F.; Lu, C.N.; Feeder reconfiguration for load factor improvement; IEEE Power Engineering Society Winter Meeting, 2002, Volume 2, 27-31 Jan. 2002, Pages:980 - 984 vol.2.

[46] Xiaohui, Y.; Yanbin, Y.; Cheng, W.; Xiaopan, Z.; An Improved PSO Approach for Profit-based Unit Commitment in Electricity Market; IEEE/PES 2005 Asia and Pacific Transmission and Distribution Conference and Exhibition, 15-18 Aug. 2005, Pages:1 – 4.

[47] Gaing, Z.W.; Discrete particle swarm optimization algorithm for unit commitment; IEEE 2003 Power Engineering Society General Meeting, Volume 1, 13-17 July 2003.

[48] Ting, T.O.; Rao, M.V.C.; Loo, C.K.; A novel approach for unit commitment problem via an effective hybrid particle swarm optimization; IEEE Transactions on Power Systems, Volume 21, Issue 1, Feb. 2006, Pages:411 – 418.

[49] Sriyanyong, P.; Song, Y.H.; Unit commitment using particle swarm optimization combined with Lagrange relaxation; IEEE 2005 Power Engineering Society General Meeting, 12-16 June 2005 Pages:2752 - 2759 Vol. 3.

[50] Victoire, T.A.A.; Jeyakumar, A.E.; Unit commitment by a tabu-search-based hybrid-optimisation technique; IEE Proceedings on Generation, Transmission and Distribution, Volume 152, Issue 4, 8 July 2005, Pages:563 – 574.

[51] Al-Rashidi, M. R.; El-Hawary, M. E.; Emission-economic dispatch using a novel constraint handling particle swarm optimization strategy; IEEE Canadian Conference on Electrical and Computer Engineering 2006, CCECE 2006, Volume 1, 7-10 May 2006, Pages:1315 - 1320 vol.1.

[52] Zimmerman, R.D.; Murillo-Sánchez, C.E.; Gan, D.; Matpower, a Matlab power system simulation package; August 2005, http://www.pserc.cornell.edu/matpower/

[53] Christie, R.D.; University of Washington - College of Engineering Power Systems Test Case Archive; August, 1999, http://www.ee.washington.edu/research/pstca/

[54] California ISO; The California Independent Systems Operator; 2005, http://www.caiso.com/

[55] Newfoundland Power; Power Connection – A Newsletter for The Customers of Newfoundland Power; July 2005, http://www.newfoundlandpower.com/content/images/July_2005_Rate_information_vrs2.pdf

Appendix A SCUC Software – Main Program

The software below consists of the main SCUC program. The Input Module, the Minimum Generators Module, The Particle Swarm Module, and the Output Module are included in this section.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR Thesis - SCUC.m
% Created: 2005-12-02
% Modified: 2006-06-30
% Robert Collett - 9907908
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function SCUC

% Initialization
clear all;
close all;
delete('temp*.*')
startTime = clock;

%------------------------------------------------------
% Definition of Global Variables
%------------------------------------------------------
global settings;
global combos;
global fuelCosts;
global pfCosts;
global baseMVA;
global bus;
global origBus;
global origBranch;
global gen;
global origGen;
global branch;
global areas;
global gencost;
global penaltyValue;
global fileCount;
global lfCount;
global bestCostsForIter;
global minGens;
global comboIndices;
global comboCosts;
global maxCombos;
global MWrate;
global opt;
global lightHour;
global hourFactors;
global allOn;
global popSize;
global iterCount;
global w;
global c1;
global c2;
global Vmax;
global caseFile;
global hourCount;
global thisHour;
global startTime;
global bestCost;
global bestPath;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Input Module
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Swarm Parameters
popSize = 20;
iterCount = 50;
w = .1;
c1 = .5;
c2 = .5;
Vmax = 4;

%Options
casefile = 'case57';
allOn = 0;

%Hourly Load Data Source File Info
excelFile = 'hourFactors.xls';
excelSheet = 'VLight24';

%Time Limit Information
timeLimit = 16*3600; %seconds

%Energy Costs
MWrate = .08458*1000;

if allOn == 1
   swarmData(1) = 1; %popSize
   swarmData(2) = 1; %iterCount
   saveFile = strcat(date,'-', casefile, ' ALLON');
else
   saveFile = strcat(date,'-', casefile, ' SWARM');
end

% Extract System Data
opt = mpoption('ENFORCE_Q_LIMS', 1, 'OUT_SYS_SUM', 0, 'OUT_ALL_LIM', 0, ...
           'OUT_BRANCH', 0, 'OUT_BUS', 0, 'VERBOSE', 0);

hourFactors = xlsread(excelFile, excelSheet);
hourCount = size(hourFactors, 1);

[baseMVA, bus, gen, branch, areas, gencost] = loadcase(casefile);

fileCount = 0;
origBus = bus;
origGen = gen;
origBranch = branch;
genCount = size(gen,1);
busCount = size(bus,1);
bestCostsForIter = zeros(1, iterCount);
genSettings = zeros(1, genCount);
minGens = zeros(hourCount, genCount);
penaltyValue = 1e9;

pfDataFile = strcat(casefile, ' LFDATA-', excelSheet);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Minimum Generators Module
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Get generator max/min power settings
genMaxes = gen(:,9);
genMins = gen(:,10);
genRange = genMaxes-genMins;

thisHour = 1;
while thisHour <= hourCount

   %Setup Buses
   bus = origBus;
   branch = origBranch;
```

```
  thisBus = 1;
  totalLoad = 0;
  while (thisBus <= busCount)
    bus(thisBus, 3) = origBus(thisBus, 3)* hourFactors(thisHour);
    bus(thisBus, 4) = origBus(thisBus, 4)* hourFactors(thisHour);
    totalLoad = totalLoad + bus(thisBus, 3);
    thisBus = thisBus + 1;
  end

  genNo = 1;
  gen = origGen;
  while(genNo <= genCount)
    gen(genNo, 2) = gen(genNo, 10)+(gen(genNo, 9)-gen(genNo, 10))/2;
    gen(genNo, 8) = 1;
    genNo = genNo + 1;
  end

  fileCount = fileCount + 1;
  tempFile = strcat('tempCase',int2str(fileCount));

  savecase(tempFile, baseMVA, bus, gen, branch, areas, gencost);
  [baseMVA, bus, gen, gencost, branch, f, success, et] = runopf(tempFile, opt);

  %Check if generator is a minimum: 0 for min, 1 for others
  genNo = 1;
  while(genNo <= genCount)
    if gen(genNo, 2) == gen(genNo, 10)
      minGens(thisHour, genNo) = 0;
    else
      minGens(thisHour, genNo) = 1;
    end
    genNo = genNo + 1;
  end
  thisHour = thisHour + 1;
end

%Identify hour with most min gens
leastGens = sum(minGens(1,:));
i = 1;
lightHour = i;
while i <= hourCount;
  if leastGens > sum(minGens(i,:));
    lightHour = i;
    leastGens = sum(minGens(i,:));
  end
  i = i+1;
end

%*********************************************************************
%Particle Swarm Module
%*********************************************************************

%Setup Matrices for Data Storage
maxCombos = popSize*iterCount;
saveFile = strcat(date,'-', casefile, ' PSO');
settings = zeros(1, maxCombos*genCount); %gen settings for each hour(e.g. 1101011)
fuelCosts = zeros(1, maxCombos*genCount); %fuel costs for each combo for each hour
pfCosts = zeros(1, maxCombos); %pf costs for each combo for each hour
comboCosts = zeros(1, maxCombos);
comboIndices = zeros(1, maxCombos);

combos = 0; %a count of the number of combos that are counted

%Initialize Swarm Parameters
velocities = zeros(1, popSize*genCount);
bestONOFF = zeros(1, popSize*genCount);
bestOverallONOFF = zeros(1,genCount);

%If allOn mode is active, activate all generators
%Otherwise use a random initial population
```

```
if allOn == 0
   genONOFF = round(rand(1,popSize*genCount));
else
   genONOFF = ones(1,popSize*genCount);
end

%Initialize best costs
bestPopCosts = ones(popSize,1)*9e9;
bestOverallCost = 9e9;

%Initialize Critical Flag
criticalFlag = 0;

iterNo = 1;
while(iterNo <= iterCount)

   popNo = 1;
   while(popNo <= popSize)

      %Ensure all Non-Minimum Gens will be on
      genONOFF((popNo-1)*genCount+1:popNo*genCount) = ...
         or(genONOFF((popNo-1)*genCount+1:popNo*genCount),minGens(lightHour,:));

      % Get popIndex
      popIndex = 0;
      i = 1;
      while i <= genCount
         if genONOFF((popNo-1)*genCount+i) > 0
            popIndex = popIndex + 2^(i-1);
         end
         i = i + 1;
      end

      %Setup Buses
      bus = origBus;
      branch = origBranch;

      thisBus = 1;
      totalLoad = 0;
      while (thisBus <= busCount)
         bus(thisBus, 3) = origBus(thisBus, 3)* hourFactors(lightHour);
         bus(thisBus, 4) = origBus(thisBus, 4)* hourFactors(lightHour);
         totalLoad = totalLoad + bus(thisBus, 3);
         thisBus = thisBus + 1;
      end


      %Turn gens on/off according to genONOFF
      genNo = 1;
      gen = origGen;
      while(genNo <= genCount)
         if genONOFF((popNo-1)*genCount+genNo) < 1
            gen(genNo, 2) = 0;
            gen(genNo, 8) = 0;
         else
            gen(genNo, 2) = gen(genNo, 10)+(gen(genNo, 9)-gen(genNo, 10))/2;
            gen(genNo, 8) = 1;
         end
         genNo = genNo + 1;
      end

      %Perform new loadflow if solution is reasonable
      totalLoad = sum(bus(:,3));
      availGen = sum(gen(:,9).*gen(:,8));
      if(availGen > totalLoad)
         success = 1;
      else
         success = 0;
      end
```

```
%Check if combo has already been checked
found = 0;
done = 0;
i = 1;
fileCount = fileCount + 1;
while (done == 0 && i <= maxCombos && success > 0)
  if popIndex == comboIndices(i) || popIndex == 0
    found = 1;
    done = 1;
  end
  i = i + 1;
end

%If combo has not been checked, perform load flow
if (done == 0 && success > 0)
  [tempSettings, tempFuelCosts, tempPFCost] = lfCheck;

  %Calulate Losses
  k = 1;
  losses = 0;
  while(k <= size(branch,1))
    losses = losses + abs(abs(branch(k,12))-abs(branch(k,14)))*MWrate;
    k = k+1;
  end
  currentCost = tempPFCost + sum(tempFuelCosts)+losses;
end

%Store combo in costing order
i = 1;
while (currentCost < penaltyValue && done == 0 && success > 0)
  %Case 1 - (Special for first hour)
  %Solution is too expensive -> Don't Save
  if (i > maxCombos)
    done = 1;

    %Case2 - Save Solution
  elseif(currentCost < comboCosts(i) || comboCosts(i) == 0)
    done = 1;
    if combos < maxCombos
      combos = combos + 1;
    end

    %Make room for new settings
    if i < maxCombos
      comboCosts(i+1:size(comboCosts,2)) = comboCosts(i:size(comboCosts,2)-1);
      comboIndices(i+1:size(comboIndices,2)) = comboIndices(i:size(comboIndices,2)-1);
      fuelCosts((i*genCount)+1:size(fuelCosts,2)) = fuelCosts((i-1)*genCount+1:size(fuelCosts,2)-genCount);
      settings((i*genCount)+1:size(settings,2)) = settings((i-1)*genCount+1:size(settings,2)-genCount);
    end

    comboCosts(i) = currentCost;
    comboIndices(i) = popIndex;
    settings((i-1)*genCount+1:i*genCount) = tempSettings;
    fuelCosts((i-1)*genCount+1:i*genCount) = tempFuelCosts;
    pfCosts(i) = tempPFCost + losses;

    %Compare with popNo's best fitness to date
    if(currentCost < bestPopCosts(popNo) || bestPopCosts(popNo) < 0)
      bestPopCosts(popNo) = currentCost;
      bestONOFF((popNo-1)*genCount+1:popNo*genCount) = genONOFF((popNo-1)*genCount+1:popNo*genCount);

      if (currentCost < bestOverallCost)
        bestOverallCost = currentCost;
        bestPopNo = popNo;
        bestOverallONOFF = genONOFF((popNo-1)*genCount+1:popNo*genCount);
      end
    end
  end

  i = i + 1;
```

```
    end

    %After all population members have been evalauated ->
    %Update Swarm

    if popNo == popSize

      %Store bestCost
      bestCostsForIter(1, iterNo) = bestOverallCost;
      rhoVector = rand(genCount*popSize);

      popCounter = 1;
      while(popCounter <= popSize)
        genNo = 1;
        while(genNo <= genCount)

          %Calculate new velocities
          velocities((popCounter-1)*genCount+genNo) = w*velocities((popCounter-1)*genCount+genNo)...
            + c1*rand*(bestONOFF((popCounter-1)*genCount+genNo)-genONOFF((popCounter-1)*genCount+genNo))...
            + c2*rand*(bestOverallONOFF(genNo)-genONOFF((popCounter-1)*genCount+genNo));

          %Enforce Vmax Limits
          if (abs(velocities((popCounter-1)*genCount+genNo)) > Vmax)
            if velocities((popCounter-1)*genCount+genNo) < -Vmax
              velocities((popCounter-1)*genCount+genNo) = -Vmax;
            else
              velocities((popCounter-1)*genCount+genNo) = Vmax;
            end
          end

          %Update each dimension
          sigVector = 1/(1+exp(-velocities((popCounter-1)*genCount+genNo)));
          if (sigVector > rhoVector((popCounter-1)*genCount+genNo))
            genONOFF((popCounter-1)*genCount+genNo) = 1;
          else
            genONOFF((popCounter-1)*genCount+genNo) = 0;
          end

          genNo = genNo +1;
        end
        popCounter = popCounter + 1;
      end
    end
    popNo = popNo + 1;
  end
  delete 'temp*.m';
  iterNo = iterNo + 1;
end

if(combos) == 0
  combos = 1;
  criticalFlag = 1;
end

save(pfDataFile, 'settings', 'minGens','fuelCosts', 'pfCosts', 'combos', ...
  'lightHour', 'criticalFlag', 'bestCostsForIter');


%-----------------------------------------------------------------------
%  Pass control to Pathfinder Module by calling "pathfinder" function
%-----------------------------------------------------------------------
comboGenIDs = zeros(hourCount, max(combos));
timePenalty = zeros(hourCount,genCount);

fields = 7;
%UCpath = [comboNumber, runningTotal, comboCost, comboFuelCost, comboPFCost, comboSimilarity, comboSettings]
UCpath = zeros(hourCount, fields+genCount);
bestCost = 0;

hour = 1;
```

```
dpstart = clock;
[UCpath] = pathfinder(UCpath, hourCount, fields, timeLimit);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Output Module
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%End of program if path not found
if criticalFlag == 1;
   disp('Critical Flag Error')
   return
end

%Get Correct Order
i = 1;
while (i <= hourCount)
   orderedPath(lightHour,:) = UCpath(i,:);
   if (lightHour == hourCount)
      lightHour = 1;
   else
      lightHour = lightHour + 1;
   end
   i = i + 1;
end

%Recalculate Running Totals
i = 1;
runningTotal = 0;
while (i <= hourCount)
   runningTotal = runningTotal + orderedPath(i,3);
   orderedPath(i,2) = runningTotal;
   i = i + 1;
end

%-------------------------------------------------------------------------
%Calculate total costs
%-------------------------------------------------------------------------
%Get data from
bestOverallCost = orderedPath(hourCount, 2);
bestOverallSettings = orderedPath(:, fields+1:fields+genCount);
bestHourlyFuelCosts = orderedPath(:, 4);
bestHourlyTimeCosts = orderedPath(:, 5);
bestHourlyPFCosts = orderedPath(:, 6);

%Get Results and finish
totalTime = etime(clock, startTime);
save(saveFile)
delete('temp*.*')

%-----------------------------------------------------------
% Plot Results
%-----------------------------------------------------------
close all
figure('DefaultAxesColorOrder',[0 0 0], ...
'DefaultAxesLineStyleOrder','-|:|--|-.')
plot(1:hourCount, bestOverallSettings);
Title('Generator Settings for the 24-Hour Operating Period');
xlabel('Time (Hours)');
ylabel('Generator Active Power Outputs (MW)');
figure('DefaultAxesColorOrder',[0 0 0], ...
'DefaultAxesLineStyleOrder','-|:|--|-.')
hold on
plot(1:hourCount, bestHourlyTimeCosts);
plot(1:hourCount, bestHourlyFuelCosts, '-.');
plot(1:hourCount, bestHourlyPFCosts, '--');
Title('Operating Costs for the 24-Hour Operating Period');
xlabel('Time (Hours)');
ylabel('Operating Costs ($)');
Legend('Gen Start-up Costs', 'Fuel Costs', 'System Losses');
hold off
```

Appendix B SCUC Software – Pathfinder Module

This section includes the code for the pathfinder.m Matlab file containing the Pathfinder Module.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR Thesis - pathfinder.m Version 4
% Created: 2005-12-02
% Modified: 2005-06-30
% Robert Collett - 9907908
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [UCpath] = pathfinder(UCpath, hourCount, fields, timeLimit)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pathfinder Module
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global settings;
global combos;
global fuelCosts;
global pfCosts;
global baseMVA;
global origGen;
global branch;
global bus;
global origBus;
global origBranch;
global gen;
global areas;
global gencost;
global fileCount;
global bestCost;
global bestPath;
global pathCount;
global minGens;
global MWrate;
global opt;
global lightHour;
global hourFactors;
global startTime;
global allOn;
global penaltyValue;

%Min on/off time settings
limitHours = 3;
minOnTime = limitHours; %genOnOff(genNo,1);
minOffTime =limitHours; %genOnOff(genNo,2);

busCount = size(bus,1);
genCount = size(gen,1);

%Cycle through all combos for first hour
thisCombo = 1;
runningCost = 0;

if allOn
  combos == 1;
end

while (timeLimit > etime(clock,startTime) && thisCombo <= combos)
  UCpath(1, fields+1:fields+genCount) = settings((thisCombo-1)*genCount+1:thisCombo*genCount);
  UCpath(1,4) = sum(fuelCosts((thisCombo-1)*genCount+1:thisCombo*genCount));
  UCpath(1,6) = pfCosts(thisCombo);
  UCpath(1,3) = sum(UCpath(1,4:6));
```

```
UCpath(1,2) = UCpath(1,3);

%Cycle through remaining hours for current combo
hourCounter = 2;
criticalFlag = 0;
while (timeLimit > etime(clock,startTime) && hourCounter <= hourCount && criticalFlag < 1)

  if hourCounter+lightHour <= hourCount;
    thisHour = hourCounter+lightHour;
  else
    thisHour = mod(hourCounter+lightHour, hourCount);
  end

  runningCost = UCpath(hourCounter-1, 2);
  comboTimeCosts = zeros(1,genCount);
  %-----------------------------------------------------------------
  %Get Bus Settings
  %-----------------------------------------------------------------
  bus = origBus;
  branch = origBranch;

  thisBus = 1;
  totalLoad = 0;
  while (thisBus <= busCount)
    bus(thisBus, 3) = origBus(thisBus, 3)* hourFactors(thisHour);
    bus(thisBus, 4) = origBus(thisBus, 4)* hourFactors(thisHour);
    totalLoad = totalLoad + bus(thisBus, 3);
    thisBus = thisBus + 1;
  end

  %-----------------------------------------------------------------
  %Get Generator Settings
  %-----------------------------------------------------------------

  %Ensure all Non-Minimum Gens will be on
  if allOn
    genONOFF = ones(1, genCount);
  else
    genONOFF = or(UCpath(1, fields+1:fields+genCount), minGens(thisHour,:));
  end

  %-----------------------------------------------------------------
  %Ensure min on/off settings are met
  %-----------------------------------------------------------------
  genNo = 1;
  gen = origGen;
  while(genNo <= genCount)
    if (hourCounter > minOnTime && hourCounter > minOffTime)
      lastOutput = UCpath(hourCounter-1, fields+genNo);
      currentOutput = genONOFF(genNo);

      wasOn = 0;
      if (lastOutput > 0)
        wasOn = 1;
      end
      isOn = 0;
      if (currentOutput > 0)
        isOn = 1;
        % Check if device is being started
        if (wasOn == 0)
          % Check if device was off for t > minOffTime
          t = 1;
          stillOff = 1;
          while (stillOff && t-1 < minOffTime)
            %Initialize counter to scroll back through time
            counter = hourCounter - t;
            if (counter < 1)
              counter = hourCount+counter;
            end
            if(UCpath(counter, fields+genNo) == 0)
```

```
            stillOff = 1;
            t = t + 1;
          else
            stillOff = 0;
          end
        end

        if (t-1 < minOffTime)
          %Violation - gen cannot be started
          genONOFF(genNo) = 0;
        else
          %Add start-up cost
          comboTimeCosts(genNo) = comboTimeCosts(genNo) + gencost(genNo, 2);
        end
      end

    else %if device is off, check if it has been just switched off
      if (wasOn == 1)
        % Check if device was on for t > minOnTime
        t = 1;
        stillOn = 1;
        while (stillOn && t-1 < minOnTime)
          %Initialize counter to scroll back through time
          counter = hourCounter - t;
          if (counter < 1)
            counter = hourCount+counter;
          end
          if(UCpath(counter, fields+genNo) > 0)
            stillOn = 1;
            t = t + 1;
          else
            stillOn = 0;
          end
        end
        if (t-1 < minOnTime)
          %Violation - gen cannot be switched off
          genONOFF(genNo) = 1;
        end
      end
    end

    %-------------------------------------------------------------------------
    %Check First Few Hours
    %-------------------------------------------------------------------------
    if(hourCounter == hourCount)
      iterator = 1;
      while(iterator <= minOnTime || iterator <= minOffTime)
        if iterator == 1
          lastOutput = genONOFF(genNo);
          currentOutput = UCpath(1, fields+genNo);
        else
          lastOutput = UCpath(iterator-1, fields+genNo);
          currentOutput = UCpath(iterator, fields+genNo);
        end

        wasOn = 0;
        if (lastOutput > 0)
          wasOn = 1;
        end
        isOn = 0;
        if (currentOutput > 0)
          isOn = 1;
          % Check if device is being started
          if (wasOn == 0)
            % Check if device was off for t > minOffTime
            t = 1;
            stillOff = 1;
            while (stillOff && t-1 < minOffTime)
              %Initialize counter to scroll back through time
              counter = iterator - t;
```

```
      if (counter < 1)
        counter = hourCount+counter;
      end
      if(UCpath(counter, fields+genNo) == 0)
        stillOff = 1;
        t = t + 1;
      else
        stillOff = 0;
      end
    end

    if (t-1 < minOffTime)
      %Violation - not off long enough
      %Check if state can saftely be changed
      if (iterator == 1)
        if UCpath(minOffTime, fields+genNo) == UCpath(minOffTime+1, fields+genNo)
          UCpath(1, fields+genNo) = 0;
          UCpath(1, 1) = 1;
        else
          comboTimeCosts(genNo) = comboTimeCosts(genNo)+ penalties(1);
        end
      else
        if UCpath(iterator+minOffTime, fields+genNo) == UCpath(iterator+minOffTime+1, fields+genNo);
          UCpath(iterator, fields+genNo) = 0;
          UCpath(iterator, 1) = 1;
        else
          comboTimeCosts(genNo) = comboTimeCosts(genNo)+ penaltyValue;
        end
      end
    end
  end

else %if device is off, check if it has been just switched off
  if (wasOn == 1)
    % Check if device was on for t > minOnTime
    t = 1;
    stillOn = 1;
    while (stillOn && t-1 < minOnTime)
      %Initialize counter to scroll back through time
      counter = iterator - t;
      if (counter < 1)
        counter = hourCount+counter;
      end
      if(UCpath(counter, fields+genNo) > 0)
        stillOn = 1;
        t = t + 1;
      else
        stillOn = 0;
      end
    end
    if (t-1 < minOnTime)
      %Violation - not on long enough
      %Check if state can saftely be changed
      if (iterator == 1)
        if UCpath(minOnTime, fields+genNo) == UCpath(minOnTime+1, fields+genNo)
          UCpath(1, fields+genNo) = 1;
          UCpath(1, 1) = 1;
        else
          comboTimeCosts(genNo) = comboTimeCosts(genNo)+ penaltyValue;
        end
      else
        if UCpath(iterator+minOnTime, fields+genNo) == UCpath(iterator+minOnTime+1, fields+genNo);
          UCpath(iterator, fields+genNo) = 1;
          UCpath(iterator, 1) = 1;
        else
          comboTimeCosts(genNo) = comboTimeCosts(genNo)+ penaltyValue;
        end
      end
    end
  end
```

```
            end
            iterator = iterator + 1;
          end
        end
      end

      %-----------------------------------------------------------------------
      %Turn gens on/off according to genONOFF
      %-----------------------------------------------------------------------
      if genONOFF(genNo) < 1
        gen(genNo, 2) = 0;
        gen(genNo, 8) = 0;
      else
        %Enforce Ramp Limits
        rampScale = 1;
        if (gen(genNo, 9) > 149)
          maxRamp = gen(genNo, 9)*rampScale*.4; %genOnOff(genNo,3);
        elseif (gen(genNo, 9) > 99)
          maxRamp = gen(genNo, 9)*rampScale*.5; %genOnOff(genNo,3);
        elseif (gen(genNo, 9) > 25)
          maxRamp = gen(genNo, 9)*rampScale*.75; %genOnOff(genNo,3);
        else
          maxRamp = gen(genNo, 9)*rampScale; %genOnOff(genNo,3);
        end

        lastOutput = UCpath(hourCounter-1, fields+genNo);
        gen(genNo, 9) = min(lastOutput+maxRamp, gen(genNo, 9));
        gen(genNo, 10) = max(lastOutput-maxRamp, gen(genNo, 10));

        gen(genNo, 2) = gen(genNo, 10)+(gen(genNo, 9)-gen(genNo, 10))/2;
        gen(genNo, 8) = 1;
      end
      genNo = genNo + 1;
    end

    %-----------------------------------------------------------------------
    %Perform new loadflow if solution is reasonable
    %-----------------------------------------------------------------------
    totalLoad = sum(bus(:,3));
    availGen = sum(gen(:,9).*gen(:,8));
    fileCount = fileCount + 1;
    if(availGen > totalLoad)
      [tempSettings, tempFuelCosts, tempPFCost] = lfCheck;
      currentCost = tempPFCost + sum(tempFuelCosts);

      %If loadflow produces acceptable solution -> store solution
      if(currentCost < penaltyValue)
        UCpath(hourCounter, fields+1:fields+genCount) = tempSettings;

        %Calulate Losses
        k = 1;
        losses = 0;
        while(k <= size(branch,1))
          losses = losses + abs(abs(branch(k,12))-abs(branch(k,14)))*MWrate;
          k = k+1;
        end

        %Store Settings
        UCpath(hourCounter, 4) = sum(tempFuelCosts);
        UCpath(hourCounter, 6) = tempPFCost + losses;

      %If solution is unacceptable -> raise flag
      else
        criticalFlag = 1;
      end
    else
      criticalFlag = 1;
    end

    %-----------------------------------------------------------------------
```

```
      %Add up time costs and check for violations
      %-------------------------------------------------------------------------
      if(sum(comboTimeCosts) > penaltyValue)
         criticalFlag = 1;
      else
         %Get totals
         UCpath(hourCounter,5) = sum(comboTimeCosts);
         UCpath(hourCounter,3) = sum(UCpath(hourCounter,4:6));
         UCpath(hourCounter,2) = UCpath(hourCounter-1,2) + UCpath(hourCounter,3);

         %At last hour -> Check if cost is better than bestCost
         if hourCounter == hourCount
            if(UCpath(hourCounter,2) < bestCost || bestCost < 1)
               bestCost = UCpath(hourCounter,2);
               bestPath = UCpath;
            end
            pathCount = pathCount+1;
         end
      end
      hourCounter = hourCounter + 1;
   end
   delete('temp*.*')
   thisCombo = thisCombo + 1;
end
%Return best path
UCpath = bestPath;
```

Appendix C SCUC Software – Load Flow Verification Module

This section includes the code for the lfcheck.m Matlab file containing the Load Flow Verification Module.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR Thesis - lfcheck.m
% Created: 2005-12-02
% Modified: 2006-06-30
% Robert Collett - 9907908
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [tempSettings, tempFuelCosts, tempPFCost] = lfCheck

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load Flow Verification Module
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global settings;
global combos;
global fuelCosts;
global pfCosts;
global baseMVA;
global bus;
global origBus;
global gen;
global origGen;
global branch;
global areas;
global gencost;
global penaltyValue;
global fileCount;
global opt;
global lfCount;
global hourCount;
global thisHour;

busCount = size(bus,1);
genCount = size(gen,1);
branchCount = size(branch,1);

%--------------------------------------------------------
%Setup Penalty Costs
%--------------------------------------------------------
successPenalty = penaltyValue;
voltPenalty = penaltyValue;
genQPenalty = penaltyValue;
branchPenalty = penaltyValue;
minOnPenalty = penaltyValue;
minOffPenalty = penaltyValue;
rampPenalty = penaltyValue;

tempFile = strcat('tempCase',int2str(fileCount));

savecase(tempFile, baseMVA, bus, gen, branch, areas, gencost);
%[bus, gen, branch, f, success, et] = mopf(tempFile, opt);
[baseMVA, bus, gen, gencost, branch, f, success, et] = runopf(tempFile, opt);
lfCount = lfCount+1;

%Get Setttings
tempSettings = gen(:, 2).';
tempFuelCosts = zeros(1, genCount);
tempPFCost = 0;

%--------------------------------------------------------
% Check PF Contraints
%--------------------------------------------------------
```

```
pfPenalty = 0; %hourly Penalty Factor for PF violations
%Check if system was solved
pfFlags = zeros(hourCount, 5);
if (success == 0)
   pfFlags(thisHour, 1) = 1;
   flags = 1;
   pfPenalty = pfPenalty + successPenalty;
else
   %Solution is acceptable

   %Check Bus Voltages
   thisBus = 1;
   while (thisBus < busCount)
      busVoltage = bus(thisBus, 8);
      maxVolts = bus(thisBus, 12)+.001;
      minVolts = bus(thisBus, 13)-.001;

      if (busVoltage > maxVolts) ||(busVoltage < minVolts)
         pfFlags(thisHour, 2) = 1;
         flags = 1;
         pfPenalty = pfPenalty + voltPenalty;
      end
      thisBus = thisBus + 1;
   end

   %Check Gen Q Output
   thisGen= 1;
   while (thisGen < genCount)
      genQ = gen(thisGen, 3);
      maxQ = gen(thisGen, 4)+2;
      minQ = gen(thisGen, 5)-2;

      if (gen(thisGen, 8) > 0 && (genQ > maxQ) ||(genQ < minQ))
         pfFlags(thisHour, 3) = 1;
         flags = 1;
         pfPenalty = pfPenalty + genQPenalty;
      end
      thisGen = thisGen + 1;
   end

   %Check Gen P Output
   thisGen= 1;
   while (thisGen < genCount)
      genP = round(gen(thisGen, 2));
      maxP = gen(thisGen, 9)+2;
      minP = gen(thisGen, 10)-2;

      if (abs(genP) > 0)&&((genP > maxP) ||(genP < minP))
         pfFlags(thisHour, 4) = 1;
         flags = 1;
         pfPenalty = pfPenalty + genPPenalty;
      end
      thisGen = thisGen + 1;
   end

   %Check Line Power Flow
   thisBranch = 1;
   while (thisBranch < branchCount)
      branchVAF = sqrt(branch(thisBranch, 12)^2+branch(thisBranch, 13)^2);
      branchVAT = sqrt(branch(thisBranch, 14)^2+branch(thisBranch, 15)^2);
      branchVA = max(branchVAF, branchVAT);
      maxVA = branch(thisBranch, 6)+5;

      if (branchVA > maxVA)
         pfFlags(thisHour, 5) = 1;
         flags = 1;
         pfPenalty = pfPenalty + branchPenalty;
      end
      thisBranch = thisBranch + 1;
```

```
  end

  %Get Fuel Costs
  genNo = 1;
  while (genNo <= genCount)
    tempFuelCosts(genNo)          =          gen(genNo,          8)*(gen(genNo,          2)*gencost(genNo,5)+gen(genNo,
2)*gencost(genNo,6)+gencost(genNo,7));
    genNo = genNo + 1;
  end
end
%Get updated penalty cost
tempPFCost = pfPenalty;
```

Appendix D System Case File Format

The fields described below define the matrices contained within the system case files. These matrices are used by the implemented software and relate to system buses, branches, generators, areas, and generator costs.

The Bus Matrix contains information relating to all network buses. This matrix contains thirteen columns that are described as follows:

1. *Bus ID* – A reference number that is assigned to each bus within the power system.
2. *Bus Type* – Defines if the bus is a PV bus (Type 1), a PQ bus (Type 2), or a System Slack Bus (Type 3) for the purposes of system load flows.
3. *Pd* – The active load connected to the system bus.
4. *Qd* – The reactive load connected to the system bus.
5. *Gs* – The per-unit shunt conductance connected to the system bus.
6. *Bs* – The per-unit shunt susceptance connected to the system bus.
7. *Area Number* – A grouping number with which system buses may be associated. The Area Number allows for busses to be segregated for financial reasons if required.
8. *Vm* – The per-unit magnitude of the voltage at the system bus.
9. *Va* – The phase angle of the voltage at the system bus in degrees.
10. *Base kV* – The base voltage for the system bus.

11. *Zone* – A grouping number with which a system bus may be associated. The Zone Number allows for buses to be segregated if a system has multiple rates associated with system losses.
12. *Vmax* – The maximum voltage permitted at the system bus.
13. *Vmin* – The minimum voltage permitted at the system bus.

The Branch Matrix contains information relating to the transmission lines contained within the power system. This matrix has eleven columns that are defined as follows:

1. *From Bus* – The reference number of one of the two busses to which the transmission line is connected. From a load flow perspective, power flows from this bus.
2. *To Bus* – The reference number of one of the two busses to which the transmission line is connected. From a load flow perspective, power flows to this bus.
3. *R* – The per-unit series resistance of the transmission line.
4. *X* – The per-unit series reactance of the transmission line.
5. *B* – The per-unit shunt susceptance of the transmission line.
6. *Long Term Rating* – Long-term MVA rating of the transmission line.
7. *Short Term Rating* – Short-term MVA rating of the transmission line.
8. *Emergency Rating* – Emergency MVA rating of the transmission line.

9. *Turns Ratio* – Voltage Ratio of the *From Bus* to the *To Bus*. May be applied if a buck-boost transformer or voltage regulator is included in the system.
10. *Phase Shift* – Phase shift (in degrees) of the *From Bus* to the *To Bus*. May be applied if a phase shift transformer is included in the system.
11. *Status* – Indicates if the transmission line is in service. This is a binary variable.

The Generator Matrix contains information relating to all power system generators. The ten columns of this matrix are defined as follows:

1. *Bus Number* – The reference number of the bus to which the generator is connected.
2. *Pgen* – The active power output of the generator.
3. *Qgen* – The reactive power output of the generator.
4. *Qmax* – The maximum reactive power output of the generator.
5. *Qmin*– The minimum reactive power output of the generator.
6. *Vset* – The per-unit voltage set point for the bus to which the generator is connected.
7. *MVA Base* – The base MVA for the generating unit.
8. *Status* – Indicates if the generating unit in service. This is a binary variable.
9. *Pmax* – The maximum active power output of the generator.
10. *Pmin* – The minimum active power output of the generator.

The Area Matrix allows for the SCUC software to develop generator scheduling solutions for transmission networks that contain multiple power systems. These neighbouring power systems may be grouped into areas that are defined using the two-column matrix below:

1. *Area ID* – A reference number that is assigned to each power system within the transmission network.
2. *Reference Bus ID* – The reference number of the slack bus within the transmission system.

The Generator Cost Matrix allows for the SCUC software to calculate the hourly operating cost for the power system generating units based on each unit's individual heat rate. As described in Chapter 4, these heat rates are defined by a three-term polynomial (4.1). To provide added functionality, however, the software also allows for the heat rate to be expressed as a piecewise linear function. Various elements in the Generator Cost Matrix may be modified to select the desired mode of operation. The columns of this matrix are described as follows:

1. *Model Type* – Permits the user to select a piecewise linear model (Type 1) or a polynomial expression (Type 2) to define a unit's heat rate.
2. *Start-up Cost* – Unit start-up cost in dollars.
3. *Shutdown Cost* – Unit shutdown cost in dollars.
4. *Model Order* – This value represents (a) the number of data points for a piecewise linear function if the unit model is of Type 1 or (b) the order of the polynomial used to represent the heat rate for a Type 2 unit.

5. *Cost Data* – All remaining table elements represent parameters specific to the heat rate of the system's thermal unit. If the model is of Type 1, the elements of the piecewise linear cost function are stored in the matrix using the format:

$$[\ldots \quad x_0 \quad y_0 \quad x_1 \quad y_1 \quad \ldots \quad x_n \quad y_n] \tag{5.1}$$

where $x_0 < x_1 < x_2 \ldots$ and (x_0, y_0), (x_1, y_1), ... are the breakpoints of the piecewise functions. Alternatively, if the model is of Type 2, the following format is applied:

$$[\ldots \; c_{(n-1)} \quad \ldots \quad c_1 \quad c_0] \tag{5.2}$$

where the unit's cost is a function of the hourly megawatt output (P) and defined by the polynomial: $c_0 + c_1P + \ldots + c_{(n-1)}P^{(n-1)}$. It should be noted that for both types of units, the size of the Generator Cost Matrix will vary depending on the number of terms used to provide an accurate model. As defined in Chapter 4, models used in this investigation will be defined using a second order polynomial expression.

Appendix E Test Case File – 57-Bus Power System

```
function [baseMVA, bus, gen, branch, areas, gencost] = case57

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  CASE57a  Power flow data for IEEE 57 bus test case.
%  Last modified 2006-02-26 by RCollett
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Converted from IEEE CDF file from:
%      http://www.ee.washington.edu/research/pstca/
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%-----  Power Flow Data  -----%%
%% system MVA base
baseMVA = 100;

%% bus data
%Bus ID  BusType  Pd  Qd  AreaNum  Vm  Va  BasekV  Zone  Vmax  Vmin
bus = [
        1    3    55    17    0    0    1    1.04    0      0    1    1.06    0.94;
        2    2    3     88    0    0    1    1.01    -1.18  0    1    1.06    0.94;
        3    2    41    21    0    0    1    0.985   -5.97  0    1    1.06    0.94;
        4    1    0     0     0    0    1    0.981   -7.32  0    1    1.06    0.94;
        5    1    13    4     0    0    1    0.976   -8.52  0    1    1.06    0.94;
        6    2    75    2     0    0    1    0.98    -8.65  0    1    1.06    0.94;
        7    1    0     0     0    0    1    0.984   -7.58  0    1    1.06    0.94;
        8    2    150   22    0    0    1    1.005   -4.45  0    1    1.06    0.94;
        9    2    121   26    0    0    1    0.98    -9.56  0    1    1.06    0.94;
        10   1    5     2     0    0    1    0.986   -11.43 0    1    1.06    0.94;
        11   1    0     0     0    0    1    0.974   -10.17 0    1    1.06    0.94;
        12   2    377   24    0    0    1    1.015   -10.46 0    1    1.06    0.94;
        13   1    18    2.3   0    0    1    0.979   -9.79  0    1    1.06    0.94;
        14   1    10.5  5.3   0    0    1    0.97    -9.33  0    1    1.06    0.94;
        15   1    22    5     0    0    1    0.988   -7.18  0    1    1.06    0.94;
        16   1    43    3     0    0    1    1.013   -8.85  0    1    1.06    0.94;
        17   1    42    8     0    0    1    1.017   -5.39  0    1    1.06    0.94;
        18   1    27.2  9.8   0    0    1    .001    -11.71 0    1    1.06    0.94;
        19   1    3.3   0.6   0    0    1    0.97    -13.2  0    1    1.06    0.94;
        20   1    2.3   1     0    0    1    0.964   -13.41 0    1    1.06    0.94;
        21   1    0     0     0    0    1    1.008   -12.89 0    1    1.06    0.94;
        22   1    0     0     0    0    1    1.01    -12.84 0    1    1.06    0.94;
        23   1    6.3   2.1   0    0    1    1.008   -12.91 0    1    1.06    0.94;
        24   1    0     0     0    0    1    0.999   -13.25 0    1    1.06    0.94;
        25   1    6.3   3.2   0    0    1    0.982   -18.13 0    1    1.06    0.94;
        26   1    0     0     0    0    1    0.959   -12.95 0    1    1.06    0.94;
        27   1    9.3   0.5   0    0    1    0.982   -11.48 0    1    1.06    0.94;
        28   1    4.6   2.3   0    0    1    0.997   -10.45 0    1    1.06    0.94;
        29   1    17    2.6   0    0    1    1.01    -9.75  0    1    1.06    0.94;
        30   1    3.6   1.8   0    0    1    0.962   -18.68 0    1    1.06    0.94;
        31   1    5.8   2.9   0    0    1    0.936   -19.34 0    1    1.06    0.94;
        32   1    1.6   0.8   0    0    1    0.949   -18.46 0    1    1.06    0.94;
        33   1    3.8   1.9   0    0    1    0.947   -18.5  0    1    1.06    0.94;
        34   1    0     0     0    0    1    0.959   -14.1  0    1    1.06    0.94;
        35   1    6     3     0    0    1    0.966   -13.86 0    1    1.06    0.94;
        36   1    0     0     0    0    1    0.976   -13.59 0    1    1.06    0.94;
        37   1    0     0     0    0    1    0.985   -13.41 0    1    1.06    0.94;
        38   1    14    7     0    0    1    1.013   -12.71 0    1    1.06    0.94;
        39   1    0     0     0    0    1    0.983   -13.46 0    1    1.06    0.94;
        40   1    0     0     0    0    1    0.973   -13.62 0    1    1.06    0.94;
        41   1    6.3   3     0    0    1    0.996   -14.05 0    1    1.06    0.94;
        42   1    7.1   4.4   0    0    1    0.966   -15.5  0    1    1.06    0.94;
        43   1    2     1     0    0    1    1.01    -11.33 0    1    1.06    0.94;
        44   1    12    1.8   0    0    1    1.017   -11.86 0    1    1.06    0.94;
        45   1    0     0     0    0    1    1.036   -9.25  0    1    1.06    0.94;
        46   1    0     0     0    0    1    1.05    -11.89 0    1    1.06    0.94;
```

```
        47      1       29.7    11.6    0       0       1       1.033   -12.49  0       1       1.06    0.94;
        48      1       0       0       0       0       1       1.027   -12.59  0       1       1.06    0.94;
        49      1       18      8.5     0       0       1       1.036   -12.92  0       1       1.06    0.94;
        50      1       21      10.5    0       0       1       1.023   -13.39  0       1       1.06    0.94;
        51      1       18      5.3     0       0       1       1.052   -12.52  0       1       1.06    0.94;
        52      1       4.9     2.2     0       0       1       0.98    -11.47  0       1       1.06    0.94;
        53      1       20      10      0       0       1       0.971   -12.23  0       1       1.06    0.94;
        54      1       4.1     1.4     0       0       1       0.996   -11.69  0       1       1.06    0.94;
        55      1       6.8     3.4     0       0       1       1.031   -10.78  0       1       1.06    0.94;
        56      1       7.6     2.2     0       0       1       0.968   -16.04  0       1       1.06    0.94;
        57      1       6.7     2       0       0       1       0.965   -16.56  0       1       1.06    0.94;
];

%% generator data
%       Bus#  Pgen    Qgen    Qmax    Qmin    Vset    MVA Base  Status  Pmax    Pmin
gen = [
        1       128.9   -16.1   200     -140    1.04    100     1       575.88  50;
        2       0       -0.8    50      -17     1.01    100     1       100     20;
        3       40      -1      60      -10     0.985   100     1       140     20;
        6       0       0.8     25      -8      0.98    100     1       100     20;
        8       450     62.1    200     -140    1.005   100     1       550     50;
        9       0       2.2     9       -3      0.98    100     1       100     20;
        12      310     128.5   155     -150    1.015   100     1       410     30;
];

%% branch data
%FrBus  ToBus  R       X       B       LTermRat.  ShTerm Rat  EmergRat  TurnRat  PhShift  Status
branch = [
        1       2       0.0083  0.028   0.129   9900    0       0       0       0       1;
        2       3       0.0298  0.085   0.0818  9900    0       0       0       0       1;
        3       4       0.0112  0.0366  0.038   9900    0       0       0       0       1;
        4       5       0.0625  0.132   0.0258  9900    0       0       0       0       1;
        4       6       0.043   0.148   0.0348  9900    0       0       0       0       1;
        6       7       0.02    0.102   0.0276  9900    0       0       0       0       1;
        6       8       0.0339  0.173   0.047   9900    0       0       0       0       1;
        8       9       0.0099  0.0505  0.0548  9900    0       0       0       0       1;
        9       10      0.0369  0.1679  0.044   9900    0       0       0       0       1;
        9       11      0.0258  0.0848  0.0218  9900    0       0       0       0       1;
        9       12      0.0648  0.295   0.0772  9900    0       0       0       0       1;
        9       13      0.0481  0.158   0.0406  9900    0       0       0       0       1;
        13      14      0.0132  0.0434  0.011   9900    0       0       0       0       1;
        13      15      0.0269  0.0869  0.023   9900    0       0       0       0       1;
        1       15      0.0178  0.091   0.0988  9900    0       0       0       0       1;
        1       16      0.0454  0.206   0.0546  9900    0       0       0       0       1;
        1       17      0.0238  0.108   0.0286  9900    0       0       0       0       1;
        3       15      0.0162  0.053   0.0544  9900    0       0       0       0       1;
        4       18      0       0.555   0       9900    0       0       0.97    0       1;
        4       18      0       0.43    0       9900    0       0       0.978   0       1;
        5       6       0.0302  0.0641  0.0124  9900    0       0       0       0       1;
        7       8       0.0139  0.0712  0.0194  9900    0       0       0       0       1;
        10      12      0.0277  0.1262  0.0328  9900    0       0       0       0       1;
        11      13      0.0223  0.0732  0.0188  9900    0       0       0       0       1;
        12      13      0.0178  0.058   0.0604  9900    0       0       0       0       1;
        12      16      0.018   0.0813  0.0216  9900    0       0       0       0       1;
        12      17      0.0397  0.179   0.0476  9900    0       0       0       0       1;
        14      15      0.0171  0.0547  0.0148  9900    0       0       0       0       1;
        18      19      0.461   0.685   0       9900    0       0       0       0       1;
        19      20      0.283   0.434   0       9900    0       0       0       0       1;
        21      20      0       0.7767  0       9900    0       0       1.043   0       1;
        21      22      0.0736  0.117   0       9900    0       0       0       0       1;
        22      23      0.0099  0.0152  0       9900    0       0       0       0       1;
        23      24      0.166   0.256   0.0084  9900    0       0       0       0       1;
        24      25      0       1.182   0       9900    0       0       1       0       1;
        24      25      0       1.23    0       9900    0       0       1       0       1;
        24      26      0       0.0473  0       9900    0       0       1.043   0       1;
        26      27      0.165   0.254   0       9900    0       0       0       0       1;
        27      28      0.0618  0.0954  0       9900    0       0       0       0       1;
        28      29      0.0418  0.0587  0       9900    0       0       0       0       1;
        7       29      0       0.0648  0       9900    0       0       0.967   0       1;
        25      30      0.135   0.202   0       9900    0       0       0       0       1;
```

```
        30      31      0.326   0.497   0       9900    0       0       0       0       1;
        31      32      0.507   0.755   0       9900    0       0       0       0       1;
        32      33      0.0392  0.036   0       9900    0       0       0       0       1;
        34      32      0       0.953   0       9900    0       0       0.975   0       1;
        34      35      0.052   0.078   0.0032  9900    0       0       0       0       1;
        35      36      0.043   0.0537  0.0016  9900    0       0       0       0       1;
        36      37      0.029   0.0366  0       9900    0       0       0       0       1;
        37      38      0.0651  0.1009  0.002   9900    0       0       0       0       1;
        37      39      0.0239  0.0379  0       9900    0       0       0       0       1;
        36      40      0.03    0.0466  0       9900    0       0       0       0       1;
        22      38      0.0192  0.0295  0       9900    0       0       0       0       1;
        11      41      0       0.749   0       9900    0       0       0.955   0       1;
        41      42      0.207   0.352   0       9900    0       0       0       0       1;
        41      43      0       0.412   0       9900    0       0       0       0       1;
        38      44      0.0289  0.0585  0.002   9900    0       0       0       0       1;
        15      45      0       0.1042  0       9900    0       0       0.955   0       1;
        14      46      0       0.0735  0       9900    0       0       0.9     0       1;
        46      47      0.023   0.068   0.0032  9900    0       0       0       0       1;
        47      48      0.0182  0.0233  0       9900    0       0       0       0       1;
        48      49      0.0834  0.129   0.0048  9900    0       0       0       0       1;
        49      50      0.0801  0.128   0       9900    0       0       0       0       1;
        50      51      0.1386  0.22    0       9900    0       0       0       0       1;
        10      51      0       0.0712  0       9900    0       0       0.93    0       1;
        13      49      0       0.191   0       9900    0       0       0.895   0       1;
        29      52      0.1442  0.187   0       9900    0       0       0       0       1;
        52      53      0.0762  0.0984  0       9900    0       0       0       0       1;
        53      54      0.1878  0.232   0       9900    0       0       0       0       1;
        54      55      0.1732  0.2265  0       9900    0       0       0       0       1;
        11      43      0       0.153   0       9900    0       0       0.958   0       1;
        44      45      0.0624  0.1242  0.004   9900    0       0       0       0       1;
        40      56      0       1.195   0       9900    0       0       0.958   0       1;
        56      41      0.553   0.549   0       9900    0       0       0       0       1;
        56      42      0.2125  0.354   0       9900    0       0       0       0       1;
        39      57      0       1.355   0       9900    0       0       0.98    0       1;
        57      56      0.174   0.26    0       9900    0       0       0       0       1;
        38      49      0.115   0.177   0.003   9900    0       0       0       0       1;
        38      48      0.0312  0.0482  0       9900    0       0       0       0       1;
        9       55      0       0.1205  0       9900    0       0       0.94    0       1;
];

%%----- OPF Data -----%%
%% area data
%AreaID  RefBusID
areas = [
        1       1;
];

%% generator cost data
%       1       start-up    shutdown    n       x0          y0      ...     xn      yn
%       2       start-up    shutdown    n       c(n-1)      ...     c0
gencost = [
        2       100     0       3       0.0775795   20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.25        20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.0222222   20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.0322581   20      0;
];

return;
```

Appendix F Test Case File – 118-Bus Power System

```
function [baseMVA, bus, gen, branch, areas, gencost] = case118a

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  CASE118a  Power flow data for IEEE 118 bus test case.
%  Last modified 2006-02-26 by RCollett
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Converted from IEEE CDF file from:
%      http://www.ee.washington.edu/research/pstca/
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%-----  Power Flow Data  -----%%
%% system MVA base
baseMVA = 100;

%% bus data
%Bus ID  BusType  Pd  Qd  AreaNum  Vm  Va  BasekV  Zone  Vmax  Vmin
bus = [
        1    2    51    27    0    0    1    0.955    10.67    0    1    1.06    0.94;
        2    1    20    9     0    0    1    0.971    11.22    0    1    1.06    0.94;
        3    1    39    10    0    0    1    0.968    11.56    0    1    1.06    0.94;
        4    2    39    12    0    0    1    0.998    15.28    0    1    1.06    0.94;
        5    1    0     0     0    0    1    1.002    15.73    0    1    1.06    0.94;
        6    2    52    22    0    0    1    0.99     13       0    1    1.06    0.94;
        7    1    19    2     0    0    1    0.989    12.56    0    1    1.06    0.94;
        8    2    28    0     0    0    1    1.015    20.77    0    1    1.06    0.94;
        9    1    0     0     0    0    1    1.043    28.02    0    1    1.06    0.94;
        10   2    0     0     0    0    1    1.05     35.61    0    1    1.06    0.94;
        11   1    70    23    0    0    1    0.985    12.72    0    1    1.06    0.94;
        12   2    47    10    0    0    1    0.99     12.2     0    1    1.06    0.94;
        13   1    34    16    0    0    1    0.968    11.35    0    1    1.06    0.94;
        14   1    14    1     0    0    1    0.984    11.5     0    1    1.06    0.94;
        15   2    90    30    0    0    1    0.97     11.23    0    1    1.06    0.94;
        16   1    25    10    0    0    1    0.984    11.91    0    1    1.06    0.94;
        17   1    11    3     0    0    1    .995     13.74    0    1    1.06    0.94;
        18   2    60    34    0    0    1    0.973    11.53    0    1    1.06    0.94;
        19   2    45    25    0    0    1    0.963    11.05    0    1    1.06    0.94;
        20   1    18    3     0    0    1    0.958    11.93    0    1    1.06    0.94;
        21   1    14    8     0    0    1    0.959    13.52    0    1    1.06    0.94;
        22   1    10    5     0    0    1    0.97     16.08    0    1    1.06    0.94;
        23   1    7     3     0    0    1    1        21       0    1    1.06    0.94;
        24   2    13    0     0    0    1    0.992    20.89    0    1    1.06    0.94;
        25   2    0     0     0    0    1    1.05     27.93    0    1    1.06    0.94;
        26   2    0     0     0    0    1    1.015    29.71    0    1    1.06    0.94;
        27   2    71    13    0    0    1    0.968    15.35    0    1    1.06    0.94;
        28   1    17    7     0    0    1    0.962    13.62    0    1    1.06    0.94;
        29   1    24    4     0    0    1    0.963    12.63    0    1    1.06    0.94;
        30   1    0     0     0    0    1    0.968    18.79    0    1    1.06    0.94;
        31   2    43    27    0    0    1    0.967    12.75    0    1    1.06    0.94;
        32   2    59    23    0    0    1    0.964    14.8     0    1    1.06    0.94;
        33   1    23    9     0    0    1    0.972    10.63    0    1    1.06    0.94;
        34   2    59    26    0    0    1    0.986    11.3     0    1    1.06    0.94;
        35   1    33    9     0    0    1    0.981    10.87    0    1    1.06    0.94;
        36   2    31    17    0    0    1    0.98     10.87    0    1    1.06    0.94;
        37   1    0     0     0    0    1    0.992    11.77    0    1    1.06    0.94;
        38   1    0     0     0    0    1    0.962    16.91    0    1    1.06    0.94;
        39   1    27    11    0    0    1    0.97     8.41     0    1    1.06    0.94;
        40   2    66    23    0    0    1    0.97     7.35     0    1    1.06    0.94;
        41   1    37    10    0    0    1    0.967    6.92     0    1    1.06    0.94;
        42   2    96    23    0    0    1    0.985    8.53     0    1    1.06    0.94;
        43   1    18    7     0    0    1    0.978    11.28    0    1    1.06    0.94;
        44   1    16    8     0    0    1    0.985    13.82    0    1    1.06    0.94;
        45   1    53    22    0    0    1    0.987    15.67    0    1    1.06    0.94;
```

46	2	28	10	0	0	1	1.005	18.49	0	1	1.06	0.94;
47	1	34	0	0	0	1	1.017	20.73	0	1	1.06	0.94;
48	1	20	11	0	0	1	1.021	19.93	0	1	1.06	0.94;
49	2	87	30	0	0	1	1.025	20.94	0	1	1.06	0.94;
50	1	17	4	0	0	1	1.001	18.9	0	1	1.06	0.94;
51	1	17	8	0	0	1	0.967	16.28	0	1	1.06	0.94;
52	1	18	5	0	0	1	0.957	15.32	0	1	1.06	0.94;
53	1	23	11	0	0	1	0.946	14.35	0	1	1.06	0.94;
54	2	113	32	0	0	1	0.955	15.26	0	1	1.06	0.94;
55	2	63	22	0	0	1	0.952	14.97	0	1	1.06	0.94;
56	2	84	18	0	0	1	0.954	15.16	0	1	1.06	0.94;
57	1	12	3	0	0	1	0.971	16.36	0	1	1.06	0.94;
58	1	12	3	0	0	1	0.959	15.51	0	1	1.06	0.94;
59	2	277	113	0	0	1	0.985	19.37	0	1	1.06	0.94;
60	1	78	3	0	0	1	0.993	23.15	0	1	1.06	0.94;
61	2	0	0	0	0	1	0.995	24.04	0	1	1.06	0.94;
62	2	77	14	0	0	1	0.998	23.43	0	1	1.06	0.94;
63	1	0	0	0	0	1	0.969	22.75	0	1	1.06	0.94;
64	1	0	0	0	0	1	0.984	24.52	0	1	1.06	0.94;
65	2	0	0	0	0	1	1.005	27.65	0	1	1.06	0.94;
66	2	39	18	0	0	1	1.05	27.48	0	1	1.06	0.94;
67	1	28	7	0	0	1	1.02	24.84	0	1	1.06	0.94;
68	1	0	0	0	0	1	1.003	27.55	0	1	1.06	0.94;
69	3	0	0	0	0	1	1.035	30	0	1	1.06	0.94;
70	2	66	20	0	0	1	0.984	22.58	0	1	1.06	0.94;
71	1	0	0	0	0	1	0.987	22.15	0	1	1.06	0.94;
72	2	12	0	0	0	1	0.98	20.98	0	1	1.06	0.94;
73	2	6	0	0	0	1	0.991	21.94	0	1	1.06	0.94;
74	2	68	27	0	0	1	0.958	21.64	0	1	1.06	0.94;
75	1	47	11	0	0	1	0.967	22.91	0	1	1.06	0.94;
76	2	68	36	0	0	1	0.943	21.77	0	1	1.06	0.94;
77	2	61	28	0	0	1	1.006	26.72	0	1	1.06	0.94;
78	1	71	26	0	0	1	1.003	26.42	0	1	1.06	0.94;
79	1	39	32	0	0	1	1.009	26.72	0	1	1.06	0.94;
80	2	130	26	0	0	1	1.04	28.96	0	1	1.06	0.94;
81	1	0	0	0	0	1	0.997	28.1	0	1	1.06	0.94;
82	1	54	27	0	0	1	0.989	27.24	0	1	1.06	0.94;
83	1	20	10	0	0	1	0.985	28.42	0	1	1.06	0.94;
84	1	11	7	0	0	1	0.98	30.95	0	1	1.06	0.94;
85	2	24	15	0	0	1	0.985	32.51	0	1	1.06	0.94;
86	1	21	10	0	0	1	0.987	31.14	0	1	1.06	0.94;
87	2	0	0	0	0	1	1.015	31.4	0	1	1.06	0.94;
88	1	48	10	0	0	1	0.987	35.64	0	1	1.06	0.94;
89	2	0	0	0	0	1	1.005	39.69	0	1	1.06	0.94;
90	2	163	42	0	0	1	0.985	33.29	0	1	1.06	0.94;
91	2	10	0	0	0	1	0.98	33.31	0	1	1.06	0.94;
92	2	65	10	0	0	1	0.993	33.8	0	1	1.06	0.94;
93	1	12	7	0	0	1	0.987	30.79	0	1	1.06	0.94;
94	1	30	16	0	0	1	0.991	28.64	0	1	1.06	0.94;
95	1	42	31	0	0	1	0.981	27.67	0	1	1.06	0.94;
96	1	38	15	0	0	1	0.993	27.51	0	1	1.06	0.94;
97	1	15	9	0	0	1	1.011	27.88	0	1	1.06	0.94;
98	1	34	8	0	0	1	1.024	27.4	0	1	1.06	0.94;
99	2	42	0	0	0	1	1.01	27.04	0	1	1.06	0.94;
100	2	37	18	0	0	1	1.017	28.03	0	1	1.06	0.94;
101	1	22	15	0	0	1	0.993	29.61	0	1	1.06	0.94;
102	1	5	3	0	0	1	0.991	32.3	0	1	1.06	0.94;
103	2	23	16	0	0	1	1.001	24.44	0	1	1.06	0.94;
104	2	38	25	0	0	1	0.971	21.69	0	1	1.06	0.94;
105	2	31	26	0	0	1	0.965	20.57	0	1	1.06	0.94;
106	1	43	16	0	0	1	0.962	20.32	0	1	1.06	0.94;
107	2	50	12	0	0	1	0.952	17.53	0	1	1.06	0.94;
108	1	2	1	0	0	1	0.967	19.38	0	1	1.06	0.94;
109	1	8	3	0	0	1	0.967	18.93	0	1	1.06	0.94;
110	2	39	30	0	0	1	0.973	18.09	0	1	1.06	0.94;
111	2	0	0	0	0	1	0.98	19.74	0	1	1.06	0.94;
112	2	68	13	0	0	1	0.975	14.99	0	1	1.06	0.94;
113	2	6	0	0	0	1	0.993	13.74	0	1	1.06	0.94;
114	1	8	3	0	0	1	0.96	14.46	0	1	1.06	0.94;
115	1	22	7	0	0	1	0.96	14.46	0	1	1.06	0.94;

```
        116     2       184     0       0       0       1       1.005   27.12   0       1       1.06    0.94;
        117     1       20      8       0       0       1       0.974   10.67   0       1       1.06    0.94;
        118     1       33      15      0       0       1       0.949   21.92   0       1       1.06    0.94;
];

%% generator data
%       Bus#  Pgen  Qgen  Qmax  Qmin  Vset  MVA Base  Status  Pmax  Pmin
gen = [
        1       0       0       15      -5      0.955   100     1       100     30;
        4       0       0       300     -300    0.998   100     1       100     30;
        6       0       0       50      -13     0.99    100     1       100     30;
        8       0       0       300     -300    1.015   100     1       100     30;
        10      450     0       200     -147    1.05    100     1       550     75;
        12      85      0       120     -35     0.99    100     1       185     50;
        15      0       0       30      -10     0.97    100     1       100     30;
        18      0       0       50      -16     0.973   100     1       100     30;
        19      0       0       24      -8      0.962   100     1       100     30;
        24      0       0       300     -300    0.992   100     1       100     30;
        25      220     0       140     -47     1.05    100     1       320     60;
        26      314     0       1000    -1000   1.015   100     1       414     60;
        27      0       0       300     -300    0.968   100     1       100     30;
        31      7       0       300     -300    0.967   100     1       107     30;
        32      0       0       42      -14     0.963   100     1       100     30;
        34      0       0       24      -8      0.984   100     1       100     30;
        36      0       0       24      -8      0.98    100     1       100     30;
        40      0       0       300     -300    0.97    100     1       100     30;
        42      0       0       300     -300    0.985   100     1       100     30;
        46      19      0       100     -100    1.005   100     1       119     30;
        49      204     0       210     -85     1.025   100     1       304     30;
        54      48      0       300     -300    0.955   100     1       148     30;
        55      0       0       23      -8      0.952   100     1       100     30;
        56      0       0       15      -8      0.954   100     1       100     30;
        59      155     0       180     -60     0.985   100     1       255     50;
        61      160     0       300     -100    0.995   100     1       260     40;
        62      0       0       20      -20     0.998   100     1       100     30;
        65      391     0       200     -67     1.005   100     1       491     50;
        66      392     0       200     -67     1.05    100     1       492     50;
        69      516.4   0       300     -300    1.035   100     1       805.2   100;
        70      0       0       32      -10     0.984   100     1       100     30;
        72      0       0       100     -100    0.98    100     1       100     30;
        73      0       0       100     -100    0.991   100     1       100     30;
        74      0       0       9       -6      0.958   100     1       100     30;
        76      0       0       23      -8      0.943   100     1       100     30;
        77      0       0       70      -20     1.006   100     1       100     30;
        80      477     0       280     -165    1.04    100     1       577     60;
        85      0       0       23      -8      0.985   100     1       100     30;
        87      4       0       1000    -100    1.015   100     1       104     30;
        89      607     0       300     -210    1.005   100     1       707     50;
        90      0       0       300     -300    0.985   100     1       100     30;
        91      0       0       100     -100    0.98    100     1       100     30;
        92      0       0       9       -3      0.99    100     1       100     30;
        99      0       0       100     -100    1.01    100     1       100     30;
        100     252     0       155     -50     1.017   100     1       352     50;
        103     40      0       40      -15     1.01    100     1       140     30;
        104     0       0       23      -8      0.971   100     1       100     30;
        105     0       0       23      -8      0.965   100     1       100     30;
        107     0       0       200     -200    0.952   100     1       100     30;
        110     0       0       23      -8      0.973   100     1       100     30;
        111     36      0       1000    -100    0.98    100     1       136     40;
        112     0       0       1000    -100    0.975   100     1       100     30;
        113     0       0       200     -100    0.993   100     1       100     30;
        116     0       0       1000    -1000   1.005   100     1       100     30;
];

%% branch data
%FrBus  ToBus  R  X  B  LTermRat.  ShTerm Rat  EmergRat  TurnRat  PhShift  Status
branch = [
        1       2       0.0303  0.0999  0.0254  9900    0       0       0       0       1;
        1       3       0.0129  0.0424  0.01082 9900    0       0       0       0       1;
        4       5       0.00176 0.00798 0.0021  9900    0       0       0       0       1;
```

3	5	0.0241	0.108	0.0284	9900	0	0	0	0	1;
5	6	0.0119	0.054	0.01426	9900	0	0	0	0	1;
6	7	0.00459	0.0208	0.0055	9900	0	0	0	0	1;
8	9	0.00244	0.0305	1.162	9900	0	0	0	0	1;
8	5	0	0.0267	0	9900	0	0	0.985	0	1;
9	10	0.00258	0.0322	1.23	9900	0	0	0	0	1;
4	11	0.0209	0.0688	0.01748	9900	0	0	0	0	1;
5	11	0.0203	0.0682	0.01738	9900	0	0	0	0	1;
11	12	0.00595	0.0196	0.00502	9900	0	0	0	0	1;
2	12	0.0187	0.0616	0.01572	9900	0	0	0	0	1;
3	12	0.0484	0.16	0.0406	9900	0	0	0	0	1;
7	12	0.00862	0.034	0.00874	9900	0	0	0	0	1;
11	13	0.02225	0.0731	0.01876	9900	0	0	0	0	1;
12	14	0.0215	0.0707	0.01816	9900	0	0	0	0	1;
13	15	0.0744	0.2444	0.06268	9900	0	0	0	0	1;
14	15	0.0595	0.195	0.0502	9900	0	0	0	0	1;
12	16	0.0212	0.0834	0.0214	9900	0	0	0	0	1;
15	17	0.0132	0.0437	0.0444	9900	0	0	0	0	1;
16	17	0.0454	0.1801	0.0466	9900	0	0	0	0	1;
17	18	0.0123	0.0505	0.01298	9900	0	0	0	0	1;
18	19	0.01119	0.0493	0.01142	9900	0	0	0	0	1;
19	20	0.0252	0.117	0.0298	9900	0	0	0	0	1;
15	19	0.012	0.0394	0.0101	9900	0	0	0	0	1;
20	21	0.0183	0.0849	0.0216	9900	0	0	0	0	1;
21	22	0.0209	0.097	0.0246	9900	0	0	0	0	1;
22	23	0.0342	0.159	0.0404	9900	0	0	0	0	1;
23	24	0.0135	0.0492	0.0498	9900	0	0	0	0	1;
23	25	0.0156	0.08	0.0864	9900	0	0	0	0	1;
26	25	0	0.0382	0	9900	0	0	0.96	0	1;
25	27	0.0318	0.163	0.1764	9900	0	0	0	0	1;
27	28	0.01913	0.0855	0.0216	9900	0	0	0	0	1;
28	29	0.0237	0.0943	0.0238	9900	0	0	0	0	1;
30	17	0	0.0388	0	9900	0	0	0.96	0	1;
8	30	0.00431	0.0504	0.514	9900	0	0	0	0	1;
26	30	0.00799	0.086	0.908	9900	0	0	0	0	1;
17	31	0.0474	0.1563	0.0399	9900	0	0	0	0	1;
29	31	0.0108	0.0331	0.0083	9900	0	0	0	0	1;
23	32	0.0317	0.1153	0.1173	9900	0	0	0	0	1;
31	32	0.0298	0.0985	0.0251	9900	0	0	0	0	1;
27	32	0.0229	0.0755	0.01926	9900	0	0	0	0	1;
15	33	0.038	0.1244	0.03194	9900	0	0	0	0	1;
19	34	0.0752	0.247	0.0632	9900	0	0	0	0	1;
35	36	0.00224	0.0102	0.00268	9900	0	0	0	0	1;
35	37	0.011	0.0497	0.01318	9900	0	0	0	0	1;
33	37	0.0415	0.142	0.0366	9900	0	0	0	0	1;
34	36	0.00871	0.0268	0.00568	9900	0	0	0	0	1;
34	37	0.00256	0.0094	0.00984	9900	0	0	0	0	1;
38	37	0	0.0375	0	9900	0	0	0.935	0	1;
37	39	0.0321	0.106	0.027	9900	0	0	0	0	1;
37	40	0.0593	0.168	0.042	9900	0	0	0	0	1;
30	38	0.00464	0.054	0.422	9900	0	0	0	0	1;
39	40	0.0184	0.0605	0.01552	9900	0	0	0	0	1;
40	41	0.0145	0.0487	0.01222	9900	0	0	0	0	1;
40	42	0.0555	0.183	0.0466	9900	0	0	0	0	1;
41	42	0.041	0.135	0.0344	9900	0	0	0	0	1;
43	44	0.0608	0.2454	0.06068	9900	0	0	0	0	1;
34	43	0.0413	0.1681	0.04226	9900	0	0	0	0	1;
44	45	0.0224	0.0901	0.0224	9900	0	0	0	0	1;
45	46	0.04	0.1356	0.0332	9900	0	0	0	0	1;
46	47	0.038	0.127	0.0316	9900	0	0	0	0	1;
46	48	0.0601	0.189	0.0472	9900	0	0	0	0	1;
47	49	0.0191	0.0625	0.01604	9900	0	0	0	0	1;
42	49	0.0715	0.323	0.086	9900	0	0	0	0	1;
42	49	0.0715	0.323	0.086	9900	0	0	0	0	1;
45	49	0.0684	0.186	0.0444	9900	0	0	0	0	1;
48	49	0.0179	0.0505	0.01258	9900	0	0	0	0	1;
49	50	0.0267	0.0752	0.01874	9900	0	0	0	0	1;
49	51	0.0486	0.137	0.0342	9900	0	0	0	0	1;
51	52	0.0203	0.0588	0.01396	9900	0	0	0	0	1;
52	53	0.0405	0.1635	0.04058	9900	0	0	0	0	1;

53	54	0.0263	0.122	0.031	9900	0	0	0	0	1;
49	54	0.073	0.289	0.0738	9900	0	0	0	0	1;
49	54	0.0869	0.291	0.073	9900	0	0	0	0	1;
54	55	0.0169	0.0707	0.0202	9900	0	0	0	0	1;
54	56	0.00275	0.00955	0.00732	9900	0	0	0	0	1;
55	56	0.00488	0.0151	0.00374	9900	0	0	0	0	1;
56	57	0.0343	0.0966	0.0242	9900	0	0	0	0	1;
50	57	0.0474	0.134	0.0332	9900	0	0	0	0	1;
56	58	0.0343	0.0966	0.0242	9900	0	0	0	0	1;
51	58	0.0255	0.0719	0.01788	9900	0	0	0	0	1;
54	59	0.0503	0.2293	0.0598	9900	0	0	0	0	1;
56	59	0.0825	0.251	0.0569	9900	0	0	0	0	1;
56	59	0.0803	0.239	0.0536	9900	0	0	0	0	1;
55	59	0.04739	0.2158	0.05646	9900	0	0	0	0	1;
59	60	0.0317	0.145	0.0376	9900	0	0	0	0	1;
59	61	0.0328	0.15	0.0388	9900	0	0	0	0	1;
60	61	0.00264	0.0135	0.01456	9900	0	0	0	0	1;
60	62	0.0123	0.0561	0.01468	9900	0	0	0	0	1;
61	62	0.00824	0.0376	0.0098	9900	0	0	0	0	1;
63	59	0	0.0386	0	9900	0	0	0.96	0	1;
63	64	0.00172	0.02	0.216	9900	0	0	0	0	1;
64	61	0	0.0268	0	9900	0	0	0.985	0	1;
38	65	0.00901	0.0986	1.046	9900	0	0	0	0	1;
64	65	0.00269	0.0302	0.38	9900	0	0	0	0	1;
49	66	0.018	0.0919	0.0248	9900	0	0	0	0	1;
49	66	0.018	0.0919	0.0248	9900	0	0	0	0	1;
62	66	0.0482	0.218	0.0578	9900	0	0	0	0	1;
62	67	0.0258	0.117	0.031	9900	0	0	0	0	1;
65	66	0	0.037	0	9900	0	0	0.935	0	1;
66	67	0.0224	0.1015	0.02682	9900	0	0	0	0	1;
65	68	0.00138	0.016	0.638	9900	0	0	0	0	1;
47	69	0.0844	0.2778	0.07092	9900	0	0	0	0	1;
49	69	0.0985	0.324	0.0828	9900	0	0	0	0	1;
68	69	0	0.037	0	9900	0	0	0.935	0	1;
69	70	0.03	0.127	0.122	9900	0	0	0	0	1;
24	70	0.00221	0.4115	0.10198	9900	0	0	0	0	1;
70	71	0.00882	0.0355	0.00878	9900	0	0	0	0	1;
24	72	0.0488	0.196	0.0488	9900	0	0	0	0	1;
71	72	0.0446	0.18	0.04444	9900	0	0	0	0	1;
71	73	0.00866	0.0454	0.01178	9900	0	0	0	0	1;
70	74	0.0401	0.1323	0.03368	9900	0	0	0	0	1;
70	75	0.0428	0.141	0.036	9900	0	0	0	0	1;
69	75	0.0405	0.122	0.124	9900	0	0	0	0	1;
74	75	0.0123	0.0406	0.01034	9900	0	0	0	0	1;
76	77	0.0444	0.148	0.0368	9900	0	0	0	0	1;
69	77	0.0309	0.101	0.1038	9900	0	0	0	0	1;
75	77	0.0601	0.1999	0.04978	9900	0	0	0	0	1;
77	78	0.00376	0.0124	0.01264	9900	0	0	0	0	1;
78	79	0.00546	0.0244	0.00648	9900	0	0	0	0	1;
77	80	0.017	0.0485	0.0472	9900	0	0	0	0	1;
77	80	0.0294	0.105	0.0228	9900	0	0	0	0	1;
79	80	0.0156	0.0704	0.0187	9900	0	0	0	0	1;
68	81	0.00175	0.0202	0.808	9900	0	0	0	0	1;
81	80	0	0.037	0	9900	0	0	0.935	0	1;
77	82	0.0298	0.0853	0.08174	9900	0	0	0	0	1;
82	83	0.0112	0.03665	0.03796	9900	0	0	0	0	1;
83	84	0.0625	0.132	0.0258	9900	0	0	0	0	1;
83	85	0.043	0.148	0.0348	9900	0	0	0	0	1;
84	85	0.0302	0.0641	0.01234	9900	0	0	0	0	1;
85	86	0.035	0.123	0.0276	9900	0	0	0	0	1;
86	87	0.02828	0.2074	0.0445	9900	0	0	0	0	1;
85	88	0.02	0.102	0.0276	9900	0	0	0	0	1;
85	89	0.0239	0.173	0.047	9900	0	0	0	0	1;
88	89	0.0139	0.0712	0.01934	9900	0	0	0	0	1;
89	90	0.0518	0.188	0.0528	9900	0	0	0	0	1;
89	90	0.0238	0.0997	0.106	9900	0	0	0	0	1;
90	91	0.0254	0.0836	0.0214	9900	0	0	0	0	1;
89	92	0.0099	0.0505	0.0548	9900	0	0	0	0	1;
89	92	0.0393	0.1581	0.0414	9900	0	0	0	0	1;
91	92	0.0387	0.1272	0.03268	9900	0	0	0	0	1;

```
        92      93      0.0258  0.0848  0.0218  9900    0       0       0       0       1;
        92      94      0.0481  0.158   0.0406  9900    0       0       0       0       1;
        93      94      0.0223  0.0732  0.01876 9900    0       0       0       0       1;
        94      95      0.0132  0.0434  0.0111  9900    0       0       0       0       1;
        80      96      0.0356  0.182   0.0494  9900    0       0       0       0       1;
        82      96      0.0162  0.053   0.0544  9900    0       0       0       0       1;
        94      96      0.0269  0.0869  0.023   9900    0       0       0       0       1;
        80      97      0.0183  0.0934  0.0254  9900    0       0       0       0       1;
        80      98      0.0238  0.108   0.0286  9900    0       0       0       0       1;
        80      99      0.0454  0.206   0.0546  9900    0       0       0       0       1;
        92      100     0.0648  0.295   0.0472  9900    0       0       0       0       1;
        94      100     0.0178  0.058   0.0604  9900    0       0       0       0       1;
        95      96      0.0171  0.0547  0.01474 9900    0       0       0       0       1;
        96      97      0.0173  0.0885  0.024   9900    0       0       0       0       1;
        98      100     0.0397  0.179   0.0476  9900    0       0       0       0       1;
        99      100     0.018   0.0813  0.0216  9900    0       0       0       0       1;
        100     101     0.0277  0.1262  0.0328  9900    0       0       0       0       1;
        92      102     0.0123  0.0559  0.01464 9900    0       0       0       0       1;
        101     102     0.0246  0.112   0.0294  9900    0       0       0       0       1;
        100     103     0.016   0.0525  0.0536  9900    0       0       0       0       1;
        100     104     0.0451  0.204   0.0541  9900    0       0       0       0       1;
        103     104     0.0466  0.1584  0.0407  9900    0       0       0       0       1;
        103     105     0.0535  0.1625  0.0408  9900    0       0       0       0       1;
        100     106     0.0605  0.229   0.062   9900    0       0       0       0       1;
        104     105     0.00994 0.0378  0.00986 9900    0       0       0       0       1;
        105     106     0.014   0.0547  0.01434 9900    0       0       0       0       1;
        105     107     0.053   0.183   0.0472  9900    0       0       0       0       1;
        105     108     0.0261  0.0703  0.01844 9900    0       0       0       0       1;
        106     107     0.053   0.183   0.0472  9900    0       0       0       0       1;
        108     109     0.0105  0.0288  0.0076  9900    0       0       0       0       1;
        103     110     0.03906 0.1813  0.0461  9900    0       0       0       0       1;
        109     110     0.0278  0.0762  0.0202  9900    0       0       0       0       1;
        110     111     0.022   0.0755  0.02    9900    0       0       0       0       1;
        110     112     0.0247  0.064   0.062   9900    0       0       0       0       1;
        17      113     0.00913 0.0301  0.00768 9900    0       0       0       0       1;
        32      113     0.0615  0.203   0.0518  9900    0       0       0       0       1;
        32      114     0.0135  0.0612  0.01628 9900    0       0       0       0       1;
        27      115     0.0164  0.0741  0.01972 9900    0       0       0       0       1;
        114     115     0.0023  0.0104  0.00276 9900    0       0       0       0       1;
        68      116     0.00034 0.00405 0.164   9900    0       0       0       0       1;
        12      117     0.0329  0.14    0.0358  9900    0       0       0       0       1;
        75      118     0.0145  0.0481  0.01198 9900    0       0       0       0       1;
        76      118     0.0164  0.0544  0.01356 9900    0       0       0       0       1;
];

%%----- OPF Data -----%%
%% area data
%AreaID RefBusID
areas = [
        1       1;
];

%% generator cost data
%       1       start-up    shutdown    n       x0          y0      ...     xn      yn
%       2       start-up    shutdown    n       c(n-1)      ...     c0
gencost = [
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.0222222   20      0;
        2       100     0       3       0.117647    20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.0454545   20      0;
        2       100     0       3       0.0318471   20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       1.42857     20      0;
```

```
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.526316    20      0;
        2       100     0       3       0.0490196   20      0;
        2       100     0       3       0.208333    20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.0645161   20      0;
        2       100     0       3       0.0625      20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.0255754   20      0;
        2       100     0       3       0.0255102   20      0;
        2       100     0       3       0.0193648   20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.0209644   20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       2.5         20      0;
        2       100     0       3       0.0164745   20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.0396825   20      0;
        2       100     0       3       0.25        20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.277778    20      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
        2       100     0       3       0.01        40      0;
];

return;
```