OPTIMAL SWITCHING IN POWER DISTRIBUTION NETWORKS BY SIMULATING ANNEALING

by

Michael H. C. Lau

A Thesis
Presented To
The Faculty of Graduate Studies
University of Manitoba
In Partial Fulfillment
of the Requirements for the Degree of

Master of Science

in the
Department of Electrical Engineering
University of Manitoba
Winnipeg, Manitoba
Canada

January 1989
© Michael Lau, 1989

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-48120-X

OPTIMAL SWITCHING IN POWER DISTRIBUTION NETWORKS BY SIMULATING ANNEALING

BY

MICHAEL H.C. LAU

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

© 1989

Permission has been granted to the LIBRARY OF THE UNIVERSITY OF MANITOBA to lend or sell copies of this thesis, to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film, and UNIVERSITY MICROFILMS to publish an abstract of this thesis.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize anyone to reproduce this thesis by photocopying or by other means, in total or in part.

Michael H. C. Lau

The University of Manitoba requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

This thesis investigates the use of simulated annealing to optimize the performance of the distribution system by changing the on/off status of the switches. The simulated annealing algorithm is theoretically based on the analogy of statistical physics of molecules during the process of cooling. The algorithm involves the finding of a solution that minimizes unacceptable conditions such as equipment overload and bus voltage violations. The network configuration is modelled in detail, and the parameters such as current and voltages are evaluated using an ac load flow. The optimization of distribution network under operating, contingency and switch adding studies was a success. The behavior of the algorithm used in the optimization resembled those in statistical physics.

Acknowledgements

The author wishes to express his deep gratitude to Professor A. Gole who proposed and supervised the research throughout.

Special acknowledgement go to Manitoba Hydro's feeder optimization project team members, especially Chuck Hasselfield and Paul Wilson for many helpful discussions.

Financial assistance from the Manitoba Hydro Research Committee and the Department of Electrical Engineering of the University of Manitoba are gratefully acknowledged.

Table of Contents

Abstract iv
Acknowledgements v
Table of Contents vi
List of Figures viii
List of Tables x

CHAPTER 1: INTRODUCTION **1**
1.1 Statement of the Problem 5
1.2 Proposed Method 5
1.3 Motivation 6
1.4 Outline 6

CHAPTER 2: OPTIMIZATION TECHNIQUES **7**
2.1 Introduction 7
2.2 Combinatorial Optimization 10
2.2.1 Algorithms and Problems 10
2.2.2 Complexity 11
2.2.3 Classes P and NP 12
2.2.4 NP-Completeness 13
2.2.5 Problems of P and NP 14
2.3 Simulated Annealing 15

CHAPTER 3: PROBLEM FORMULATION **20**
3.1 Introduction 20
3.2 Network Model 21
3.2.1 Feeder Segment Model 22
3.2.2 Switch Model 23
3.2.3 Load Bus Model 23
3.2.4 Fixed Tap Setting Transformer 24
3.2.5 Nonlinear Ladder Network 25
3.3 Objective Function 26
3.4 Numerical Technique 28

3.4.1 Network Data 28
3.4.2 Network Simulation 29
3.4.3 Load Flow 36

CHAPTER 4: TESTINGS 38
4.1 Motivation 38
4.2 Applications 38
4.3 Description of Network 39
4.4 Applications Study 40
4.4.1 Operating Study 40
4.4.2 Study with Switch Addition 41
4.4.3 Contingency Studies 41
4.5 Statistical Physics Parameters 44
4.5.1 Switch Addition versus Switching 53
4.6 Annealing Schedule 57
4.6.1 Initial Value of T 57
4.6.2 Exit Criteria 60
4.6.3 Number of Transitions 61
4.6.4 Decrement Rule for T 62
4.7 Objective Function 64
4.8 Summary 66

CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS 68
5.1 Conclusions 68
5.2 Recommendations 69

APPENDIX A: Load Flow Algorithm 70

APPENDIX B: Figures and Tables 74

APPENDIX C: Program Listing 90

REFERENCES 99

List of Figures

Figure		page
2.1	Objective function of a multi-modal system	16
3.1	Example of an open-loop distribution system	21
3.2	Feeder segment model	23
3.3	Load model	24
3.4	Transformer model. (a) Equivalent circuit; (b) Equivalent π-circuit with parameters expressed in terms of admittance and off-nominal turn ratios	24
3.5	Example of a distribution system network (a) using single line diagram representation (b) per phase equivalent circuit	25
3.6	Input data files	28
3.7	Flow chart of the simulation program	30
3.8	Parameters in bus and line records	30
3.9	Linked list connection (a) across a closed switch (b) across an open switch	31
3.10	Annealing parameters menu	33
3.11	Flow chart for the optimization routine	34
4.1	Simplified one-line diagram of network 1 after optimization	43
4.2	Single-line diagram of feeder Fr650 in network 2	45
4.3	Location of new sectionalizing switches in feeder Fr650	45
4.4	Fluctuations of $F_{\min}$ at various temperatures	47
4.5	Variation of $\overline{F}_{acc}$ with T	48
4.6	Normalized $\overline{F}_{acc}$ in a 100-city TSP[18]	48
4.7	Acceptance ratio.(a) in this study, (b) of Metropolis	50
4.8	Variation of specific heat with T	50

Figure		page
4.9	Effect of random number on $F_{\min}$	52
4.10	Comparison of $\overline{F}_{acc}$ in operating and switch addition studies	54
4.11	Comparison of α in operating and switch addition studies	55
4.12	Comparison of $F_{\min}$ in operating and switch addition studies	56
A1	Sequence of branch visited. (a) by layers using branch numbering, (b) by preorder traversal.	71
B1	Single-line diagram of network 1	82
B2	Single-line diagram of network 2	83
B3	Schematics of banks and feeders used in network 1 and network 2	84
B4	Effect of temperature reduction factor on $F_{\min}$	85
B5	Effect of temperature reduction factor on α	86
B6	Effect of temperature reduction factor on $\overline{F}_{acc}$	87
B7	Effect of n on $F_{\min}$	88
B8	Effect of n on $\overline{F}_{acc}$	89

List of Tables

Table		page
2.1	Comparisons between exponential and polynomial time complexity	12
3.1	Modelling of a distribution system using tree structure	21
4.1	Description of the tested networks	40
4.2	Comparison of the three contingency studies	44
4.3	Effect of random number on $F_{\min}$	51
4.4	Parameters in switch addition and operating studies	53
4.5	Effect of T_o in the optimization	59
4.6	Effect of number of accepted moves and moves allowed on $F_{\min}$	62
4.7	Results of varying β from 0.80 to 0.35	63
4.8	Effect of weighing factors on $F_{\min}$	64
4.9	Effect of n on the objective function	65
B1	Initial subsystem information of network 1	75
B2	Initial subsystem information of network 2	75
B3	Optimal subsystem information of network 2 in switch addition study	76
B4	Optimal branch status of network 2 in switch addition study	76
B5	Initial subsystem information of network 1 in contingency study	77
B6	Optimal subsystem information of network 1 in contingency study solved by the switching of the existing switches	77
B7	Optimal branch status of network 1 in contingency study	78
B8	Initial subsystem information of network 2 in contingency study	79
B9	Optimal subsystem information of network 2 in contingency study solved by the switching of the existing switches	79

Table		page
B10	Optimal subsystem information of network 2 in contingency study solved by switch addition	80
B11	Optimal branch status of network 2 in contingency study solved by switch addition	81

CHAPTER 1

Introduction

Since the development of power systems in the late nineteenth century, the electric power industry has constantly upgraded and improved the methods of supplying energy to the customers. These improvements in areas such as efficiency of generation, the reliability of system operation and increased transmission capabilities have provided us with a reliable and economical source of energy.

Although the generation system is considered the most important part of the electric power system, the distribution system makes up to about half of the total power system investment[1]. In a distribution network, the apparatus components are small, but there are a large number of them. A typical distribution network might contain hundreds of feeder segments, switches and distribution transformers. Expansion planning of such a large distribution system becomes very difficult. One method of distribution system expansion is to build new substations and new feeders to accommodate the load growth in the network. The planning includes the choosing the sites to locate substations; determining the optimum size of the transformer; selecting the optimum capacity and routing of the conductor and determining the best installation date of the equipment. The design also requires the consideration of losses and voltage drop in the network, and the selection of the plan with the minimum expansion and maintenance cost. Traditionally, expansion planning of a distribution system is based on future load forecasts, an extensive amount of network analyses and the past experience of the planning engineers.

Fortunately, the emergence of digital computer technology in the past forty years have provided a fast and accurate analytic tool for power system study. Computer applications in planning, engineering analysis, operations and maintenance have

enhanced the human capability to monitor, diagnose and control the system.

In the area of distribution network planning, extensive work has been done to optimize the planning of new substations and primary feeders from the existing network by minimizing cost[2-15]. However, the size and complexity of the problem of simultaneously designing all aspects of a distribution network exceeds the limitations of these methods. Most techniques attempt to solve only one particular aspect of the plan, such as substation locations and capacities[2-3] or feeder system switching and load transfers[4]. In the early seventies, Masud[2] presented an approach to optimize the capacity of proposed substations by using branch and bound optimization method. Meanwhile, Crawford and Holt[3] developed an approach based on the analysis of loads and feeders on a grid. In his work, the location of the distribution substation was optimized by varying the assignments of loads to the substations. A more detailed representation of distribution feeders and loads was presented by Wall et al.[4]. Load transfer between substations by the switching of the feeder segments was implemented in his approach. In the early eighties, Thompson and Wall[5], Fawzi et al.[6], Sun et al.[7] and Gonen and Foote[8] combined the substation siting and capacity problem with the feeder switching problem in their analyses. Later, El-Kady[9] improved on Fawzi's approach[6] by implementing a time-dependent fixed and variable cost in the problem. Although most models[2-8] considered the planning period as a single stage, some models[7-9] also extended the planning period over a number of years. A multistage planning model which solved the substation siting and sizing problem and feeder switching problem was developed by Gonen and Ramirez-Rasado[10]. Youssef et al.[11] also extended the previous work by taking inflation and interest rate into account in the time-phased calculation of substation and feeder costs. An approach to optimize the distribution network by reconductoring feeder segments on a fixed network was developed by Ponnavaikko and Rao[12]. In more recent work, Willis et al.[13] developed a distribution planning

model with load forecast uncertainty by adding a probability factor in the future load calculation. Willis et al.[14] also introduced an approach of substation siting and sizing based on diversity maximization by using a 24-hour load profile in his analysis. Meanwhile, Tram and Wall[15] improved on Ponnavaikko's model[12] by adding regulators to resolve voltage problem in his feeder reconductoring problem.

In normal operations on a distribution system, the main goal is to provide a reliable and continuous supply of power to the customers. Abnormal conditions such as feeder and transformer overload, and voltage violations, may develop on a daily basis due to scheduled or random events. Interruption of power can be caused either by natural phenomenon or by mechanical failure of the equipment. In the event of a fault or other contingencies, equipment such as fuses or fuse switches is interrupted by abnormal voltages or currents. The blowing open of these sectionalizing equipments cuts off the power supply to the loads away from the substation. Emergency ties and other normally open disconnecting switches are closed automatically or manually to modify the circuit configuration and to restore power supply to the out-of-service area. In an automated distribution system, the configuration alternation is based on real-time analysis of the contingency. If the system is not automated, system operators have to use their own judgement to rearrange the network configuration. Circuit switching is necessary to redistribute the loads of the interrupted area to another circuit and prevent further power outages.

In some cases, the problem of overloading of banks and feeder segments can be identified by load flow studies of the circuit. Information describing the bus voltages, line flows can be calculated to determine the system performance of the present network. Abnormal conditions such as low voltage, feeder segments and banks overload of the circuit can be diagnosed. Corrective actions such as circuit switching of the network configuration may be used to protect the circuit equipments and to prevent the occurrence of faults. Circuit switching of the network involves the transfer

of loads between phases or between substations to relieve overloads or potential overload and to improve the voltage condition. The application of circuit switching is useful not only in the prevention of outages or the re-energizing of out-of-service area. It is also used to transfer loads for scheduled interruption and for the change of load pattern during summer and winter seasons.

The use of circuit switching of remote-controllable switches to minimize the total loss in the distribution system was introduced by Aoki et al.[16]. In his approach, the minimization of the losses is carried out subject to voltage drop, station capacity and line capacity. The voltage drop along the feeder is estimated by the use of a piecewise convex quadratic function. The calculation of this quadratic function is based on a constant current model along the feeder. The minimization process involves the determination of the optimal location of each individual open switch on a continuous feeder segment model and rounding off of the continuous solution to the nearest actual automatic remote-controllable switches. The optimal solution of this approach is not fully justified because of the individual solution of the location of the open switches and the estimation of the rounding off.

The use of circuit switching to balance the loads of the transformers and feeders was discussed by Aoki et al.[17]. In his approach, an iterative process is used to shift loads between two transformers with the highest and lowest load indices. After the loads between all transformers have been balanced, the loads in the feeders connecting to each transformer are balanced by shifting loads within feeders of the same transformer. The feeder segments capacity and voltage drop along the distribution system network are not considered in his approach. The individual optimization of load balancing between transformers and feeders may not lead to a global optimum. The problem should be studied at a global basis.

1.1. Statement of the Problem

The use of circuit switching to optimize the performance of an existing distribution system is investigated in this thesis. The switch positions are chosen to attempt to minimize the overloading of transformers and feeder segments while trying to maintain the bus voltage within prescribed limits.

Due to the discrete nature of the switch position, the optimization problem falls into a class of optimization problem called combinatorial optimization problem[18]. The variables associated with the switching of the feeder segments on and off can be modelled as "0" and "1" integers. Specifically, this problem of "0 *or* 1 *integer programming*" have been proven to be *NP–hard*[19]. The class of *NP–hard* problems is considered to be the hardest problem in combinatorial optimization. No efficient optimization method have yet been found to solve this type of problem in polynomial time. The execution time of the polynomial time algorithm increases at a rate bounded by a polynomial function that depends on the number of variables.

1.2. Proposed Method

Recently, Kirkpatrick et al.[20] and Cerny[21] had succeeded in finding a near-optimal solution of a *NP–hard* problem in polynomial time. This method known as "simulated annealing" is based on the analogy of statistical physics of molecules during the process of cooling.

This method will be applied to the switching problem in power distribution network. A digital simulation of the power distribution system in steady-state has been implemented for the analysis. A phase-independent network representation and a three-phase load flow method are used in the problem.

1.3. Motivation

The purposes of this thesis are to apply the method of simulated annealing to optimize the switch positions in a primary distribution network and to examine its behavior that is analogous with statistical physics.

1.4. Outline

An overview of distribution system analysis on planning and operating studies has been introduced briefly in this chapter. Chapter 2 reviews some of the optimization methods used in distribution system planning. Background information on combinatorial optimization and simulated annealing is also discussed. The modelling of the distribution system and the formulation of the algorithm are discussed in Chapter 3. Testing of the algorithm's performance and its annealing schedule are investigated in Chapter 4. Finally, conclusions and recommendations of this study are presented. A brief description of the load flow algorithm is discussed in Appendix A. Appendix B shows some of the figures and tables in the study. A listing of some of the subroutines used in the optimization algorithm is given in Appendix C.

CHAPTER 2

Optimization Techniques

2.1. Introduction

Optimization involves the maximizing or minimizing an objective function f that depends on one or more independent variables.

$$f(\mathbf{x}) = f(x_1, x_2, \ldots\ldots, x_n) \tag{2.1}$$

This class of problems where the variables can take on any value is referred as unconstrained optimization problem. In most applications, inequality and equality constraints are presented in the problem and those problems are classified as constrained optimization problems. The inequality constraint can be stated in the form:

$$g(\mathbf{x}) \geq 0 \tag{2.2}$$

The equality constraint is usually in the form:

$$g(\mathbf{x}) = 0 \tag{2.3}$$

Direct and indirect optimization methods are often used in solving unconstrained and constrained optimization problems. The direct search methods are those whose exploration strategy is based on the comparison of the objective function only. These methods are useful when the function is continuous and non-differentiable. In unconstrained optimization, direct methods such as the direct search, Rosenbrock's method, and the downhill simplex method of Nelder and Mead are often used. In constrained optimization, a direct method such as sequential unconstrained minimization technique(SUMT) is commonly used. The SUMT method by Fiacco and McCormick[22] is based on transforming a constrained problem into a sequence of

unconstrained problems. The indirect methods, on the other hand, are useful if the Jacobian and Hessian matrices can be easily calculated. In unconstrainted optimization, indirect methods such as steepest descent methods and conjugate gradient method require the calculation of the first and second order derivatives of the objective function. In constrained optimization, Lagrange and Kuhn-Tucker multipliers are used to solve problems with equality and inequality constraints. However, the most commonly used method developed in the area of linear constrainted optimization, where the objective function and constraints are all linear functions of the independent variables is linear programming.

The linear programming problem may be stated in the standard form:

$$\text{Minimize } y = \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to } \sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i=1,2...,m$$

$$x_j \geq 0 \qquad j=1,2,....,n \tag{2.4}$$

$$b_i \geq 0 \qquad i=1,2,....,m$$

Any solution for which the constraints are satisfied is located within the feasible region. Since the feasible region is convex, the extremum found in the solution is guaranteed to be the global optimum. The most well-known method for solving linear programs is the simplex algorithm by G. Dantzig. The algorithm requires a pivotal transformation of the objective function and constraint equations to check for the range of variables. The process is continued until the lower and upper bounds converge to a value.

An extension of the linear programming problem where the variables take on only integer values are called integer programming problems. The equation can be written in the form of Equation (2.4) with the addition of the constraint:

$$x_j = 0 \ mod(1) \qquad j \varepsilon J \tag{2.5}$$

where it implies that x_j is an integer. If only some of the variables are integer, the Constraint (2.5) is replaced by the constraint:

$$x_j = 0 \ mod(1) \qquad j \varepsilon K \tag{2.6}$$

where $K \subset J$, and the problem becomes a mixed integer programming problem.

One of the applicable methods used in solving mixed integer programming problem is the branch and bound method. In this method, the simplex method is used to solve for all the variables. The real value of the integer variables x_j are then rounded normally one at a time to the nearest integer value by the addition of an extra constraint in the next stage of the calculation. The underlying concept of branch and bound strategies is the observation that at any values of x_j, one can always state:

$$[x_j] + 1 \geq x_j \geq [x_j] \tag{2.7}$$

where $[x_j]$ is the truncated value of x_j. The branch and bound method is like a tree consisting of vertices and arcs. Each vertex consists of solving a linear programming problem and each arc consists of imposing new integer constraints to the problem. The branching of the tree terminates when the solution at a vertex is worse than the best feasible solution. This process of adding new branches creates a tree of enormous configuration. Explicit enumeration of every branch is not practical since the number of solutions increases combinatorially with the number of integer variables. Implicit enumeration of the tree structure is often used to determine the order of variable selecting and the elimination of branching.

In distribution feeder planning, the branch and bound technique has been commonly used in solving mixed integer problems in distribution system planning[5-7]. Sun et al.[6] used special heuristic or logic to determine the node and branch selection process in his implicit enumeration of branch and bound. Most of the planning models[2-15] in the past have used some type of simplex algorithm in solving

transshipment transportation and branch and bound problems. In real applications, most practical problems are actually nonlinear. Since linear programming and mixed integer programming packages are commercially available and very reliable, the approximate solution by solving the linear model may be the best alternative in solving large nonlinear problem.

2.2. Combinatorial Optimization

The feeder switching in the distribution system network belongs to a special type of discrete problem called combinatorial optimization. Combinatorial analysis is the mathematical study of the arrangement, grouping, ordering, or selection of discrete objects, usually finite in number[23]. Combinatorial optimization involves the development of efficient algorithms for finding the maximum or minimum of an objective function with variables that are combinatorial in nature. Many combinatorial optimization problems can be found in areas such as computer science, operations research and engineering.

2.2.1. Algorithms and Problems

An algorithm is a step-by-step procedure for solving a problem. A problem can be considered to be composed of a domain containing the instances of the problem, together with a question that can be asked about any one of the instances[19]. Consider an instance of the traveling salesman problem(TSP) composes of N cities, and a N x N distance matrix C, and the question is "What is the minimum distance in the tour?". In a brute force approach, an algorithm can be used to examine all permutations of the tour and then pick out the best as the optimal one. A typical N-city traveling salesman problem has $(N-1)!/2$ possible tours, and a solving algorithm will have running time $O(n!)$. Since a factorial function grows faster than an exponential function, the solution requires $O(n^k)$ operations, and this number of

operations increases faster than any power of N.

The running time and execution steps can be easily determined in the above well-behaved algorithm. However, the number of operations in some algorithms such as the simplex algorithm used in linear programming varies widely in different problems. Worst-case analysis is usually used to determine the complexity of the algorithm. In large problems, the information resulting from the worst-case analysis is useless since the worst case often never arises. On the other hand, average-case analysis requiring statistical methods to find the complexity of the algorithm is not often used because it is impossible to find a mathematical model to represent all instances of the problems. Finally, worst-case analysis at least provides a 'guarantee' to the user of an algorithm, average-case analysis often provides a prediction that one can have confidence only if one has to solve a large number of instance[19]. Studies shows worst-case analysis of the simplex algorithm for linear programming gives an exponential complexity, but its average behavior exhibits a polynomial complexity[24].

2.2.2. Complexity

So far, the execution steps as a function of the size of the variable n have been used to evaluate the performance of the algorithm, but it is not possible to tell an algorithm is a good one or not. An algorithm is considered to be good if its worst-case complexity is bounded by a polynomial function of N. For example, given three algorithms with complexities $O(2^n)$, $O(n!)$ and $O(n^3)$, the one with complexity $O(n^3)$ is a polynomial time algorithm and is good. Any algorithms which are not bounded by any polynomial functions are considered to be exponential. Most exponential algorithms are believed to be variations of exhaustive searches and are therefore inherently bad. The difference between polynomial and exponential time complexity functions is shown in Table 2.1. Examination of this table clearly shows that the polynomial-time algorithm is the more efficient of the two.

Table 2.1 Comparisons between exponential and polynomial time complexity

Function	Approximate Values		
n	10	100	1000
nlogn	33	664	9966
n^3	1000	10^6	10^9
$10^5\ n^8$	10^{13}	10^{21}	10^{29}
2^n	1024	1.27×10^{30}	1.05×10^{301}
$n^{\log(n)}$	2099	1.93×10^{13}	7.89×10^{29}
n!	3,628,800	9×10^{157}	4×10^{2567}

The computation time of exponential algorithm grows so rapidly that the current fastest computer can only solve a small instance of the problem in reasonable time. Suppose an algorithm with a complexity of $O(n!)$ is available and a computer that can execute a step in 1 nano-second, a problem for $N = 14$ can be solved in 87 seconds. If the size is increased by 50 percent, the same problem for $N = 21$ will take 1.62 centuries to solve. A problem is not considered to be solved unless the 'solver' lives long enough to see the solution. In most cases, it is desired to identify algorithms that have polynomially complexity.

2.2.3. Classes P and NP

A class P problem is one which is included in the set of all problems solved by deterministic algorithms in polynomial time. Deterministic means whatever the algorithm is doing, there is only one thing that it could do next. In an nondeterministic algorithm, there is more than one possible path that the algorithm can take. The algorithm makes a guess at the solution and then checks it to see if it is correct. The sets of all problems which can be solved by nondeterministic algorithms

in polynomial time is called *NP*. Since deterministic algorithms are a special case of nondeterministic algorithms, it can be concluded that $P \subset NP$. In the past, researchers have failed to find polynomial time algorithms for solving problems such as TSP. If such an algorithm is found to solve these problems, then $P = NP$ will be true and all hard problems can be solved in polynomial time. From present knowledge on solving exponential algorithms, it is more appropriate to expect $P \neq NP$ even though it cannot be proved.

2.2.4. NP-Completeness

A problem has not been 'well-solved' until a polynomial time algorithm has been found to solve it. A problem is intractable if it is so hard that no known polynomial time algorithms can solve it. Many of these decision problems that ask for only 'yes-no' answer have proved to be *NP*-complete. Garey and Johnson[25] have presented a list of 300 *NP*-complete problems. A decision problem A is said to be *NP*-complete if (a) $A \in NP$ and (b) every problem in *NP* is polynomially transformable to A [2]. A polynomial transformation from a language $L_1 \subseteq \Sigma_1$ to language $L_2 \subseteq \Sigma_2$ is a function $f: \Sigma_1 \rightarrow \Sigma_2$ that satisfies the following conditions:

[1] f is computable by a polynomial time algorithm

[2] for all $x \in \Sigma_1$, $x \in L_1$ if and only if $f(x) \in L_2$.

There are many problems that do not belong to the class of *NP*-complete problems due to technical definition. The TSP (optimization version) that finds the minimum tour cannot be *NP*-complete because it is not a decision problem, even though the TSP decision problem is *NP*-complete. On the other hand, if a decision problem only satisfies one of the criteria, it cannot be classified to be *NP*-complete. All these problems are referred only to be *NP*-hard, and are at least as hard as the corresponding *NP*-complete problems.

2.2.5. Problems of P and NP

Combinatorial optimization problems are divided into classes of P and NP. Although many problems look alike, there is a fine line between problems in P and NP. One class has been solved by using efficient algorithms in polynomial time, while the other class of problems is so hard that it is not likely to be solved by any polynomial time algorithms. Let look at four combinatorial problems:

1. Shortest Path Problem:

   Given a graph G that contains a set of vertices V and edges E and length of each edge and a positive integer B, is there a simple path between specified vertices having total length B or less?

2. Edge Cover Problem:

   Given graph $G=(V,E)$, what is the shortest tour (a closed path) that passes through each edge in G at least once?

3. Longest Path Problem:

   Given graph $G=(V,E)$ and length of each edge and a positive integer B, is there a simple path between specified vertices having total length B or more?

4. Vertex Cover Problem:

   Given graph $G=(V,E)$, what is the shortest tour (a closed path) that passes through each vertex in G at least once?

Of all the previous problems listed above, the first two problems can be solved by linear programming and other polynomial time algorithms. The last two problems belong to the class of NP. There are many similarities between the shortest and longest path problems, and between the edge and vertex cover problems. However, the longest path and vertex cover problems are hard to solve.

2.3. Simulated Annealing

In dealing with *NP*-hard problems where an efficient algorithm is unavailable, the approach is to use approximation technique or heuristics to obtain a good local optimum with reasonable computational cost. Heuristics are problem specific and unpredictable. They work most of the time, but sometimes fail to find an optimal or near-optimal solution[26]. The two basic strategies for heuristics are 'divide and conquer' and iterative improvement. An algorithm such as dynamic programming decomposes the problem into small subproblems, solves all the subproblems and then the solutions of the subproblems are used to solve the larger one. There are two major difficulties in implementing this technique. First, the combination of the solution of the subproblems does not guarantee the overall solution due to the disjoint nature of the subproblems. Second, there may be a countably infinite number of small subproblems to solve. In large combinatorial problems such as TSP, dynamic programming requires the storage space equal to $O(n2^n)$ locations and operations equal to $O(n^2 2^n)$[27].

In the iterative improvement algorithm, the objective function for the initial configuration is calculated. The configuration is rearranged by a standard perturbation and a new objective function is obtained. The old configuration is replaced by the new one only if the new objective function is lower. This iterative improvement approach generates changes that monotonically increase or decrease the objective function. Since there are many local optimums in a multi-modal system(Figure 2.1), this algorithm usually terminates in a local optimum. Since the problem does not guarantee an optimum, the algorithm is usually repeated several times from different starting point and the best solution is used as the optimal solution.

Some success of solving the traveling salesman problem by neural network simulation has been reported[28]. A moderate result is observed on a 30-city system due to slow simulation of the differential equations by the digital computer. The

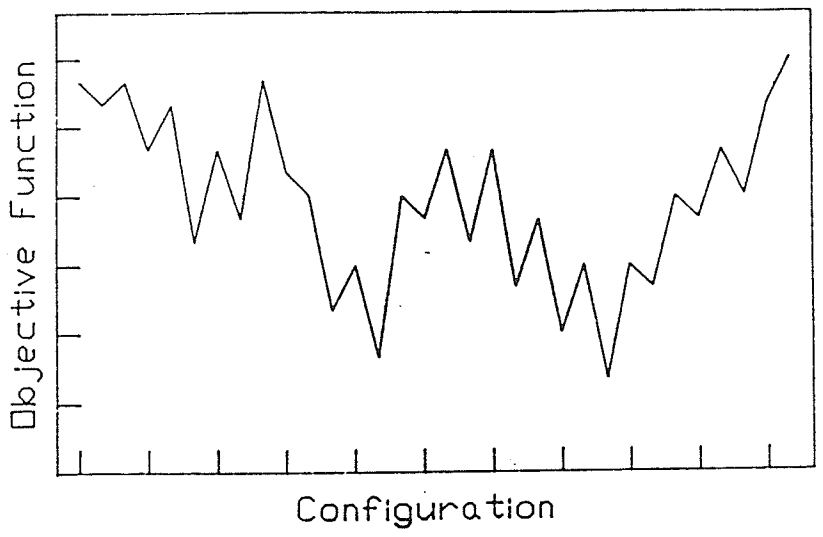


Figure 2.1: Objective function of a multi-modal system

embedding of discrete problems into continuous problems gains the ability to compute a gradient direction in the differentiable systems. However, Baum[29] points out that embedding removes hills from the landscape and leads to poor local optimum.

A probabilistic optimization method proposed independently by Kirkpatrick et al.[20] and Cerny[21] called simulated annealing has been used to find a near-optimal solution of the traveling salesman problem. In a minimization problem, this algorithm allows the acceptance of configurations with higher objective function so that it can escape the local optimum. The solution obtained by simulated annealing is independent of the initial configuration and can be approximated in polynomial time. Although the solution obtained by simulated annealing is better than the iterative improvement method, the quality of the solution is achieved at a higher computational cost.

The method derives its name from the annealing process of metals. In the cooling of molten metal to freezing temperatures, the atoms tend to assume a relative lattice position such that the potential energy of the mutual forces is minimized. The minimization algorithm is based on the Boltzmann probability distribution:

$$P(E) \approx exp(-E/kT) \tag{2.8}$$

The Boltzmann distribution expresses the probability of the system configuration at different energy state E. The Boltzmann constant, k, is used to relate temperature and energy. According to the Boltzmann distribution, there is still a small chance for the system to excite into a high energy state even at very low temperatures. The formation of low energy crystalline lattice structures requires a sufficiently slow cooling process. If the cooling is too rapid, such as in the case of quenching, the solid is not allowed to reach thermal equilibrium and a higher energy metastable amorphous structure is formed. Since there are a very large number of atoms ($10^{23}/cm^3$) and possible arrangements, the final arrangement is likely to be in one of the local optima.

The method was first introduced by Metropolis et al.[30] in the statistical mechanical simulation of atoms at a given temperature. In the Metropolis algorithm, each particle is allowed to move in a random direction in the heat bath. The change in energy of the system, ΔE, which is caused by the move is calculated. If $\Delta E \leq 0$, the move is allowed and the particle is put in its new position. If $\Delta E > 0$ (the Metropolis criterion), the move is allowed with a Boltzmann's probability $P(\Delta E,T) = \exp(-\Delta E/kT)$. A random number ξ between 0 and 1 is generated, if $\xi < \exp(-\Delta E/kT)$, the particle is moved to its new position, otherwise it is returned to its old position. By repeating the basic steps many times, it simulates the thermal motions of atoms in a heat bath.

The Metropolis algorithm was applied to the field of combinatorial optimization by Kirkpatrick et al.[20] and Cerny[21]. In the simulated annealing algorithm, the Metropolis criterion is extended over a sequence of decreasing temperature, and the energy function is replaced by a cost function. Random sampling of the variables is used to transform the configuration. At each temperature T, the process is continued until equilibrium is reached. The temperature is further reduced and the process is repeated until a freezing temperature is reached. The temperature is only a control parameter to control the acceptance of a configuration with higher energy when $\Delta E >$

0. At high temperatures, the process simulates the melting of metals and thus accepts all the configuration changes since $P \approx 1$. As the cooling process begins by gradually reducing the temperature, transitions to high energy configuration are taken with a probability P which depends on both the change of energy ΔE and temperature T. When T approaches zero, the probability function P approaches zero and the optimization process becomes gradient based like those of the iterative improvement method.

The simulated annealing algorithm has been proven to converge asymptotically to the global minimum with probability one by using the theory of Markov chains[18]. Asymptotically means it requires the algorithm to perform an infinite number of iterations every time the temperature is updated. In any implementation, the algorithm can only execute a finite number of steps and does not guarantee to converge to the global optimum.

Since simulated annealing is based on the analogy between statistical mechanics and combinatorial optimization, some of the statistical parameters may be used to improve the optimization process. The specific heat $C(T)$ is defined as the rate of change of the average energy with respect to the temperature T. A large value in C signals a phase transition such as the start of freezing ,and therefore slow cooling is recommended.

The annealing schedule $[T_0,n_0]$, $[T_1,n_1]$, ..., $[T_m,n_m]$ gives the sequence of decreasing temperature and the number of accepted moves associated with each temperature. Various annealing or cooling schedules have been studied to yield better results with less computational time[18,31-32]. A multiprocessor system[33-34] and a variable selection heuristic[35] have also been used to improve the computational speed of the simulated annealing algorithm.

Simulated annealing was originally applied in the optimization of circuit design[20,33-39], and in the solving of the traveling salesman problem[20,21,24].

More importantly, the technique has found to be very useful in other fields such as the restoration of images[40], construction of evolutionary trees[41] and pilot plant selection for secondary voltage control in power systems[42]. The formulation of simulated annealing to the switching of primary distribution system is discussed in the upcoming chapter.

CHAPTER 3

Problem Formulation

3.1. Introduction

In order to apply digital simulation to a distribution network, the components in a distribution system network must first be modelled mathematically. Algorithms and numerical techniques are then used to solve for the parameters in the simulated network. The performance of the network is evaluated by using a value called the objective function. The optimization process involves the finding of a network arrangement that minimizes this objective function.

In this chapter, the models of the components used in the simulation are first described. Next, the formulation of the objective function is reviewed. Finally, the methodology of the optimization technique and numerical technique used in the problem is discussed.

The distribution system used in the study is called an "open-loop distribution system". An example of an open-loop distribution system is shown in Figure 3.1. This type of system is operated as a radial system in which each load point is supplied by exactly one substation transformer. Normally open sectionalizing switches and open ties are used to divide the loop distribution system into many radial subsystems. In Figure 3.1, open ties S2 to S7 are used to separate the connection between different feeders and sectionalizing switch S1 is used to open up a closed loop in one of the feeders. Since an open-loop distribution system always operates as a radial network, it can also be referred as a radial system.

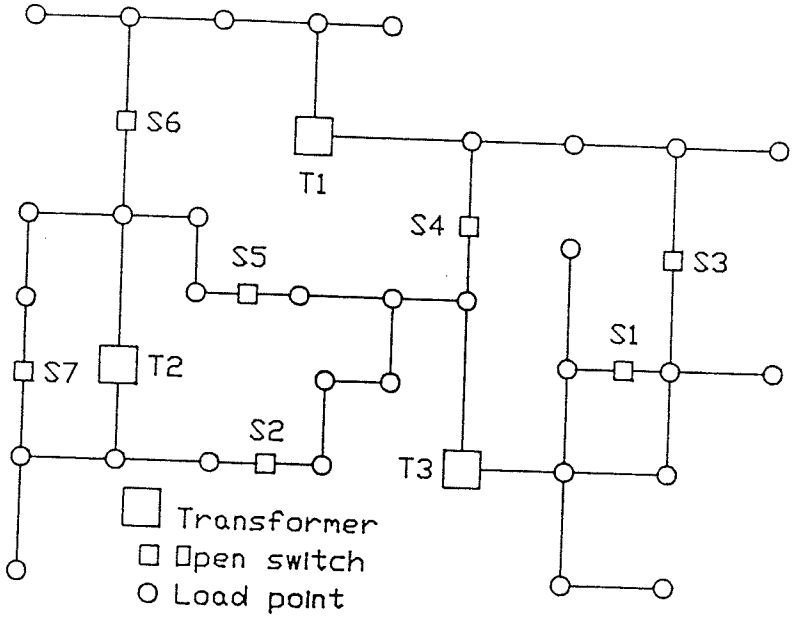


Figure 3.1: Example of an open-loop distribution system

3.2. Network Model

A distribution network can be represented by a graph consisting of nodes and branches. More specifically, a radial distribution system can be modelled as a tree. A tree is a graph with no cycles. Table 3.1 represents the correspondence between a radial distribution system and a tree.

Table 3.1: Modelling of a distribution system using tree structure

Distribution system	Graph
bus	node
feeder segment	branch
closed switch	branch
end node	leaf

In this model, the buses are treated as nodes and the wires connecting them are treated as branches. The root of the tree begins at the low voltage side of the transformer and the first level of branching corresponds to the main feeders radiating out of the substations. A node may represent a source (high-voltage side of the transformer), a root (low-voltage side of the transformer), a branching point with no load or a load point. A branch may represent a line segment, a switch or a transformer. The leaves represent the ends of the feeders or nodes across an open switch. Although the open switch is not depicted as a branch in a tree, it is considered to be an invisible branch.

Since the purpose of this distribution system study is to examine the steady-state characteristic of the network, a fairly accurate model of the distribution network must be developed for the load flow studies. The components which constitute a distribution network are tap-changing transformers, loads, feeders and switches.

3.2.1. Feeder Segment Model

A distribution feeder which corresponds to a branch emanating from the root node, consists of many segments with various ampacities (current carrying capacities) and construction types. These feeders are comprised of a series of overhead lines and underground cables. Since different segments have different series impedances, shunt admittances and ampacities, a detailed feeder model of the distribution network is used in the problem formulation.

A nominal π-circuit shown in Figure 3.2 is used to represent the line model. Each phase is represented individually without any inductive and capacitive mutual couplings between phases and circuits. This model consisting of two capacitive shunt reactances, is commonly used for the modelling of medium-length transmission lines with distances less than 200 miles. Since the present optimization problem may be used in other levels of the power system, an accurate line model has been implemented. Another reason for the use of this model is the per unit length of the

series impedance and shunt admittance of various conductors are readily available in the conductor file.

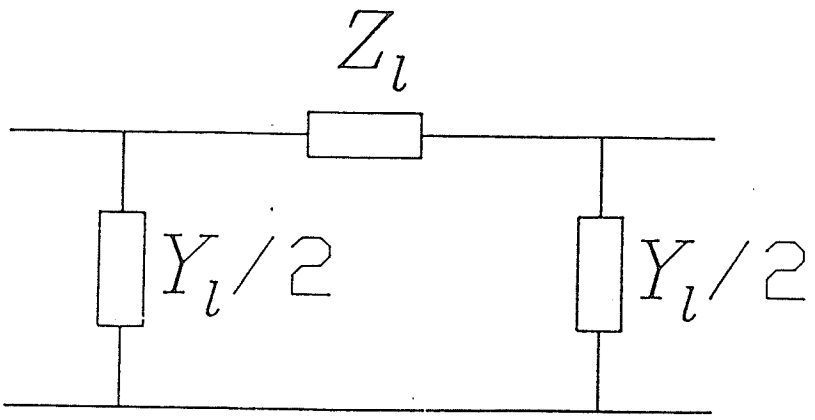


Figure 3.2: Feeder segment model

3.2.2. Switch Model

A switch is modelled as a feeder segment. A switch is different from a feeder segment in that it has zero length, and no impedance or shunt admittance. The switch ampacity is the only parameter contained in the switch model. The opening or closing of a switch connecting from a three-phase line represents the simultaneous opening or closing of the three switches.

3.2.3. Load Bus Model

All generation, slack and load buses are modelled as a terminal across a feeder segment. All loads are defined to be fixed complex power ($S=P+jQ$) by phase. All three-phase loads are wye-connected and all double-phase and single-phase loads are connected line to grounded neutral. The three-phase load model is shown in Figure 3.3.

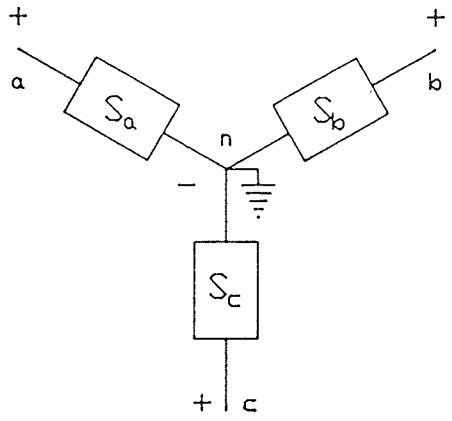


Figure 3.3: Load model

3.2.4. Fixed Tap Setting Transformer

A transformer with the fixed tap setting is represented by its admittance Y_t in series with an ideal autotransformer. The one-line diagram of a transformer with fixed tap setting having a ratio t is shown in Figure 3.4a. The corresponding transformer is modelled as a π-circuit and is depicted in Figure 3.4b.

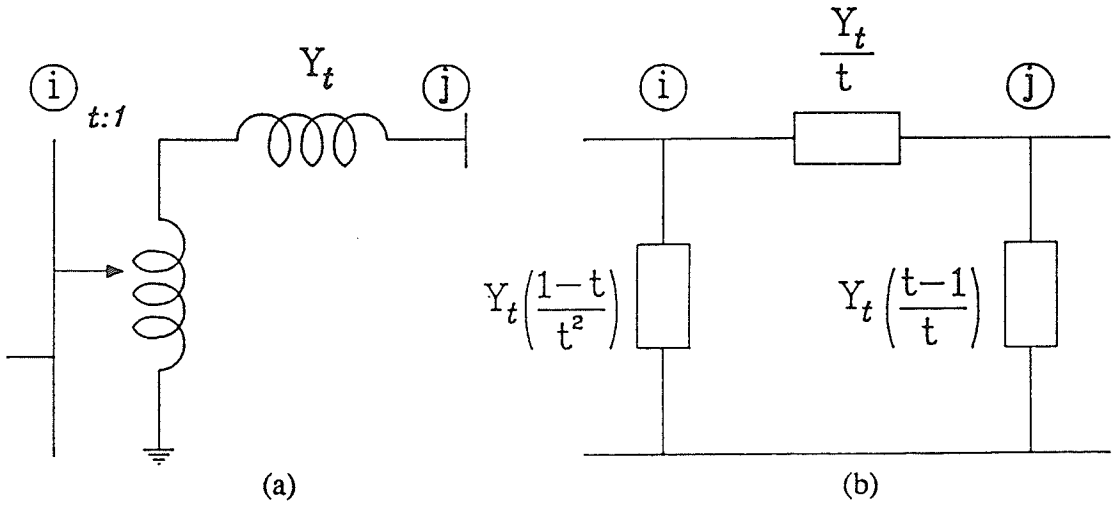


Figure 3.4: Transformer model. (a)Equivalent circuit; (b)Equivalent π-circuit with parameters expressed in terms of admittance and off-nominal turn ratios.

3.2.5. Nonlinear Ladder Network

A distribution system is a nonlinear network because loads are assumed to be constant complex power. A system can be considered as an electrical network consisting of one voltage source and many circuit elements. These elements are connected either in series or shunt with one another. The voltage source is defined at the low voltage side of the transformer bank. The three line to neutral voltages at the source are balanced with *a* phase as the zero reference. Figure 3.5a shows a simplified one-line diagram of a hypothetical distribution network, the corresponding nonlinear ladder network model is shown in Figure 3.5b.

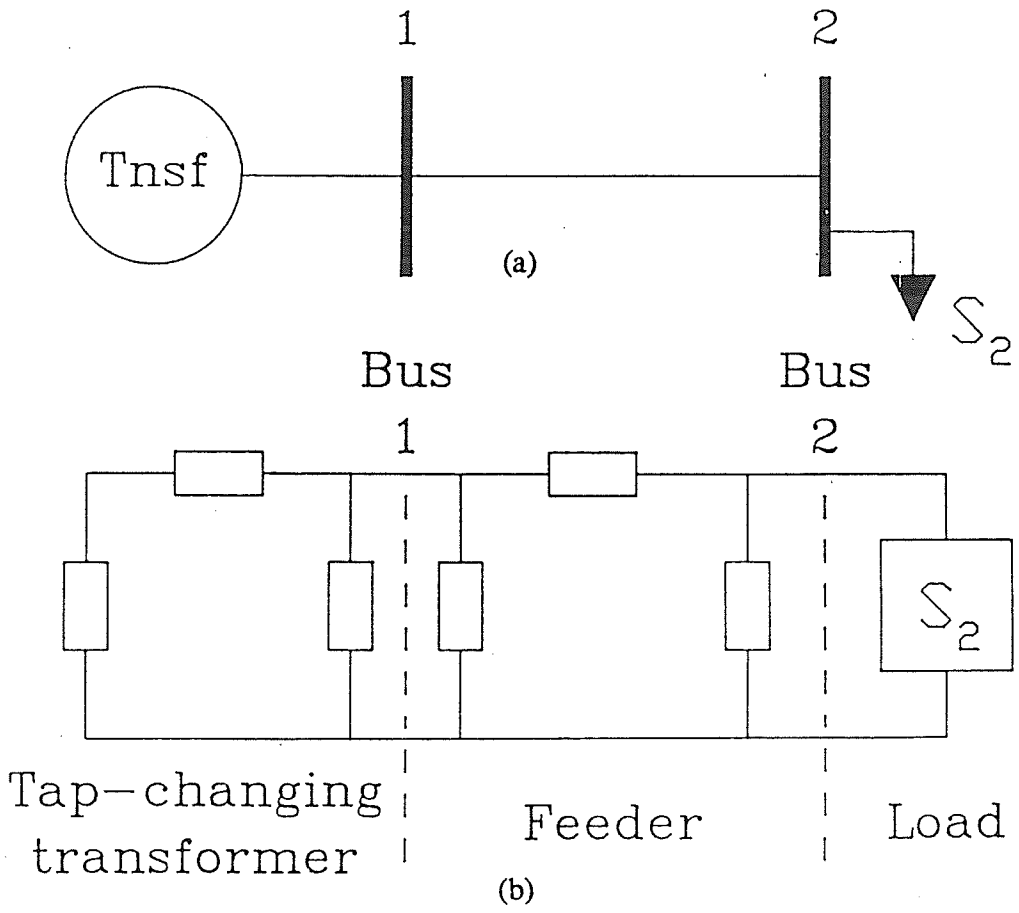


Figure 3.5: Example of a distribution system network (a) using single line diagram representation (b) per phase equivalent circuit

3.3. Objective Function

In an operating study on a distribution network, it is more important in alleviating unacceptable conditions such as equipment overloading and voltage violations, than in minimizing cost. The simulated annealing algorithm is applicable to a large class of nonlinear objective functions. The constraints are explicitly handled and modelled as penalty terms in the objective function. The objective function consists of the sum of three terms. These three terms are penalties for the transformer bank and the feeder segment overload, and voltage deviation of the system. Since the values of these quantities may vary, normalized penalty functions are used in the three penalty terms. Firstly, the overloading of each transformer bank is to be minimized. A transformer bank overload penalty function F_t is formulated as follows:

$$\mathrm{F}_t = \max \left\{ \frac{\sum_{p=1}^{3} | I_{kp} | \, | V_k |}{C_k} \right\} \quad \text{for } k=1,..,m \tag{3.1}$$

where I_{kp} is the current at phase p flowing in transformer k, m is the number of transformers, V_k is the low side voltage of transformer k, and C_k is the capacity of transformer k. Similarly a penalty function F_s is used to penalize the most overloaded feeder segment in the system. The penalty function used is as follows:

$$\mathrm{F}_s = \max \frac{| I_{ip} |}{A_{ip}} \quad \text{for } i=1,..,n \tag{3.2}$$

where I_{ip} is the current at phase p in branch i, n is the number of feeder segments and A_{ip} is the ampacity of branch i. A third penalty function for bus voltage deviation is as follows:

$$\mathrm{F}_v = \max \left| \frac{V_R - | V_{lp} |}{\Delta V} \right| \quad \text{for } l=1,..,s \tag{3.3}$$

where V_R is the desired voltage, V_{lp} is the phase voltage at bus l, s is the number of buses, and ΔV is the maximum allowable voltage deviation. The above functions are

minimax functions, and the optimization will attempt to minimize the most penalized elements in the problem. The functions F_s, F_t and F_v are the normalized terms relating to the overloading of feeders and transformer, and voltage violations. A value of more than one indicates constraint violations. The objective function is expressed as a combination of these three terms and is written as follows:

$$F = w_t F_t^{n_t} + w_s F_s^{n_s} + w_v F_v^{n_v} \tag{3.4}$$

where w_s, w_t and w_v are weights of the three penalties, and the terms n_s, n_t and n_v are the powers of F_s, F_t and F_v respectively. In the problem, the bus voltage constraint is modelled as $F_v^{n_v}$. A value of n_v much greater than one, makes the penalty severe for constraint violation, and adds very little to the function value in the feasible region ($F_v \leq 1$). The users of the program felt that it was more important to penalize constraint violations fairly heavily and minimize the effect of the penalty in the feasible region without the penalty rising too steeply at the border of the feasible region. Therefore an intermediate value of two was chosen for n_v. Similar reasoning was applied in assigning a value of two for n_t and n_s.

The weights w_s, w_t and w_v are used to scale the penalty terms $F_s^{n_s}$, $F_t^{n_t}$, and $F_v^{n_v}$ in the objective function. It is decided to give equal importance to all three terms, so C_s, C_t and C_v have been chosen to be equal to one without any loss of generality.

Although the objective function F may be minimized, one or more of the individual components F_s, F_t and F_v may have a value greater than one indicating a constraint violation. In that case, the weight of the corresponding function may be increased.

3.4. Numerical Technique

3.4.1. Network Data

Before the simulation of the network takes place, the data from the distribution system have to be supplied to create the network. The files used in creating the network are shown in Figure 3.6.

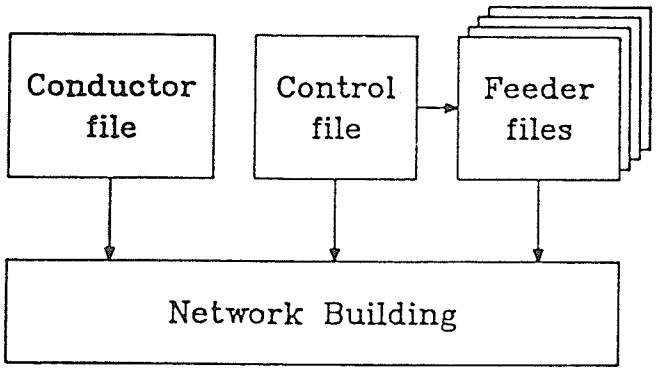


Figure 3.6: Input data files

The conductor file contains the per unit length value of the series impedance and the shunt charging admittance of the various type of conductors. The ampacity of overhead lines is calculated based on the construction and conductor type. On the other hand, conductor ampacities for underground cables are dependent on the duration of peak load current. In such a case, manufacturer ampacity data for those underground cables are used. This conductor file also contains the ampacity of the switches.

The control file contains information on the transformer bank, the main feeder and the locations of the normally open ties. The acceptable voltage range is given in this file. Transformer bank information such as the high and low side phase to neutral voltages, and bank size is described. It also gives main feeder information such as the transformer bank to feeder connection, and path of the feeder files.

Each feeder file contains its own node, branch and load data. The node data consist of a list of unique feeder node names with an associated voltage in kV, an (x,y) coordinate on a map, and other comments. Branch data give the names of the from and to nodes, distance, type of connecting conductor and transformer description. Load data were entered in kW with a default power factor depending on the load type (i.e. residential, commercial, industrial, rural, etc.). The data format is compatible with that used by PSS/U[1].

The feeder data are compiled manually from several sources[2]. Distribution E size maps are used to collect the node point lists and branch connections. The Transformer Load Management System (TLMS) is used to estimate the power load on different points from the meter readings. Station single lines, and transformer data sheets were used to collect the required bank sizes, impedances and tap-settings. Tie-point schematics are used to give the tie-point locations and switch ampacities.

3.4.2. Network Simulation

Although simulation programs differ significantly in size and capacity, the structure of the programs is very similar. This simulation program consists of four major subprograms: input, setup, optimizer and output (see Figure 3.7).

The input subprogram[3] reads in data from the control, conductor and PSS/U feeder files. Parameters associated with the network elements are calculated and stored in a data structure. In a feeder segment, the impedance and admittance are calculated by multiplying the per unit value with the distance. The conductor ampacities are found in the conductor file. At a load bus, the reactive power Q is calculated as follows:

1. PSS/U is a distribution network analysis program developed by Power Technologies Inc., Schenectady, NY.
2. The feeder data is compiled by Paul Wilson of Manitoba Hydro.
3. The input subprogram is written by Paul Wilson.

$$Q = P \tan(\cos^{-1}(power\ factor)) \tag{3.5}$$

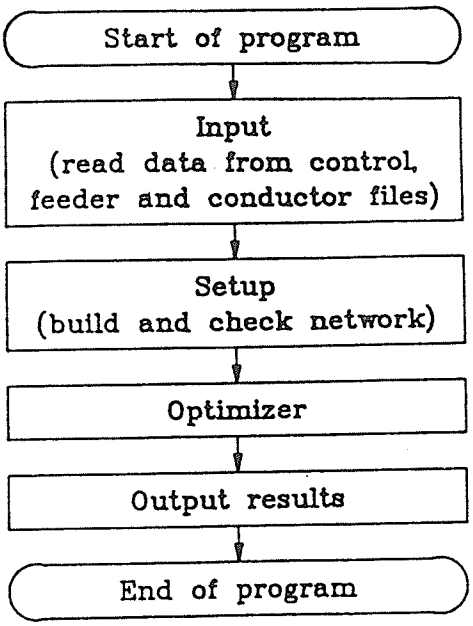


Figure 3.7: Flow chart of the simulation program

The parameters in the bus and feeder segment are stored in a data structure called a record. Figure 3.8 shows the parameters associated with the bus and line records.

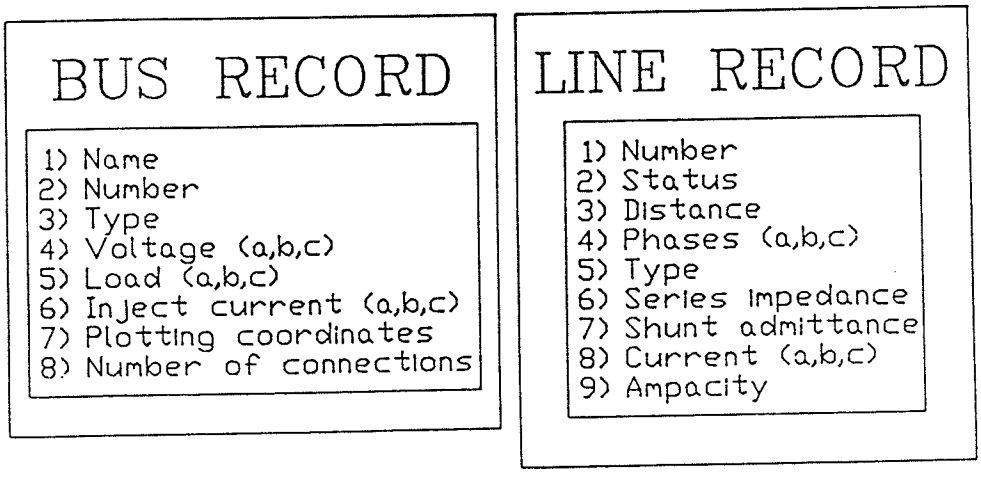


Figure 3.8: Parameters in bus and line records

Parameters of each bus and line are stored locally in each respective record. Depending on the number of buses and lines, these records are dynamically allocated when needed. Some quantities such as bus voltages and line currents are calculated

later in the load flow. The data structure, after the input phase, contains a complete description of the buses and lines in the distribution network.

In the setup subprogram, these bus and feeder records are connected together by pointers to form a linked list. This data structure is used because it allows easy traversal of the network. A line record is always connected between two bus records. The direction of the pointers between these records dictates the power flow in the load flow calculation. The forward and backward pointers are the two types of pointers in a bus or line record. Figures 3.9a and 3.9b illustrate the connection of pointers across a closed and open feeder segment respectively. The forward pointer(s) of the bus record always point in the direction of power flow. Since the power flow terminates at a leaf node such as an open switch, the bus records across an open switch record have both of their forward pointers pointing toward the open switch record (see Figure 3.9b). The backward pointer of the root node record (high voltage bus of the transformer) is pointed to a *NIL* pointer. In the problem, the maximum number of line connections to a bus is limited to sixteen. A maximum number is specified to reduce the amount of memory allocated for the forward pointers.

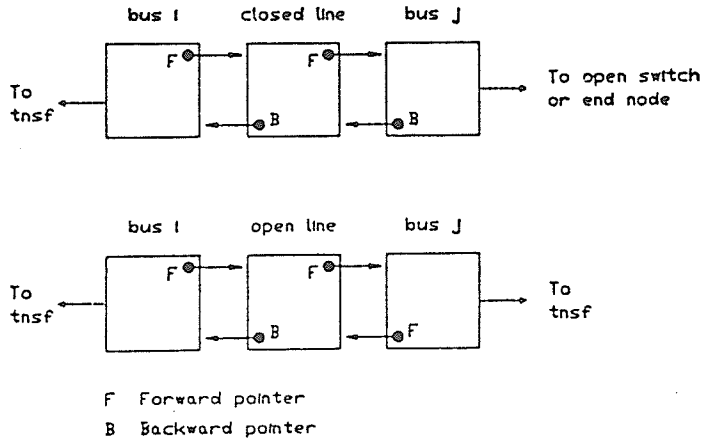


Figure 3.9: Linked list connection (a) across a closed switch, (b) across an open switch

The direction of these pointers must be correct since the tree traversal in the load flow algorithm relies on the structure of this linked list. The next step in the setup

procedure is to detect and correct any connection error in the linked list. In the PSS/U feeder files, the order of the from and to nodes is not always correct. In those files, the two buses across a feeder segment are stated in any order. In the implementation, the backward pointer of a closed line record must point to the "from node" and the forward pointer must point to the "to node". An error checking and correcting subroutine is used to correct specifically for this type of error. Other errors due to incorrect specification of the network connection are reported to the user for manual correction.

Two recursive routines are used to check for errors in the linked list. First, the forward checking subroutine checks the connection from the root node record down the hierarchy to the leaf node records. If the leaf node record is an open switch, a second recursive checking routine is used to check from that open switch record back up to another root node via a second path. During the traversal of the linked list, the directions of the pointers of the bus and line records are checked accordingly. If the program finds any error in the connection, it displays a message describing where the error has occurred and attempts to correct it by the reversal of pointers. The checking and correcting scheme is continued until all feeders are checked. The two paths connecting across any open switch are always checked twice. The purpose of this checking routine is not to correct all connection errors, but to ensure that the connection of the linked list is absolutely correct.

As noted earlier, the optimization program uses the method of simulated annealing to minimize the objective function stated in Equation 3.4. A load flow calculation is performed on the initial configuration, the magnitude of the voltages and currents are then used to calculate the initial value of the objective function. The program then initializes the simulated annealing parameters such as starting temperature, temperature reduction factor, and the maximum number of iterations allowed. A parameter menu is provided to the user if some of the parameters have to

be changed. This annealing parameter menu is shown in Figure 3.10. After the parameters have been initialized, the iterative optimization begins.

```
Enter changes that you want to make:

 1. Initial Acceptance                  2. Temperature reduction factor
 3. Exit Acceptance                     4. Maximum # of iterations
 5. Scale on line overloading           6. Scale on real power loss
 7. Scale on reactive power loss        8. Scale on voltage constraint
 9. Scale on trnsf. overloading        10. Optimize option (brch or sw)
11. Change seed for RANDOM             12. Exit Ratio
13. Quit
==>
```

Figure 3.10: Annealing parameters menu

At each temperature, the algorithm performs a fixed number of iterations in order to reach a thermal equilibrium. The thermal equilibrium is considered to be reached after a number of successful transitions is made. The algorithm used in the branch switching is illustrated in Figure 3.11, and described in the following steps:

1. Randomly select one of the open switches. For example, if there are 20 open switches, a random integer N generated between 1 and 20 is used to select the Nth open switch.

2. For an operating study, this randomly chosen open switch is exchanged with a randomly chosen closed switch in either of the two paths leading from the open switch to the supplying root nodes. In studies where the introduction of new switches is allowed, every segment along the two paths is a candidate for the exchange with the open switch. The selection of the closed switches along the two paths is to keep the system radial.

3. On the system(s) modified in step 2, the bus and line pointers between the two affected switches are modified. A load flow calculation is carried out and the new voltage and current are obtained. A new objective function results from Equations 3.1 to 3.4.

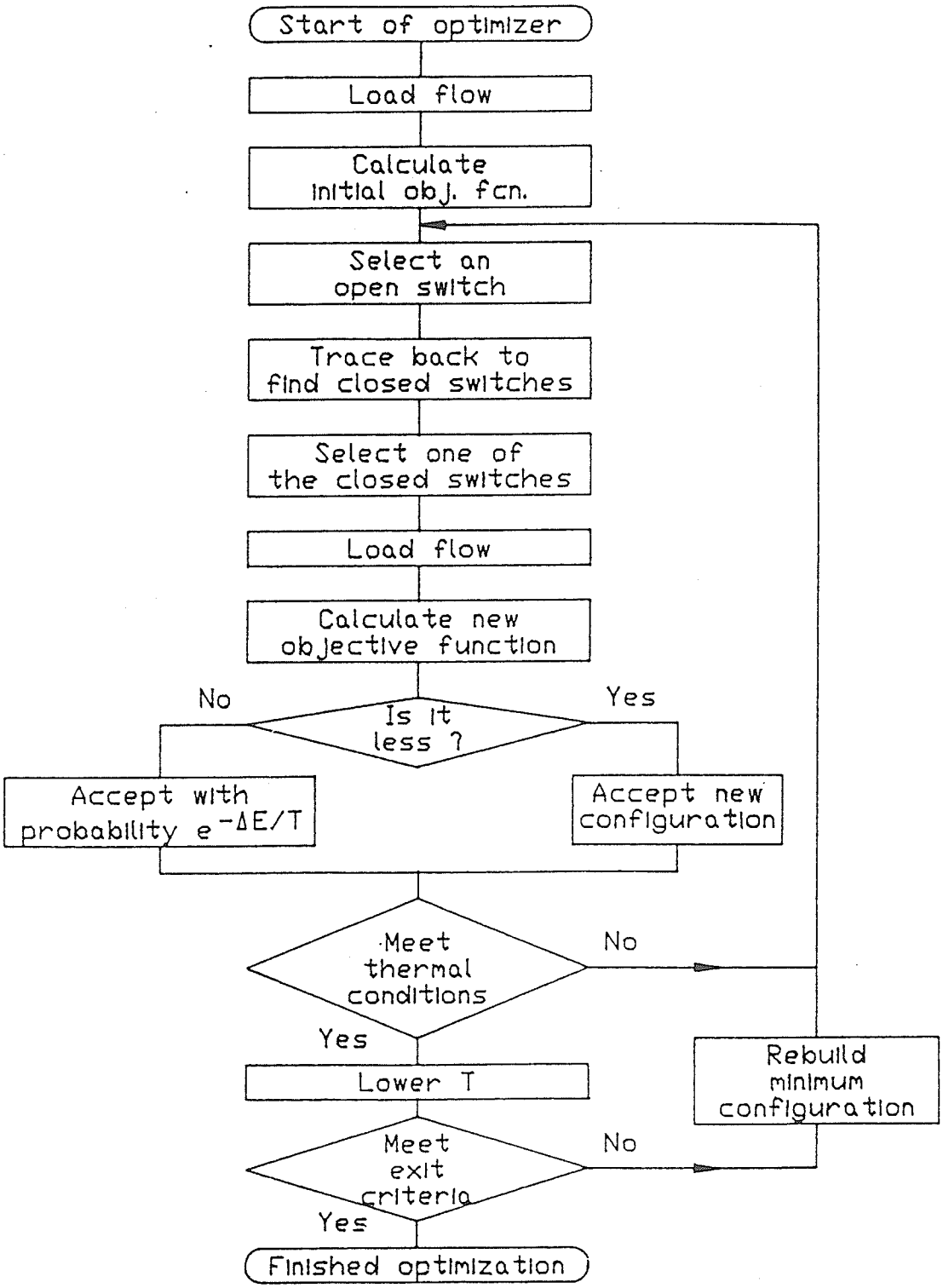


Figure 3.11: Flow chart for the optimization routine

4. If the new objective function is less than or equal to the old objective function, the new configuration is accepted. Otherwise, the configuration is accepted with a probability of $e^{-\Delta F/T}$, where $\Delta F = F_{new} - F_{\min}$, T being the temperature, F_{new} the new function value, and $F_{\min}$ the minimum objective function found so far.

5. If the configuration is rejected, the status of the two chosen switches are reversed. The directions of pointers between the two affected line records are replaced by the initial directions.

The above steps are repeated until thermal equilibrium is reached. A thermal equilibrium is determined when the value of the objective function reaches steady state. The condition is attained after a sufficiently large number of successful moves have been made. In this problem, the number of successful moves is set to three times the number of open switches and is determined empirically. An improper exit from this loop is also used when the total number of steps attempted exceeds a large number. This number is set to ten times the number of open switches. This exit criteria is used when the number of successful moves is small.

Once the program exits this inner loop, the temperature T is reduced by multiplying with a constant reduction factor of 0.65. The program repeats the previous branch switching steps until one of the exit criteria is met. This happens when:

1. The temperature T is less than the exit temperature, or
2. The number of iterations is equal to the maximum allowable limit, or
3. The objective function averaged over the successful moves at that temperature does not differ significantly from the recorded minimum, or
4. The number of accepted moves is equal to zero.

After the optimization is completed, the node and the branch information corresponding to the original network and the optimized network is output graphically

or in the form of tables.

3.4.3. Load Flow

Load flow calculations are important in the evaluation of current performance of a distribution system. Since the load is not equal across the three phases, the influence of load unbalance requires the use of three-phase load flow analysis. The objective of the load flow calculation is to determine the steady state operating characteristics of the power system given a set of busbar loads. The solution is expected to calculate the magnitude and phase angle of voltages at each bus, the real and reactive current flow in each line, and the losses in the system. The four quantities associated with each bus are:

(i) Real power P

(ii) Reactive power Q

(iii) Magnitude of voltage $|V|$

(iv) Phase angle of the voltage ϕ

In the source bus, the $|V|$ and ϕ are specified. The phase angles are set to 0, -120^o, $+120^o$ for the three phases respectively. The three-phase voltages at the generation bus are assumed to be equal and balanced. The values of P and Q are specified at the load bus.

The total current flowing in bus p is described as:

$$I_p = \frac{P_p - jQ_p}{E_p^*} - y_p E_p \tag{3.6}$$

where y_p is the total admittance at the bus, $y_p E_p$ is the current flowing across the shunt element connecting between ground and bus p, P_p and Q_p are the real and reactive power at bus p, and E_p^* is the complex conjugate of voltage E_p. According to Kirchoff's voltage law (KVL), the current flowing from bus i to bus j is described as:

$$I_{ij} = \frac{E_i - E_j}{Z_{ij}} \tag{3.7}$$

where E_i and E_j are the voltage at bus i and j, and Z_{ij} is the series impedance between the two buses. At each bus, Kirchhoff's current law (KCL) states that the sum of all phasor currents equals zero must be satisfied. A mathematical expression of KCL is:

$$\sum_{n=1}^{N} I_n = 0 \tag{3.8}$$

Since the voltage and current are not known in Equations 3.6 and 3.7, iterative numerical techniques have to be used to solve the $6n$ unknowns in a n bus system. An ac load flow method proposed by Shirmohammadi et al.[43] for solving radial system equations is implemented in the optimization algorithm. This load flow is used in to evaluate I_{kp}, I_{ip}, and V_{lp} in Equations 3.1 to 3.3. Appendix A gives more detailed information on this load flow algorithm.

CHAPTER 4

Testing

4.1. Motivation

The purpose of testing is twofold: (i) to determine the overall performance of the optimization program in distribution network studies, and (ii) to compare the statistical parameters of simulated annealing with the predicted ones. Although the tests conducted were not exhaustive enough to determine the optimal annealing parameters, some valuable observations were made. The tests conducted were to determine the performance of this optimization program in operating, contingency, and switch addition studies. Annealing parameters such as average energy, acceptance ratio and overall energy are modelled and examined in these studies. The testing of the parameters in the annealing schedule is important to understand the behavior of the algorithm and is essential to improve the present optimization process.

4.2. Applications

The optimization program can be used for various studies in distribution engineering. The three studies which were performed are as follows:

(1) Operating study

In normal operation, it is desired to protect the distribution system by balancing the load so that the feeders and transformers are evenly loaded. The optimal configuration is achieved by the switching of the existing switches in the distribution system so that the loadings on the feeders and transformers are evenly shared and within their respective capacities.

(2) Contingency study

Scheduled outages can be caused either by the repair of equipment or by changing of the load pattern. Failure outages are caused by a fault in the distribution system. In both outages, the switching of the existing switches is required to re-supply power to the outage area. The new switching profile to reconfigure the system is optimized by the program.

(3) Study with switch additions

New switches can be added onto the existing networks to re-distribute the load. In this case, open switches can be added along any feeder segments in the network. Thus switches can be placed optimally for better future performance of the system.

4.3. Description of Network

Two Manitoba Hydro 7.2kV distribution networks in Winnipeg have been used for the testing of the optimization program. In Table 4.1, network 1 consists of three substations: Springfield, Emerson, and Peguis; and network 2 consists of two substations: Dakota and Frobisher. The simplified single-line diagrams of networks 1 and 2 are shown in Figures B1 and B2 of appendix B respectively. The dotted lines in both diagrams represent the underground cables and the solid lines represent the overhead lines. Figure B3 shows the normal connection between primary feeders and transformers in the substations of the two networks.

Network 1 was tested earlier on a PRIME 9955 computer at Manitoba Hydro before the present annealing schedule was implemented. The purpose of testing network 1 was to show the effectiveness of the algorithm in solving operating and contingency problems. Network 2 was tested on a SUN3 computer at the University of Manitoba using a newer version of the program. Besides using network 2 to demonstrate its performance for the three types of studies, the performance of the algorithm was also observed by changing the annealing schedule and statistical

parameters. Network 2 was primarily used to observe the analogous relationship between combinatorial optimization and statistical physics.

Table 4.1 Description of the tested networks

Components	Network 1	Network 2
Substations	3	2
# of feeders	16	13
Feeder segments	664	652
Buses	628	646
Closed switches	68	23
Open switches	18	10

4.4. Applications Study

4.4.1. Operating Study

Networks 1 and 2 have both been tested under operating studies where the configuration can only be altered by the switching of the existing switches. In both networks, no better solution was found indicating the initial arrangement of each network is probably optimal. Tables B1 and B2 of appendix B show the initial subsystem information for networks 1 and 2 respectively. Even though the initial configuration of network 2 is optimal, the subsystem information originating from bank 3 indicates a problem of voltage violation and feeder overload. In the implementation, the optimal solution is based on the minimization of the objective function, and is not always feasible. In these two tables, the columns labelled 'F_s', 'F_t' and and 'F_v' are the penalties evaluated for that radial subsystem only and represent the most violated penalty term in the subsystem. The objective function is calculated by summing the square of the maximum term of these three columns.

4.4.2. Study with Switch Addition

Network 2 has been extensively studied with the option of switch additions where any branch is a candidate for an open switch. The objective function is lowered from 3.642 to 1.646 by adding 9 open switches into the network. In studying this network under the switch addition option with various annealing schedules, the optimal solutions found in those cases range from 1.6465 to 1.6487 and are within 0.1% of the lowest solution found(1.6464). This small marginal error is caused by the slight misplacement of the additional switches from the optimal solution. Most of these switches are only one to two branches away from their respective optimal positions. In general, all the solutions agree with the general rough partition of the network. Subsystem information in the optimization studied with the switch addition option is shown in Table B3. The solution shows the alleviation of the voltage violation and feeder segment overload of the subsystem originated from bank 3. Table B4 shows the status of those branches in the optimized network where switching occurs.

4.4.3. Contingency Studies

Two contingency studies on the two networks were examined. In the first study using network 1, a 50 MVA transformer (bank 6 in Table B5) was removed from service by opening the switch at the substation. The entire 20.67 MVA load was transferred to another 31.25 MVA transformer at the same substation(bank 3 in Table B5). At the initial configuration, the 31.25 MVA transformer has a total load of 34.27 MVA and is overloaded by 9.65%. Table B5 shows the initial subsystem information of the network used in this contingency study.

This problem is optimized by the opening and closing of the existing switches in the system. The results of the optimization shown in Table B6 indicate the alleviation of the overload of the subsystem originating from bank 3, and show more even loading of the transformer banks in the load index column(F_t). The objective function is

reduced from 2.32 to 1.74 after optimization by changing the status of 11 pairs of switches as shown in Table B7. Figure 4.1 shows the simplified single line diagram of the network after optimization. The 22 labelled switches corresponding to those in Table B7, are the only ones that have changed status from off to on or vice versa.

In the second study, a 10 MVA transformer (bank 4 in Table B8) was removed from service by opening the switch at the substation. The entire 5.41 MVA load was transferred to another 10 MVA transformer (bank 3 in Table B8) at the same substation. This arrangement caused severe overload in the transformer bank and feeder segments, and voltage violation in the network. The initial subsystem information of the network used in this contingency study is depicted in Table B8.

The objective function of network 2 studied under the switching option is lowered from 4.398 to 3.855 by changing the status of 5 pairs of switches. The optimal subsystem information of network 2 depicted in Table B9 shows the overload of the transformer banks and feeders, and voltage violation persists. A closer study of Table B9 indicates the feeder segment overload and voltage violation originating from bank 3 did not improve at all. Further examination of the location of feeder overload and voltage violation in Figure 4.2 indicates the problem cannot be solved using the switching of the existing switches alone. The network configuration of feeder Fr650 cannot be altered since no closed switch exists between the two open ties and the transformer bank. A feasible solution in which F_s, F_t, and F_v of the objective function are all less than one cannot be found with switching alone. One of the solutions which is available in the optimization program is to plan for new additional open switches in the network.

The same contingency study was further optimized with the option of switch additions. The objective function was lowered from the initial outage value of 4.4 to an optimized value of 2.0. The subsystem information of the optimized network shown in Table B10 shows the previous abnormal conditions were alleviated. A total

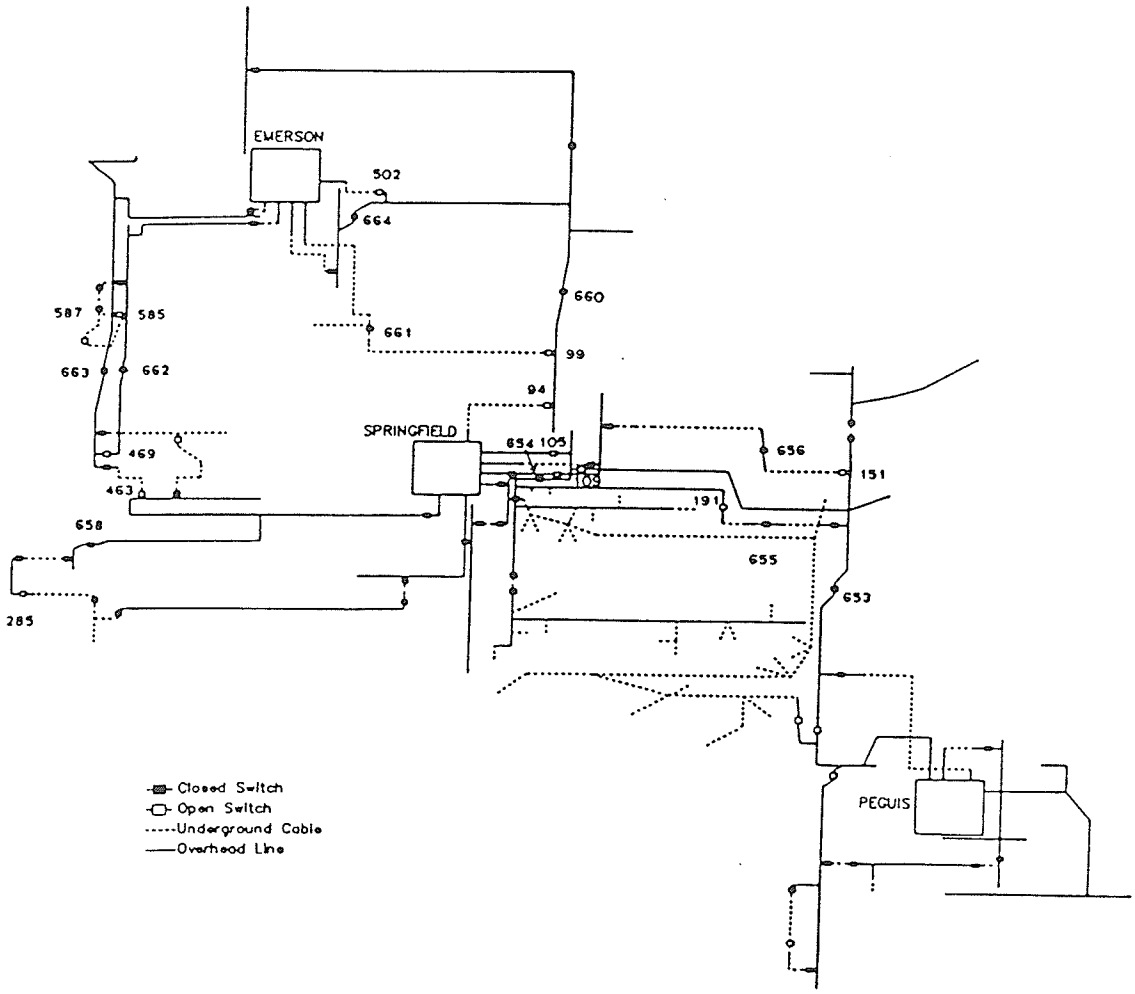


Figure 4.1: Simplified one-line diagram of network 1 after optimization

of 8 additional open switches were added to the network and 8 other previously open switches were closed to shift the loads between feeders. The status of these 8 pairs of segments are shown in Table B11. The previously overloaded feeder segment and bus voltage violations in feeder Fr650 are alleviated by closing both open ties and adding two new sectionalizing switches as shown in Figure 4.3.

The number of iterations, function evaluations and the real losses of the three contingency studies are shown in Table 4.2. Since the total losses of the network were not modelled into the minimization of the objective function, the losses in the optimal network may increase or decrease. A loss penalty may be added into the objective function if required to handle the problem.

Table 4.2 Comparison of the three contingency studies

Study	# of Iterations	Function Evaluations	Real loss	
			Initial	Final
Network 1(operating)	23	2303	491	565
Network 2(operating)	41	3725	454	571
Network 2(switch addition)	54	6059	454	328

4.5. Statistical Physics Parameters

As mentioned in the introduction, the simulated annealing algorithm is modelled from the physical annealing process of materials. Various statistical physics parameters that are used to describe a physical many-particle system at equilibrium can be formulated in a combinatorial optimization problem. The parameters being evaluated are as follows:

(1) The parameter T is modelled as the temperature in the physical annealing of materials. The purpose of this parameter is to control the acceptance in the Metropolis criterion. At high temperatures, the probability of acceptance given

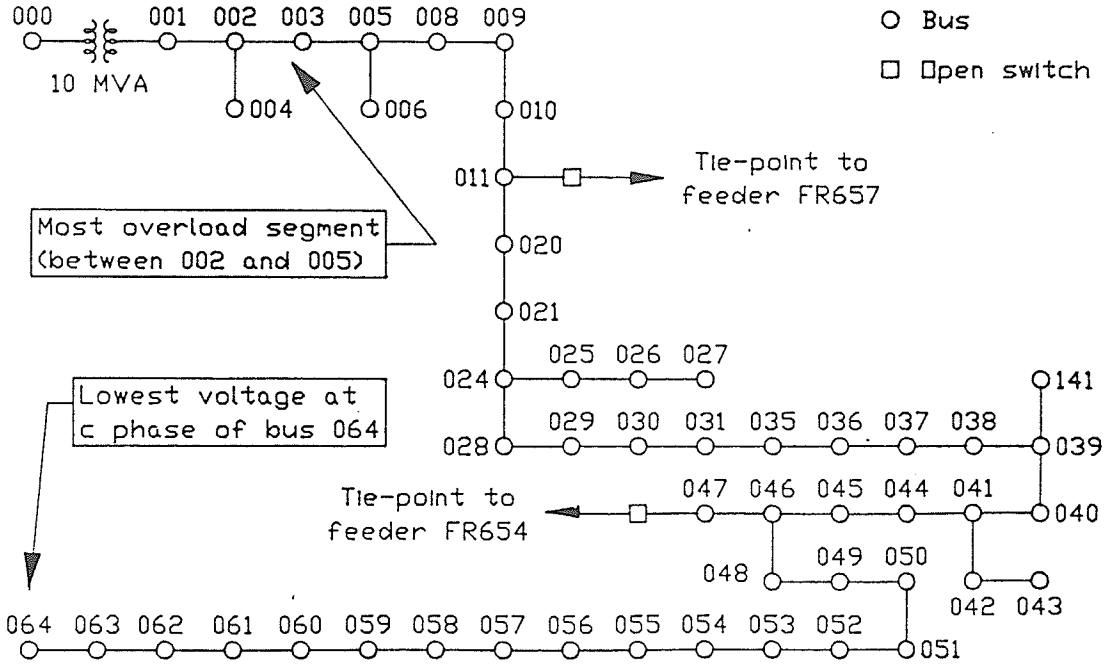


Figure 4.2: Single-line diagram of feeder Fr650 in network 2

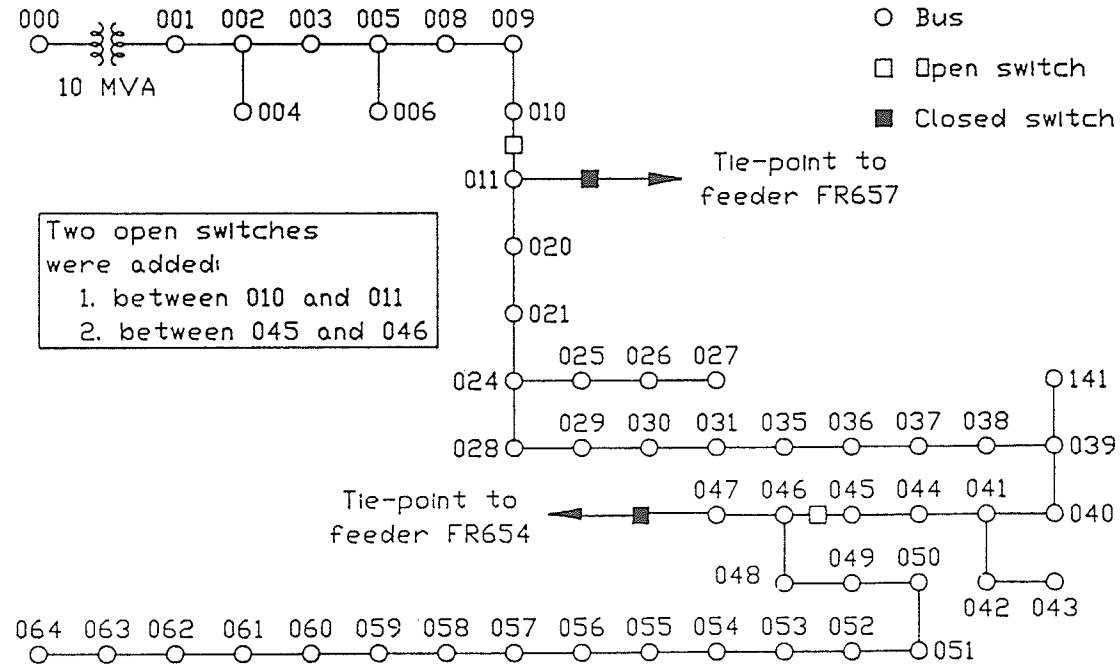


Figure 4.3: Location of new sectionalizing switches in feeder Fr650

by $e^{(-dF/T)}$ is close to one. The objective function of these accepted moves (F_{acc}) is depended largely on T. Laarhoven and Aarts[18] have suggested the fluctuation of F_{acc} would decrease after a number of function evaluations and can be used as a termination criterion for the number of iterations performed at that temperature. A series of tests were performed to see if this fluctuation of the objective function of the accepted moves (F_{acc}) would decrease and settle into steady state. Network 2 was examined under the switch addition option at three different temperatures ranging from T=1000 to T=0.1. All three tests started with the same initial configuration and used the same control parameters. The objective function of the first 300 accepted moves is averaged every five moves and is depicted in Figure 4.4.

The results of the study show that the fluctuation of the objective function does not settle to a steady state. The high spikes on the three figures occur at random intervals. As expected, the magnitude of these fluctuations decreases with temperature indicating only near-optimal configurations are accepted at lower temperatures. The three figures also show the effect of parameter T on the Metropolis criterion of $e^{-dF/T}$. At high temperature such as T=1000, hill climbing moves that increase the objective function up to a value of 480 are accepted. At lower temperature, only moves that are closed to the minimum objective function are accepted.

(2) The average objective function of the accepted moves ($\overline{F}_{acc}$) is calculated by dividing the sum of the objective function of the accepted moves by the number of accepted moves. The $\overline{F}_{acc}$ at different temperatures is depicted in Figure 4.5. At high temperatures, F_{acc} shown in Figure 4.4a causes similar fluctuations in the curve of $\overline{F}_{acc}$. As temperature approaches zero, $\overline{F}_{acc}$ approaches the minimum objective function since only configurations with objective function close to the minimum objective function value are being accepted. A normalized average cost

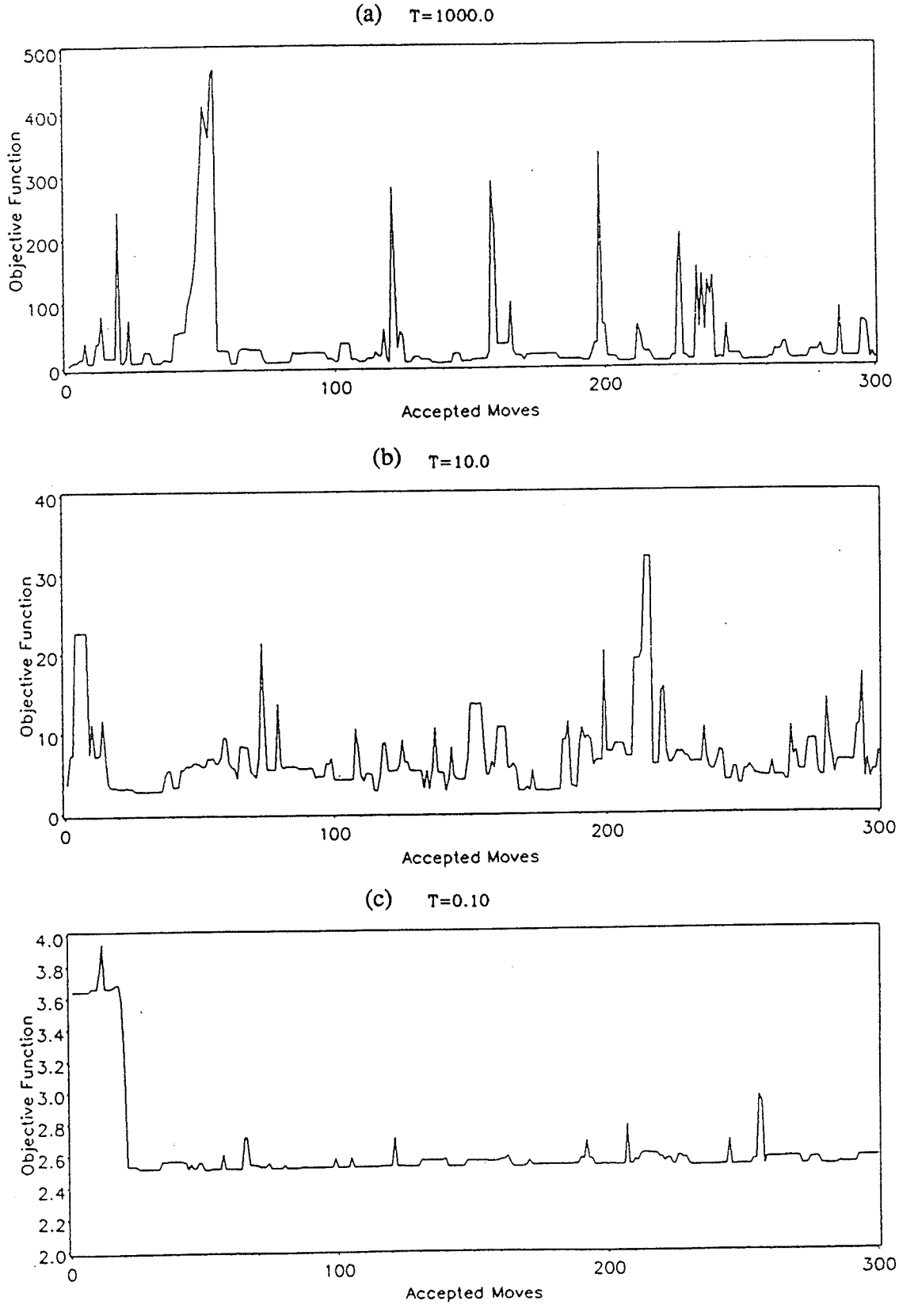


Figure 4.4: Fluctuations of F_{min} at various temperatures

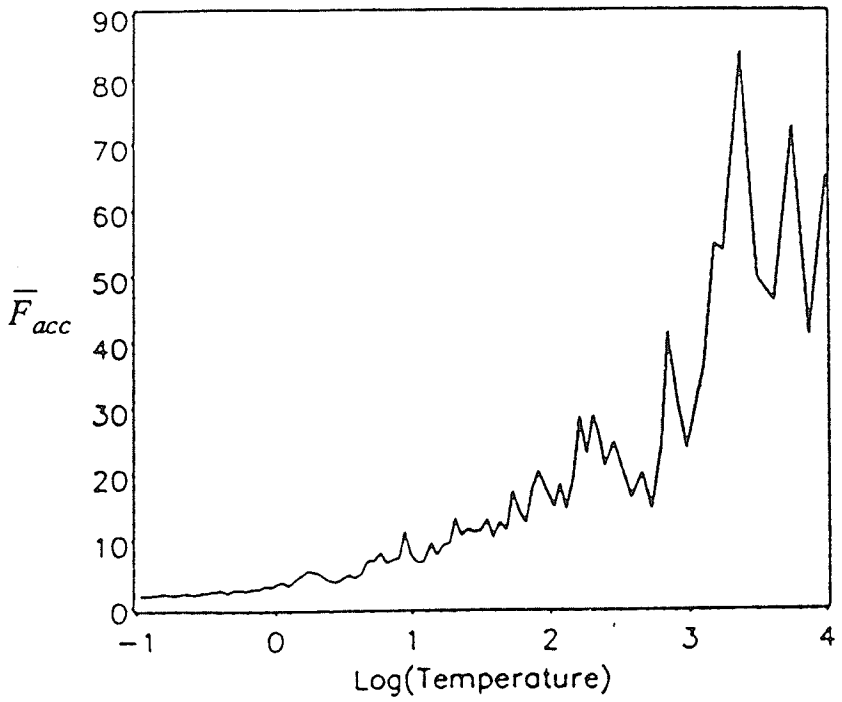


Figure 4.5: Variation of $\overline{F}_{acc}$ with T

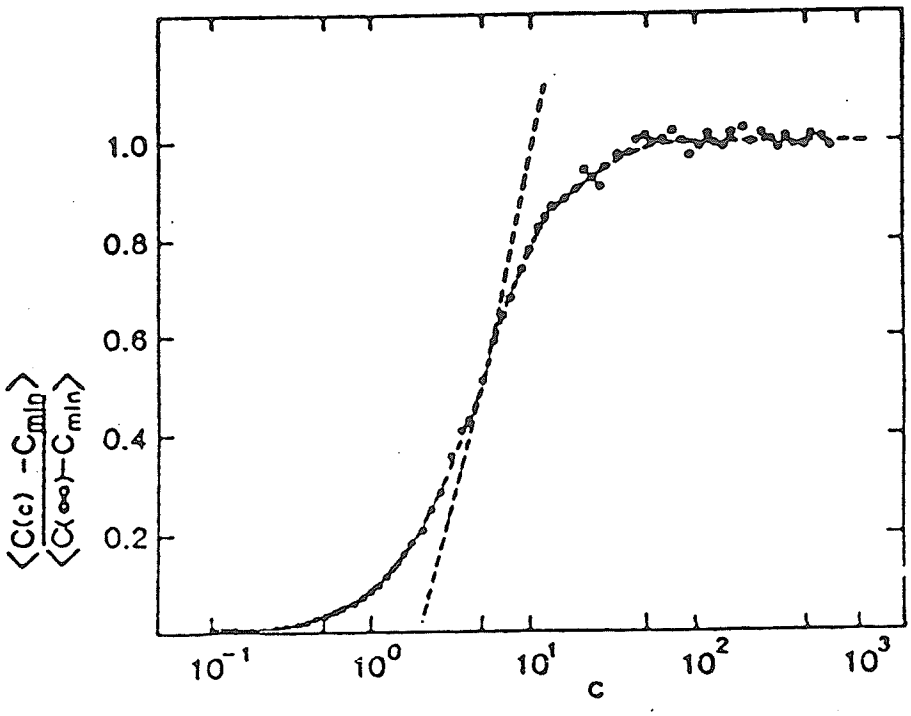


Figure 4.6: Normalized $\overline{F}_{acc}$ in a 100-city TSP[18]

function for a 100-city TSP[18] is shown in Figure 4.6. The average objective function at high temperature is found to be constant whereas in this problem, $\overline{F}_{acc}$ varies drastically due to the relationships between and properties of the three penalty functions. An infeasible configuration with feeder overloading will likely contain severe voltage violation and bank overload. In such cases, the three penalty functions will penalize the configuration very heavily and cause an enormous change in the objective function. At high temperatures, these moves with their drastic changes in the objective function are sometimes accepted and can cause large fluctuations in the $\overline{F}_{acc}$.

(3) Acceptance ratio (α) is defined as the number of accepted transitions divided by proposed transitions. A typical curve of the acceptance ratio in this problem is depicted in Figure 4.7a and it resembles the curve of the Metropolis criterion ($\exp^{(-dF/T)}$) in Figure 4.7b. In this problem, the acceptance ratio does not approach zero as T approaches zero. The reason is that the moving of an open switch between two radial networks may not affect the overall objective function if the most penalized elements are located in a radial subsystem away from these two networks. The value of the acceptance ratio at low temperatures depends on the locations of the most violated elements in the network and the frequency of selecting an open switch that does not affect the overall objective function.

(4) Specific heat $C(T)$ is defined as the rate of change of $\overline{F}_{acc}$ with temperature T. In statistical mechanics, a large value of C indicates a phase transition such as the start of freezing, and therefore slow cooling is recommended. The specific heat of a typical problem is shown in Figure 4.8. The value of C changes drastically at high temperature because of the large fluctuations of $\overline{F}_{acc}$, therefore observing phase change by monitoring the C is not practical and useful. For these reasons, the specific heat is not used in the annealing schedule for the control of the cooling process in the optimization.

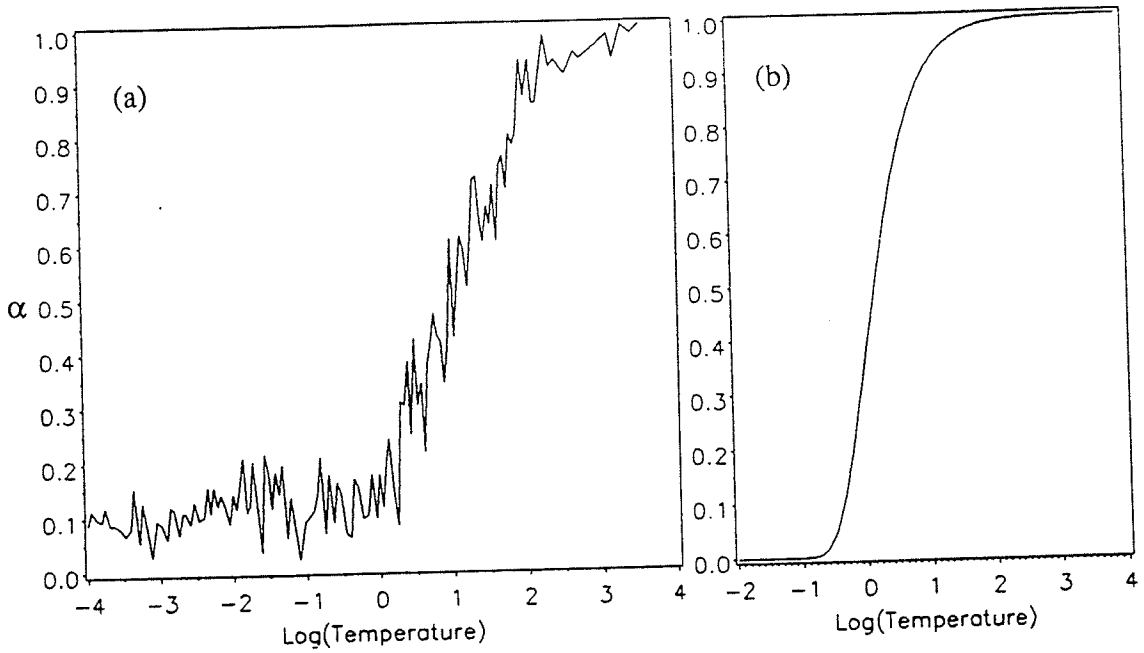


Figure 4.7: Acceptance ratio. (a) in this study, (b) of Metropolis

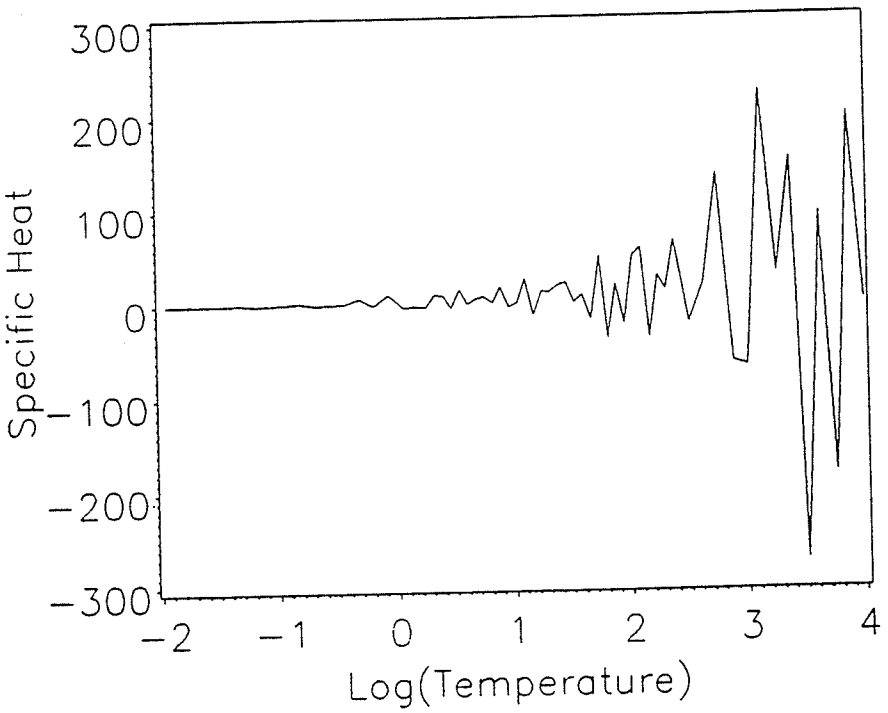


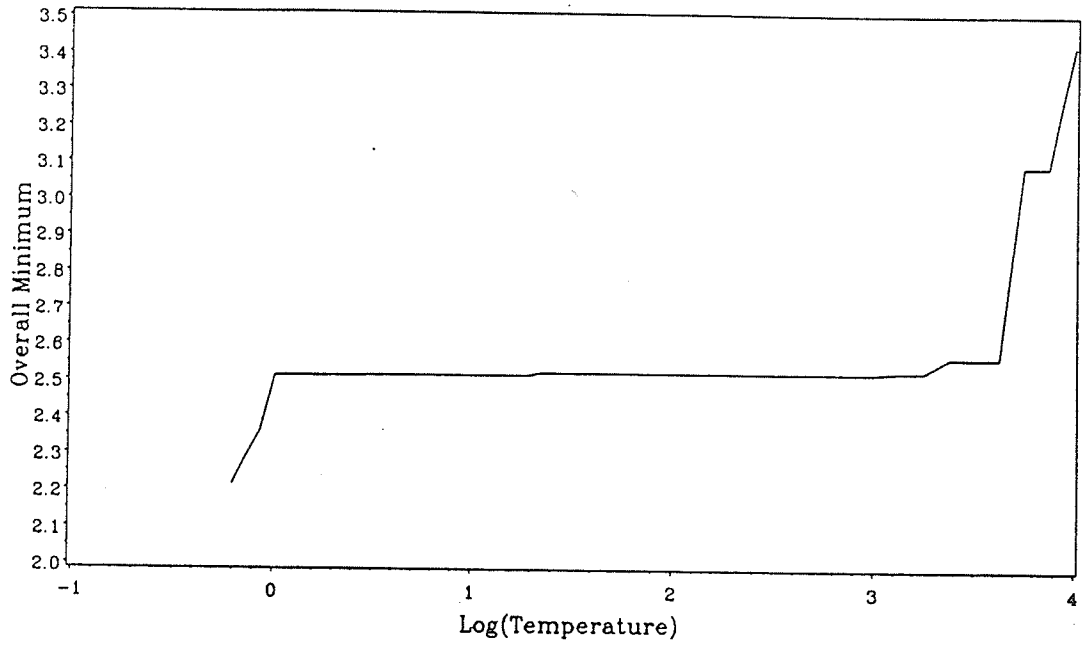
Figure 4.8: Variation of specific heat with T

(5) The overall minimum (F_{min}) is the minimum objective function found in the optimization. Since the algorithm only converges to a near-optimal solution, the value of the overall minimum is a good indicator of the algorithm's performance. The minimum reached will depend on the effectiveness of the annealing schedule and sequence of random number. The sequence of the random number between 0 and 1 is cyclic, and generated by a subroutine. The sequence of the number generated can be varied by using different seeds in the random number generating subroutine. Two switch addition studies with two different sequences of random numbers terminate with two different solutions. The value of the two seeds used and the overall minimum reached is shown in Table 4.3. Examining the minimum objective function in Figure 4.9(a) and (b) indicate that two plateau exist in the problem. Although the objective function in Figure 4.9(a) does not reach the lower function value, it clearly suggests that a lower value may be reached if the program is allowed to continue to a lower temperature. The result in this experiment shows that the sequence of the random number can affect the speed at which the algorithm converges to the solution.

Table 4.3 Effect of random number on F_{min}

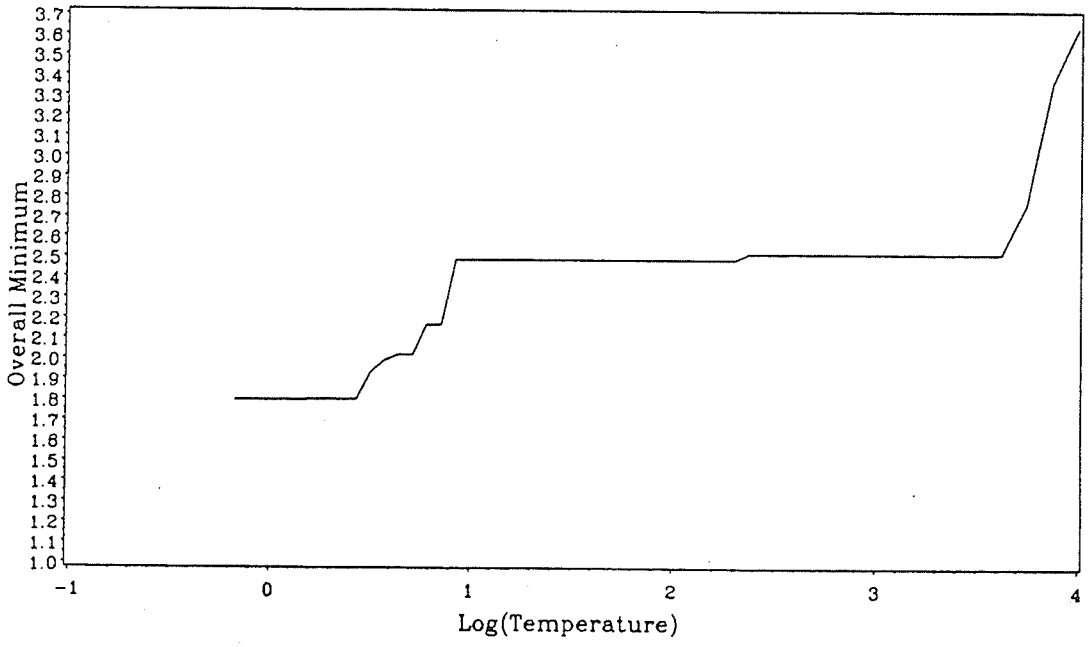
Seed	# of iterations	F_{min}
14	49	2.5139
61061	47	1.7977

RANDOM NUMBER SEED = 14



(a)

RANDOM NUMBER SEED = 61061



(b)

Figure 4.9: Effect of random number on F_{min}

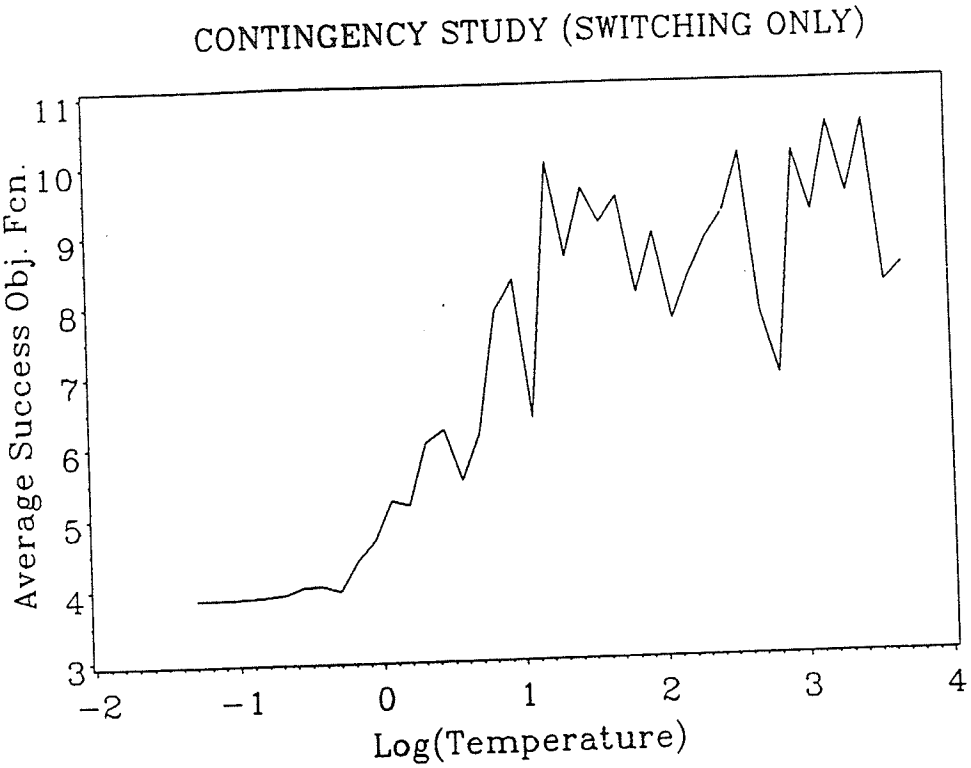
4.5.1. Switch Addition versus Switching

After the parameters were discussed, experiments were carried out to analyze the behavior of the algorithm under both operating and switch addition studies. Using the contingency network of network 2 mentioned in previous section as the initial configuration, the above parameters were observed as shown in Figures 4.10 to 4.12. A comparison of the parameters in these two studies is depicted in Table 4.4.

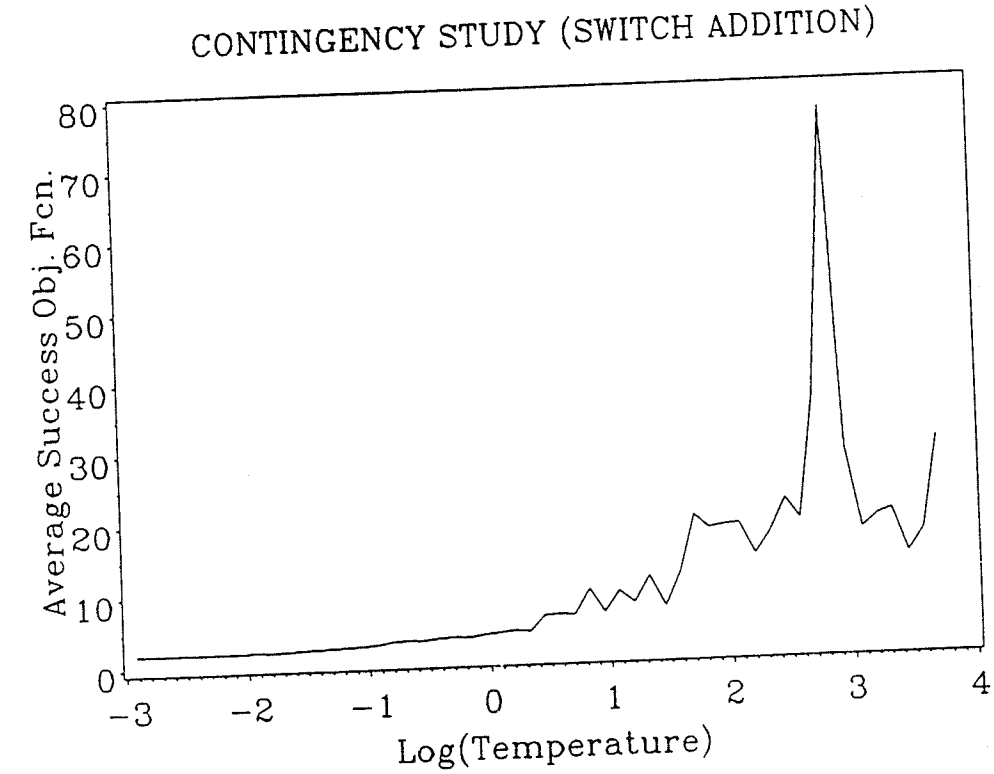
Table 4.4 Parameters in switch addition and operating studies

Parameter	Switch Addition	Switching Only
$\overline{F}_{acc}$	- Fluctuations are more frequent and greater in magnitude (max. fluctuation of 72).	- Small fluctuations for higher temperature (max. fluctuation of 6.5).
α	- Resembles to Metropolis curve and terminates with average acceptance ratio of 0.2	- Terminates with a slightly higher average acceptance ratio
$F_{\min}$	- Lower in value and found in iterations at lower temperature	- Higher in value and found after a few iterations

The problem studied under the switch addition option allows drastic alternation of the network configuration, thus a large fluctuation of $\overline{F}_{acc}$ is expected more often in the study. Since the acceptance ratio is related very closely to the parameter T, the two curves in Figure 4.11 look very similar. The acceptance ratio at low temperature is largely dependent on the locations of the most penalized elements in the network. The optimal solutions in these two studies vary very differently. In the study of the network by changing the status of the existing switches, the results in Figure 4.12a indicate that the number of possible solutions is few, and near-optimal solution can be found in just a few iterations. On the other hand, the number of possible configurations in a switch addition study is much greater than that in the switching

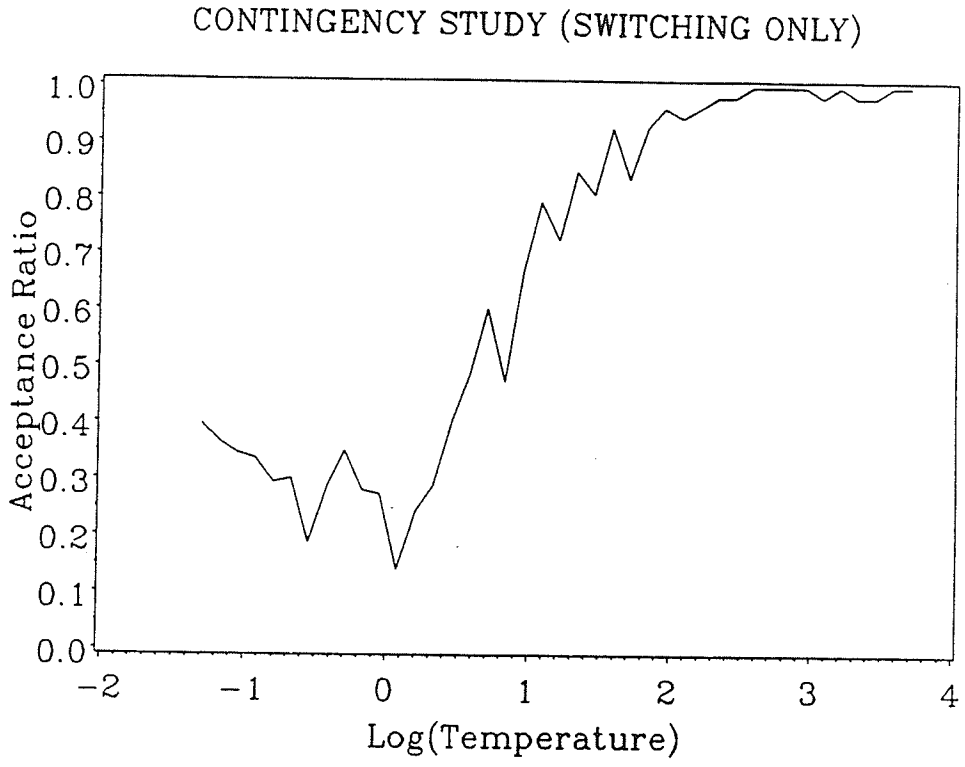


(a)

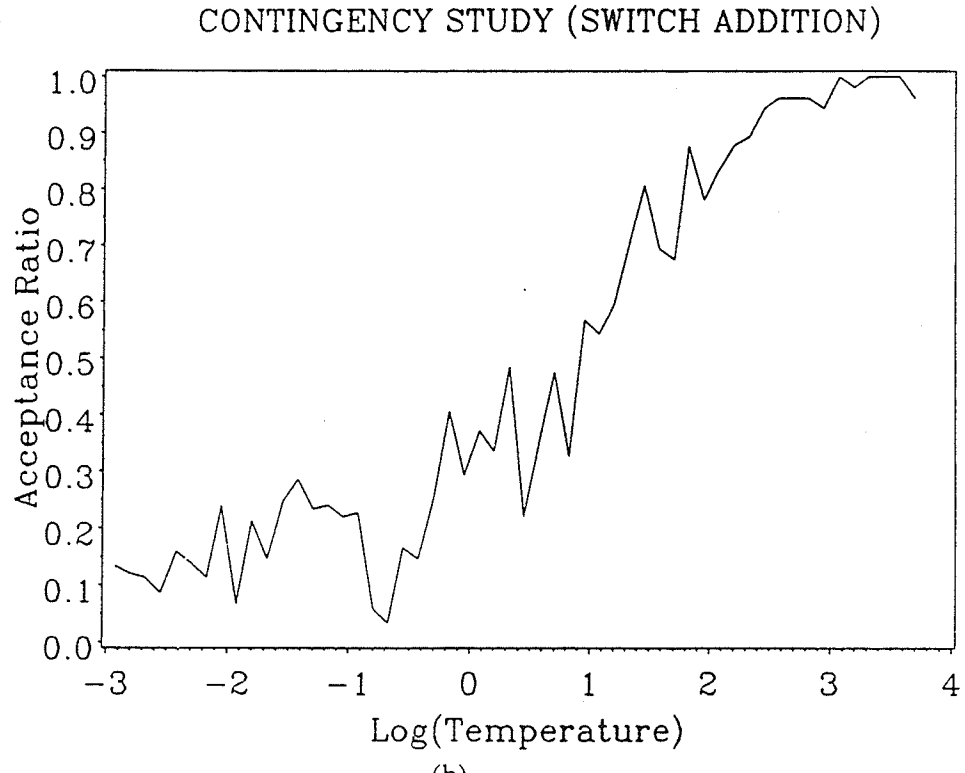


(b)

Figure 4.10: Comparison of $\overline{F}_{acc}$ in operating and switch addition studies

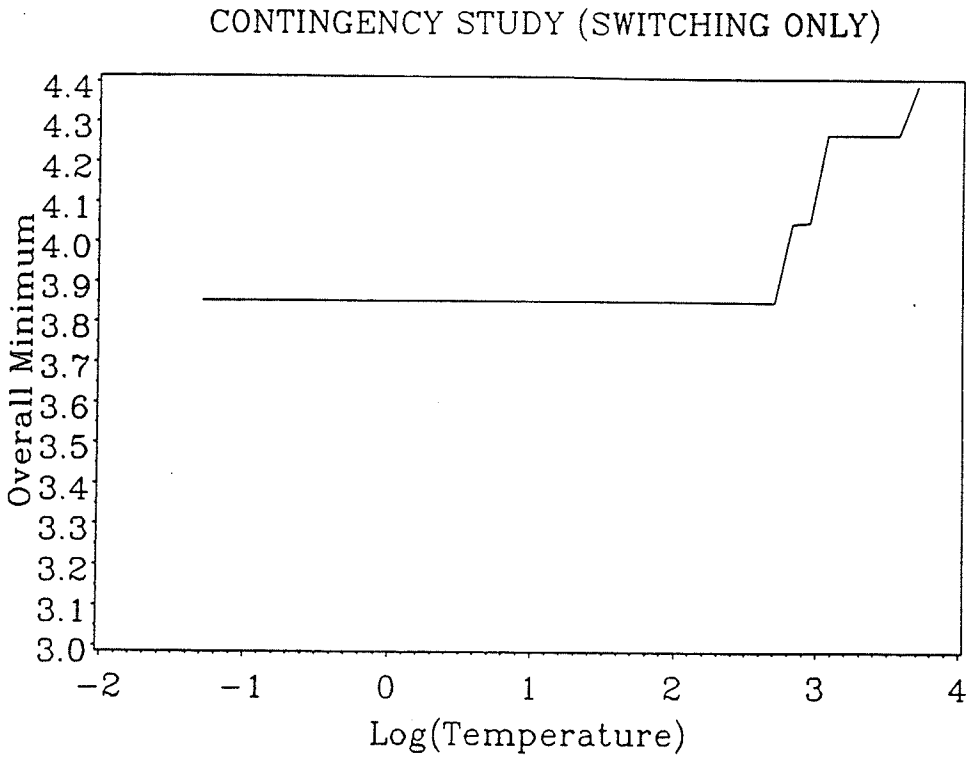


(a)

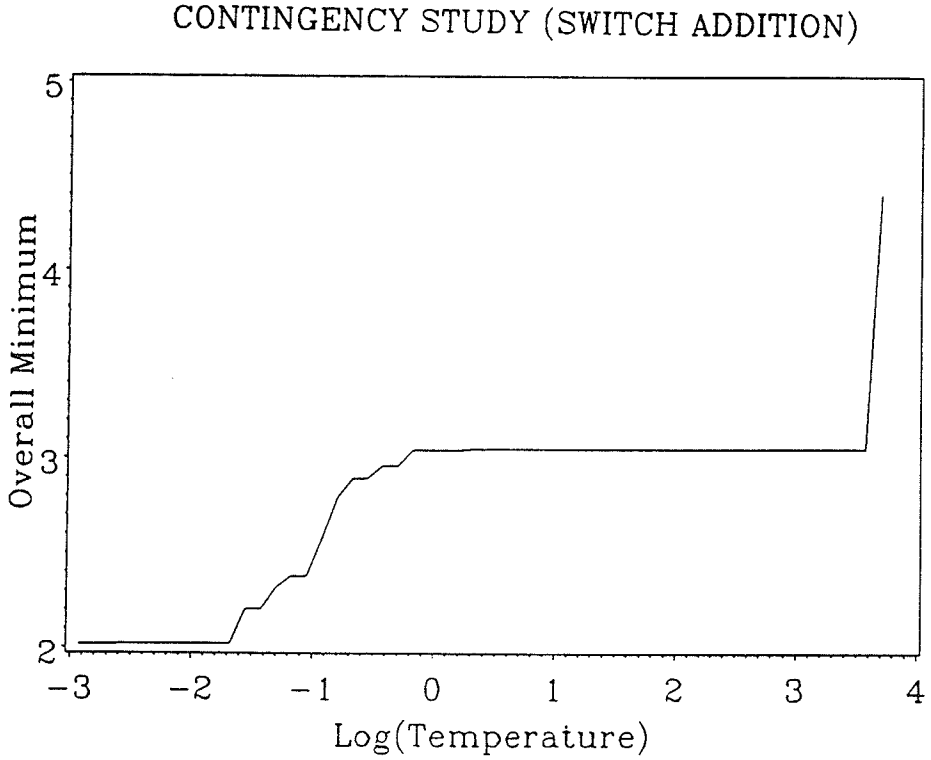


(b)

Figure 4.11: Comparison of α in operating and switch addition studies



(a)



(b)

Figure 4.12: Comparison of F_{min} in operating and switch addition studies

study. Figure 4.12b indicates that the optimum is harder to find, and the problem requires a series of gradient-oriented moves at low temperature to reach an optimal solution. This experiment indicates that the switching study requires only a small number of iterations whereas switch addition study requires a longer annealing process that terminates at a low temperature.

4.6. Annealing Schedule

The annealing schedule governs the sequence of decreasing temperature and the number of function evaluations performed in the simulated annealing algorithm. The annealing schedule (or cooling schedule) involves the choosing of the following parameters:

1. Initial value of temperature, T_o
2. Exit criteria for terminating the algorithm
3. Number of iterations performed at each temperature
4. A rule for calculating the sequence of temperature

The performance of the algorithm has been studied by varying these parameters in the annealing schedule so that an optimal annealing schedule may be determined empirically. Many theoretical-based annealing schedules have been reported by Laarhoven and Aarts[18], but the present schedule used is proposed by Kirkpatrick et al.[20] and is a simple one that based on empirical rules.

4.6.1. Initial Value of T

Initial temperature should be chosen such that all transitions are accepted. In other words, the value of the Metropolis criterion for all transitions should be close to one. Kirkpatrick et al.[20] have proposed an empirical rule to determine the initial temperature. The steps involve the choosing of a large T_o, and performing a small

number of function evaluations. If the acceptance ratio (α) is less than 0.8, the initial T_o is doubled and the steps are repeated until the α at T_o is greater than 0.8.

The purpose of determining the initial temperature T_o is to allow the algorithm to step out from the initial configuration to a random one with a higher objective function. An acceptance of one simply means that the algorithm can escape every minimum and search the configuration randomly.

Testing of the starting temperature, T_o was studied using the contingency case of network 2. The initial temperature used in the algorithm are varied from 1000 to 0.001. The results of this testing are summarized in Table 4.5.

Results show the optimal configuration with an objective function of 2.017 can sometimes be reached when T_o = 0.001. Four tests with different sequence of random number were carried at this temperature and near-optimal solutions were found half of the time. All tests with an initial temperature above 0.1 reached the near-optimal configuration. As we lower T_o, the configuration may sometimes get trapped in a local minima and cannot escape from it. The finding of a near-optimum solution with T_o=0.0001 indicates the problem is quite convex since most of the accepted moves are gradient-oriented. Although results show a value of 0.1 is adequate for the algorithm to reach the near-optimal configuration in the present network, a higher T_o is preferred to ensure the optimal solution can always be reached. A value of 10 is determined empirically for the default value of T_o in the optimization program.

Table 4.5 Effect of T_o in the optimization

T_o	T_{exit}	α_{max}	# of iterations	F_{min}	random seed	CPU (min)
1000	0.000755	1.00	50	2.016976	11	186
10	0.00065	0.56	35	2.018541	11	138
0.1	0.000564	0.41	19	2.019274	11	75
0.1	0.001002	0.39	17	2.017473	27	71
0.01	0.000317	0.44	13	2.019578	11	53
0.01	0.004219	0.44	4	3.031758	27	15
0.01	0.001335	0.25	8	2.017500	6016	34
0.001	0.000750	0.26	2	3.042750	11	8
0.001	0.000563	0.42	3	3.040532	27	11
0.001	0.000100	0.26	9	2.018912	6016	38
0.001	0.000316	0.30	5	2.084441	88	19

Note: A constant reduction factor of 0.75 was used and the algorithm terminates when the difference between F_{acc} and F_{min} are within 0.1% of F_{min}.

4.6.2. Exit Criteria

Many exit criteria have been used to terminate the algorithm and they are as follows:

(1) Exit temperature

The algorithm is terminated when the parameter T is lower than the exit temperature. This method is based directly from the physical annealing and does not necessarily work in an optimization problem since the freezing temperature of a particular problem will depend on the size of the problem and objective function used.

(2) Maximum number of iterations

The algorithm is allowed to continue for a certain number of iterations. An iteration is the number of times the algorithm lowers the parameter T. The maximum number ranges from six, used by Nahar et al.[44] to fifty by Bonomi and Lutton[24]. If the optimal solution can be reached in only a few iterations, the continuation of the algorithm until a maximum number of iterations is reached, is very time consuming.

(3) Acceptance ratio

An acceptance ratio smaller than α_f can be used to terminate the process. This exit criterion is based on the Metropolis criterion that α approaches zero as T goes to zero. As shown in the study, the acceptance ratio does not go to zero at low temperature; and the use of acceptance ratio may not be applicable in this problem.

(4) $\overline{F}_{acc}$

From the discussion on average objective function of the accepted moves ($\overline{F}_{acc}$) in the previous section, results show $\overline{F}_{acc}$ approaches $F_{\min}$ as T decreases. From this observation, an exit criterion based on $\overline{F}_{acc}$ is used as one of the rules in terminating the algorithm. Since this convergence of $\overline{F}_{acc}$ to $F_{\min}$ is consistent in all of the studies, the following rule is used to terminate the problem:

$$\frac{\overline{F}_{acc} - F_{\min}}{F_{\min}} < \zeta$$

The algorithm is terminated when the difference of $\overline{F}_{acc}$ and $F_{\min}$ in that iteration are within a small tolerance ζ of $F_{\min}$.

(5) Number of accepted moves

This exit criterion is directly related to that of the acceptance ratio. If the number of accepted moves equals to zero, the process is terminated. In the problem, the acceptance ratio and the number of accepted moves do not go to zero, so this exit criterion may not be satisfied.

In the implementation, a value of 0.001 for ζ is shown to be adequate for the problem to terminate with a near-optimal solution. The algorithm also terminates when the number of accepted moves is zero, and it is used to halt the algorithm in calculating the acceptance ratio and $\overline{F}_{acc}$. An exit temperature of 0.0001 and a maximum number of 50 iterations are also used as the other exit criteria and their values are determinated empirically. If any one of the following four criteria is violated, the algorithm is automatically terminated. In most cases, the algorithm is terminated by the tolerance criteria of ζ.

4.6.3. Number of Transitions

The most commonly used method of choosing a limit for the number of transitions is a value depending polynomially on the number of variables. The maximum bound on the number of transitions attempted is used to limit the search. On the other hand, the maximum bound on the number of accepted transitions is sometimes used to terminate the search at high temperatures. This criterion is used to save computation time since most of the transitions are accepted. In the implementation, the two maximum bounds on the number of accepted transitions and transitions attempted are proportional to the number of variables in the problem.

Several tests by varying the number of accepted moves between 40 and 400, and the transition attempts between 100 and 500 were examined. Results in Table 4.6 show that all these tests reached a configuration with objective function very close to the optimum value. Values of 4 times and 10 times the number of open switches are determined to be adequate for determining the number of accepted transitions and transitions attempted.

Table 4.6 Effect of number of accepted moves and moves allowed on F_{min}

# of		F_{min}
accepted transition	transition allowed	
40	100	1.646402
60	100	1.647219
160	200	1.648194
400	500	1.646893

4.6.4. Decrement Rule for T

In the process of the freezing of a liquid, a gradual cooling is required to anneal the system into a crystalline form. If the cooling is too rapid, the process is quenched into an intermediate glass state with a higher energy state. Similarly, a gradual lowering of the parameter T in simulated annealing is required to simulate the natural process of freezing.

A decrement rule given in Equation 4.1 has been widely used[20,24].

$$T_{k+1} = \beta T_k \quad , \; k = 0,1,.. \tag{4.1}$$

Kirkpatrick et al.[20] have suggested a $\beta \leq 0.95$ for the TSP. In the problem, this decrement ratio (β) was varied to study the effect on the objective function. The operating study of network 2 was used for these tests. A constant temperature

decrement ratio of 0.70 was used if the acceptance ratio (α) was greater than 0.85 and less than 0.15. A second decrement ratio was used if α was between 0.15 and 0.85. This different decrement ratio was used to vary the temperature range where most changes of the objective function are occurred. The results of the testings are shown in Table 4.7. Figures B4 to B6 of Appendix B show the effect of temperature reduction factor on the overall minimum, acceptance ratio and average objective function of the accepted moves.

Slow cooling with a large value in the decrement factor shows many small changes in the value of the overall minimum ($F_{\min}$), while rapid decrease of $F_{\min}$ is observed with small β. An algorithm with large β performs more iterations and reaches an optimal solution earlier at a slightly higher temperature. The results also show a reduction factor of 0.5 or more gives a minimum value of 1.65, whereas rapid cooling with $\beta = 0.35$ gives only a local minimum value of 2.51. Since an algorithm with large decrement factor increases computation time, a small decrement factor is recommended. The default value of β is set to 0.65 in the implementation.

Table 4.7 Results of varying β from 0.80 to 0.35

β	# of iterations	$F_{\min}$	T_{exit}
0.80	52	1.646886	0.001761
0.65	38	1.647163	0.000313
0.50	26	1.646901	0.001207
0.35	20	2.514261	0.000278

4.7. Objective Function

The objective function in Equation 3.4 of the previous chapter can be stated as follows:

$$F = w_t F_t^{n_t} + w_s F_s^{n_s} + w_v F_v^{n_v}$$

Experiments were carried out to see the algorithm performance with various weighing factors and powers of the three penalty functions.

The three penalty functions F_t, F_s and F_v are used to represent three constraints in the problem. Since these constraints may be conflicting, it is possible that no arrangement can simultaneously minimize the three penalty functions. Experiments were carried out to observe the interaction between penalty functions by setting only one of the weighing factors to one and the others to zero for all three weights. Table 4.8 summaries the result of this experiment.

Table 4.8 Effect of weighing factors on $F_{\min}$

Weights			Penalty Functions			Objective Function
w_s	w_t	w_v	F_s	F_t	F_v	$F_{\min}$
1	0	0	0.7800	0.9582	1.0754	0.60839
0	1	0	1.6159	0.3878	1.5878	0.15041
0	0	1	1.3390	0.8778	0.8063	0.65010
1	1	1	0.8038	0.7287	0.9165	2.01698

The lowest values of F_s, F_t and F_v are 0.78, 0.39 and 0.81 respectively, but none of the three solutions is feasible with all F's less than one. On the other hand, these

penalty functions of the optimal solution with all three weights set equal to one were 0.80, 0.73 and 0.92. Results indicate that the optimal values of F_t, F_s and F_v do not occur simultaneously, and one must find a feasible solution that minimizes the sum of the three penalty functions. For this reason, equal values for the three penalty functions are recommended.

The powers of n_t, n_s and n_v are used to give a nonlinear property to the penalty functions that will encourage values less than one and penalize an infeasible value greater than one. Experiments were carried out to see the effect of the algorithm performance by varying these three powers between two and six. The curves of $F_{\min}$ and $\overline{F}_{acc}$ versus parameter T with different value of n are depicted in Figures B7 and B8 of appendix B respectively. Table 4.9 shows the result of the experiments with different value of n.

Table 4.9 Effect of n on the objective function

Powers	Penalty Functions			Objective Function
n	F_s	F_t	F_v	$F_{\min}$
2	0.8038	0.7287	0.9165	2.016976
4	0.8038	0.7287	0.9167	1.405672
6	0.8038	0.7538	0.9046	1.001245

Results shows that the change of n's does not affect the value of three penalty terms. Figure B.7 shows the change of $F_{\min}$ versus $\log(T)$ for n=2, 4 and 6. These three curves looks very similar and the minimum objective function found in each case is reached when $\log(T) \approx 1.6$.

4.8. Summary

Many worthwhile observations were made in the testing and they can be briefly summarized as follows:

(1) The algorithm has been tested under operating, contingency and switch addition studies and shows very satisfactory results. No solution was found in the operating studies indicating the original network cannot be optimized further by switching of the existing switches. The finding of the solution in this problem does not seem to be very difficult. The solutions found in contingency and switch addition studies were consistent and close to their respective optimal solutions.

(2) Statistical physics parameters such as acceptance ratio and average energy were observed and compared with the theoretical ones from other optimization problems. The acceptance ratio at low temperature did not approach zero in these problems. Testings also showed large fluctuations of $\overline{F}_{acc}$ due to the nonlinearity of the penalty function and the interaction of the three penalty terms. The $F_{\min}$ in operating study was found in only a few iterations while the solution in the switch addition study was found at a very low T and after a large number of iterations.

(3) The annealing schedule requires the choosing of the four parameters:

   (i) starting value of T

   (ii) a rule for lowering T

   (iii) number of transitions

   (iv) exit criteria

   Large reduction of temperature analogous to rapid cooling of materials resulted in a configuration with a local optimum. A low starting value for T reduced the number of hill-climbing moves and resulted in a solution with a local optimum.

An annealing schedule should be optimized so that a near-optimal solution can be found with the least computation time.

(4) Since the constraints may be conflicting, no arrangement can simultaneously minimize the three penalty function in the objective function. Tests show unequal values of weighing factors in the objective function could cause one or more penalty functions to be violated. An objective function which minimizes the sum of the three penalty functions with equal weighing factors gives a feasible solution. Tests show the three penalty functions raised to different powers do not change the value of the penalty functions F_s, F_t and F_v.

CHAPTER 5

Conclusions and Recommendations

5.1. Conclusions

The performance of the optimization program as we observed is summarized below:

(1) A Manitoba Hydro 7.2kV distribution network was extensively tested under operating, contingency and switch adding studies and showed very promising results. The solution obtained by simulated annealing was consistent and near-optimal.

(2) Although tremendous efforts and a massive amount of data were required to collect the input data, the detailed model of the distribution network allows for easy future addition of other network components such as regulators and capacitors.

(3) The ac load flow gives accurate values of the three-phase current and bus voltage, but it is estimated to account for 90% of the CPU time used in the optimization. It is extremely inefficient from a computational standpoint.

(4) Simulated annealing accepts configurations with higher value of the objective function in an effort to search the configuration space more thoroughly. The hill-climbing characteristic reduces the possibility of the configuration being trapped in a local minimum.

(5) The present program is impractical to be used as a real-time feeder reconfiguration strategy in an automated distribution network. With further enchancements, it may be a useful tool for distribution engineers in the planning and analyzing of existing distribution networks.

5.2. Recommendations

To complete this study, the following recommendations to enchance the program are suggested:

(1) Further testing on other networks should be investigated to ensure the effectiveness of the algorithm.

(2) A speed up of the program may be possible if a simplified load flow technique is used. Other load flow methods such as the linearized dc load flow which converts the nonlinearized dc load flow by ignoring the reactive power calculation should be investigated. The accuracy of the parameters and speed improvement should be compared to find a more effective technique.

(3) Since the distribution system's loss accounts for more than half of the total loss in the power system, a penalty for total system loss should be implemented.

(4) Heuristics for choosing a random closed switch may be used to accelerate the convergence to the optimal solution. At high temperature, all closed segments on both sides of an open switch are candidates for switching. As T approaches zero, the positions of the open switches are only a few switches away from the optimal solution. In other words, only segments close to an open switch are required to be considered at lower temperature.

(5) Several feeder segments with the same conductor type with no load connecting between them should grouped together to form one feeder segment. The feeder segments generated by digitizing create many fragmented segments with no load between them. The amount of search performed by the algorithm can be greatly reduced if only segments connected to a load bus or a junction are allowed to switch.

APPENDIX A

Load Flow Algorithm

This appendix describes the load flow method used to calculate the steady-state characteristic of the distribution network. The method proposed by Shirmohammadi et al.[43] was shown to be more efficient than the Newton-Raphson power flow method when studying radial and weakly meshed distribution networks.

In each radial subsystem, a root node with a known voltage is selected. The branches are then assigned to layers depending on their proximity to the root node. The numbering of the branches in a new layer starts only if the branches in the layer above it have been numbered. The reason for the branch numbering is to ensure the order of branch examination is either from top to bottom or vice versa. In this tree structure, the node of a branch L closest to the root node is denoted by L_1 and the node farther away is denoted by L_2. A tree using this branch numbering strategy is shown in Figure A1a. In the implementation, the numbering of the branches is not used since the data structure allows the traversal of the tree from root to leaf node and vice versa.

Traversal is a method of examining the nodes of a tree systematically so that each node is processed exactly once[45]. A preorder traversal of a tree operates as follows:

1. Visit the root.
2. Perform a preorder traversal of the first subtree,
3. Perform a preorder traversal of other subtrees.

The sequence of a preorder traversal of the tree in Figure A1a is illustrated in Figure A1b.

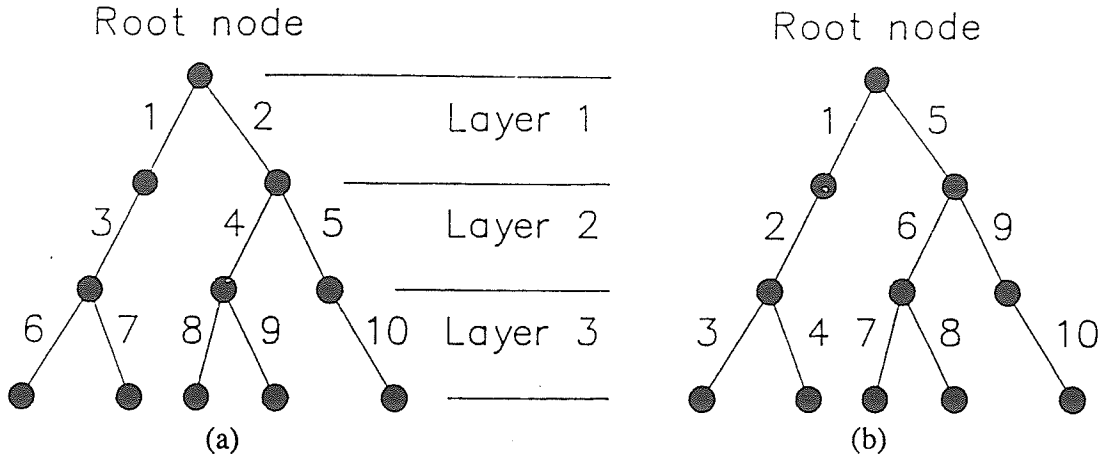


Figure A1: Sequence of branch visited. (a) by layers using branch numbering, (b) by preorder traversal.

The voltage profile of the root node is assumed for the other nodes in the network. The iterative algorithm consists of the following steps:

(1) Forward sweep: At iteration k, the nodal current injection $I_i(k)$ at node i is calculated as:

$$I_i^{(k)} = (S_i / V_i^{(k-1)})^* - Y_i V_i^{(k-1)} , \quad i = \text{Root, ..., Leaves} \tag{A.1}$$

where $V_i^{(k-1)}$ is the voltage at node i during the $(k-1)^{th}$ iteration; S_i is the specified complex power injection at node i, and Y_i is the sum of all incident shunt admittances at node i. This calculation is performed on the forward sweep of the preorder traversal.

(2) Backward sweep: At iteration k, starting from the branches farthest from the root node and moving towards the root node, the current in any branch L (denoted by J_L is:

$$J_L^{(k)} = -I_{L_2}^{(k)} + \sum_{L_2} (I_{in}) , \quad L = \text{Leaves, ..., Root} \tag{A.2}$$

where I_{in} is the current in branch emanating from a node and $I_{L_2}^{(k)}$ is the current injection at node L_2 as calculated in step 1. This backward sweep is performed on the backtracking of the preorder traversal. When the traversal algorithm leaves an node during backtracking, it returns to the branch connecting to that node and step 2 is performed on that branch.

(3) Forward sweep: Once all currents are calculated, a second forward sweep starting from root node is used to update the node voltages:

$$V_{L_2}^{(k)} = V_{L_1}^{(k)} + Z_L J_L^{(k)} \ , \ L = \text{Root, ..., Leaves} \tag{A.3}$$

where Z_L is the series impedance of branch L. This is the direct application of the KVL. In the implementation, a second preorder traversal is used to update the voltage and to calculate the real and reactive power mismatches. There is no operation performed on the backward sweep of the second preorder traversal.

These three steps are repeated until the exit criterion is met. The known complex load at node i is S_i. However, the complex power using the voltages and currents calculated in steps 1 to 3 in the k^{th} iteration is:

$$S_i^{(k)} = V_i^{(k)} I_i^{(k)*} - Y_i \mid V_i^{(k)} \mid^2 \tag{A.4}$$

The mismatches are then defined as:

$$\Delta P_i^{(k)} = \text{Re}[\ S_i^{(k)} - S_i\ ] \tag{A.5}$$

$$\Delta Q_i^{(k)} = \text{Im}[\ S_i^{(k)} - S_i\ ] \ , \ i = \text{Root, ..., Leaves}$$

The load flow is terminated when the maximum real and reactive power mismatches fall below a specified tolerance ξ. The exit criterion of the load flow is as follows:

$$\text{Max}\left[\frac{\Delta P_i^{(k)}}{\text{Re}[S_i]}\right] \leq \xi \quad \text{and} \quad \text{Max}\left[\frac{\Delta Q_i^{(k)}}{\text{Im}[S_i]}\right] \leq \xi \tag{A.6}$$

The above load flow algorithm represents the steps performed for a single-phase system. A three-phase system requires each steps to be executed three times.

APPENDIX B

Figures and Tables

Table B1: Initial subsystem information of network 1

Subsystem		Penalty Functions			Losses	
#	Bank Name	F_s	F_t	F_v	Real(kW)	Reactive(kvar)
1	Peguis B1	0.3567	0.4468	0.7778	30.31	67.64
2	Peguis B2	0.8400	0.6234	0.7778	161.76	369.87
3	Springfield B3	0.6573	0.4352	0.7778	92.56	193.97
4	Springfield B4	0.7411	0.3444	0.7778	127.96	302.65
5	Emerson B1	0.6002	0.5589	0.7778	45.44	101.66
6	Emerson B2	0.6271	0.3612	0.7778	32.78	52.21

Note: F_v is defined differently in this study

$$F_v = \left| \frac{V_{lp} - V_{avg}}{\Delta V} \right| \text{ for } l=1,..,s$$

where V_{avg} is the average of the voltage limits, V_{lp} is the p-phase voltage at bus l and ΔV is the maximum allowable voltage deviation.

Table B2: Initial subsystem information of network 2

Subsystem		Penalty Functions			Losses	
#	Bank Name	F_s	F_t	F_v	Real(kW)	Reactive(kvar)
1	Dakota B1	0.9934	0.2555	0.7062	55.45	108.79
2	Dakota B3	0.8037	0.3069	0.5151	66.19	146.03
3	Frobisher B1	1.1515	0.9255	1.2080	223.26	270.37
4	Frobisher B2	0.5503	0.5407	1.0387	101.45	141.49

Table B3: Optimal subsystem information of network 2 in switch addition study

Subsystem		Penalty Functions			Losses	
#	Bank Name	F_s	F_t	F_v	Real(kW)	Reactive(kvar)
1	Dakota B1	0.7800	0.2495	0.5886	43.76	89.45
2	Dakota B3	0.8038	0.4271	0.9044	209.55	458.88
3	Frobisher B1	0.1989	0.3493	0.5336	2.16	2.19
4	Frobisher B2	0.7988	0.4204	0.6923	59.75	86.43

Table B4: Optimal branch status of network 2 in switch addition study

Feeder segment		Status	
From	To	From	To
DK730.520	DK730.510	Closed	Open
DK744.270	DK744.280	Closed	Open
DK744.400	DK744.410	Closed	Open
DK746.480	DK746.490	Closed	Open
FR650.005	FR650.003	Closed	Open
FR650.046	FR650.045	Closed	Open
FR654.003	FR654.002	Closed	Open
FR654.120	FR654.024	Closed	Open
FR657.033	FR657.026	Closed	Open
DK746.810	DK730.570	Open	Closed
DK731.310	DK744.280	Open	Closed
DK745.370	DK744.410	Open	Closed
FR654.650	FR650.047	Open	Closed
DK745.610	DK746.490	Open	Closed
DK746.450	FR657.053	Open	Closed
FR657.008	FR650.011	Open	Closed
DK744.450	FR654.460	Open	Closed
FR650.004	FR654.006	Open	Closed

Table B5: Initial subsystem information of network 1 in contingency study

Subsystem		Penalty Functions			Losses	
#	Bank Name	F_s	F_t	F_v	Real(kW)	Reactive(kvar)
1	Peguis B1	0.3566	0.4468	0.2555	30.3	67.6
2	Peguis B2	0.8400	0.6233	0.6440	161.7	369.8
3	Springfield B3	0.7411	1.0965	0.4068	220.5	496.6
4	Springfield B4	-	-	-	-	-
5	Emerson B1	0.6002	0.5589	0.7778	45.4	101.7
6	Emerson B2	0.6271	0.3612	0.7778	32.8	52.2

Note: Subsystem 4 originating from Springfield B4 is out of service

Table B6: Optimal subsystem information of network 1 in contingency study solved by the switching of the existing switches

Subsystem		Penalty Functions			Losses	
#	Bank Name	F_s	F_t	F_v	Real(kW)	Reactive(kvar)
1	Peguis B1	0.7940	0.7281	0.5560	103.4	207.8
2	Peguis B2	0.8433	0.6249	0.6249	162.8	372.1
3	Springfield B3	0.7399	0.7929	0.3748	135.7	303.9
4	Springfield B4	-	-	-	-	-
5	Emerson B1	0.5941	0.5724	0.2782	50.9	108.4
6	Emerson B2	0.8506	0.7281	0.4427	112.3	199.1

Table B7: Optimal branch status of network 1 in contingency study

Feeder segment			Status	
#	From	To	From	To
94	SD790.040	SD790.030	Closed	Open
99	SD790.070	SD790.090	Closed	Open
105	SD791.030	SD791.020	Closed	Open
109	SD791.050	SD791.070	Closed	Open
151	SD791.240	SD791.460	Closed	Open
191	SD792.351	SD792.350	Closed	Open
285	SD775.340	SD775.330	Closed	Open
463	SD777.180	SD777.080	Closed	Open
469	SD777.222	SD777.224	Closed	Open
502	EN481.040	EN481.030	Closed	Open
585	EN483.160	EN483.170	Closed	Open
587	EN483.140	EN483.150	Open	Closed
653	PS433.160	SD791.190	Open	Closed
654	SD776.030	SD791.060	Open	Closed
655	SD791.450	SD792.352	Open	Closed
656	SD774.140	SD791.470	Open	Closed
658	SD777.490	SD775.400	Open	Closed
660	EN481.240	SD790.080	Open	Closed
661	EN482.060	SD790.130	Open	Closed
662	EN483.180	SD777.290	Open	Closed
663	EN486.370	SD777.270	Open	Closed
664	EN484.080	EN481.060	Open	Closed

Table B8: Initial subsystem information of network 2 in contingency study

Subsystem		Penalty Functions			Losses	
#	Bank Name	F_s	F_t	F_v	Real(kW)	Reactive(kvar)
1	Dakota B1	0.9934	0.2555	0.7062	55.45	108.79
2	Dakota B3	0.8037	0.3069	0.5151	66.19	146.03
3	Frobisher B1	1.1515	1.4663	1.2080	324.97	412.21
4	Frobisher B2	-	-	-	-	-

Note: Subsystem 4 originating from Frobisher B2 is out of service

Table B9: Optimal subsystem information of network 2 in contingency study solved by the switching of the existing switches

Subsystem		Penalty Functions			Losses	
#	Bank Name	F_s	F_t	F_v	Real(kW)	Reactive(kvar)
1	Dakota B1	0.4576	0.2080	0.3268	9.93	22.79
2	Dakota B3	1.1460	0.4313	1.1971	333.22	682.79
3	Frobisher B1	1.1550	1.0277	1.2103	228.03	277.44
4	Frobisher B2	-	-	-	-	-

Note: Subsystem 4 originating from Frobisher B2 is out of service

Table B10: Optimal subsystem information of network 2 in contingency study solved by switch addition

Subsystem		Penalty Functions			Losses	
#	Bank Name	F_s	F_t	F_v	Real(kW)	Reactive(kvar)
1	Dakota B1	0.7800	0.2488	0.5886	42.71	87.55
2	Dakota B3	0.8038	0.4349	0.9165	225.88	491.17
3	Frobisher B1	0.7989	0.7287	0.7078	59.04	84.68
4	Frobisher B2	-	-	-	-	-

Note: Subsystem 4 originating from Frobisher B2 is out of service

Table B11: Optimal branch status of network 2 in contingency study solved by switch addition

Feeder segment			Status	
#	From	To	From	To
51	DK730.510	DK730.500	Closed	Open
111	DK744.250	DK744.260	Closed	Open
126	DK744.410	DK744.400	Closed	Open
301	FR650.010	FR650.011	Closed	Open
328	FR650.046	FR650.045	Closed	Open
479	FR654.002	FR650.001	Closed	Open
487	FR654.010	FR654.009	Closed	Open
563	FR657.009	FR657.008	Closed	Open
642	DK746.810	DK730.570	Open	Closed
644	DK731.310	DK744.280	Open	Closed
645	DK745.370	DK744.410	Open	Closed
646	FR654.650	FR650.047	Open	Closed
648	DK746.450	FR657.053	Open	Closed
649	FR657.008	FR650.011	Open	Closed
650	DK744.450	FR654.460	Open	Closed
651	FR650.004	FR654.006	Open	Closed

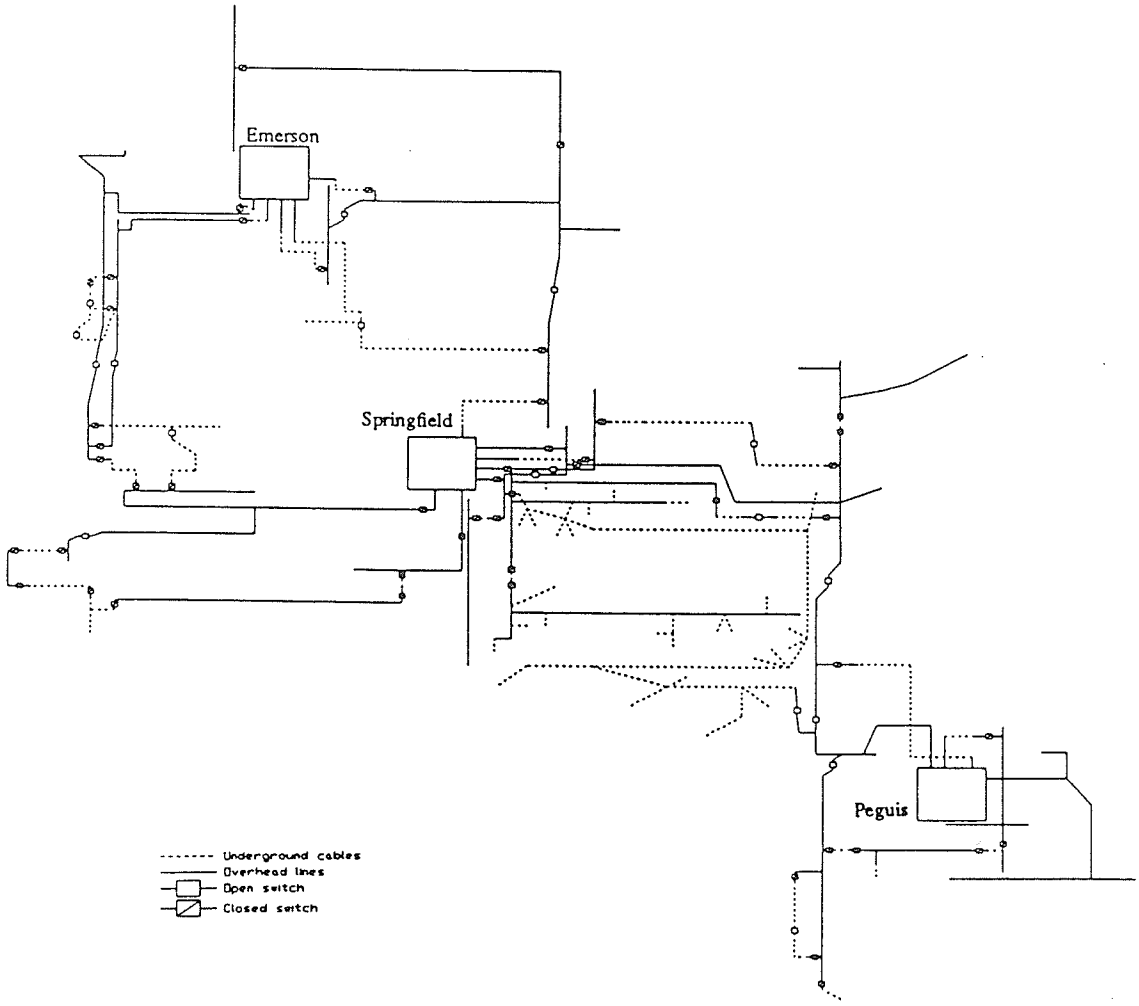


Figure B1: Single-line diagram of network 1

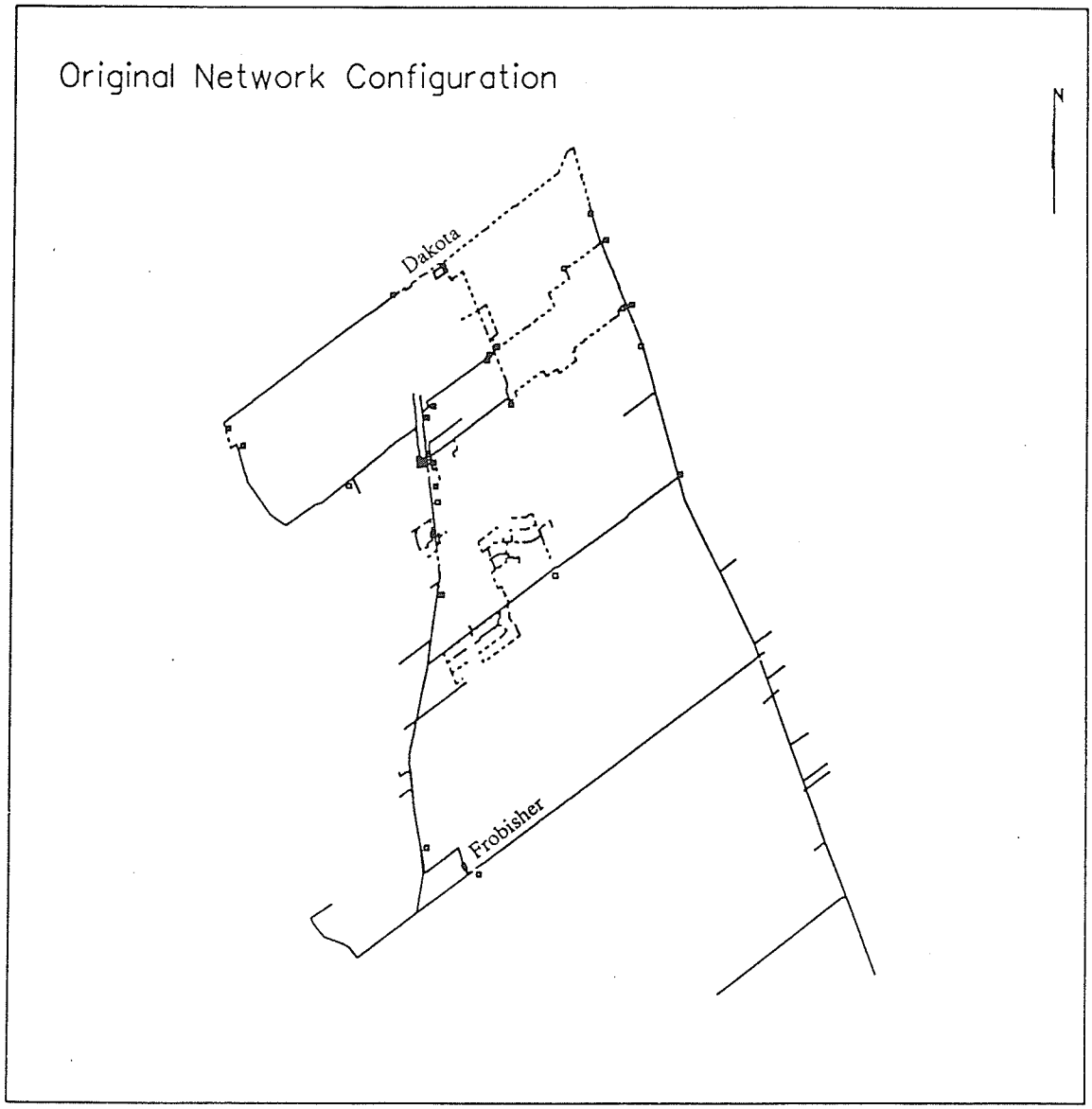


Figure B2: Single-line diagram of network 2

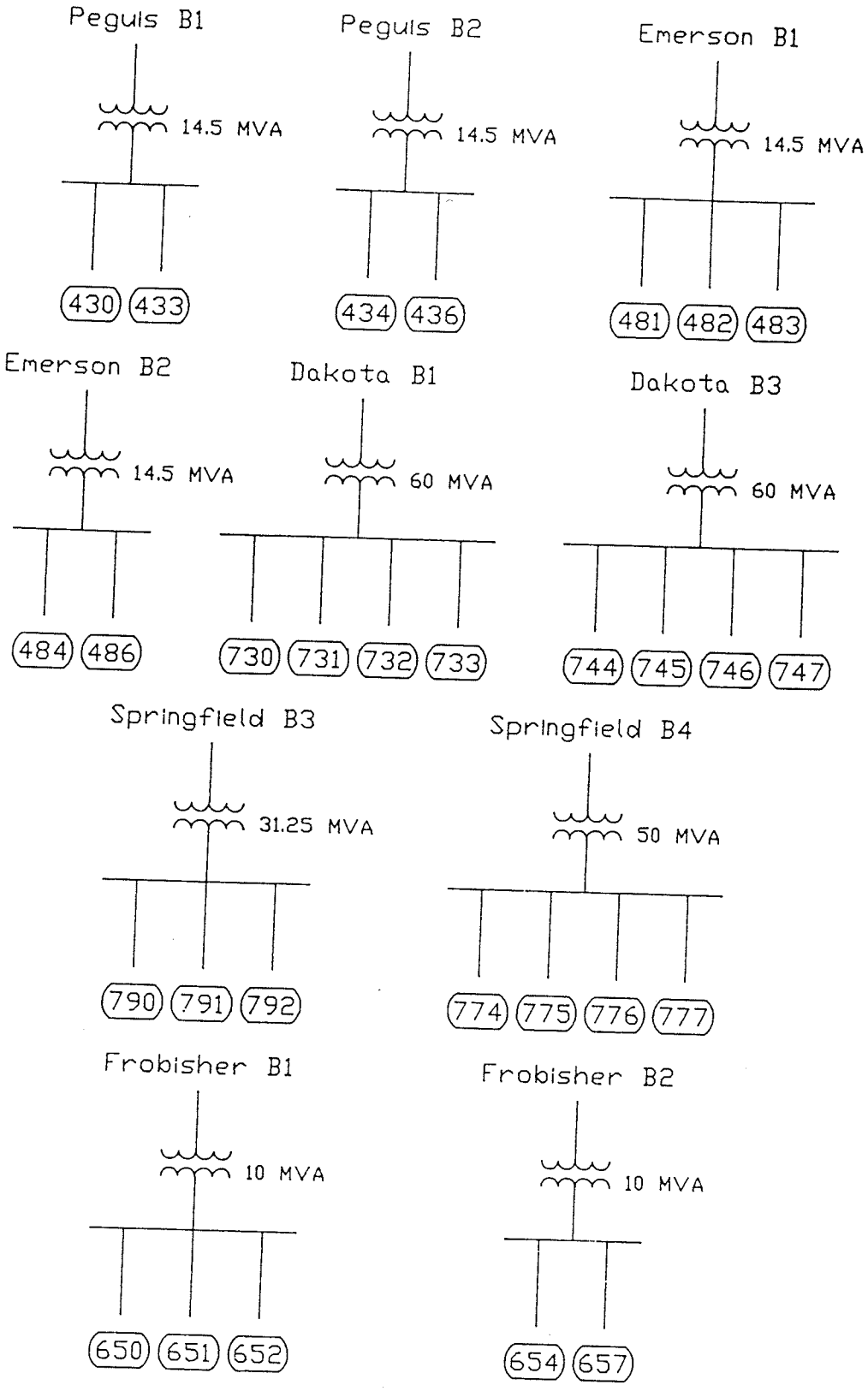


Figure B3: Schematics of banks and feeders used in network 1 and network 2

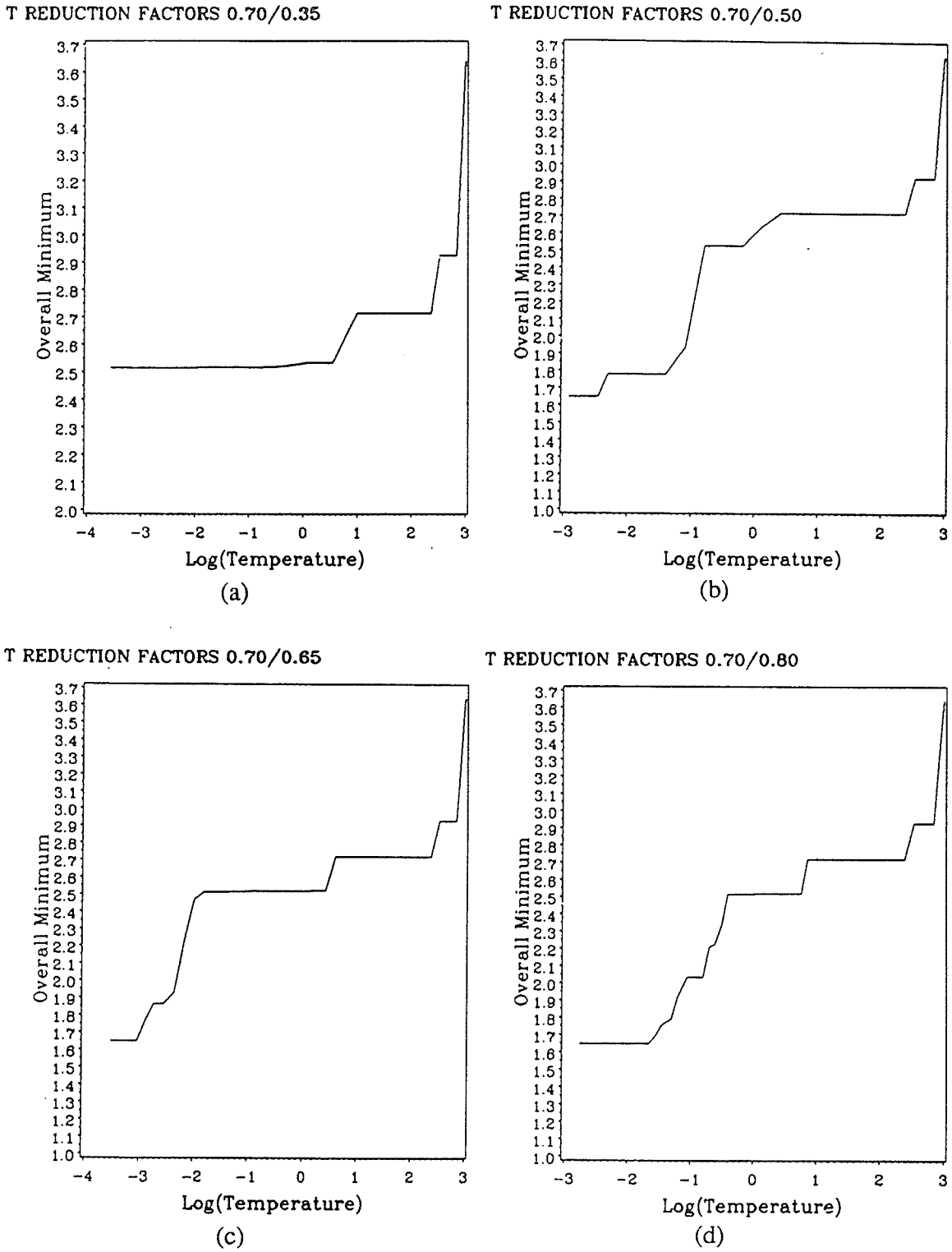


Figure B4: Effect of varying temperature reduction factor on $F_{\min}$

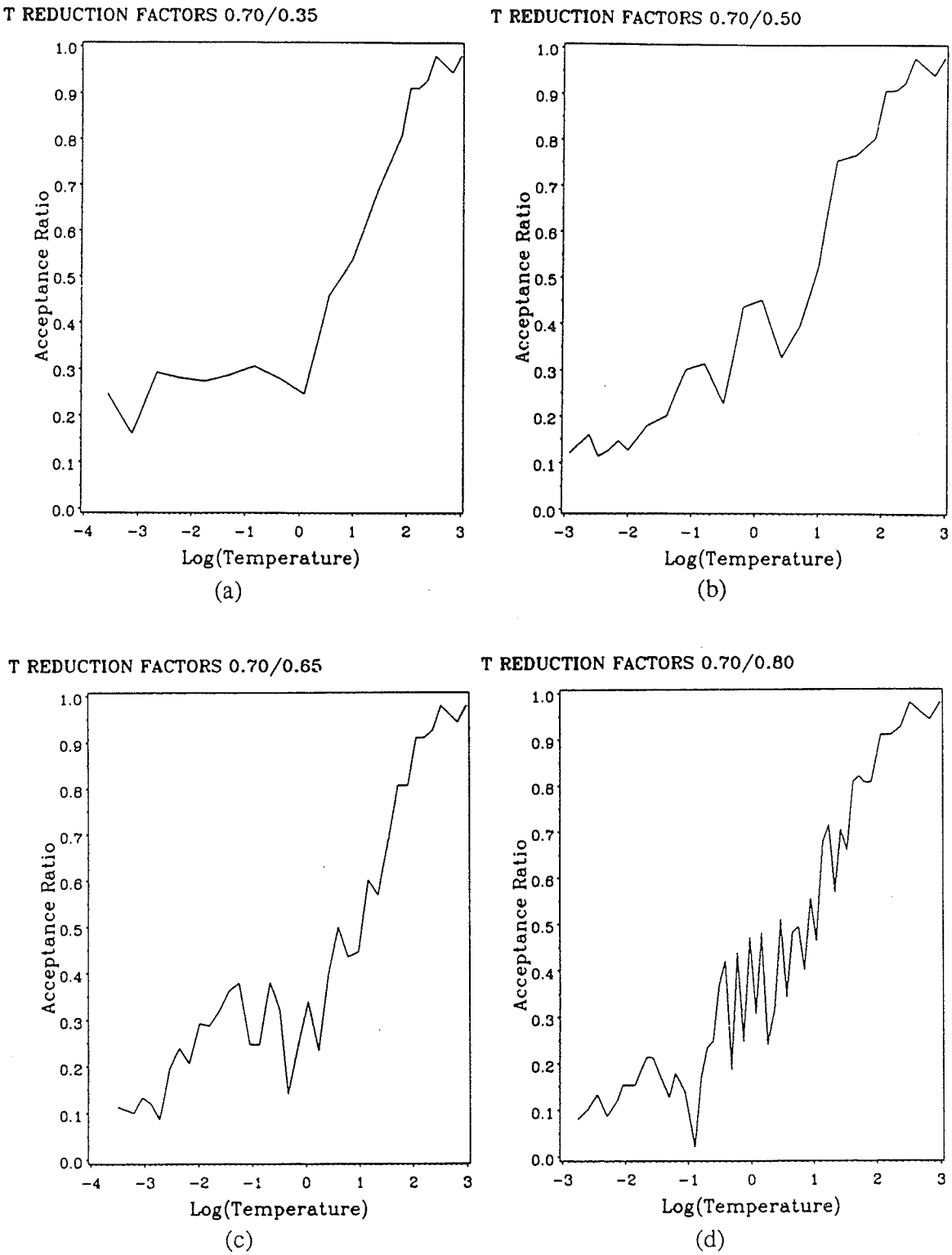


Figure B5: Effect of varying temperature reduction factor on α

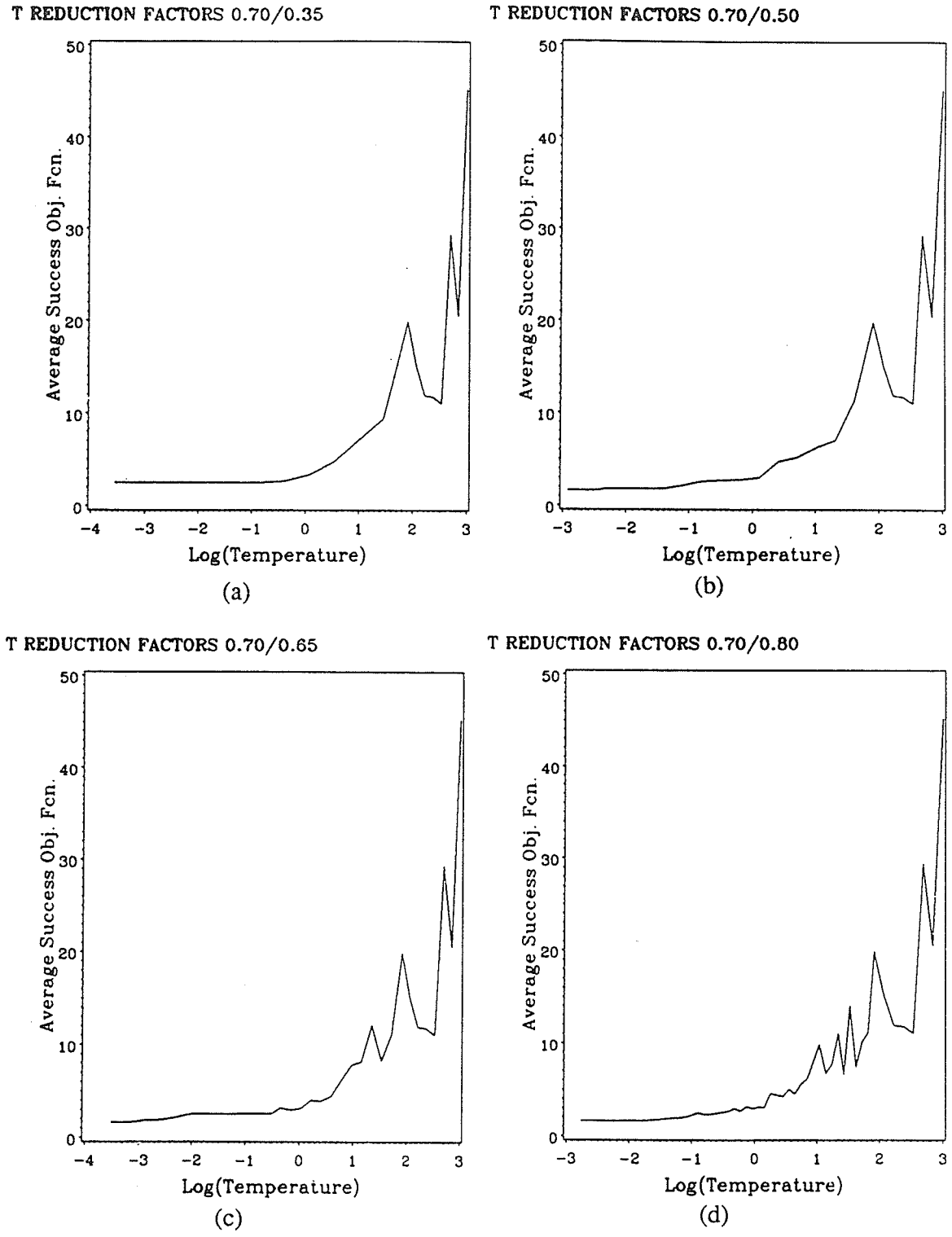


Figure B6: Effect of varying temperature reduction factor on $\overline{F}_{acc}$

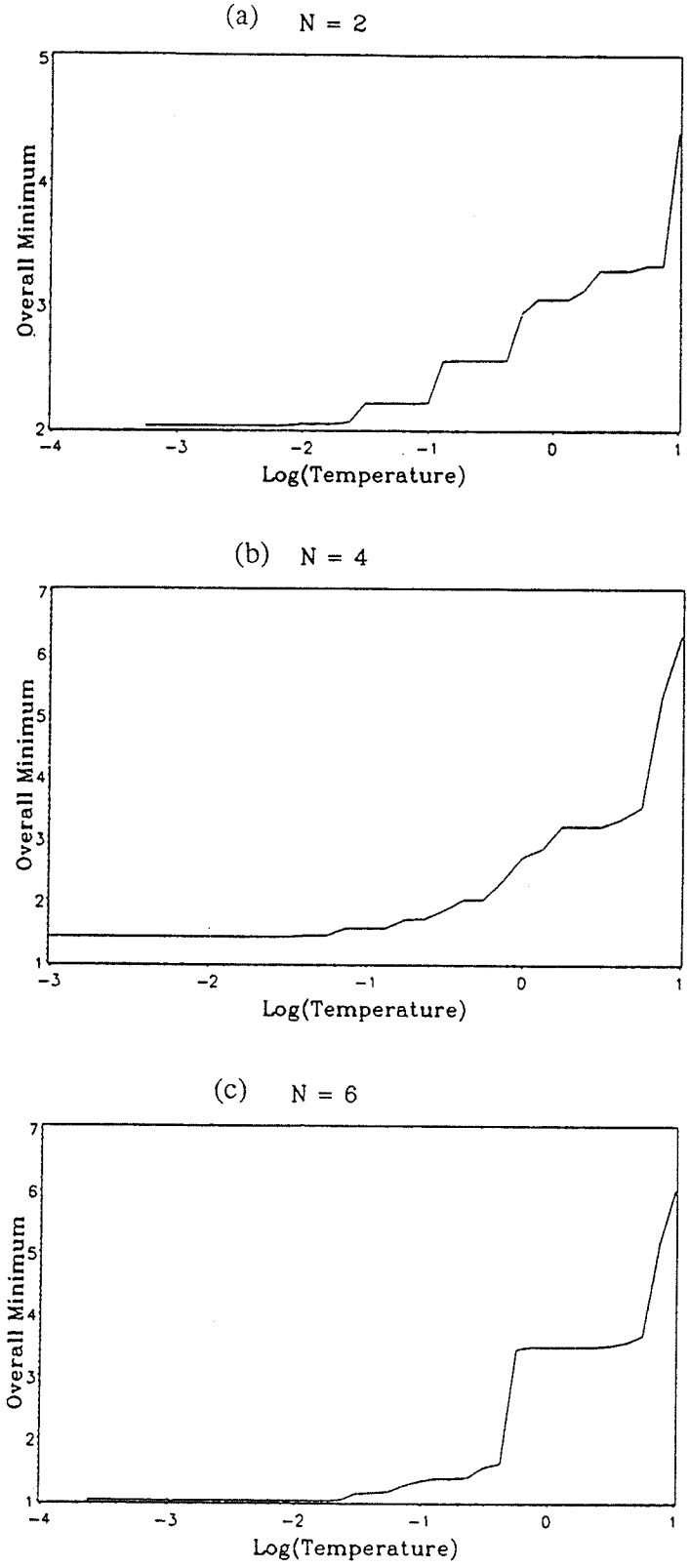
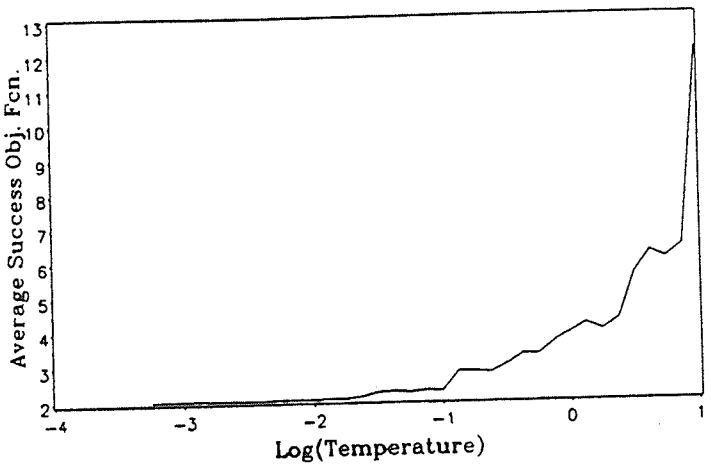
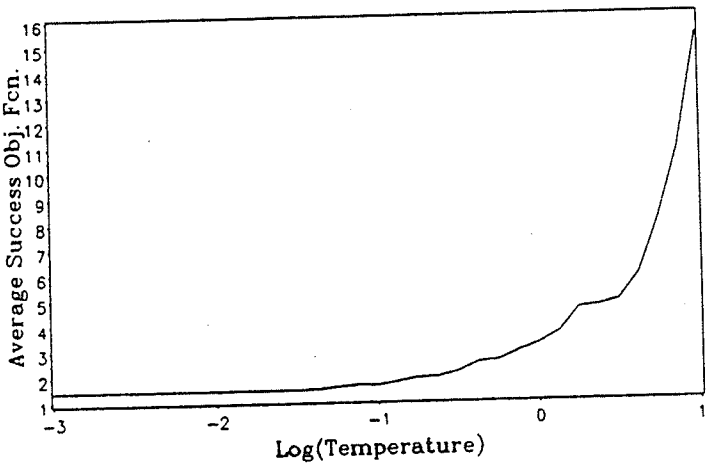


Figure B7: Effect of varying n on $F_{\min}$

(a) N = 2



(b) N = 4



(c) N = 6

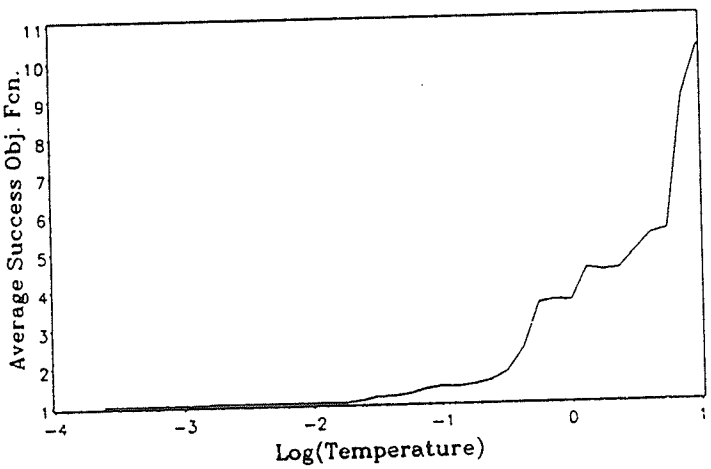


Figure B8: Effect of varying n on $\overline{F}_{acc}$

APPENDIX C

Program Listings

This appendix shows only the source code of the annealing subroutine depicted in Figure 3.11.

```
#include 'global.ins.h';
#include 'annealsubs.h';
#include 'annealing3.h';
#include 'loadflow.h'  ;
#include 'printsubs.h' ;
#include 'syssubs.h'   ;
#include 'netsubs.h'   ;
#include 'mathsubs.h'  ;

{=========================================================================}
{== external routine declarations                                       ==}
{=========================================================================}
{ 1. procedure cpu_pas ( var seconds : real ) ; external ;
  2. function  random ( dummy : integer ) : real ; external ;
  3. procedure store ( root:node_pointer; var brch_array:min_brch_type );
                     external ;
  4. procedure rebuild ; external ;
  5. function  ysno(msg:pack80; key:integer): boolean; external ; }
{=========================================================================}

procedure cal_obj;

{ This function returns an array of sub_obj. fcn of an open tree by
  finding the maximum of square of load percent, the maximum voltage
   penalty constraints.  It also sums the real and reactive losses of
    the tree. }

   var
      count    : 0..max_node_brchs ;       { branches counter          }
      p_count  : num_phases;               { phase counter             }
      cur_bran : brch_pointer;             { current branch examined   }
      cur_mag  : real;                     { current magnitude         }
      ireal    : real;                     { real current              }
      iimag    : real;                     { imaginary current         }
      vmag     : real;                     { voltage magnitude         }
      vtemp    : real;                     { used in voltage penalt cal }
      tioa2    : real;                     { present branch (i/a) index }

   begin

   for count := 1 to bus^.bac_ptr^.n_phases do
      begin
      p_count := bus^.bac_ptr^.i_phases[count];
      rect_to_polar(bus^.voltage[p_count],vmag,vtemp);
      v_fact := bus^.system_voltage / 120;
      vtemp := (v_upper - (vmag/v_fact) )/(v_upper - v_lower ) ;
      if vtemp > tobj[4] then tobj[4] := vtemp
      end ;

   for count := 1 to bus^.brannum do     { tobj[1] = (i/a) index       }
      begin                              { tobj[2] = r. losses obj fcn }
      cur_bran := bus^.for_ptr[count] ;  { tobj[3] = i.   "     "   "  }
      with cur_bran^ do                  { tobj[4] = voltage penalty   }
         for p_count := a to c do
            begin
            ireal := current[p_count].r ;
            iimag := current[p_count].i ;
            cur_mag := sqrt( ireal*ireal + iimag*iimag ) ;
            percent_ld[p_count] := cur_mag/ampacity ;
            tioa2   := percent_ld[p_count] * percent_ld[p_count] ;
            if tioa2 > tobj[1] then tobj[1] := tioa2 ;
            tobj[2] := tobj[2] + cur_mag*cur_mag*r1 ;
            tobj[3] := tobj[3] + cur_mag*cur_mag*x1
            end ;
      if cur_bran^.status >= 1 then        {closed}
         cal_obj(cur_bran^.for_ptr,tobj)
      end
   end ;


procedure anneal ;

{ performs the annealing routine by using the metropolis algorithm.   }
```

```
var
   niter      : integer;        { maximum # of iterations in each t  }
   nlimit     : integer;        { maximum # of success moves in t    }
   temp       : real;           { temperature at this iteration      }
   num1       : integer;        { number in the open branch array    }
   num2       : integer;        { number in the closed branch array  }
   branch1    : brch_pointer;   { the current open branch            }
   branch2    : brch_pointer;   { the current closed branch          }
   nclose     : integer;        { number of closed branches          }
   x          : real;           { x     to initialize random         }
   dummy      : real;           { dummy to initialize random         }
   count      : integer;        { level 2 iteration counter          }
   count2     : integer;        { general counter                    }
   loop       : integer;        { loop counter                       }
   object     : real;           { the objective function             }
   oldobjmax  : real;           { old max objective function so far  }
   newobjmax  : real;           { new max objective function so far  }
   nmin       : integer;        { number of minimum moves            }
   nsucc      : integer;        { number of successful moves         }
   closebran  : open_sw_type;   { array of closed branches pointer   }
   contloop1  : boolean;        { continue executing loop 1 ?        }

   bank1_obj  : obj_array;      { objective array for bank 1         }
   bank2_obj  : obj_array;      { objective array for bank 2         }
   temp_obj   : obj_array;      { temporary objective array          }

   iter1      : integer;        { level 1 iteration counter          }
   dobj       : real;           { difference in objective function   }
   told       : real ;          { old internal cpu time counter      }
   tpresent   : real ;          { present cpu time counter           }
   cputime    : real ;          { cpu time used in level 1 iteration }
   success    : boolean;        { successful moves or not            }
   obj        : obj_matrix;     { store all components of obj. fcn   }
   prev_obj   : obj_matrix;     { store previous values of var "obj" }
   objmat     : obj_matrix;     { store minimum obj fcn matrix       }
   losses     : complex_type;   { losses of the system               }
   scale      : obj_array;      { scaling factor for obj function    }
   root1_num  : banks_range;    { number of root1 in bank array      }
   root2_num  : banks_range;    { number of root2 in bank array      }
   sum1       : real;           { objective sum of all moves         }
   sum2       : real;           { objective sum of all success moves }

   root1      : node_pointer;   { one of the root nodes              }
   root2      : node_pointer;   { another root node                  }
   tinit      : real;           { initial temperature for annealing  }
   tfact      : real;           { reduction of temperature           }
   texit      : real;           { exit temperature                   }
   maxiter    : integer;        { maximum of iteration at t levels   }
   end_changes: boolean;        { finished making changes ?          }
   itermin    : real ;          { minimum obj. fcn. at that iteration}
   eq0,ls0,mr0:integer ;        { success from =0 or <0 or >0 moves  }
   minopenbran : open_sw_type;{ minimum open branch array for rebui}
   succ_ratio : real ;          { ratio of success and attempts      }
   succ_e : real ;              { average energy of successful moves }
   newavg_e : real ;            { new average of objective function  }
   oldavg_e : real ;            { previous aver. of obj. function    }
   old_tfact : real ;           { previous temperature reduct. factor}
   spec_heat : real ;           { specific heat dE/dT * important *  }
   anneal_stats : array[1..250,1..6] of real ; { store anneal stats}
   conv_ratio : real ;
   exit_ratio : real ;

procedure set_parameters ;

   var
      i : integer ;

   begin
   tinit := 100.0 ;
   tfact := 0.80 ;
   texit := 0.001 ;
   maxiter := 23 ;
   exit_ratio := 0.0001 ;
   capmax := 1.0 ;
   sw_mode := plan ;
```

```
      scale[1] := 1.0 ;                    { scale on (i/a)**2 = 1.0     }
      for i := 2 to max_num_obj do
         scale[i] := 0.0 ;
      scale[5] := 1.0                      { scale on bank load = 1.0    }
      end ; { proc set_parameters }

procedure print_change_menu ;

{ prints out the change menu }

var
   choice , newseed , oldseed : integer ;

begin
writeln('Enter changes that you want to make:');
writeln;
writeln(' 1. Initial Acceptance             2. Reduction Factor');
writeln(' 3. Exit Acceptance                4. Maximum # of iterations');
writeln(' 5. Scale on line overloading      6. Scale on real power loss');
writeln(' 7. Scale on reactive power losses 8. Scale on voltage constraint');
writeln(' 9. Scale on trnsf. overloading   10. Optimize option (brch or sw)');
writeln('11. Change seed for RANDOM        12. Exit ratio');
writeln('13. Quit');
write('==>'); readln(choice);
case choice of
   1 : begin
       write('Enter new initial temperature (Default ',tinit:5:2,') ==> ');
       readln(tinit)
       end ;
   2 : begin
       write('Enter new reduction factor (Default ',tfact:5:2,') ==> ');
       readln(tfact)
       end ;
   3 : begin
       write('Enter new exit temperature (Default ',texit:7:5,') ==> ');
       readln(texit)
       end ;
   4 : begin
       write('Enter new max. # of iterations (Default ',maxiter:2,') ==> ');
       readln(maxiter)
       end ;
   5 : begin
       write('Enter new scale on (i/a)**2 (Default ',scale[1]:5:2,') ==> ');
       readln(scale[1])
       end ;
   6 : begin
       write('Enter new scale on Real power losses (Default ',
             scale[2]:5:2,') ==> ');
       readln(scale[2])
       end ;
   7 : begin
       write('Enter new scale on Reactive power losses (Default ',
             scale[3]:5:2,') ==> ');
       readln(scale[3])
       end ;
   8 : begin
       write('Enter new scale on voltage constraint(Default ',
             scale[4]:5:2,') ==> ');
       readln(scale[4])
       end ;
   9 : begin
       write('Enter new scale on bank load (Default ',
             scale[5]:5:2,') ==> ');
       readln(scale[5])
       end ;
  10 : begin
       msg[1] := ' ';
       write('Change optimizer (branches to switches only) ==>');
       if ysno(msg,a_dno) then
          sw_mode := operate
       end ;
  11 : begin
       oldseed := seed(7774755) ;
       write('Enter new seed for random generator (Default ',
             oldseed:8,') ==> ');
```

```
      readln(newseed) ;
      oldseed := seed(newseed)
      end ;
 12 : begin
      write('Enter new exit ratio (Default ',
            exit_ratio:9:6,') ==> ');
      readln(exit_ratio)
      end ;
 13 : end_changes := true  { do nothing }
 end
end ; { proc print_change_menu }

  procedure change_parameters ;

     begin
     write('Change any parameters (default=no)');
     msg[1] := ' ' ;
     if ysno(msg,a_dno) then
        while not end_changes do
           print_change_menu
     end ; { proc change_parameters }

  function find_max_obj( obj:obj_matrix ): real ;

  { This finds out the maximum objective in the objective matrix }

  var
     bank_count : integer ;
     temp_array : obj_array ;

  begin
  for bank_count := 1 to max_num_obj do
     temp_array[bank_count] := zero ;
  for bank_count := 1 to last_bank do
     begin
     if obj[bank_count,1] > temp_array[1] then
        temp_array[1] := obj[bank_count,1] ;
     if obj[bank_count,4] > temp_array[4] then
        temp_array[4] := obj[bank_count,4] ;
     if obj[bank_count,5] > temp_array[5] then
        temp_array[5] := obj[bank_count,5]
     end ;
  { ****** objective function calculation ****** }
  find_max_obj := temp_array[1] * temp_array[1] +
                  temp_array[4] * temp_array[4] +
                  temp_array[5] * temp_array[5]
  end ; { proc find_max_obj }

  function log( x : real ) : real ;

  { This returns the log (base 10) of x }

  begin
  log := ln(x) / 2.302585093
  end ; { func log }

  procedure store_anneal_stats ;

  { This stores the annealing statistics into an array }

  begin
  anneal_stats[iter1,1] := log(temp) ;
  anneal_stats[iter1,2] := succ_ratio ;
  anneal_stats[iter1,3] := newavg_e ;
  anneal_stats[iter1,4] := succ_e ;
  anneal_stats[iter1,5] := spec_heat ;
  anneal_stats[iter1,6] := itermin
  end ; { proc store_stats }

  procedure dump_anneal_stats ;

  { This dumps the annealing statistics to file 'anneal.stats' }

  var i,j : integer ;
```

```
begin
rewrite(statfile,statfile_name) ;
for i := 1 to iter1 do
   begin
   for j := 1 to 6 do
      write(statfile,anneal_stats[i,j]:11:6) ;
   writeln(statfile)
   end
end ; { proc dump_anneal_stats }


begin  { proc anneal }
cpu_pas(told);

set_parameters ;                { initialize annealing parameters  }
end_changes := false ;
change_parameters ;             { change parameter option          }

writeln;
writeln;
writeln('-----------------------------------------');
writeln(' S t a r t i n g   O p t i m i z a t i o n');
writeln('-----------------------------------------');
writeln;


temp := tinit ;
dummy := random(x) ;
writeln('Number of open switches : ',last_open_sw) ;
niter := 150 ; { 16 * last_open_sw ;   change from 12 }
nlimit := 50 ; { 10 * last_open_sw ;   change from 3  }
object := zero ;                { calculate the objective function }
losses.r := zero ;
losses.i := zero ;
clear_subobj(temp_obj);
writeln('** Objective calculation of') ;
writeln(outfile,'Bank  (i/a)**2  bankcap   v.penal   r.losses  i.losses');
writeln(outfile,'-------------------------------------------------------');
for count := 1 to last_bank do
   begin
   for count2 := 1 to max_num_obj do
      obj[count,count2] := zero ;
   clear_subobj(bank1_obj);
   cal_obj(root_list[count],bank1_obj);
   find_stn_capac(count,bank1_obj);
   assign_obj(obj,bank1_obj,count);
   write(outfile,' ',count:3);
   write(outfile,'  ',bank1_obj[1]:8:4) ;  { ampacity }
   write(outfile,'  ',bank1_obj[5]:8:4) ;  { bank capacity }
   write(outfile,'  ',bank1_obj[4]:8:4) ;  { voltage constraint }
   write(outfile,'  ',bank1_obj[2]/1000:8:3) ;
   write(outfile,'  ',bank1_obj[3]/1000:8:3) ;
   writeln(outfile) ;
   if bank1_obj[1] > temp_obj[1] then
      temp_obj[1] := bank1_obj[1] ;
   if bank1_obj[4] > temp_obj[4] then
      temp_obj[4] := bank1_obj[4] ;
   if bank1_obj[5] > temp_obj[5] then
      temp_obj[5] := bank1_obj[5] ;
   losses.r := losses.r + bank1_obj[2];
   losses.i := losses.i + bank1_obj[3]
   end ;

report_bank_stats ;
report_feeder_stats ;

{ ************************************************************************ }
scale[2] := scale[2] * 10 / losses.r ; { five sub-objective functions of  }
scale[3] := scale[3] * 10 / losses.i ; { the main objective function.     }
{ ****** objective function calculation ***** }
object := temp_obj[1] * temp_obj[1]
        + temp_obj[4] * temp_obj[4]
        + temp_obj[5] * temp_obj[5];
```

```
   objmat := obj ;
   writeln;
   writeln('objective function = ',object:10:7) ; writeln;
   writeln(outfile,'objective function = ',object:10:7) ;
   writeln(outfile,'max. (current/ampacity)**2 = ',temp_obj[1]:10:7);
   writeln(outfile,'total losses = ',losses.r/1000:8:2,'kw + j',
                   losses.i/1000:8:2,'kvar.');
   writeln(outfile,'max. voltage constraint = ',temp_obj[4]:8:3);
   writeln(outfile,'max. bank load (in %)   = ',temp_obj[5]:8:3);
   writeln(outfile) ;
{  printdata(outfile,in_brch,full); }
   oldobjmax := object ;
   minopenbran := openbran ;
   iter1 := zero ;
   contloop1 := true ;
   while contloop1 do                                          { level 1 loop }
      begin
      count := zero ;
      nsucc := zero ;
      sum1:= zero ;
      sum2:= zero ;
      nmin  := zero ;
      iter1 := iter1 + 1 ;

      itermin := object ;
      eq0 := zero ;
      ls0 := zero ;
      mr0 := zero ;
      {max_trace := 2 + round(100*(1-exp(-temp)));}
      writeln('** start of iteration ',iter1:3); writeln;
      while (count < niter) and (nsucc < nlimit) do     { level 2 loop }
         begin
         nclose := 0 ;
         while nclose = 0 do { if can not find any switch to close }
            begin
            dummy := random(x) ; { change from 0 }
            num1 := 1 + trunc( last_open_sw * dummy ) ;
            branch1 := openbran[num1] ;
            trace_to_subs(branch1,closebran,nclose,root1,root2)
            end ;
         dummy := random(x) ; { change from 0 }
         num2 := 1 + trunc( nclose * dummy ) ;
         branch2 := closebran[num2] ;
         root1_num := find_bank(root1) ;
         if root1 <> root2 then
            root2_num := find_bank(root2) ;
         setup(branch1,branch2) ;
         rearrange(branch1,branch2) ;
         load_flow(root1) ;
         clear_subobj(bank1_obj);
         cal_obj(root1,bank1_obj);
         find_stn_capac(root1_num,bank1_obj);


         prev_obj := obj ;          { store old matrix if not success }
         assign_obj(obj,bank1_obj,root1_num) ; { assign bank 1 in obj }
         if root1 <> root2 then
            begin
            load_flow(root2) ;
            clear_subobj(bank2_obj);
            cal_obj(root2,bank2_obj);
            find_stn_capac(root2_num,bank2_obj);
            end ;

         object := find_max_obj(obj) ;
         sum1 := sum1 + object ;
         dobj := object - oldobjmax ;
         success := metrop(dobj,temp) ;
         if success then
            begin

            { keep track of what kind of success moves }

            if dobj = 0.0 then
               eq0 := eq0 + 1
```

```
            else if dobj < 0.0 then
               ls0 := ls0 + 1
            else mr0 := mr0 + 1 ;

            nsucc := nsucc + 1 ;

            newobjmax := object ;
            sum2 := sum2 + newobjmax ;        { newobjmax as above    }
            openbran[num1] := branch2 ;
            if newobjmax < itermin then
               begin
               itermin := newobjmax ;
               nmin := nmin + 1 ;
               oldobjmax := newobjmax ;
               objmat := obj ;
               writeln('**** New maximum objective function (',
                        (count+1):3,') = ',newobjmax:9:5);
               for loop := 1 to last_bank do
                  store(bank[loop].source_pointer,out_brch) ;
               minopenbran := openbran
               end
            end
         else
            begin
            reverseb(branch2) ;
            rearrange(branch2,branch1);
            obj := prev_obj
            end ;
         count := count + 1
         end ; { level 2 loop }
      if nmin <> 0 then
        begin
    writeln(outfile,'bank  (i/a)**2  bankindx  vdroppen r.losses  i.losses');
    writeln(outfile,'-------------------------------------------------------');
        for loop := 1 to last_bank do
           begin
           write(outfile,' ',loop:3);
           write(outfile,'  ',objmat[loop,1]:8:5) ;
           write(outfile,'  ',objmat[loop,5]:8:5) ;
           write(outfile,'  ',objmat[loop,4]:8:5) ;
           write(outfile,'  ',objmat[loop,2]/1000:8:3) ;
           write(outfile,'  ',objmat[loop,3]/1000:8:3) ;
           writeln(outfile) ;
           end ;
         end ;
      cpu_pas(tpresent) ;
      cputime := tpresent - told ;
      told := tpresent ;
      write(outfile,'Iteration =',iter1:2) ;
      write(outfile,', T =',temp:9:6) ;

      succ_ratio := nsucc / count ;

      writeln(outfile,', CPU used =',cputime:8:3,'s') ;

      if nsucc <> zero then { defined new avg energy as avg. succ energy }
         succ_e := sum2 / nsucc
      else succ_e := 0.0 ;
      newavg_e := sum1 / count ;
      if iter1 <> 1 then { skip first iteration }
         begin
         {spec_heat := (newavg_e - oldavg_e) / (temp - (temp/old_tfact)) ;}
         spec_heat := (succ_e - oldavg_e) / log(old_tfact) ;
         writeln(outfile,'Specific Heat = ',spec_heat:9:5)
         end
      else spec_heat := 0.0 ;
      oldavg_e := succ_e ; { newavg_e ; }
      old_tfact := tfact ;

      write(outfile,'Min. at this iter =',itermin:9:6) ;
      writeln(outfile,', Min. objective = ',oldobjmax:9:6) ;
      write(outfile,'<E> =',newavg_e:9:6) ;
      if nsucc <> zero then
         writeln(outfile,', <Esuccess> =',succ_e:9:6)
      else
```

```
        writeln ;
      writeln(outfile,'#Min/#Succ/Attempts =',nmin:3,'/',nsucc:3,'/',count:4);
      writeln(outfile,'#Succ due </=/> 0.0 =',ls0:3,'/',eq0:3,'/',mr0:4);
      writeln(outfile) ;
      store_anneal_stats ;
      temp := temp * tfact ;

      rebuild ;                  { rebuild the network from out-branch }
      object := itermin ;        { use the previous minimum as obj. fcn. }
      openbran := minopenbran ; { store minimum open branch array }
      obj := objmat ;            { store minimum obj. matrix as starting obj }

      tpresent := tstart - tpresent ; { use tpresent to test exit criteria }
      conv_ratio := (succ_e - itermin) / itermin ;
      if (nsucc = zero) or (temp < texit) or
         (conv_ratio < exit_ratio) or (iter1 >= maxiter) then
         contloop1 := false
      end ; { while contloop1 }

   clear_subobj(bank1_obj);
   for count := 1 to last_bank do
      begin
      load_flow(root_list[count]) ;                { recalulate current     }
      sum_i_to_src(root_list[count]);              { sum current to source  }
      cal_obj(root_list[count],bank1_obj);         { calculate % load       }
      end ;
   report_bank_stats ;
   report_feeder_stats ;
   printdata(outfile,out_brch,quick);
   dump_anneal_stats
   end ; { procedure anneal }

{=============================================================================}
```

REFERENCES

[1] A. J. Pansini, *Electrical Distribution Engineering,* McGraw-Hill, 1983.

[2] E. Masud, "An Interactive Procedure for Sizing and Timing Distribution Substations Using Optimization Techniques," *IEEE Trans. PAS,* vol. PAS-93, no. 5, pp. 1281-1286, September/October 1974.

[3] D. M. Crawford and S. B. Holt, Jr., "A Mathematical Optimization Technique for Locating and Sizing Distribution Substations, and Deriving Their Optimal Service," *IEEE Trans. PAS,* vol. PAS-94, no. 2, pp. 230-235, March/April 1975.

[4] D. L. Wall, G. L. Thompson and J. E. D. Northcote-Green, "An Optimization Model for Planning Radial Distribution Networks," *IEEE Trans. PAS,* vol. PAS-98, no. 3, pp. 1061-1068, May/June 1979.

[5] G. L. Thompson and D. L. Wall, "A Branch and Bound Model for Choosing Optimal Substation Locations," *IEEE Trans. PAS,* vol. PAS-100, no. 5, pp. 2683-2687, May 1981.

[6] T. H. Fawzi, K. F. Ali and S. M. El-Sobki, "A New Planning Model for Distribution Systems," *IEEE Trans. PAS,* vol. PAS-102, no. 9, pp. 3010-3017, September 1983.

[7] D. I. Sun, D. R. Farris, P. J. Cote, R. R. Shoults and M. S. Chen, "Optimal Distribution Substation and Primary Feeder Planning via the Fixed Charge Network Formulation," *IEEE Trans. PAS,* vol. PAS-98, no. 3, pp. 602-609, March 1982.

[8] T. Gonen and B. L. Foote, "Distribution System Planning Using Mixed Integer Programming," *Proc IEE,* vol. 128, Pt. C, no. 2, pp. 70-79, March 1981.

[9] M. A. El-Kady, "Computer Aided Planning of Distribution Substations and Primary Feeders," *IEEE Trans. PAS,* vol. PAS-103, no. 6, pp. 1183-1189, June 1984.

[10] T. Gonen and I. Ramirez-Rosado, "Review of Distribution System Planning Models: A Model for Optimal Multistage Planning," *Proc IEE,* vol. 133, Pt. C, no. 7, pp. 397-408, November 1986.

[11] H. K. Youssef, R. Hackam and M. A. Abu-El-Magd, "Novel Optimization Model for Long Range Distribution Planning," *IEEE Trans. PAS,* vol. PAS-104, no. 11, pp. 3195-3202, November 1985.

[12] M. Ponnavaikko and K. S. Prakasa Rao, "An Approach to Optimal Distribution System Planning through Conductor Gradation," *IEEE Trans. PAS,* vol. PAS-101, no. 6, pp. 1735-1742, June 1982.

[13] H. L. Willis, R. W. Powell and H. N. Tram, "Long-Range Distribution with Load Forecast Uncertainty," *IEEE Trans. PS,* vol. PWRS-2, no. 3, pp. 684-691, August 1987.

[14] H. L. Willis, R. W. Powell and H. N. Tram, "Substation Siting and Capacity Selection Based on Diversity Maximization," *IEEE Trans. PS,* vol. PWRS-2, no. 3, pp. 692-699, August 1987.

[15] H. N. Tram and D. L. Wall, "Optimal Conductor Selection in Planning Radial Distribution System," *IEEE Trans. PS,* vol. PWRS-3, no. 1, pp. 200-206, February 1988.

[16] K. Aoki, T. Ichimori and M. Kanezashi, "Normal State Optimal Load Allocation in Distribution Systems," *IEEE Trans PD,* vol. PWRD-2, no. 1, pp. 147-155, January 1987.

[17] K. Aoki, H. Kuwabara, T. Satoh and M. Kanezashi, "An Efficient Algorithms for Load Balancing of Transformers and Feeders by Switch Operation in Large Scale Distribution Systems," Paper No. 87 SM 543-2, IEEE PES Summer Meeting, San Francisco, CA., July 1987.

[18] P. J. M. van Laarhoven and E. H. Aarts, *Simulated Annealing: Theory and Applications,* D. Reidel Publishing Company, Dordrecht, Holland, 1987.

[19] E. Lawler, J. Lenstra, A. Kan and D. Schmoys, *The Traveling Salesman Problem,* John Wiley & Sons, 1986.

[20] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science,* vol. 220, no. 4598, pp. 671-680, May 1983.

[21] V. Cerny, "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm," *J. Opt. Theory Appl.,* vol. 45, pp. 41-51, 1985.

[22] A. V. Fiacco and G. P. McCormick, *Non-linear Programming:Sequential Unconstrainted Minimization Techniques,* John Wiley, New York, 1968.

[23] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids,* Holt, Rinehart and Winston, 1976.

[24] E. Bonomi and J. Lutton, "The N-city Traveling Salesman Problem: Statistical Mechanics and the Metropolis Algorithm," *SIAM Review,* vol. 26, no. 4, pp. 551-568, October 1984.

[25] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman and Company, 1979.

[26] J. Pearl, "Some Recent Results in Heuristic Search Theory," *IEEE Trans on Pattern Anal. Machine Intell.,* vol. PAMI-6, no. 1, pp. 1-12, January 1984.

[27] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity,* Prentice-Hall, 1982.

[28] J. Hopfield and D. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics,* vol. 52, pp. 147-152, 1985.

[29] E. Baum, "Intractable Computation without Local Minima," *Phys. Rev. Lett.,* vol. 57, no. 21, pp. 2764-2767, November 24 1986.

[30] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth and A. H. Teller, "Equation of State Calculations by Fast Computing Machines," *J. Chem. Phys.,* vol. 21, no. 6, pp. 1087-1092, June 1953.

[31] E. H. L. Aarts and P. J. M. van Laarhoven, "A New Polynomial-Time Cooling Schedule," Proc. IEEE Int. Conference on Computer-Aided Design, Santa Clara, pp. 206-208, November 1985.

[32] S. R. White, "Concepts of Scale in Simulated Annealing," Proc IEEE Int. Conference on Computer Design, Port Chester, NY., pp. 646-651, November 1984.

[33] S. A. Kravitz and R. A. Rutenbar, "Placement by Simulated Annealing on a Multiprocessor," *IEEE Trans. CAD,* vol. CAD-6, no. 4, pp. 534-549, July 1987.

[34] A. Casotto, F. Romeo and A. Sangiovanni-Vincentelli, "A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells," *IEEE Trans. CAD,* vol. CAD-6, no. 5, pp. 838-847, September 1987.

[35] J. W. Greene and K. J. Supowit, "Simulated Annealing Without Rejected Moves," *IEEE Trans. CAD,* vol. CAD-5, no. 1, pp. 221-228, January 1986.

[36] M. P. Vecchi and S. Kirkpatrick, "Global Wiring by Simulated Annealing," *IEEE Trans. CAD,* vol. CAD-2, no. 4, pp. 215-222, October 1983.

[37] F. Romeo, A. Sangiovanni Vincentelli and C. Sechen, "Research on Simulated Annealing at Berkeley," Proc of IEEE Int. Conference on Computer Design:VLSI

in Computers, Port Chester, NY., pp. 652-657, 1984.

[38] J. G. Gay and R. Richter, "Component Placement in VLSI Circuits Using a Constant Pressure Monte Carlo Method," *Integration,* vol. 3, pp. 271-282, 1985.

[39] P. Siarry, L. Bergonzi and G. Dreyfus, "Thermodynamic Optimization of Block Placement," *IEEE Trans. CAD,* vol. CAD-6, no. 2, pp. 211-221, March 1987.

[40] S. Geman and D. Geman, "Stochastic Relaxation, Gribbs Distributions, and Bayersian Restoration of Images," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PARM-6, no. 6, pp. 721-741, November 1983.

[41] M. Lundy, "Applications of the Annealing Algorithm to Combinatorial Problems in Statistics," *Biometrika,* vol. 72, pp. 191-198, 1985.

[42] M. Ilic'-Spong, J. Christensen and K. L. Eichorn, "Secondary Voltage Control Using Pilot Plant Information," 1987 Power Industry Computer Applications Conference, Montreal, Quebec, Canada, pp. 128-135, May 1987.

[43] D. Shirmohammadi, H. Hong, A. Semlyen and G. Luo, "A Compensation-Based Power Flow Model for Weakly Meshed Distribution and Transmission Networks," 1987 Power Industry Computer-Applications Conference, Montreal, Quebec, Canada, pp. 370-376, May 1987.

[44] S. Nahar, S. Sahni and E. Shragowitz, "Experiments with Simulated Annealing," Proc 22nd Des. Automation Conference, Las Vegas, pp. 293-299, June 1985.

[45] J. Bate, M. Doyle and H. Ferch, *Computer Programming and Data Structures using Pascal,* The Charles Babbage Research Centre, 1982.