7
System Optimization

Optimal, optimize and optimization are terms prone to misuse. Though the concept of optimization and the techniques to achieve it are powerful, few people truly understand their implications and are able to apply them in a rigorous and systematic manner. System optimization has always been a favorite topic in academia, but until recently has had little applicability to distribution systems. With the exponential increase in computation power, applying optimization techniques to practical distribution reliability problems is becoming feasible, and a deeper understanding of optimization is warranted.

What does it imply when an engineer claims to have the optimal solution? Is it reasonable for top management to request a solution that optimizes both reliability and cost? Do the members of a distribution optimization team truly understand their mandate? The goal of this chapter is to allow the reader to understand and answer such questions. More importantly, it will provide a theoretical foundation so that optimization techniques can be properly applied to distribution systems and increase the probability of providing higher reliability for lower cost.

7.1 OVERVIEW OF OPTIMIZATION

The goal of system optimization is to minimize an objective function without violating any constraints. To do this, of course, an appropriate objective function must be defined and all applicable constraints must be identified. In addition, the objective function must be computable and all constraints must be testable for all

possible solutions, which are characterized by a set of controllable variables. Mathematically, an optimization problem is expressed as follows:

Optimization Problem Formulation (7.1)

Minimize:

$$\Omega(x_1, x_2, \ldots, x_L) \qquad \text{Objective Function}$$

Subject to:

$$\begin{aligned} C_1(x_1, x_2, \ldots, x_L) &= Y_1 \qquad \text{Equality Constraints} \\ C_2(x_1, x_2, \ldots, x_L) &= Y_2 \\ &\vdots \\ C_M(x_1, x_2, \ldots, x_L) &= Y_M \\ C_{M+1}(x_1, x_2, \ldots, x_L) &\le Y_{M+1} \qquad \text{Inequality Constraints} \\ C_{M+2}(x_1, x_2, \ldots, x_L) &\le Y_{M+2} \\ &\vdots \\ C_N(x_1, x_2, \ldots, x_L) &\le Y_N \end{aligned}$$

L = Number of Optimizable Variables
M = Number of Equality Constraints
N = Total Number of Constraints

From this formulation, it can be seen that the objective of an optimization problem is to identify a feasible set of variables that minimizes an objective function. The objective function can be any combination of formulae, algorithm or process as long as it returns a single value for each possible combination of parameters. Objective functions may also be formulated to be maximized, but distribution reliability optimization typically deals with cost and interruptions, which are generally more desirable as they decrease in magnitude. Feasible solutions are defined as solutions that satisfy all equality constraints and do not violate any inequality constraints.

For distribution system reliability, optimization problems generally take one of two forms. The first is to minimize cost while satisfying all reliability constraints. For example, minimize the net present value of reliability improvement projects while achieving a $MAIFI_E$ of 10 /yr, a SAIFI of 1 /yr and a SAIDI of 60 min/yr. The second is to minimize customer interruptions subject to cost constraints. For example, minimize SAIDI without exceeding an annualized project cost of $5 million per year.

The same factors that appear in the objective function can also appear in constraints. For example, if the goal is to minimize the net present value of reliability improvement projects while achieving specific $MAIFI_E$, SAIFI and SAIDI targets, there could also be a stipulation that net present value cannot exceed a certain budget limit. In this situation, it becomes much more likely that there will be no combination of optimization variables that will satisfy all con-

straints. If it is impossible to satisfy all constraints, the problem is not well-formulated and there is no feasible solution to the optimization problem.

Equality and inequality constraints, in general, make an optimization problem more difficult to solve. Even if they do not make all solutions infeasible, they may make the state space of feasible solutions difficult to search. For these reasons, constraints can be temporarily relaxed while performing an optimization, allowing the state space of infeasible solutions to be explored with the ultimate goal of finding a better solution. Alternatively, constraints can be incorporated into the objective criteria as penalty factors. If a constraint is violated, a penalty is added to the objective function based on the degree that the constraint is violated. In some cases, relaxed constraints in the final solution may be acceptable and small violations can translate into small penalties. In other cases, all constraints must be satisfied and care must be taken so that all solutions that violate constraints have a net objective function (base objective function plus all penalty factors) that is higher than all solutions that do not violate any constraints.

To illustrate the concept of penalty factors, consider an optimization problem that seeks to minimize cost while achieving a SAIDI of 4 hr/yr without overloading any equipment. In this case, the SAIDI target is set by state regulators and is considered a hard constraint—values greater than 4 hr/yr are not acceptable. Equipment overloads, however, are a soft constraint since small overloads are not strictly prohibited. Since the cost values for feasible solutions are expected to be in the $10 million range, all SAIDI violations are assigned a penalty factor of $1 billion plus an additional amount based on the degree of the violations. Doing this ensures that any solution that violates the SAIDI constraint will have an objective function higher than any solution that satisfies the SAIDI constraint. Loading violations, however, are assigned a penalty factor in strict proportion to the degree of each overload, allowing solutions with small loading violations to have a net objective function lower than other solutions without any loading violations.

The purpose of this chapter is to address system optimization in a rigorous manner. Doing so requires the consistent use of precisely defined terms. Definitions of terms corresponding to optimization problem formulation include:

Optimizable Variables — a set of variables that describe potential solutions to an optimization problem. One or more combinations of optimizable variables corresponds to each feasible solution, and all other combinations correspond to infeasible solutions.

Solution Space — all possible combinations of optimizable variables. A solution space can be divided into a feasible solution space and an infeasible solution space, each of which may be non-contiguous.

Objective Function — a mathematical expression, sometimes referred to as a criterion, that indicates the relative quality of a potential solution corresponding to a combination of optimizable variables. It is standard practice to define objective functions so that lower values are more desirable than higher values.

Constraint — a mathematical expression that corresponds to the feasibility of a potential solution in one particular aspect. Constraints are typically categorized into equality constraints and inequality constraints, with inequality constraints typically being formulated so that feasible values are less than or equal to a threshold value. Constraints can also be grouped into hard constraints, which cannot be violated, and soft constraints, which may be violated under certain conditions.

Relaxed Constraint — a soft constraint that is violated on purpose, typically to facilitate solution space investigation or to identify potential solutions that otherwise have desirable objective function values.

Binding Constraint — an inequality constraint that is at its threshold value for a particular combination of optimizable variables.

Penalty Factor — A number that is added to the objective function when a constraint is allowed to be violated. Penalty factors can be constant or can vary based on the degree of the violation. Large penalty factors are typically used for constraints that cannot be relaxed, and small penalty factors are typically used for constraints that can be relaxed.

As stated previously, the goal of system optimization is to minimize an objective function without violating any constraints. It would seem, then, that the optimal solution would correspond to the combination of optimizable parameters in the feasible solution space that results in the lowest objective function value. It may surprise the reader to learn that this is not the case, and that a feasible solution space may contain many optimal solutions depending upon which type of optimality is being considered.

7.1.1 Locally Versus Globally Optimal

Consider a reliability problem that seeks to find the optimal value of a single variable x based on an objective function f(x). To further simplify the problem, assume that x must have a value between x_{min} and x_{max}, and that f(x) is a continuous function for all values of x between x_{min} and x_{max}. A conceptual representation of this problem is shown in Figure 7.1.

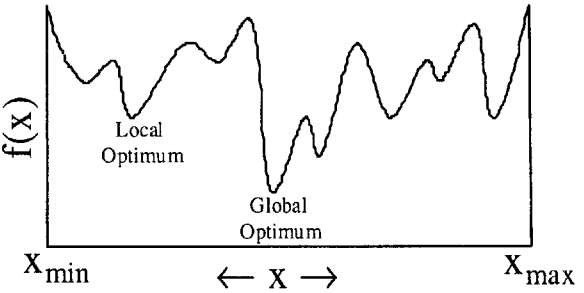


Figure 7.1. A conceptual representation of local and global optimality. Locally optimal solutions become worse for any small change in optimization variables. Globally optimal solutions have lower objective function values than all other feasible solutions. For large optimization problems such as distribution reliability, globally optimal solutions are not generally obtainable and locally optimal solutions must be pursued.

Now consider an optimization strategy that starts with $x=x_{min}$ and slowly increases x by infinitesimal values until it reaches x_{max}. After performing this exhaustive search of the solution space, the value of x that results in the lowest objective function is referred to as the globally optimal solution.

Global optimality is a fine concept, but is difficult to obtain for most reliability optimization problems since it is not typically feasible to test all possible solutions. Consider the impact of eliminating the x_{max} limit for the problem represented in Figure 7.1. As the value of x increases from x_{min}, it will periodically encounter troughs where increasing or decreasing x by a small amount will make the objective function worse. These points are referred to as local optima, and can be identified in twice differentiable functions by a zero first order derivative and a positive second order derivative. As the value of x increases towards infinity, the best locally optimal solution can be remembered, but there is no guarantee that higher values of x will not result in lower objective function values. Hence, most optimization algorithms can guarantee local optimality, but cannot guarantee global optimality. Precise definitions for these two types of optimality are:

Local Optimum — a solution that will produce a worse objective function value if any optimization variable is perturbed.

Global Optimum — the solution that produces the best objective function value when compared to all other solutions in the feasible solution space.

The study of optimization problems is referred to as mathematical programming. There are two primary types of optimization problems that can be solved with guaranteed global optimality. The first are problems that are small

enough for all possible solutions to be exhaustively tested. The second are problems that have a continuous and convex feasible solution space, such as linear programming problems and quadratic programming problems. A summary of the major classes of mathematical programming is:

Mathematical Programming — the study of problems that seek to optimize a function of variables subject to constraints. The term programming does not refer to computer programming, and was coined before the word became closely associated with computer software development.

Linear Programming — the study of optimization problems with a linear objective function and linear constraints. Linear programming problems have convex solution spaces, and well-known solution methods are generally able to guarantee global optimality.

Quadratic Programming — the study of optimization problems with a quadratic objective function and linear constraints. Quadratic programming problems have convex solution spaces, and well-known solution methods are generally able to guarantee global optimality but are more computationally intensive that those used for linear programming.

Non-Linear Programming — the study of optimization problems where at least one of the functions (objective or constraints) does not vary linearly with all optimization variables. Unless convexity of the solution space can be demonstrated, solution methods cannot generally guarantee global optimality for non-linear programming problems.

Up to this point, it has been implicitly assumed that optimization problems consist of a single objective function and utilize real numbers for variables. Though this is an appropriate way to introduce the concept of optimization, neither of these assumptions is necessary and, in fact, they are often not appropriate for distribution reliability optimization. For this reason, Sections 7.1.2 and 7.1.3 discuss the concepts of multi-objective optimization and discrete variable optimization, respectively.

7.1.2 Multi-Objective Optimization

The first step in approaching an optimization problem is to clearly define the metric that is being optimized. If this criterion can be represented as a single scalar value, the goal of the optimization process is clear—minimize this value. However, many utilities are not comfortable reducing their reliability problems to a single measure of goodness, and multiple objectives are required.

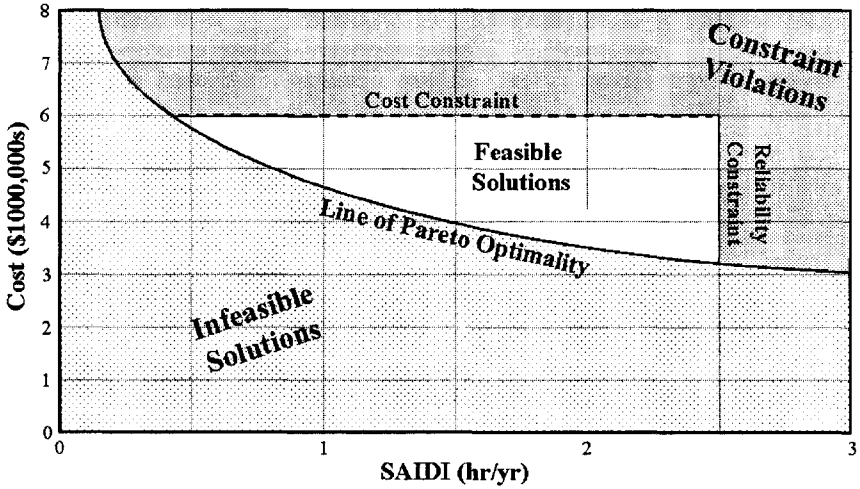


Figure 7.2. Classes of solutions for a multi-objective optimization problem considering reliability (SAIDI) and cost. For each value of cost, there exists a solution resulting in the lowest possible SAIDI. The set of these points is referred to as the line of Pareto optimality, and solutions on this line can only improve cost or reliability at the expense of the other.

To illustrate the concept of multi-objective optimization, consider a utility that is concerned about reliability and cost. Specifically, they would like to minimize net present value of reliability improvement projects while minimizing the expected value of SAIDI. Consider a specific cost, C. Theoretically, there is a best possible way to spend this money to result in the lowest possible SAIDI. Similarly, there is theoretically a lowest possible cost to achieve each specific SAIDI value. Each of these combinations of reliability and cost is said to be a Pareto optimum. This term is derived from the famous Italian economist and sociologist Vilfredo Pareto, who defined total societal welfare as an improvement in a person's condition that was not achieved at any other person's expense. The definition of Pareto optimality is:

> **Pareto Optimality** — a term used to describe a solution to a multi-objective optimization problem if any improvement to any objective function comes at the expense of one or more other objective functions. A solution with Pareto optimality is sometimes referred to as an efficient solution.

The concept of Pareto optimality is shown in Figure 7.2. The line of Pareto optimality represents the best possible combinations of reliability and cost. Any combination below this line cannot be achieved, and any combination above this line is sub-optimal. In addition, this figure represents a maximum acceptable SAIDI value and a maximum acceptable cost value. These constraints result in infeasible combinations of cost and reliability on and above the line of Pareto optimality.

When first examining reliability and cost from a Pareto optimality perspective, most utilities will find that the cost and reliability of their system lies above the line of Pareto optimality. This means that it is possible to achieve the same level of reliability for lower cost, which is inarguably an improvement. Similarly, it is possible to achieve a higher level of reliability for the same cost, also an unambiguous improvement. There will also be designs that can improve reliability and reduce cost at the same time, which is often the desired outcome. These unambiguous improvements in reliability and cost can only continue until the line of Pareto optimality is reached. At this point, cost cannot be further reduced without increasing SAIDI, and SAIDI cannot be reduced without increasing cost. When Pareto optimality is reached, it becomes imperative for reliability engineers to be able to communicate the implications of budget reductions and reliability improvement goals to those that make these decisions. If budget is cut by X%, a reliability engineer should be able to respond that this cut will result in a SAIDI increase of Y%. Similarly, if a corporate mandate states that SAIDI shall be reduced by Y%, a reliability engineer should respond that achieving this goal requires a budget increase of X%.

The concept of Pareto optimality is just as valid for optimization problems with more than two objective functions. An optimization problem with 1 objective function has a single (one-dimensional) optimal solution and an optimization problem with 2 objective functions has a two-dimensional curve of optimal solutions. Similarly, an optimization problem with 3 objective functions has a three-dimensional surface of optimal solutions and an optimization problem with N objective functions has an N-dimensional manifold of optimal solutions.

Consider a utility that wishes to optimize three values: cost, SAIDI and SAIFI. For this scenario, there will be a lowest possible cost for any selected combination of SAIDI and SAIFI. Similarly, there will be a lowest possible SAIDI for and selected combination of cost and SAIFI and a lowest possible SAIFI for and selected combination of cost and SAIDI. The set of optimal combinations forms a three-dimensional surface of Pareto optimality (see Figure 7.3). This surface has all of the implications of the two-dimensional line of Pareto optimality discussed in the reliability versus cost scenario. Utilities will generally find themselves above this surface, and can improve any objective function without sacrificing others. After a the surface of Pareto optimality is reached, the only way to improve one objective function is at the expense of one or both of the others.

Dealing with optimization problems with more than three objective functions becomes a bit more arcane since manifolds with more than three dimensions cannot generally be visualized. This is not a problem for computers, which can identify whether solutions are located on the manifold of Pareto optimality, what the trade-offs are between different objective functions if they are, and how far away they are if they are not. To present this information in a method comprehensible to humans, dimensionality must generally be reduced.

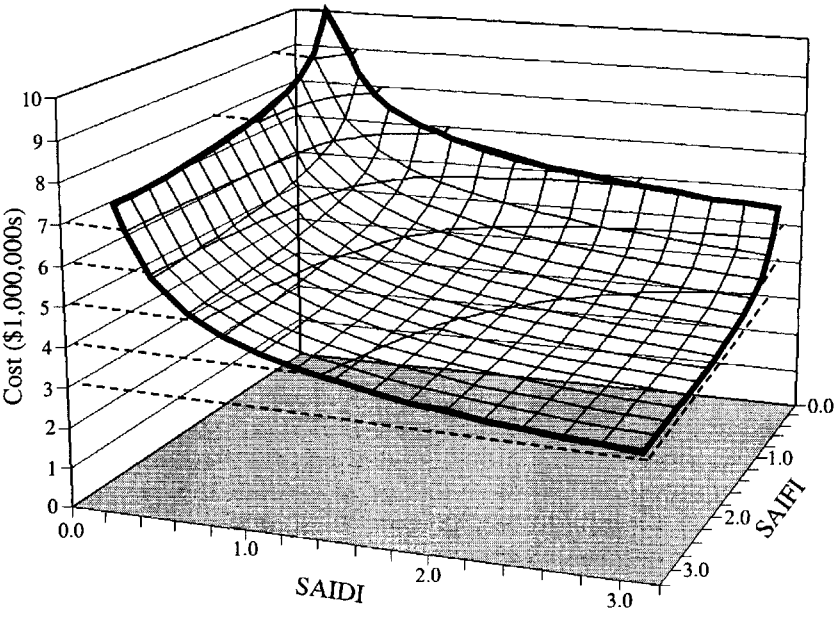


Figure 7.3. A surface of Pareto optimality for an optimization problem with three objective functions: cost, SAIDI and SAIFI. If a solution is on the surface of Pareto optimality, no objective function can be improved without causing one or both of the others to become worse.

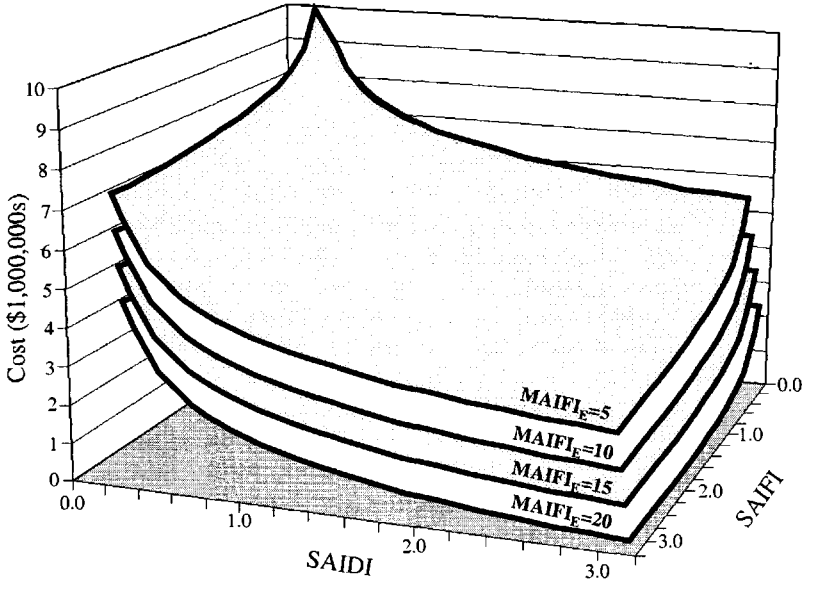


Figure 7.4. For optimization problems with more than three objective functions, Pareto optimality can be visualized by taking three-dimensional cross sections. In this case, optimizing MAIFI$_E$, SAIFI, SAIDI and cost is shown by taking cross sections for different fixed values of MAIFI$_E$.

It is often beneficial to examine Pareto optimality of optimization problems with more than three objective functions by taking two-dimensional or three-dimensional cross sections. Consider a problem that seeks to optimize cost, $MAIFI_E$, SAIFI and SAIDI. The surface of Pareto optimality will consist of a four-dimensional manifold that is impossible to visualize. A three-dimensional surface, however, can be generated by fixing one of the objective functions at a specific value and creating a manifold based on the remaining three objective functions. This can be done for several fixed values to create a set of three-dimensional cross sections similar to those shown in Figure 7.4.

7.1.3 Discrete Variable Optimization

Sections 7.1.1 and 7.1.2 have discussed optimization in terms of continuous optimization variables. Although beneficial for conceptual purposes, the vast majority of distribution reliability optimization problems consist of discrete choices rather than a continuum of choices. As an example, consider some of the choices that impact the reliability of a primary distribution feeder. There are an integer number of fuses and switches. These devices can only be placed at specific pole locations. These devices can only be purchased with specific ratings. Feeder sections can be either overhead or underground. Wire only comes in certain sizes. Switches are either automated or they are not. Optimization techniques that address discrete choices fall under the category of integer programming, and optimization techniques that address both continuous and discrete choices fall under the category of mixed integer programming[1]:

Integer Programming — the study of optimization problems with discrete optimization variables.

Mixed Integer Programming — the study of optimization problems with both continuous and discrete optimization variables.

In one sense, discrete choices are good for optimization problems. An optimal solution can be found by examining each combination of discrete possibilities and choosing the best combination. In reality, however, there are far too many possible solutions to examine each one separately. Discrete optimization suffers from a problem referred to as combinatorial explosion, which means that the number of possible solutions grows exponentially with problem size (i.e., the problem is NP hard). To illustrate, consider conductor sizing. If there are 5 possible conductor sizes and there is only one line section that is being built, the number of possible solutions is $5^1 = 5$. If there are two line sections, the number of possible solutions is $5^2 = 25$. If there are 20 line sections, the number of pos-

sible solutions is 5^{20} = 95 trillion. Clearly, even simple problems can quickly become intractable.

In the past, when computing power was much less than today, many discrete optimization problems were linearized so that computationally efficient linear programming techniques could be used. The final solution would then be selected by choosing discrete values nearest to the continuous values chosen by the continuous optimization process. For example, if the linear programming algorithm suggests placing a 567-kVAR capacitor bank 3.87 miles from the substation, the closest feasible solution may be placing a 600-kVAR capacitor bank (available for purchase) at 3.8 miles from the substation (the location of the nearest pole). Such techniques were an acceptable compromise in the past, but are generally not recommended since they can lead to sub-optimal results when compared to solution methods specifically designed for discrete optimization problems.

Even with sophisticated algorithms and fast computers, discrete optimization problems can quickly become untenable if their solution space is allowed to become excessively large. For this reason, it is vital to restrict the domain space of discrete optimization problems wherever possible. This is a divide-and-conquer approach that breaks up a large problem into several sub-problems that can be solved independent of one another. To illustrate the potential advantages, consider the conductor sizing problem with 20 conductors and 95 trillion possible solutions. If this problem can be broken down into 5 separate feeders with 4 sections each, the solution space is reduced to $5x5^4$ = 3125. This is a domain space reduction of 99.999999997%.

When dividing a reliability optimization problem into sub-problems, it is important to insure that each sub-problem is sufficiently independent of other sub-problems. For example, when placing protection devices and normally closed switches on a distribution system for reliability improvement, each feeder can be treated separately since contingencies on one feeder will not affect the reliability of other feeders. Within a feeder, however, protection device and switch placement must be treated together since their placements are interdependent: protection device placement will impact optimal switch placement and switch placement will impact optimal protection device placement.

7.2 DISCRETE OPTIMIZATION METHODS

This section discusses the techniques necessary to implement a discrete optimization algorithm for distribution reliability problems. It starts by discussing problem formulation including objective functions, constraints and penalty factors. It continues by discussing approaches to encoding potential solutions in a form that is usable by discrete optimization algorithms. It concludes by discuss-

ing the most common discrete optimization methods including local search, simulated annealing and genetic algorithms.

7.2.1 Problem Formulation

Problem formulation, as previously discussed in the introduction of this section, consists of an objective function, a set of constraints and a set of optimizable variables.

For discrete optimization problems, defining the objective function is technically simple since there are no requirements for linearity, convexity, differentiability or continuity. In fact, the objective function can usually be defined without knowing how it is computed. For example, SAIDI can be selected as the objective function without knowing precisely how SAIDI will be computed, a luxury not afforded by most continuous optimization algorithms.

In most situations, the distribution reliability optimization function and constraints will consist primarily of reliability and cost components. The three natural formulations are:

Common Distribution Reliability Problem Formulations

1. Optimize reliability subject to cost constraints
2. Optimize cost subject to reliability constraints
3. Optimize the total cost of reliability including the cost to provide reliability and the incurred costs associated with interruptions.

The first formulation is common for utilities with a fixed budget to spend on reliability improvement projects. Given a fixed amount of money, these utilities wish to achieve the best possible reliability, usually measured in terms of reliability indices. If multiple indices are of interest, it is generally appropriate to create an objective function equal to a weighted sum of multiple indices. For example, if a utility has present reliability indices of $MAIFI_{E0}$, $SAIFI_0$ and $SAIDI_0$, an appropriate objective function might be:

$$\text{Objective} = w_1\left(\frac{MAIFI_E}{MAIFI_{E0}}\right) + w_2\left(\frac{SAIFI}{SAIFI_0}\right) + w_3\left(\frac{SAIDI}{SAIDI_0}\right) \tag{7.2}$$

The weights of this objective function are selected according to the relative importance of reducing each index ratio. It is also convenient to make the sum of all weights equal to unity so that the base case objective function value is also equal to unity. For example, consider a system with reliability indices of $MAIFI_{E0}$=10 /yr, $SAIFI_0$=3 /yr and $SAIDI_0$=240 min/yr. If an objective function is defined as simply the sum of these three indices, SAIDI improvements will tend to dominate results since these particular SAIDI values are much larger than

the other indices. To solve this problem, each index can be normalized to initial index values so that a 10% increase in SAIDI is given the same value as a 10% increase in SAIFI or $MAIFI_E$. Last, weights are given to these normalized index values to reflect their relative importance. For example selecting the weights w_1=0.25, w_2=0.25 and w_3=0.5, will give SAIDI improvements twice as much importance as SAIFI or $MAIFI_E$ improvements. Further, the base case objective function value will be equal to one, allowing other scores to be viewed as per unit values.

The second formulation has a much more straightforward objective function since it is entirely based on cost. Once the cost basis is selected (e.g., net present value, annualized cost, initial cost), the objective function is simply equal to the sum of the cost of all reliability improvement projects. The third formulation is similar, but adds additional costs associated with the frequency and duration of interruptions. These incurred costs can include performance penalties, customer rebates, lawsuit awards, lost revenue and a host of other possibilities.

Constraints in discrete optimization problems are generally handled in one of three ways: (1) do not allow search algorithms to consider solutions that violate constraints, (2) have a function that modifies all solutions until all constraints are satisfied, and (3) represent constraints as penalty factors. Of these, penalty factors are generally recommended as a first approach. Treating constrained solutions as impenetrable barriers in the solution space can greatly reduce the amount of solution space that is explored, potentially resulting in lower quality solutions. Dynamic solution modifications to satisfy constraints can be effective if implemented carefully, but can bias results and reduce the computational effectiveness of optimization algorithms.

Penalty factors can be implemented in a variety of ways, but generally consist of an initial penalty that occurs when the constraint is violated and a variable penalty that changes its value based upon the degree of the violation. A general form for a penalty factor for variable x with a constraint of x_{min} is:

$$\text{Penalty Factor} = C_1 + C_2(x - x_{min})^{C_3} \tag{7.3}$$

Selection of C_1, C_2 and C_3 are important to both the type of solutions that will be identified as optimal and what their overall quality is likely to be. If constraints can be relaxed and slight violations are acceptable, C_1 should be small to nothing and the other parameters should be carefully chosen so that the tradeoffs being made between the objective function and the penalty factor are appropriate. If constraints cannot be relaxed, C_1 should be large when compared to the objective function of the worst feasible solution and the other parameters should be chosen so that solutions with a few small constraint violations are more desirable than solutions with many large constraint violations.

7.2.2 Solution Representation

An important step in discrete optimization is solution representation. This is simply a method of representing a solution in a manner that can be used by discrete optimization algorithms. The most popular way to represent a solution is to use an integer map, which is a string of bounded integer numbers that represent a solution. A common type of integer map restricts integer values to zero or one, and is called a bitmap. Consider a utility trying to decide where to place fault detectors on a system with 50 possible locations. Feasible solutions can be represented by a string of 50 binary numbers, with each number corresponding to a feasible location. A value of zero corresponds to not having a fault detector at the location and a value of one corresponds to having a fault detector at the location. For this scenario, the following bitmap represents a solution with fault detectors in locations 1, 5, 25 and 50.

```
                00000000001111111112222222222333333333344444444445
Location:       12345678901234567890123456789012345678901234567890
Value:          10001000000000000000000010000000000000000000000001
```

Sometimes discrete decisions are not binary in nature. An example is the selection of conductor size. Suppose there are five standard sizes to choose from when upgrading a conductor. For a bitmap to reflect this choice, each section has to be assigned a sub-string of three bits. The sequence 000 corresponds to no upgrade, the sequence 001 corresponds to the smallest upgrade, and so on until the sequence 101 corresponds to the largest possible upgrade. This type of representation is acceptable, but has several drawbacks. First, certain combinations may not correspond to meaningful solutions. In this case, the sequence 110 and 111 are not assigned to a conductor upgrade. Second, the changing of a single bit may result in a very different solution, making solution space exploration difficult. Consider the upgrades shown in Table 7.1, with solution 000 corresponding to no conductor upgrade. If the first bit changes, the resulting sequence of 100 corresponds to second largest possible upgrade, skipping all of the possibilities in between. Such problems can be reduced but not eliminated by using sequences of binary strings that only change one bit at a time, like the Gray code[2], but it is often more convenient to use non-binary maps.

Table 7.1. Representation of a conductor upgrade using a typical binary sequence and a Gray code sequence. Notice that the Gray code sequence only changes one bit at a time.

Typical Binary Representation		**Gray Code Representation**		
Decimal	Binary	Decimal	Binary	Upgrade
0	000	0	000	No Upgrade
1	001	1	001	Upgrade to 4/0
2	010	3	011	Upgrade to 250 kcmil
3	011	2	010	Upgrade to 500 kcmil
4	100	5	110	Upgrade to 750 kcmil
5	101	8	111	Upgrade to 1000 kcmil

A generalized integer map generally consists of a string of integers, with each integer representing each individual decision. For the conductor upgrade problem discussed earlier, solutions could be encoded by assigning to each line section an integer with a value from zero to five. The following integer map represents a solution with an upgrade of section 1 to 1000 kcmil (integer value of 5), an upgrade of sections 5 through 8 to 250 kcmil (integer value of 3) and an upgrade of sections 25 through 32 with an upgrade to 4/0 (integer value of 1).

```
                0000000000111111111222222222233333333334444444445
Location:       1234567890123456789012345678901234567890123456789O
Value:          50003333000000000000000011111111000000000000000000
```

Integer maps can also be used to represent multiple decisions. Consider a solution that consists of both fault location devices and conductor upgrades. In this situation, two sequential integers can be used to represent each section: a binary integer to represent whether a fault location device is present (A) and a six-value integer to represent the conductor upgrade (B). A solution that places fault location devices on sections [1, 5, 10, 13], upgrades section 1 to 1000 kcmil, upgrades sections 5 through 8 to 250 kcmil and upgrades sections 16 through 17 with an upgrade to 4/0 is represented as:

	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7
Location:	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB
Value:	15	00	00	00	13	03	03	03	00	10	00	00	10	00	00	01	01

Solution representation is important since it is the interface between the optimization algorithms and the reliability assessment algorithms. As such, it is important to generate an integer map that is efficient for the optimization algorithms to manipulate and for the reliability assessment algorithms to translate into a model that can be used to compute objective functions and test for constraints. The solution representation can and will have an impact on the quality of solutions for complex problems, and enough time should be spent on developing an appropriate structure before selecting and implementing a solution methodology.

7.2.3 Local Search

A simple and robust approach to solving a discrete optimization problem is referred to as a local search. Essentially, a local search starts with a solution and makes small changes until stopping criteria are met. During the algorithm, the best solution is remembered and is used as the result when the algorithm terminates.

The most common form of local search is referred to as hill climbing. This method can either begin with the existing state of the system or with a random solution. After evaluating the objective function of this solution, a single integer is increased in value by one step and the new objective function is computed. If the new value is better than the initial value, the new solution is kept. If the new value is not better than the initial value, the original value is reduced in value by one step and kept if the objective function becomes better. This process is done for each variable sequentially, looping back to the first integer after the last integer has been checked, until the design cannot be further improved. The hill climbing algorithm can be summarized as follows:

Algorithm for a Hill Climbing Local Search

1. Randomly generate an initial integer string, $C_1, C_2, \ldots, C_n$
2. Compute the initial objective function value, $\Omega(C_1, C_2, \ldots, C_n)$
3. Set x=1
4. Increase the integer value of C_x by one increment (if possible) and compute the new objective function value, Ω_{test}
5. Is Ω_{test} better than Ω? If yes, set $\Omega = \Omega_{test}$ and go to Step 10.
6. Decrease the integer value of C_x by one increment
7. Decrease the integer value of C_x by one increment (if possible) and compute the new objective function value, Ω_{test}
8. Is Ω_{test} better than Ω? If yes, set $\Omega = \Omega_{test}$ and go to Step 10.
9. Increase the integer value of C_x by one increment and set $x_{stop}=x$
10. Set x=x+1
11. Is x>n? If yes, set x=1
12. Is $x= x_{stop}$. If yes, end.
13. Go to Step 4

There are several variations to the hill climbing algorithm described above. The first variation is to alternate between tests for integer increments and integer decrements after each integer string pass rather than after each individual integer. This is a minor algorithmic change and will not tend to affect solution quality or algorithm performance by a substantial amount. The second variation is to examine all possible integer increments and integer decrements before selecting the single change that results in the best optimization function value. This type of discrete gradient search approach will tend to terminate in fewer accepted changes, but will also tend to be computationally much slower.

The hill climbing algorithm is important since it guarantees local optimality. After the algorithm terminates, a single increment or decrement to any integer is guaranteed not to produce a better solution. The hill climbing algorithm will also always result in the same solution if the same initial integer string is used. For this reason, it is often useful to run the hill climbing algorithm for many different randomly generated initial integer strings to see if solution quality or solution

characteristics vary. If the solution space is convex, each trial, regardless of the initial solution, will result in the same answer. This, unfortunately, is rarely the case and many local minima are usually encountered. If most of the solutions are nearly equivalent, this is not a problem and hill climbing is a sufficient method. If solutions vary widely, more sophisticated methods such as simulated annealing or genetic algorithms may be required.

A local search technique designed to avoid being trapped in local minima is referred to as Tabu search[3-4]. This method essentially consists of a hill climbing algorithm in the gradient form. However, while making changes, the algorithm keeps a Tabu list of integers that have recently been changed. When a local minima is reached, the algorithm selects the movement that will minimize objective function degradation, but only considers movements that are not on the Tabu list. This technique is not only effective for escaping local minima, but for escaping discontinuous islands surrounded by infeasible solutions.

The size of the Tabu list is an important parameter since too short a list can result in search cycles and too long a list may restrict interesting moves that may lead to high quality solutions[5-6]. Other features that can impact solution quality are aspiration, intensification and diversification[7]. Aspiration allows the Tabu list to be ignored if an unusually good move is detected. Intensification uses a relatively short Tabu list to find the local optimum in the local neighborhood of solutions, and diversification uses a long Tabu list to escape the local neighborhood and examine unexplored areas of the solution space. The stopping criteria for a Tabu search is often set by the number of diversification phases performed.

Local search techniques are simple to implement, are robust and do not have a large number of parameters to tune. As such, they are often sufficient for reliability optimization problems, especially when the problem is unique and the algorithm is not likely to be reused in the future. Perhaps the simplest method to implement is a hill climbing algorithm with a random initial solution. This algorithm can be repeated many times for many different initial solutions until a distribution of outcomes becomes apparent and there is confidence that the best encountered solution is near the global optimal. This method may not be as computationally efficient as some of the more sophisticated techniques that will be discussed in future sections, but is quick to implement and may produce answers that are nearly as good.

7.2.4 Simulated Annealing

Simulated annealing optimizes a function by performing a directed random search based on statistical mechanics. Consider the annealing process of metal. First, the metal is heated up so that the molecules have high energy and high mobility. Next, the metal is slowly cooled so that the atoms have a high probability of settling down in a low energy state. When the temperature becomes

low enough, the metal molecules have very low mobility and the system is "frozen" in a low energy state. In a similar manner, a discrete optimization problem can be annealed to find high quality solutions. Initially, the solution has high mobility and is allowed to randomly move about the solution space regardless of whether solution quality improves or degrades. Gradually, the system is cooled, mobility is decreased, and the probability of moving to lower quality solutions is reduced. Eventually, the system is reduced to zero mobility, referred to as being frozen, and a high quality solution has hopefully been found. Simulated annealing is attractive since it can theoretically guarantee a globally optimal solution, but the annealing process is inherently slow and formulating an efficient cost function and annealing schedule has proven difficult[8].

A suggested annealing schedule uses a single temperature parameter, T, for its annealing schedule. The schedule begins by creating a random solution and setting an initial value of T close to unity. A new solution is then generated by examining each integer in the integer map and randomly incrementing or decrementing its value with a probability of T (e.g., if T is 0.9, an integer will be reassigned 90% of the time). If this new solution is an improvement over the old solution, it is accepted. If this new solution is worse than the previous solution, it is accepted with a probability of T. The temperature is then reduced by multiplying it by an annealing rate, R, and the process is repeated. This procedure repeats until T has reduced in value a pre-specified number of times without any improvements in solution quality. At this point the solution is frozen and the annealing process is terminated. Since simulated annealing cannot guarantee that a local minimum has been reached, a hill climbing algorithm should be performed after a solution freezes. A summary of the algorithm is:

Algorithm for Simulated Annealing

1. Select a starting temperature, $T \in (0,1)$, and an annealing rate, $R \in (0,1)$
2. Randomly generate an initial solution and compute the objective function, Ω
3. Generate a new solution by changing each integer value with a probability of T. Increment or decrement the integer value with equal probability
4. Compute the objective function of the new solution, Ω_{test}
5. Is Ω_{test} better than Ω? If yes, keep the new solution and set $\Omega = \Omega_{test}$. If no, keep the new solution with a probability of T
6. Has the solution changed since a preset number of temperature changes? If no, go to Step 8
7. Multiply T by the annealing rate R and go to Step 3
8. Perform a hill climbing local search on the frozen solution to guarantee local optimality

The speed and solution quality of a simulated annealing algorithm is a strong function of both starting temperature and annealing rate, with higher quality solutions becoming much more likely with a slow annealing rate. In general, higher initial temperatures should be used with fast annealing rates, but lower starting temperatures are acceptable with slow annealing rates since the algorithm will still have sufficient time to explore the solution space. In the author's experience, a starting temperature of 0.3 with an annealing rate of 0.99 typically results in high quality solutions in a reasonable amount of time.

A common variation of simulated annealing determines the probability of accepting a worse solution based on both temperature and the difference between the objective function values of the candidate and present solution. This probability is typically formulated as follows:

$$\text{Probability of Acceptance} = \exp\left(\frac{\Omega - \Omega_{\text{test}}}{\Omega \cdot T}\right) \tag{7.4}$$

This probability of acceptance equation works well for certain objective functions, but may limit solution space exploration for problem formulations that utilize penalty factors that can be large when compared to the objective function. This is especially true for problems with hard constraints. Consider a problem where a hard constraint violation is reflected as a penalty factor approximately twice the value of the expected objective function. Even at a temperature of 0.9, the probability of accepting a solution that barely violates the constraint will be less than 10%.

Implementing a simulated annealing algorithm is somewhat more complicated than implementing a local search, and should actually include a local search so that solutions are guaranteed to be locally optimal. Therefore, simulated annealing is recommended for optimization problems that are not being handled well by a local search algorithm, or for optimization algorithms that will be required on a regular basis in the future.

7.2.5 Genetic Algorithms

In both simulated annealing algorithms and local search algorithms, each trial is independent. This means that each individual trial has the same probability of resulting in a good solution, and increasing the number of trials will not affect this probability. If a large number of trials are needed to ensure a high quality solution, it may be beneficial to use techniques in which trials are run in parallel, and each trial can share information with other trials in an attempt to improve overall solution quality. An example of such a technique is genetic algorithms.

A genetic algorithm is a form of directed random search based on natural selection and evolution. After an initial population of solutions is created, future

generations are determined by probabilistically selecting high quality parents and combining them to create children. During this process, a small amount of random mutation is used to ensure genetic diversity. Genetic algorithms have proven to be robust and efficient when applied to difficult optimization problems and their application to distribution planning problems has increased exponentially in recent years[9-12].

Genetic algorithms operate on a set of potential solutions. Using biological terminology, each individual member is referred to as a chromosome and the entire set is referred to as a population. A chromosome is usually encoded as an integer map, with each integer in the map referred to as a gene and each possible value of the genes being referred to as an allele. An example of two chromosomes, each with eight genes and binary alleles, is:

```
Chromosome A: 11010010
Chromosome B: 10011011
```

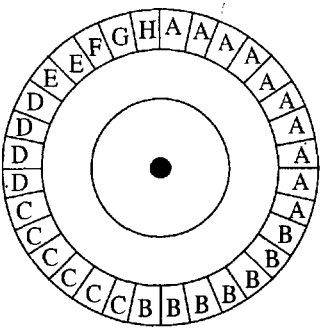
Each chromosome is characterized by a fitness function, which is typically equal to the objective function value plus all penalty factors. After all chromosomes are assigned a fitness value, the next generation is created through genetic operators such as crossover and mutation. Crossover is the fundamental genetic operator for genetic algorithms and consists of probabilistically selecting two fit parents and combining their chromosomal information to produce offspring. Mutation consists of randomly altering a small percentage of genes to ensure genetic diversity.

There are two common methods of parental selection: roulette wheel and tournament selection. The roulette wheel method allocates wheel bins based on parental fitness, with each parent receiving at least one bin and fitter parents receiving more bins. Parents are chosen by randomly selecting a bin, with each bin having an equal chance of being selected.

A simple way of assigning roulette wheel bins to parents is to allocate a small number of bins to the worst parent and a large number of bins to the best parents. The number of bins allocated to other parents can be assigned by linear interpolation. For example, consider a population with a best fitness score of Ω_{best} and a worst fitness score of Ω_{worst}. If the best member is allocated B_{max} bins and the worst member is allocated B_{min} bins, members with a fitness of Ω are allocated the following number of bins:

$$\text{Bins} = B_{min} + (B_{max} - B_{min})\frac{(\Omega - \Omega_{best})}{(\Omega_{worst} - \Omega_{best})} \tag{7.5}$$

An example of bin allocation for a population size of eight, a minimum of one bin per member and a maximum of ten bins per member is shown in Figure 7.5.



	Fitness	# Bins
A	22,142	10
B	25,389	7
C	26,203	6
D	28,489	4
E	30,532	2
F	31,581	1
G	31,908	1
H	32,189	1

Figure 7.5. Example of roulette wheel parental selection. In this case, the roulette wheel is formed from a population of eight members with the best member (A) receiving ten bins and the worst member (H) receiving a single bin. The number of bins allocated to other members is determined by linear interpolation.

A second popular method of parental selection is referred to as tournament selection. In tournament selection, a parent is selected by winning a tournament consisting of two or more randomly selected members. The winner is chosen in a similar manner to roulette wheel selection, with fitter members having a higher probability of winning the tournament than weaker members do. If a tournament consists of just two members, the stronger member can simply be given a fixed probability of being selected, such as 70% or 80%.

Once two parents are selected, the genetic operator crossover is used to create two new children. The simplest form of crossover, referred to as single point crossover, is performed by selecting a random crossover point and creating two new children by swapping genes on either side of the crossover point. An example of single point crossover is shown below (the selected crossover point is indicated by an 'x'):

Pare... A: **1101** x **0010**
Parent B: 1001 x 1011

Child 1: **1101** 1011
Child 2: 1001 **0010**

The theory addressing the computational efficiency of genetic algorithms is based schema, which is defined as a set of alleles in a chromosome. The most important aspect of a schema is its length, which is equal to the separation of the first encountered allele and the last encountered allele. An example of several different alleles with various lengths is shown in Table 7.2.

Table 7.2. Examples of schema and their corresponding length.

Schema	Alleles	Length
A	-123----	3
B	-321----	3
C	-1-2-3--	5
D	3--3--33	8

There are several important points regarding schema as illustrated in Table 7.2. First, different schema can consist of the exact same genes with different allele values as shown by Schema A and Schema B. Second, the alleles associated with a schema do not have to be sequential as shown by Schema C and Schema D (a dash indicates that the gene is not associated with the schema).

The efficiency of genetic algorithms using roulette wheel parental selection and single point crossover has been mathematically proven in what is referred to as the schema theorem[13], which states that schema with short lengths that are correlated with high fitness functions are exponentially reinforced in subsequent generations. It has been further shown that genetic algorithms are good at avoiding local minima and have inherent parallelism in computation.

The schema theorem proves the inherent computational efficiency of genetic algorithms, but has an important implication that is often overlooked. The probability of good allele combinations being reinforced is directly related to how closely these gene combinations are grouped together in the integer map. Consider a highly desirable two-gene combination. If these two genes are far apart, they have a good chance of being separated during the crossover process. Therefore, it is vitally important to group highly correlated genes as close as possible when creating a chromosome to represent a solution.

A common variation of single point crossover is referred to as two-point crossover. In two-point crossover, two crossover points are randomly chosen and the bits between these crossover points are exchanged to create children. An example of two-point crossover is:

Parent A: **0010101101** x **001000** x **000010101101**
Parent B: 1100010100 x 101101 x 001010001011

Child 1: **0010101101** x 101101 x **000010101101**
Child 2: 1100010100 x **001000** x 001010001011

In single or two-point crossover, crossover points can be chosen randomly or be restricted to a subset of gene locations. The restriction of crossover points is used to keep closely associated genes from being split during the crossover process. Sets of integers located between allowed crossover points are referred to as co-adapted alleles.

The second commonly used genetic operator is mutation. After a child is created through crossover, a small chance is given (typically less than 1%) for any particular integer to change its value. Mutation serves as a constant supply of genetic diversity and prevents the permanent loss of any particular allele. High mutation values may be necessary for small populations without a large amount of genetic information, but this reduces the computational effectiveness associated with the schema theorem[14]. Essentially, high mutation rates result in undirected random searches that will hopefully stumble upon a good solution. If high mutation rates seem to increase solution quality, it is probable that the population is too small for genetic algorithms to be effective.

The third feature commonly used in genetic algorithms is an elitist class. When a new generation is being created, a certain percentage of the future population is reserved for the fittest members of the present population. Instead of crossover, these elite members are copied directly into the new population and are shielded from mutation. At least one elite member should always be used since this ensures that the best encountered solution will always be remembered. Larger elite populations will tend to reinforce good schema before they are corrupted from crossover, but will tend to reduce genetic diversity over time and may lead to premature algorithm convergence.

The last feature of a genetic algorithm is the stopping criterion, which terminates the algorithm after certain conditions are met. The simplest method is to stop after a fixed number of generations, but this may cause the algorithm to stop while solutions are continuing to improve or stop well after solutions have stopped improving. Other possible methods are to stop after a preset number of generations have elapsed without the best member improving, the average of the elite population improving, the average of the entire population improving or the convergence of average solution quality to best solution quality.

It is difficult to know *a priori* how to best apply genetic algorithms to a specific problem—genetic operators must be selected, parameter values must be chosen, stopping criteria must be defined, and so on. All choices are ultimately interrelated and, therefore, finding the best combination can be an arduous and elusive task. Although optimal choices will vary based on the specific problem formulation, some general recommendations are as follows:

- Use a population size equal to 25% of the number of genes in the solution (e.g., a population size of 150 for a solution is represented by a chromosome with 600 genes).
- Use an elitist class equal to 2% of the total population.
- Use a two-member tournament to select parents with the more fit parent being chosen 70% of the time and the less fit parent being chosen 30% of the time.
- Use two-point crossover instead of single-point crossover.
- Avoid the use of co-adapted alleles.

- Do not perform mutation on elite members and use a mutation rate of 0.05% per gene per generation for all other members.
- Terminate the genetic algorithm when the best elitist member has not improved for 10 generations.
- Perform a hill climbing algorithm after the genetic algorithm is complete to ensure local optimality.

Genetic algorithms can come in many forms depending upon various choices that are made. The following algorithm is provided as an example for optimization problems related to distribution reliability:

Genetic Algorithm with 2-Point Crossover

1. Design solution chromosome with highly related genes being located in close proximity to each other
2. Randomly generate an initial population of solutions referred to as generation zero
3. Compute the fitness of each member in the population
4. From Generation G, create Generation G+1
   - 4.1. Copy the elite members of G directly into Generation G+1
   - 4.2. Create the remaining members for Generation G+1
     - 4.2.1. Select 2 parents using a parental selection method
     - 4.2.2. Randomly select 2 crossover points
     - 4.2.3. Create 2 children by swapping the parental genes located between the crossover points
     - 4.2.4. Perform mutation on these two children
     - 4.2.5. Place the 2 children in Generation G+1
5. Compute the fitness of each member in Generation G+1
6. Has the stopping criterion been satisfied? If no, go to Step 4.
7. Perform hill climbing on all members to ensure local optimality
8. Select the member with the best objective function

For complicated distribution reliability optimization problems of moderate size and larger, genetic algorithms will almost always outperform local search and simulated annealing, with performance gains being reflected in both solution quality and in computation time. These gains, however, come at a price. Implementing genetic algorithms is substantially more difficult and tuning the genetic algorithm parameters for optimal performance can be difficult and time consuming. Regardless, a good genetic algorithm remains one of the most effective methods of solving discrete optimization problems and the added performance will often be worth the added effort.

7.3 KNOWLEDGE-BASED SYSTEMS

The discrete optimization methods described in the previous section are powerful methods of exploring a complex solution space, but do not take advantage of human expertise that may be available from distribution planners and engineers. This section describes several techniques that are able to make reliability improvement decisions based on expert human knowledge represented as rules. The first method, referred to as an expert system, represents rules as a set of conditional statements in the form of if-then rules. The second method, referred to as a fuzzy expert system, extends this concept by accommodating vague human semantics such as small, medium and large.

7.3.1 Expert Systems

Expert systems are algorithms that perform tasks historically reserved for human experts. This includes the representation of expert domain knowledge, the ability to infer an answer based on this knowledge and the ability to explain why a particular inference has been made[15]. Expert systems can be quite valuable since they allow expert-like decisions to be made by non-experts. Further, those using the expert system are taught to think and reason like experts since all decisions made by the expert system are explained in detail.

Expert system domain knowledge is typically captured by a set of expert rules collectively referred to as a knowledge base. These rules are generally represented as if-then conditional statements with the "if clause" referred to as the antecedent and the "then clause" referred to as the consequence. The consequence can also contain a measure of confidence reflecting the percentage of time that the satisfaction of the antecedent ultimately corresponds to a recommendation of the consequence. An example of an expert rules is:

Rule	[R1]
If	[A1] the lateral tap is overhead, and
	[A2] the lateral tap is single phase, and
	[A3] the lateral tap is longer than 500 feet
Then	[C1] the lateral tap should be fused (0.95 confidence)

This rule (R1) consists of three antecedent clauses (A1-A3) and a single consequence (C1). If all of the antecedent clauses are true, then the expert that created the rule has a 95% confidence that the consequence should be recommended.

A typical expert system can consist of thousands of rules with many rules associated with the same consequences. This creates the possibility of conflicting rules. Consider the recommendation associated with rule [R1] in the above ex-

ample. There may be many other rules that recommend that the lateral tap be fused, leading to a question of what the confidence of the recommendation should be. There may also be rules that recommend that the lateral tap not be fused, leading to a question of which recommendation is preferred.

To perform rule resolution, each expert system is equipped with an inference engine that identifies active rules (rules with true antecedents) and resolves rules with conflicting consequences. A simple inference approach is to group together all active rules that have the same or opposite consequence (e.g., "the lateral tap should be fused" and "the lateral tap should not be fused"). The net consequence is taken to be the sum of all active rules with the same consequence minus the sum of all active rules with the opposite consequence divided by the total number of active rules. This is essentially an average value, with a negative result indicating that the opposite consequence should be recommended. A sample rule inference is:

Rule	Consequence	Confidence
[R1]	the lateral tap should be fused	0.95
[R5]	the lateral tap should be fused	0.70
[R12]	the lateral tap should be fused	0.55
[R47]	the lateral tap should not be fused	-0.30
[R81]	the lateral tap should not be fused	-0.10
[Final]	the lateral tap should be fused	0.36

For the above example, the final recommendation is that the lateral tap should be fused. This recommendation has a 36% confidence and is based on five active rules—three in concurrence with the final recommendation and two in contradiction with the final recommendation.

Inferring final recommendations by taking the average of active rules assumes that all rules are of equal importance. This may not be the case. Consider the following two rules: (1) if a wasp is flying towards me from the right, then run to the left with 95% confidence, and (2) if a crocodile is running towards me from the left, then run to the right with 95% confidence. If both rules are active, a simple inference engine will recommend standing still—not good expert advice. This problem can be overcome by assigning a weight to each rule based in its relative importance. The inference engine can then base the final confidence of the consequence based on the weighted average of the active rule confidence values rather than the unweighted average.

Expert systems are well suited for analyzing very large systems[16-17] and producing high level analyses that can be used as a starting point for more in-depth analyses. To illustrate, consider the expert system analysis of Commonwealth Edison's 15-kV distribution system, which consists of 3,300 feeders. This assessment was a result of the major reliability problems experienced in the summer of 1999[18], and had a goal of identifying the quickest and most cost-effective

ways of improving customer reliability while avoiding similar problems in the future[19].

The expert system rules for this project were based on the anticipated benefit-to-cost ratio of potential reliability improvement projects, with benefit being defined as reduction in customer kVA-minutes per year and cost being defined as annualized project cost. These rules were then applied to automatically generated reliability improvement projects, with an average of 1000 projects being considered per feeder (over 3 million projects in all). Done by hand, expert analysis of each potential project is obviously not feasible. Using an expert system, recommendations for each potential project (after the system was modeled and calibrated) was accomplished in less than three weeks.

The expert system makes extensive use of system topology for its recommendations. In addition, rules make use of reliability analysis results such as peak loading, reliability, root cause scores and capacity constraints. Using this wide array of information allows complicated rules to be generated such as:

Rule	[R1]
If	[A1] the lateral tap is 3ϕ overhead, and
	[A2] the number of lateral tap customers is more than 500, and
	[A3] the lateral tap is longer than 100 feet, and
	[A4] the lateral tap SAIDI is greater than 120 min/yr, then
Then	[C1] a recloser should be placed on the lateral tap (0.9 confidence)

Rule	[R2]
If	[A1] the switch operates more than once per year, and
	[A2] the number of upstream customers is more than 1000, and
	[A3] there is a downstream tie switch, and
	[A4] the transfer path has no capacity constraints, then
Then	[C1] the switch should be automated (0.8 confidence)

Several different categories of reliability improvement projects were explored, which correspond to consequences of the expert system rules. This allowed different reliability improvement approaches to be compared on the same basis. Basic categories of reliability improvement options examined include:

- **Transfer Path Upgrades** — A transfer path is an alternate path to serve load after a fault occurs. If a transfer path is capacity constrained due to small conductor sizes, reconductoring may be a cost-effective way to improve reliability.

- **New Tie Points** —Adding new tie points increases the number of possible transfer paths and may be a cost-effective way to improve reliability on feeders with low transfer capability.

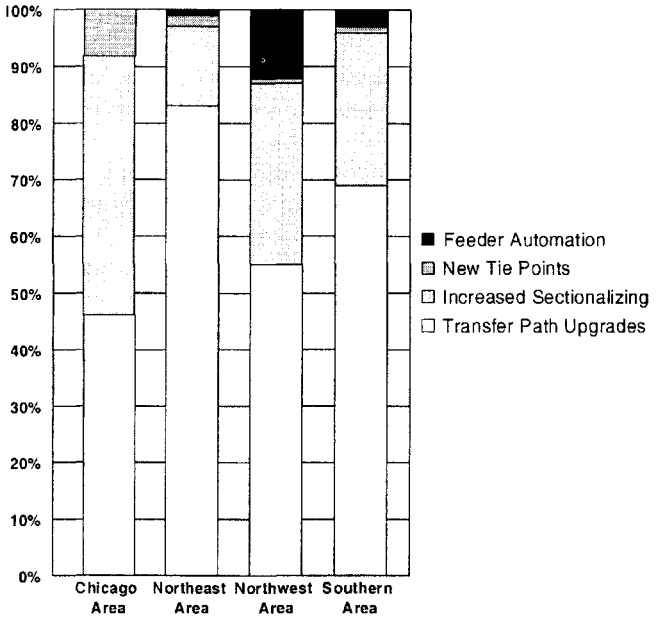


Figure 7.6. A breakdown of reliability improvement recommendations made by an expert system for the four Commonwealth Edison operating areas. This figure clearly shows that reliability improvement strategies will vary for different areas.

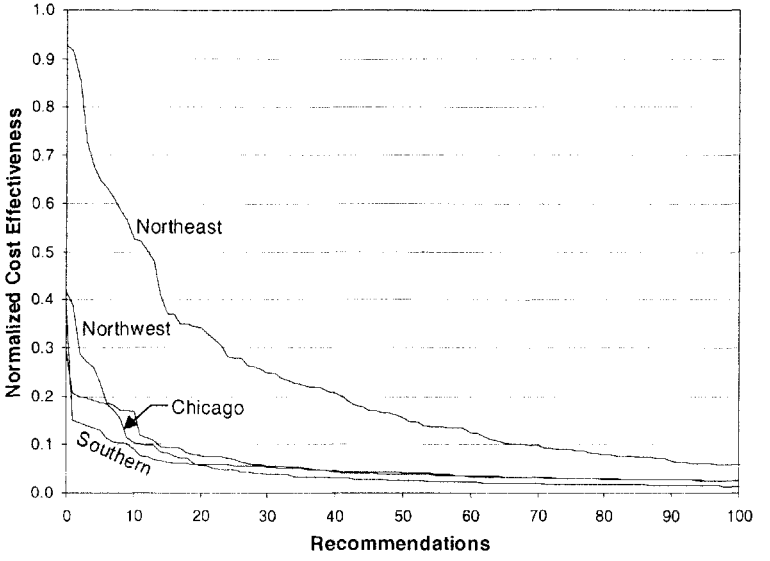


Figure 7.7. The range of project cost-effectiveness broken down by operating region. The expert system results indicate that the most cost-effective reliability gains can be made in the Northeast region and the least in the Southern region.

- **Increased Line Sectionalizing** — Increased line sectionalizing is accomplished by placing normally closed switching devices on a feeder. Adding fault interrupting devices (reclosers and fuses) improves reliability by reducing the number of customers interrupted by downstream faults. Adding switches without fault interrupting capability improves reliability by allowing more flexibility during post-fault system reconfiguration.

- **Feeder Automation** — In this study, feeder automation refers to SCADA-controlled switches on feeders. These automated switches allow post-fault system reconfiguration to occur much more quickly than with manual switches, allowing certain customers to experience a momentary interruption rather than a sustained interruption.

Recommendations were generated for each feeder and grouped into the four utility operating areas: Chicago, Northeast, Northwest and Southern. A breakdown of the type of recommendations made for each area is shown in Figure 7.6, and the ranking of projects by cost effectiveness is shown in Figure 7.7. These results show that reliability improvement recommendations vary widely for each region, and that the expected cost-effectiveness of reliability improvement projects also varies widely from project to project and from region to region. For example, the highest ranked project for the Northeast region is more than three times as cost-effective as the highest ranked project for the Southern region. Within the Northeast region, the highest ranked project is more than 5 times as cost-effective as the fiftieth ranked project.

Expert systems are an excellent tool for capturing expert knowledge and using this knowledge to obtain tentative recommendations for large systems. In a sense, they can fulfill a triage function to quickly identify problem areas and quickly identify potential cost-effective solutions. Final recommendations, however, should still be based on rigorous cost and reliability model analysis so that details not considered in the expert rules, such as project interaction, can be properly considered[20].

7.3.2 Fuzzy Expert Systems

Traditional expert systems can sometimes have difficulty capturing expert knowledge since expert knowledge is often vague whereas expert rules must be precise. Consider an expert who states that, "long feeders in areas of high tree density should have frequent tree trimming." In this statement, the imprecise modifiers "long," "high" and "frequent" are difficult to translate into an expert rule. Many expert systems handle this difficulty by assigning numerical values that approximate the intent of the rule such as, "feeders longer than 10 miles in

areas of more than 100 trees per mile should perform tree trimming at least twice per year." Although this statement may accurately reflect expert opinion for a specific case, it may not reflect expert opinion for other cases such as feeders slightly less than 10 miles long or feeders in areas of slightly less than 100 trees per mile.

Fuzzy set theory is a framework capable of representing vague human semantics and incorporating them into expert rules[21-22]. In classical set theory, membership is crisp; an item is either a full member or it is a non-member. This is practical in many scenarios. For example, the set of "protection devices" is crisp—any fuse is a full member while any capacitor is a non-member. In fuzzy sets, each item in the domain has a grade of membership for each set—a number between zero and one that represents how much the item belongs to a set (one indicates absolute membership and zero indicates non-membership). For example, a feeder 8 miles in length might have a membership of 0.5 in the set of "long feeders." Possible fuzzy sets for short, medium, long and very long are shown in Figure 7.8.

The four sets of Figure 7.8 have a domain (every possible member) of all feeders, and each feeder belongs to each fuzzy set to a certain degree. For example, the statement "a feeder 4 miles long is short" has a truth of 0.5, and the statement "a feeder 20 miles long is very long" has a truth of 1.0.

Fuzzy set theory, like crisp set theory, becomes more useful if the union, intersection, and implication of sets are defined. These operators allow the truth of more complex statements using "and," "or" and "if" to be handled. Typical definitions for these three operators are ($A \wedge B$ refers to the minimum value of A and B and $A \vee B$ refers to the maximum value of A and B):

$$\text{Intersection } (A \text{ and } B) \quad A \cap B = A \wedge B \tag{7.6}$$

$$\text{Union } (A \text{ or } B) \quad A \cup B = A \vee B \tag{7.7}$$

$$\text{Implication } (B \text{ if } A) \quad A \rightarrow B = 1 \wedge (1 - A + B) \tag{7.8}$$

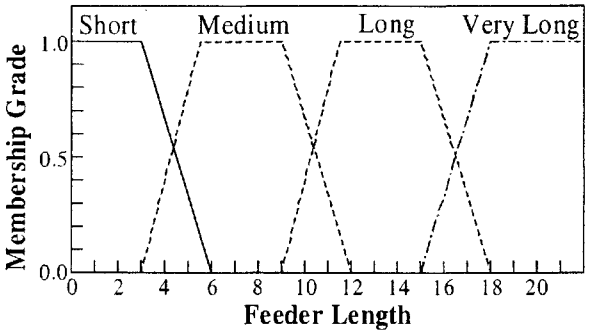


Figure 7.8. Examples of fuzzy sets related to feeder length. Each feeder, depending upon its length, belongs to each fuzzy set to a varying degree, called its grade of membership. The above example shows fuzzy sets as trapezoids, but other shapes, such as triangular and Gaussian, are also common.

Fuzzy sets can be used to create expert rules similar to those used in basic expert systems. The difference is that rules can contain modifiers corresponding to pre-defined fuzzy sets. An example fuzzy rule with three inputs and one output is:

Rule	[R1]
If	[A1] the lateral tap is long, and
	[A2] the tree density is high, and
	[A3] the lateral tap has many customers
Then	[C1] the lateral tap should be fused (high confidence)

In addition to fuzzy sets in the antecedent clauses (i.e., long, high and many), a fuzzy set is used for the consequence. This consequence, unlike those in traditional expert system rules, is not restricted to being either active or inactive. Since the antecedent clause of a fuzzy rule can be partially true, the consequences of fuzzy rules are allowed to be partially active. The degree of activeness of a fuzzy rule is determined by fuzzy inference, which returns the mean value of all consequences weighted by their respective antecedent truths. To illustrate, consider the following two fuzzy rules, each with two fuzzy inputs and one fuzzy output:

$$\text{[R1]:} \quad \text{IF } x_1 \text{ is } A_{11} \text{ and } x_2 \text{ is } A_{12} \text{ THEN } y_1 \text{ is } B_1 \tag{7.9}$$

$$\text{[R2]:} \quad \text{IF } x_1 \text{ is } A_{21} \text{ and } x_2 \text{ is } A_{22} \text{ THEN } y_1 \text{ is } B_2 \tag{7.10}$$

The first step of fuzzy inference is to compute the compatibility, C, of each antecedent, which is equivalent to the truth of the antecedent:

$$C_i = A_{i1}(x_1) \wedge A_{i2}(x_2) \tag{7.11}$$

In this equation, $A_{ij}(x_k)$ refers to the truth of "x_k is A_{ij}". The consequence of rule i, referred to as Y_i, is now computed by multiplying the set B_i by the truth of its respective antecedent. This is written as follows:

$$Y_i = C_i \cdot B_i \tag{7.12}$$

The final solution is found by combining the results of all fuzzy rules. This is done by dividing the sum of all Y_i sets by the sum of all antecedent truths. This resulting fuzzy output set, Y_{out}, is mathematically expressed as:

$$Y_{out} = \sum_{i=1}^{N} Y_i \Big/ \sum_{i=1}^{N} C_i \tag{7.13}$$

The result of a set of fuzzy rules is a fuzzy output set, Y_{out}. This fuzzy set result can be translated into a single numerical value by computing its mean value, which is equal to the integral of the moments of the function divided by the integral of the function. This process, sometimes referred to as defuzzification, is mathematically expressed as follows:

$$\bar{y} = \frac{\int_{-\infty}^{\infty} x \cdot Y_{out}(x)dx}{\int_{-\infty}^{\infty} Y_{out}(x)dx} \tag{7.14}$$

Fuzzy expert systems that utilize defuzzified outputs can be applied in exactly the same manner as traditional expert systems. This includes adding weights to rules to reflect their relative importance. The drawback to fuzzy expert systems is that rules can sound semantically correct, but will not actually represent expert knowledge unless the fuzzy sets are chosen properly. Regardless, fuzzy set theory is a powerful concept that, when applied properly, can aid in capturing expert knowledge, help to identify the reliability characteristics of large systems and generate reliability improvement options that have a high likelihood of cost-effectively improving distribution reliability.

7.4 OPTIMIZATION APPLICATIONS

The optimization techniques described previously in this chapter are not just abstract academic concepts. They are powerful computational frameworks that can be applied to practical distribution system reliability optimization problems with computing power available on a typical personal computer[23-29]. To reinforce this point and to give the reader a better feel for practical reliability optimization, this section presents several actual optimization applications and corresponding results. It concludes with a comparison of optimization methods so that their relative effectiveness can be examined.

7.4.1 Feeder Automation

A simple application of reliability optimization is the selection of existing manual feeder switches to be automated. The objective function and constraints are easily defined in terms of cost and reliability, potential solutions are easily represented as an integer map, the reliability improvements associated with potential solutions are easily modeled and locally optimal solutions are easily generated by using hill climbing techniques. In addition, this is a practical problem of in-

terest to many utilities considering the use of automated feeder switches to improve the reliability of their distribution system.

In this application, the goal of feeder automation is to achieve a SAIDI target using the fewest possible number of automated switches. As previously mentioned, this will be done by adding automation equipment to existing manual switches (both normally closed and normally open). The area under consideration is the service territory of an eight feeder substation in the midwestern US. This system serves approximately 12,000 customers with a peak demand of 50 MVA. There are a total of 40 normally closed switches and 36 normally open switches (all of the switches are manual), and the expected SAIDI value is 120 minutes per year. The geographic layout of this system (including the location of switches) is shown in Figure 7.9.

Each feasible solution for this optimization project is represented by a bitmap, with each bit corresponding to an existing switch on the system. A bit value of zero indicates that the corresponding switch is not automated, and a bit value of one indicates that the corresponding switch is automated.

The objective function is equal to the arithmetic sum of all bits in the bitmap, which is equal to the number of automated switches in the solution. The only constraint considered is a SAIDI target, which is represented as a penalty factor that linearly increases as the degree of the constraint violation increases. For a solution with N_a automated switches, a reliability target $SAIDI_T$ and a reliability value of SAIDI, the objective is to minimize the following:

$$\text{Minimize: } N_a + \begin{cases} 0 & ;\text{SAIDI} <= \text{SAIDI}_T \\ \dfrac{10^9(\text{SAIDI} - \text{SAIDI}_T)}{\text{SAIDI}_T} & ;\text{SAIDI} > \text{SAIDI}_T \end{cases} \tag{7.15}$$

The solution methodology is a simple hill climbing algorithm starting with a random bitmap and using analytical simulation to compute system reliability. This problem was solved for a wide range of SAIDI targets to determine what happens to automation requirements as reliability targets become more aggressive. The required number of automated switches to achieve SAIDI targets from 60 minutes to 120 minutes is shown in Figure 7.10. Solutions for each SAIDI target were computed multiple times to improve the probability of finding high quality solutions.

Multiple trials for the same target SAIDI sometimes result in substantial solution variation. To demonstrate, 20 solutions for a SAIDI target of 90 min/yr were generated using random initial solutions. A histogram of outcomes is shown in Figure 7.10, with the worst solution requiring 23 switches and the best solution requiring 13 switches (the switches corresponding to the best solution are shown Figure 7.9). Since each of these solutions is quite different in terms switch number and location, multiple trials are needed to help ensure that a near globally optimal solution is ultimately selected for implementation.

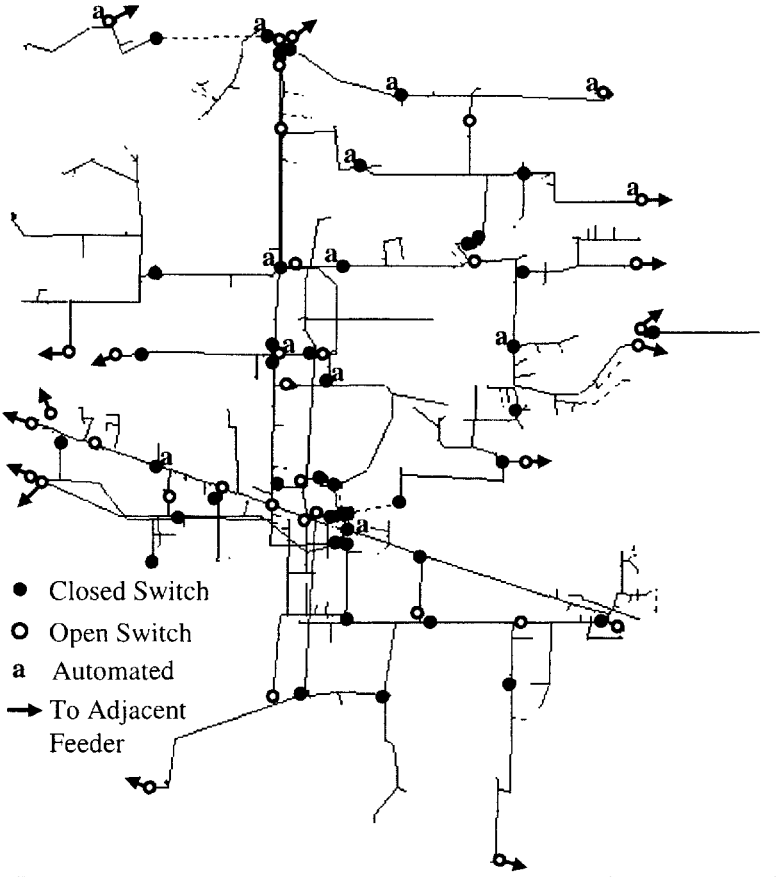


Figure 7.9 Eight-feeder system to demonstrate feeder automation optimization. Using a hill climbing algorithm, it is shown that the original SAIDI of 120 min/yr can be reduced to 90 min/yr by automating 3 of the 36 normally open switches and 10 of the 40 normally closed switches.

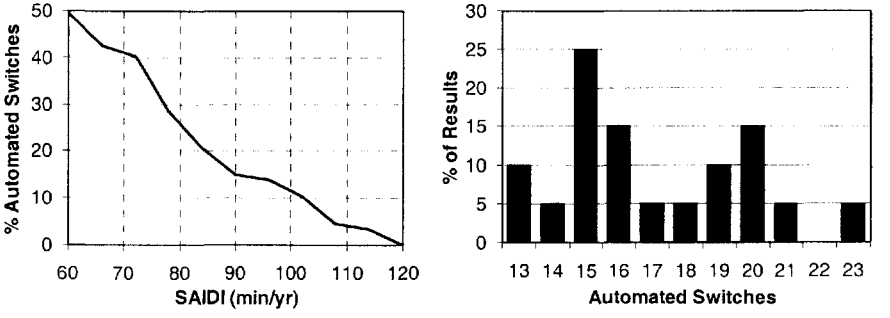


Figure 7.10. The left graph shows the number of automated switches required to improve SAIDI to various levels for the system shown in Figure 7.9. The right graph demonstrates the natural variation of multiple trials by showing a histogram of 20 solutions found for a target SAIDI of 90 min/yr.

This application of optimization techniques to feeder automation demonstrates several points. First, it is not necessarily difficult to formulate an optimization problem. In this case, the objective was simply to meet a SAIDI target using the least possible number of automated switches. Second, it is not necessarily difficult to represent potential solutions as an integer map. In this case, a simple string of bits corresponds to switches being manual or automated. Third, it is not necessarily difficult to implement an effective optimization algorithm. In this case, a simple hill climbing approach achieves good results, especially if multiple trials are performed with different random starting points.

Optimization can be an effective way to approach practical distribution reliability problems. This section has applied optimization to the problem of feeder automation, but many other problems are just as appropriate for optimization and are limited only by the creativity of the distribution engineer. With practice, more sophisticated algorithms can be implemented and more complicated problems (like switching and protection optimization, discussed in the next section) can be addressed.

7.4.2 Reclosing and Switching

Utilities usually have reliability targets for several reliability indices such as $MAIFI_E$, SAIFI and SAIDI. One way to achieve these targets is to place reclosers and switches on the primary distribution system. Reliability can be further improved by automating existing devices and automating new devices. Identifying the optimal combinations and locations of switches, reclosers and automation to achieve multiple reliability targets is a complicated optimization problem that is difficult to solve using simple optimization techniques.

This section applies a genetic algorithm to improve the reliability of an actual utility feeder located in the southern US. Reliability improvement projects can include the installation of new reclosers and the installation of new sectionalizing switches. For an additional cost, these devices can be fitted with automation equipment so that they can be remotely switched by dispatchers. Last, reliability can be improved by retrofitting existing reclosers and sectionalizing switches with automation so that they can be remotely switched by dispatchers. The existing reliability indices of this feeder are: $MAIFI_E$=8.2 /yr, SAIFI=1.2 /yr and SAIDI=133 min/yr. The objective of the optimization algorithm is to achieve the following reliability index targets: $MAIFI_E$=4 /yr, SAIFI=0.5 /yr and SAIDI=45 min/yr. The algorithm attempts to do this for the least possible cost with switches costing $1000, reclosers costing $15,000 and installing automation on any new or existing switch or recloser costing $5000. The formal problem formulation can be written as follows:

Reclosing and Switching Problem Formulation

$$
\begin{aligned}
&\text{Minimize:} \\
&\quad 1000 \cdot N_s + 15000 \cdot N_r + 5000 \cdot N_a \\
&\text{Subject to:} \\
&\quad \text{MAIFI}_E < 4.0 \text{ /yr} \\
&\quad \text{SAIFI} \quad < 0.5 \text{ /yr} \\
&\quad \text{SAIDI} \quad < 45 \text{ min/yr}
\end{aligned}
\tag{7.16}
$$

N_s = Number of new switches
N_r = Number of new reclosers
N_a = Number of devices equipped with new automation equipment

The feeder used in this example, shown in Figure 7.11, serves 1900 customers and has a peak load of nearly 10 MW. It has no line reclosers, no automation equipment, one normally closed switch and six ties to other feeders. It is divided into 300 line segments and 15 cable segments, with each segment being a candidate for a new device. Solutions are represented by using three bits for each existing switch and three bits for each line or cable section as follows:

Bit 1: Is a switch or protection device present? A 0 corresponds to "yes" and a 1 corresponds to "no."
Bit 2: If a device is present, a 0 corresponds to a switch and a 1 corresponds to a recloser.
Bit 3: If a device is present, a 0 corresponds to no automation and a 1 corresponds to automation.

For existing switches, Bit 1 is always equal to 1, a Bit 2 value of 1 represents an upgrade to a recloser and a Bit 3 value of 1 represents the installation of automation equipment. For existing reclosers Bit 1 and Bit 2 are always equal to 1, and a Bit 3 value of 1 represents the installation of automation equipment. For line and cable sections, Bit 2 and Bit 3 are only relevant if Bit 1 has a value of 1, which indicates that a new device has been installed. Since there are a total of 322 possible device locations (7 existing switches + 300 line segments + 15 cable segments), the total solution representation requires $322 \cdot 3 = 966$ bits.

This optimization problem has been solved by using analytical simulation (as described in Chapter 5) to compute system reliability and genetic algorithms (as described in Section 7.2.5) to identify the optimal solution. This algorithm uses random initial population generation, a total population of 500, an elite population of 10, a mutation rate of 0.5% per bit per generation, and a stopping criterion of 10 generations without improvement of the best solution. The best solution has a cost of $66,000 and results in the indices shown in Table 7.3.

Table 7.3. Reliability indices associated with the reclosing and switching optimization problem.

Index	Present	Target	Optimized
$MAIFI_E$	8.2	4.0	3.8
SAIFI	1.2	0.5	0.4
SAIDI	133	45	45

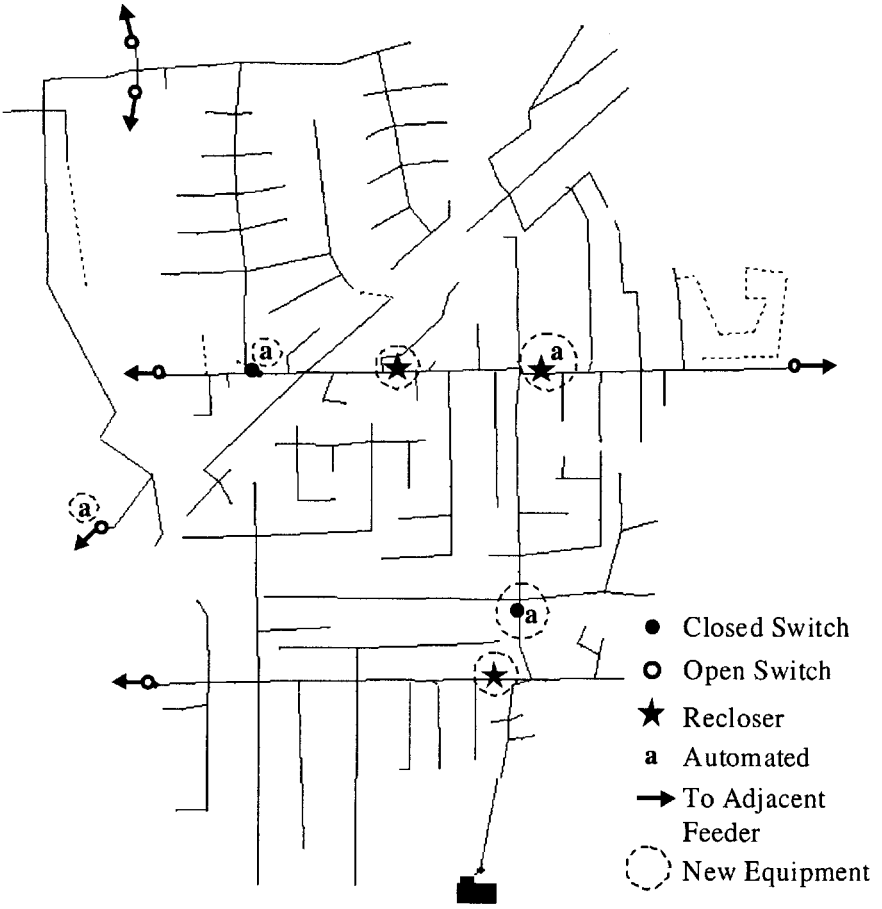


Figure 7.11. Feeder used for the reclosing and switching optimization problem. The present system has no line reclosers, one normally closed switch and six ties to adjacent feeders. The least cost solution that meets all reliability index targets, identified by a genetic algorithm, automates the existing normally closed switch, automates an existing tie switch, adds two new reclosers, adds one new automated recloser and adds one new automated switch.

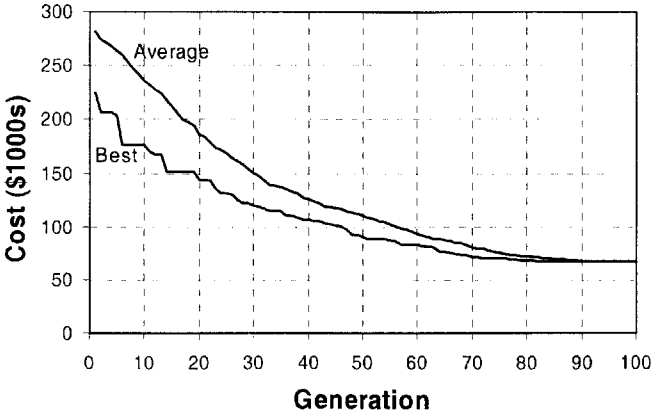


Figure 7.12. Convergence of the genetic algorithm for reclosing and switching optimization. As the average cost of all solutions approaches the cost of the best solution, genetic diversity is lost and further improvements can only be achieved through random mutations.

Convergence of a genetic algorithm can be examined by tracking the objective function score of both the best solution and the average of all solutions for each generation. The genetic algorithm is considered to have converged when average solution quality approaches the quality of the best solution. At this point, it is likely that most genetic diversity has been lost and additional improvements will only occur due to random mutations. The curves for this example, shown in Figure 7.12, indicate that convergence occurs after approximately 90 generations.

The least cost solution identified by the genetic algorithm is shown in Figure 7.11. It consists of automating the existing normally closed switch, automating the tie switch furthest away from the substation, adding two new reclosers, adding one new automated recloser and adding one new automated switch. These actions just meet the SAIDI target and slightly exceed the SAIFI and $MAIFI_E$ targets.

The solution identified by the genetic algorithm is non-obvious and would be nearly impossible to identify through manual methods. In fact, the solution identified by the genetic algorithm significantly outperformed simple hill climbing. While the genetic algorithm identified a solution with a cost of $66K, ten hill climbing runs with random initial bitmaps resulted in costs of $117K, $123K, $139K, $167K, $182K, $192K, $267K, $270K, $272K and $360K. Each hill climbing run resulted in a different locally optimal solution, with the best solution nearly twice as expensive as the genetic algorithm solution. For difficult optimization problems with complex solution spaces, sophisticated techniques such as genetic algorithms clearly outperform more simple methods such as hill climbing.

7.4.3 System Reconfiguration

A distribution system can be reconfigured by changing the location of normally open switches. This changes the allocation of customers and the flow of power for the affected feeders. Feeder reconfiguration methods have been developed to minimize losses[30-31] and operational costs[32]. This section demonstrates how to use the feeder reconfiguration concept to improve system reliability by using an annealed local search.

The algorithm described in this section is designed to optimize radially operated distribution systems. A radially operated system may be highly interconnected, but at any particular time, a component will have a unique electrical path back to a distribution substation. In a radial system, the direction of power flow is well defined. Figure 7.13 shows an example of a radial tree structure consisting of a component (A) with a parent (B) and two children (C, D). The parent/child structure is important since its modification is the goal of the feeder reconfiguration process.

A radial system is modeled by representing components as objects and parent/child relationships as pointers between objects. Each component has a pointer to its parent, and a list of pointers to its children. This data structure allows for the efficient implementation of power flow and reliability algorithms using tree navigation techniques. Upstream components such as protection devices and sources can be found by following parent pointers. Downstream components such as interrupted customers can be found using depth-first and breadth-first search techniques.

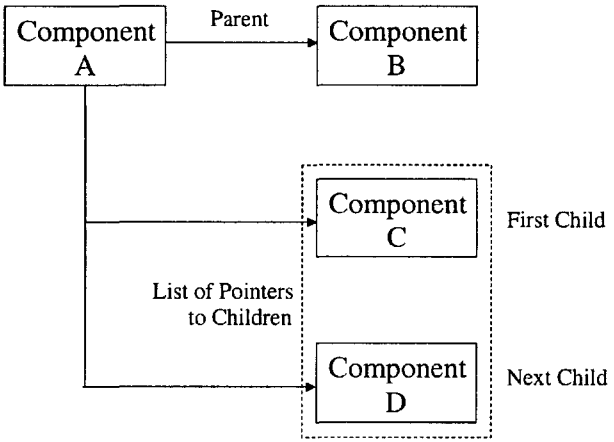


Figure 7.13. In a radial tree structure, each component is described by a parent and a set of children. The parent is the component next closest to the source of power and the children are the next furthest components from the source of power. System reconfiguration is accomplished by modifying parent/child relationships of components.

The optimization function for feeder reconfiguration is defined as a weighted sum of reliability indices. The goal of the feeder reconfiguration is to minimize this function subject to equipment overloads and voltage violations:

System Reconfiguration Problem Formulation

Minimize:

w_1SAIDI + w_2SAIFI + w_3MAIFI (7.17)

w_x: Reliability Index Weight

Subject to:

No equipment overloads

No voltage drop violations

The reliability index weights are chosen to reflect utility targets, performance based rate structures or relative cost to customers. Constraints are implemented as penalty factors added to the optimization function. The penalty factor formulation for loading violations and voltage violations is:

$$pf_{lv} = \sum_i \max\left[\frac{kVA_i - Rated\_kVA_i}{Rated\_kVA_i}, 0\right] \quad (7.18)$$

$$pf_{vv} = \sum_i \max\left[\frac{Min\_Voltage_i - Voltage_i}{Min\_Voltage_i}, 0\right] \quad (7.19)$$

pf_{lv} = Penalty Factor due to loading violations

pf_{vv} = Penalty Factor due to voltage violations

Feeder reconfiguration using existing switch locations is a discrete optimization problem. Simulated annealing and genetic algorithms, however, are problematic to apply to radial structure reconfiguration problems for two reasons. First, most of the generated switch position combinations will not represent feasible solutions. Second, generating a new radial tree structure for each combination of switch positions is computationally intensive. A local search avoids the problems of simulated annealing and genetic algorithms by making small changes in the radial structure by closing a normally open switch and opening a nearby upstream switch. This process, referred to hereafter as a tie switch shift, allows incremental changes to the tree structure as opposed to complete reformulation of the structure for each switch position change.

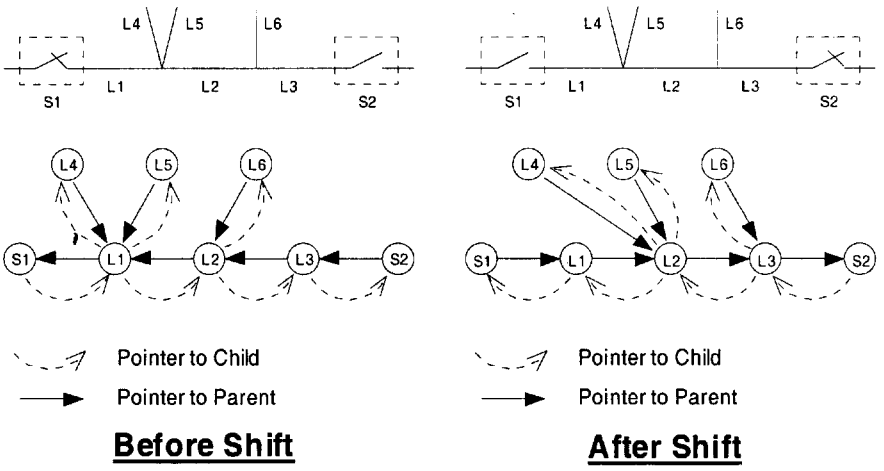


Figure 7.14. Radial tree structure before and after a tie switch shift. Before the shift, power flows from switch S1 to switch S2. After the open point is shifted from switch S2 to switch S1, the parent and child relationships associated with lines L1-L3 are reversed to reflect the shift in power flow. Similarly, the locations at which lateral branches L4-L6 are connected to the main trunk are shifted in what was a downstream direction before the shift.

To illustrate a tie switch shift, consider the simple system in Figure 7.14. It consists of a normally closed switch (S1), a normally open switch (S2), and six line segments (L1-L6). The radial tree structure is shown by solid arrows (pointers to parents) and dotted arrows (pointers to children). **P** is defined as the set of components on the path from S1 to S2 and **D** is defined as the set of components directly connected to **P**. In Figure 2, **P** $\in$ (L1,L2,L3) and **D** $\in$ (L4,L5,L6).

Consider a tie switch shift from S2 to S1. Prior to the shift, power flows from S1 to S2. After the shift, a reversal occurs and power flows from S2 to S1. This requires the following changes to the radial tree structure:

Radial Tree Structure Changes After a Tie Switch Shift

1. For P_1 and P_2 in **P** and a pointer $P_1 \rightarrow P_2$, reverse the connection to become $P_2 \rightarrow P_1$.
2. For P_1 in **P**, D_1 in **D** and a child pointer $P_1 \rightarrow D_1$, alter the pointer to become $P_2 \rightarrow D_1$ where P_2 is the new parent of P_1.
3. For P_1 in **P**, D_1 in **D** and a parent pointer connecting $D_1 \rightarrow P_1$, change the pointer to become $D_1 \rightarrow P_2$ where P_2 is the new parent of P_1.

After performing a tie switch shift, the system shown on the left side of Figure 7.14 is reconfigured to become the system shown on the right side of Figure 7.14. Power now flows in the reverse direction and terminates at the new normally open point.

The tie switch shift is well suited for hill climbing, which could simply test tie switch shifts and keep changes if the solution improves; continuing until a local optimum is reached. Unfortunately, hill climbing has difficulty identifying solutions that differ substantially from the initial condition since there is a high probability of becoming trapped in a local optimum. The concept of a local annealed search overcomes the limitations of hill climbing, allowing a greater search area to be explored by probabilistically accepting certain solutions with worse objective functions.

The primary variable that drives a local annealed search is temperature, T: $T \in [0,1]$. The temperature of a system represents the probability that a worse solution will be accepted. A temperature of 1 generalizes to an undirected random search and a temperature of zero reduces to simple hill climbing.

A local annealed search is characterized by two parameters: initial temperature T_0 and annealing rate R. The initial temperature determines how freely the algorithm will initially traverse the solution space, and the annealing rate determines the rate at which the temperature is reduced. When the temperature keeps dropping without the solution changing, the process is frozen and the algorithm terminates. The annealed local search algorithm is outlined below.

Annealed Local Search

1. Set $T = T_0$: $T_0 \in [0,1]$
2. Compute the objective function Ω_{best}
3. Select a tie switch, shift the tie switch in Direction 1, and compute the objective function Ω_{test}
4. Let R be a uniform random number: $R \in [0,1]$
5. If $\Omega_{test} < \Omega_{best}$ or $T > r$, set $\Omega_{best} = \Omega_{test}$ and go to step 8.
6. Shift the tie switch back to its previous position
7. Repeat steps 3-7 for Direction 2
8. Repeat steps 3-8 for all untested tie switches
9. Set $T = R \cdot T$
10. If Ω_{best} has changed since the last change in T, go to step 3
11. End

The annealed local search probabilistically allows system reconfigurations that result in worse reliability and/or violate constraints. Occasionally accepting worse solutions allows a greater area of the search space to be explored and increases the likelihood of a near-optimal solution being discovered. Local optimality is assured since the annealed local search reduces to hill climbing at a zero temperature.

The characteristics of an annealed local search have been examined on an eight-feeder test system. Each feeder is a multi-branch layout[33] and the total system consists of 576 load points, 1865 components, 208 fused laterals, and 200 switches. The test system is shown in Figure 7.15.

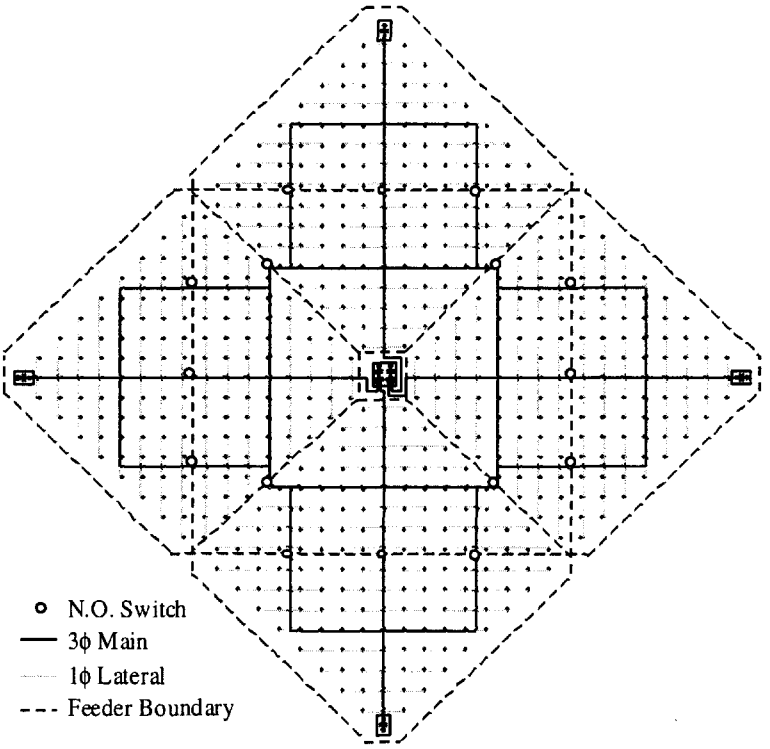


Figure 7.15. Eight-feeder test system used to demonstrate an annealed local search to optimize the location of normally open switches. This system consists of 16 open points and more than 200 switch locations, resulting in more than 10^{36} possible system configurations.

The feeder boundaries shown in Figure 7.15 represents the most reliable configuration for this system if all feeders are topologically identical and all loads are equal. The optimal system configuration changes if the loading patterns are altered sufficiently. To create a system for optimization, each feeder was assigned a random average loading level and each load point assigned a random demand ranging from zero to twice the average demand. The randomization resulted in two of the feeders being overloaded and provides an opportunity for optimization.

The impact of annealing rate on solution quality was assessed by performing ten trials for annealing rates between 0 and 0.99. The initial temperature was set equal to the annealing rate so that fast annealing rates tended to explore solutions similar to the initial conditions, and slow annealing rates tended to explore a wide range of solutions. Results for optimizing SAIDI (w_1=1, w_2=0, and w_3=0 in Eq. 7.17) are shown in Figure 7.16.

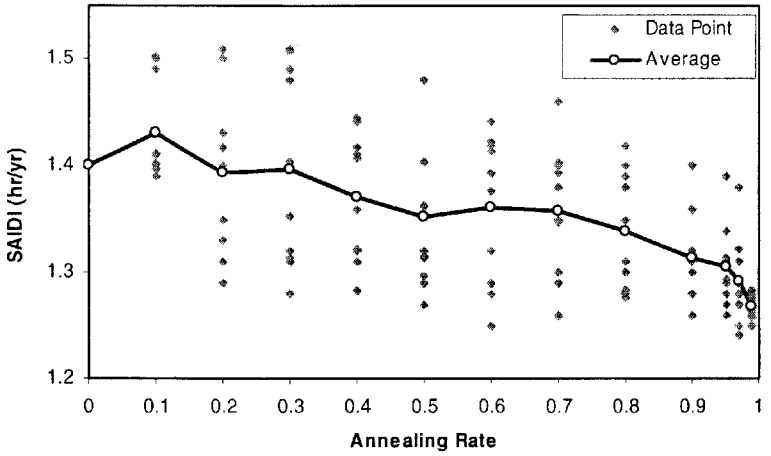


Figure 7.16. Solution quality versus annealing rate for the 8-feeder test system. Both solution quality and solution variation become better as the annealing rate becomes slower. Based on this graph, an annealing rate of 0.99 can be used to ensure a high quality solution with a single trial.

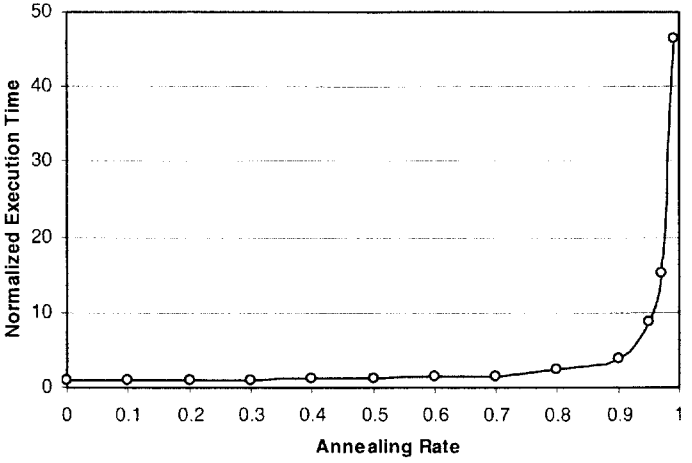


Figure 7.17. A graph of execution time versus annealing rate for the 8-feeder test system shows that the time required to produce a solution rises hyperbolically as the annealing rate approaches unity.

Figure 7.16 shows that average solution quality improves as the annealing rate increases and shows that the variance of solution quality remains high until very slow annealing rates are used. The disadvantage of high annealing rates is increasing execution time. A normalized plot of the of average execution time versus annealing rate is shown in Figure 7.17. The plot is normalized such that the execution time for an annealing rate of zero is equal to unity. The plot shows a hyperbolic relationship as the annealing rate approaches unity.

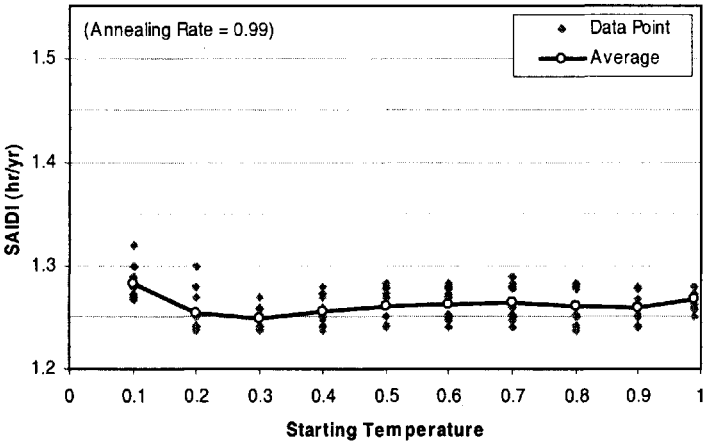


Figure 7.18. A graph of solution quality versus starting temperature for the 8-feeder test system shows that a starting temperature of 0.3 is sufficient to produce high quality results when an annealing rate of 0.99 is used.

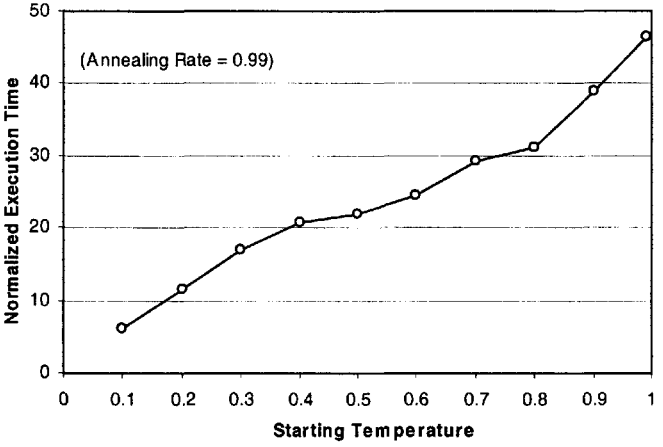


Figure 7.19. A graph of execution time versus starting temperature shows that the time required to produce a solution rises linearly with starting temperature. By choosing a starting temperature of 0.3 rather than 0.9, execution time can be reduced by approximately 67%.

Based on Figure 7.16, an annealing rate of 0.99 was selected to provide reasonable assurance of a high quality solution with a single trial. Similar trials were performed to determine the minimum required initial temperature, T_0, which consistently produces a quality solution (a low initial temperature being desirable to reduce execution time). Ten trials were performed for initial temperatures between 0.1 and 0.99. Solution quality versus T_0 is shown in Figure 7.18 and execution time versus T_0 is shown in Figure 7.19. Solution quality and solution

variance do not significantly improve for initial temperatures greater than 0.3. Execution time increases linearly for increasing starting temperatures.

The annealed local search methodology has been applied to seven actual distribution systems in order to test the ability of the algorithm to optimize reliability through feeder reconfiguration. When applying the annealed local search to actual systems, optimization parameters must be selected *a priori*. Based on the test system results, an annealing rate of 0.99 and a starting temperature of 0.3 were selected for the analyses on actual systems. These values are shown to result in high quality solutions for a single trial while avoiding unnecessary computation time. Brief descriptions of these topologically diverse systems are now provided.

Urban 1 — This system serves a commercial and hotel district in a large southeastern coastal city. It is served by an aging underground system and is moderately loaded.

Urban 2 — Serves an industrial and commercial district in a large midwestern city. It is served by an aging underground system and is heavily loaded.

Suburban 1 — This system serves a newly developing suburban area in the Southeast. There are several moderately sized industrial customers served by this system.

Suburban 2 — This system serves a quickly growing suburban area in the Southeast that is primarily residential.

Suburban 3 —This system serves a suburban area in the Southeast. The infrastructure is lightly loaded in most areas, but is heavily loaded in an area that serves several industrial customers.

Rural — This system serves a rural area in the Southeast. Load density is low and most of the circuit exposure is single phase overhead construction.

Institutional —This system serves a large institutional campus in a coastal area of the Southeast. All buildings are served from a single substation and the site is served through underground feeders and pad-mounted equipment.

These seven test systems are diverse in terms of topology, load density, construction type and customer type. The number of normally open and normally closed switches also varies considerably, resulting in varying solution space sizes (solution space grows as the number of switches grows). Larger solution spaces require more time to explore, but increase the likelihood of finding higher quality solutions. Key characteristics of the test systems are summarized in Table 7.4.

Table 7.4. Test system characteristics for optimizing system configuration.

System	Circuits	Load (MVA)	Circuit Miles	Switches
Urban 1	29	223	20.4	295
Urban 2	31	193	86.3	242
Suburban 1	15	212	545.7	102
Suburban 2	21	267	545.6	151
Suburban 3	13	85	204.9	61
Rural	8	144	496.0	46
Institutional	7	108	52.1	345

Table 7.5. Test system results for optimizing system configuration.

System	Original SAIDI (hr/yr)	Optimized SAIDI (hr/yr)	Reduction
Urban 1	1.81	1.55	14.4%
Urban 2	0.86	0.74	13.8%
Suburban 1	4.31	4.08	5.3%
Suburban 2	4.24	4.05	4.5%
Suburban 2	10.84	10.31	4.9%
Rural	5.72	5.68	0.7%
Institutional	1.23	1.20	2.4%

For each test system, the algorithm is instructed to optimize SAIDI (corresponding to weights of w_1=1, w_2=0 and w_3=0). The results of the reliability optimization for the seven test systems are summarized in Table 7.5.

The annealed local search is able to find configurations with higher reliability for each of the seven test systems without violating voltage and loading constraints. This improvement ranges from less than 1% to nearly 15%. In general, higher density systems result in greater reliability improvement. This is true for load density (MVA per circuit mile), switch density (switches per circuit mile) and switches per MVA of load. With the exception of the institutional system, the relationship appears to be nonlinear for the first two measures and linear with respect to switches per MVA of load. The sample set is small and further investigation is required to determine whether the relationship is evidenced in a broader array of systems.

Feeder reconfiguration can improve reliability for little or no capital expenditure. As such, feeder reconfiguration for reliability optimization presents electric utilities with an opportunity to become more competitive by providing higher levels of reliability at a lower cost. The results seen on these seven test systems are encouraging. In an era of deregulation and increasing customer reliability needs, utilities are under pressure to improve reliability without increasing cost. For certain types of systems, the annealed local search is able to improve reliability by nearly 15% without any capital spending. Assuming a conservative capital reliability improvement cost of $10/kVA-hr, reconfiguration optimization results in a reliability improvement that could cost nearly $580,000 for the "Urban 1" system.

7.4.4 Comparison of Methods

When selecting an optimization algorithm for distribution reliability problems, it is best to select the simplest method that will produce reasonably good answers in an acceptable amount of time[26]. To do this, a basic feel for the performance differences between various optimization methods is required. This section applies various algorithms to the same optimization problem so that these types of comparisons can be made. It begins by defining an optimization problem and a test system, continues by comparing hill climbing, simulated annealing and genetic algorithms, and concludes by examining the impact of probabilistic initialization using fuzzy rules.

The problem that will be used to benchmark algorithms is to optimize the detailed design of a pre-routed primary distribution system[34]. The solution must identify protection device locations, switch locations, tie point locations, automation strategies and underground versus overhead strategies so that overall cost is minimized. This overall cost, referred to as the total cost of reliability (TCR), is equal to the sum of the utility cost of reliability (UCR) and the customer cost of reliability (CCR):

$$TCR = UCR + CCR \tag{7.20}$$

The utility cost of reliability consists of annual operation and maintenance costs and annualized capital costs. The customer cost of reliability is equal to the sum of all customer incurred costs ($C_{incurred}$) due to momentary and sustained interruptions. This is based on both the customer's cost of interrupted power (C_{kW}, in \$/kW) and the customer's cost of interrupted energy (C_{kWh}, in \$/kWh). For a customer with an average load of P (in kW), CCR can be modeled as:

$$CCR = \sum_{i=1}^{N} P\left(C_{kW} + t_i \cdot C_{kWh}\right) \tag{7.21}$$

N: the number of interruptions
t_i : the duration of interruption i

The total cost of reliability is used as the objective function to be minimized subject to design constraints. The constraint considered in this benchmark problem is the coordination of protection devices—only three protection devices can be placed in series while maintaining coordination. If a design violates a coordination constraint, a penalty factor is added to the TCR to make it more expensive than the TCR of any valid design.

Since distribution system design is a discrete optimization problem, it is convenient to represent potential solutions as bitmaps. The bitmap for this prob-

lem is created by dividing the distribution system into line sections and assigning 7 bits to represent information concerning each section. Each of these bits is characterized by a question. If the bit value is zero, the answer to the question is "no." If the bit value is one, the answer to the question is "yes." The following bits and their corresponding questions are used:

Bit 1: Is the section underground?
Bit 2: Is a normally closed switch present?
Bit 3: Is the normally closed switch automated?
Bit 4: Is a protection device present?
Bit 5: Is the protection device a recloser?
Bit 6: Is a normally open switch present?
Bit 7: Is the normally open switch automated?

It is important to note that there is a hierarchy in the above bits. For example, Bit 7 will have no impact on a design unless Bit 6 has a value of one. Similarly, Bit 5 will have no impact on a design unless both Bit 2 and Bit 4 have a value on one.

This benchmark optimization problem requires feeder routing information. In addition, feasible tie points must be established. These are points on the feeder where it is feasible to build tie lines and install a normally open switch so that loads can be back fed during contingencies. To ensure that the benchmarking results are based on a realistic system, the routing and feasible tie points of the test system are based on an actual distribution system in the Pacific Northwest, shown in Figure 7.20.

In addition to routing information, both utility cost information and customer cost information are needed. The utility cost data used is summarized in Table 7.6 and the customer cost data used is summarized in Table 7.7. Reliability data used are taken from the recommended values presented in Chapter 4.

This optimization problem is now solved using hill climbing, simulated annealing and genetic algorithms. A description of each methodology is now provided:

Hill Climbing — The hill climbing algorithm uses random initial bit values, tests each bit sequentially and immediately accepts changes that improve the objective function score. It is executed 1000 times to obtain a distribution of results.

Simulated Annealing — The simulated annealing algorithm uses random initial bit values, an initial temperature of 0.9, an annealing rate of 0.99 and a probability of accepting detrimental changes equal to the current temperature. Hill climbing is performed on all frozen solutions to ensure local optimality. It is executed 1000 times to obtain a distribution of results.

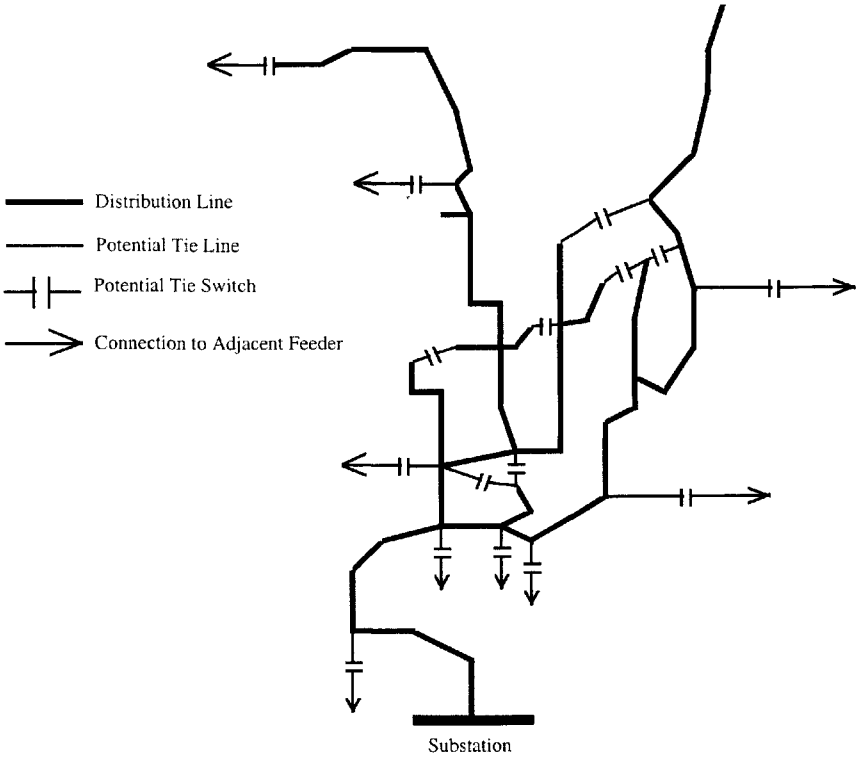


Figure 7.20. Test system primary feeder routing and feasible tie points. This test system is based on an actual utility feeder in the Pacific Northwest. The goal of the optimization problem is to identify protection device locations, switch locations, tie point locations, automation strategies and underground versus overhead strategies so that the sum of utility cost and incurred customer cost is minimized.

Table 7.6. Annualized equipment cost data.

Device	Base Cost ($/year)	Automation Cost ($/year)
Sectionalizing Switch	1,358	4,939
Tie Switch	1,358	2,488
Recloser	3,771	3,071
Fuse	1,553	4,939
OH Line (per mile)	33,937	--
UG Line (per mile)	238,918	--

Table 7.7. Incurred customer cost data.

Customer Type	% of Load	C_{kW} ($/kW)	C_{kWh} ($/kWh)
Residential	50	0.01	2.68
Commercial	20	1.19	25.61
Office Buildings	20	6.44	20.09
Small Users	10	1.00	5.08
Total System	**100**	**1.63**	**10.99**

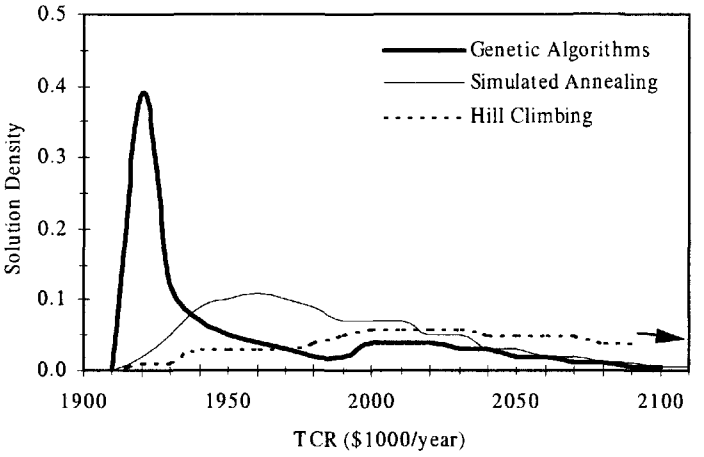


Figure 7.21. Comparison of results for three optimization methodologies. Hill climbing results in a broad array of locally optimal solutions that vary widely in total cost. Simulated annealing is less sensitive to initial conditions than integer programming, but still results in a wide range of costs. Genetic algorithms take advantage of multiple test cases by sharing information and results in better overall solution quality when compared to the other two techniques.

Genetic Algorithms — The genetic algorithm uses random initial bit values, a total population of 1000, an elite population of 50, a mutation rate of 1% per bit per generation and stops when the best member does not change for 50 generations. Hill climbing is performed on all members of the final generation to ensure local optimality. Each of the members in the final population is kept to show a distribution of results.

A summary of the solutions resulting from hill climbing, simulated annealing and genetic algorithms are shown in Figure 7.21. Integer Programming, when presented with 1000 different initial conditions, resulted in 1000 different locally optimal solutions. This illustrates the complexity of the search space, and shows that integer programming is highly sensitive to initial conditions. Simulated annealing is less sensitive to initial conditions than integer programming, but still results in a wide range of costs, requiring many trials to be performed before a distribution engineer would be confident that a near-optimal solution has been found. Genetic algorithms take advantage of multiple test cases by simultaneously running trials and constantly sharing information among trials through genetic crossover. This improves computational efficiency and results in better overall solution quality when compared to hill climbing and simulated annealing.

Up to this point, all optimization techniques have used random initialization, which essentially gives an algorithm a random starting solution from which to begin its search. This begs the question, "Can solution quality be improved by

providing starting locations that are more likely to contain elements of an optimal design when compared to random starting locations? If so, how can these new starting locations be generated?"

To date, distribution system design is performed almost exclusively by distribution planners and engineers using a few analytical tools and a large number of heuristic rules. This method cannot optimize a design, but the heuristic rules used certainly contain information about good designs. In this situation, where optimal designs are desired and large amounts of expert knowledge exist, it is often beneficial to combine optimization techniques and expert rules into a hybrid optimization method[35-36]. Since the knowledge relating to distribution system reliability optimization tends to be vague in nature, fuzzy rules are selected and used to probabilistically create distribution system designs that correspond to heuristic knowledge[37-38]. These designs are then used as initial solutions in an attempt to improve the performance of hill climbing, simulated annealing and genetic algorithms.

Before fuzzy rules can be generated, it is necessary to define the fuzzy sets that will be used by the fuzzy rules. For this particular problem, five triangular fuzzy sets will be used: very small, small, medium, large, and very large. These sets are shown in Figure 7.22. Fuzzy rules will then be used to create a design on a bit by bit basis. Since each bit corresponds to a particular section, information about this section is needed to accomplish this task.

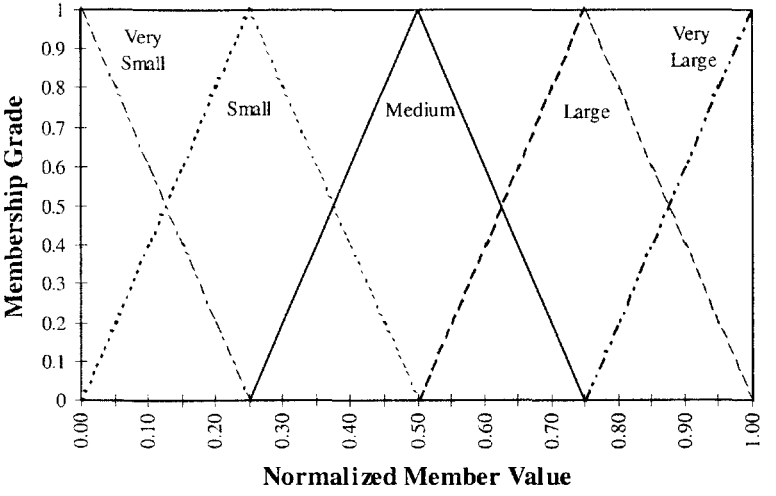


Figure 7.22. Fuzzy sets used for probabilistic initialization of optimization problems. In this case, five triangular functions are used to represent normalized values between 0 and 1. The membership grade of a number x in fuzzy set **S** corresponds to the truth of the statement "x is **S**." For example, the statement "0.6 is Small" has a truth of 0.0, the statement "0.6 is Medium" has a truth of 0.6 and the statement "0.6 is Large" has a truth of 0.4.

After examining many low-cost designs for various system and reliability data, the following features have been chosen as inputs for fuzzy rules (all features are represented by values between zero and one and can be described using the fuzzy sets shown in Figure 7.22):

Downstream Load (DSL) — This value is a measure of how much connected load is downstream of the section being considered. A per unit value is obtained by dividing the total load downstream of the section (including the load on the section) by the total feeder load.

Number of Siblings (SIB) — This is a measure of the branching that is occurring at the beginning of the section being considered. Since the maximum number of branches stemming from a node on the system being considered is 3, a per-unit value is obtained by dividing the number of other sections stemming from the upstream node of the section by 2.

Overhead Line Failure Rate (FR) — This is a measure of the per-mile failure rate of a section if it were to be of overhead construction. A normalized measure is obtained by dividing the overhead line failure rate of the section (in failures/mile/year) by 0.3.

Load Density (LD) — This is a measure of the overall loading of the system. A normalized measure is obtained by dividing the load density of the system (in kW/mile) by 2000.

Tie Point Quality (TPQ) — This is a measure of how often a tie point will be able to connect an interrupted section to a source via an alternate path. A per-unit measure is obtained by dividing the length of the overlapping part of the normal path and back feedable path by the normal path length.

Not all of the above characteristics are needed when examining each bit. For example, deciding whether or not to place a tie switch on a section requires different information than deciding whether to make the section overhead or underground. In addition, the number of fuzzy inputs should be kept to a minimum since the number of fuzzy rules required increases exponentially with the number of inputs (if there are n fuzzy input sets and x fuzzy inputs, the number of rules required is n^x). The following is a description of each bit and the corresponding fuzzy inputs that have been chosen:

Bit 1— Is the section underground? Low cost designs tend use underground cable if overhead line failure rate is high and if the system load density is high. Because of this reason, FR and LD are the fuzzy inputs used to determine Bit 1.

Bit 2 — Is a normally closed switch present? The number of normally closed switches on low cost designs increases as load density increases. In addition, there is a tendency to place normally closed switches on branching sections. Because of these observations, SIB and LD are the fuzzy inputs used to determine Bit 2.

Bit 3 — Is the normally closed switch automated? Low cost designs tend to automate normally closed switches that are close to the substation and to not automate normally closed switches that are far away from the substation. The number of automated switches also tends to increase as load density increases. Because of these observations, DSL and LD are the fuzzy inputs used to determine Bit 3.

Bit 4 — Is a protection device present? Low cost designs tend to place protection devices at points of branching. In addition, the number of protection devices increases with load density. Because of these observations, SIB and LD are the fuzzy inputs used to determine Bit 4.

Bit 5 — Is the protection device a recloser? Low cost designs tend to place reclosers near the substation. In addition, the number of reclosers increases with load density. Because of these observations, DSL and LD are the fuzzy inputs used to determine Bit 5.

Bit 6 — Is a normally open switch present? Low cost designs tend to place normally open switches towards the end of the feeder. It is also important for this location to have a high probability of connecting the section to a source via an alternate path during a fault. Because of these observations, DSL and TPQ are the fuzzy inputs used to determine Bit 6.

Bit 7 — Is the normally open switch automated? The most important determiner of whether a tie switch should be automated or not is tie point quality. In addition, more tie switches are automated as load density increases. For these reasons, TPQ and LD are the fuzzy inputs used to determine Bit 7.

Since there are 5 fuzzy sets and two fuzzy inputs corresponding to each bit, there are 25 rules associated with each bit. This rule will have an antecedent (the "if" portion of the rule) and a consequence (the "then" portion of the rule). An example of a fuzzy rule corresponding to a section being underground is:

Example fuzzy rule — If the failure rate of the section is medium and the load density of the system is very large, then the probability that the section should be underground is medium.

To determine the value assigned to a particular bit, the truth of each of its associated rule antecedents is computed. The fuzzy set corresponding to each rule consequence is then weighted by its antecedent truth. All 25 weighted consequences are then added together and the center of mass is computed. This number, a value between 0 and 1, can then be used to probabilistically determine the state of a bit. For example, if the fuzzy output number of a bit is 0.8, then that bit will have a probability of 0.8 of having a value of 1 and a probability of 0.2 of having a value of 0. Using this process, many design solutions can be quickly generated and used for initial solutions in hybrid optimization methods. The fuzzy rules corresponding to Bit 1 of this problem are shown in Table 7.8. A complete list of rules can be found in Reference 39.

A comparison of hill climbing solutions for random versus fuzzy initialization is shown in the top graph of Figure 7.23. Solution quality does improve for fuzzy initialization, but only slightly. The solution variation is still large and many trials will be needed to insure that a near-optimal solution has been found.

A comparison of simulated annealing solutions for random versus fuzzy initialization is shown in the middle graph of Figure 7.23. As with integer programming, solution quality improves, but only marginally. This result is somewhat expected since simulated annealing is relatively insensitive to initial conditions.

A comparison of genetic algorithm solutions for random versus fuzzy initialization is shown in the bottom graph of Figure 7.23. Unlike integer programming and simulated annealing, fuzzy initialization has a profound impact on genetic algorithms. By providing the genetic algorithm good initial genetic information, lower cost solutions are more likely to be found.

Assessment method performances are summarized in Table 7.9. It is evident that genetic algorithms result in lower cost solutions when compared to integer programming and simulated annealing. It is further shown that using fuzzy initialization in genetic algorithms improves solution quality, reduces the standard deviation of solutions, and even reduces computation time. (The average CPU time taken to perform integer programming with random initialization is given a relative CPU time of 1. All other methods are compared to this norm.)

In summary, this section has compared the effectiveness of optimization techniques for a complex distribution system reliability design problem. It initially compared the performance of hill climbing, simulated annealing and genetic algorithms and showed that genetic algorithms result in lower cost solutions and lower variability in results. It then examined the impact on solution quality of replacing random initialization with probabilistic initialization based on heuristic knowledge encoded as fuzzy rules.

Table 7.8. Rules for Bit 1, which determines the probability of a section initially being specified as underground. In this case, rules are based on the failure rate of the section (FR) and the load density of the overall system (LD). These features are described by the following five fuzzy sets: very small (VS), Small (S) Medium (M), Large (L) and Very Large (VL). Since there are two features and five fuzzy sets, there are twenty-five total rules corresponding to this bit (5^2). The probability of Bit 1 being equal to one, P(1), is equal to center of mass of the 25 fuzzy consequences weighted by their respective antecedent truths.

Rule 1-1:	If	FR	is	VS	and	LD	is	VS	then	P(1)	is	VS
Rule 1-2:	If	FR	is	VS	and	LD	is	S	then	P(1)	is	VS
Rule 1-3:	If	FR	is	VS	and	LD	is	M	then	P(1)	is	VS
Rule 1-4:	If	FR	is	VS	and	LD	is	L	then	P(1)	is	VS
Rule 1-5:	If	FR	is	VS	and	LD	is	VL	then	P(1)	is	S
Rule 1-6:	If	FR	is	S	and	LD	is	VS	then	P(1)	is	VS
Rule 1-7:	If	FR	is	S	and	LD	is	S	then	P(1)	is	VS
Rule 1-8:	If	FR	is	S	and	LD	is	M	then	P(1)	is	VS
Rule 1-9:	If	FR	is	S	and	LD	is	L	then	P(1)	is	S
Rule 1-10:	If	FR	is	S	and	LD	is	VL	then	P(1)	is	S
Rule 1-11:	If	FR	is	M	and	LD	is	VS	then	P(1)	is	VS
Rule 1-12:	If	FR	is	M	and	LD	is	S	then	P(1)	is	VS
Rule 1-13:	If	FR	is	M	and	LD	is	M	then	P(1)	is	VS
Rule 1-14:	If	FR	is	M	and	LD	is	L	then	P(1)	is	S
Rule 1-15:	If	FR	is	M	and	LD	is	VL	then	P(1)	is	M
Rule 1-16:	If	FR	is	L	and	LD	is	VS	then	P(1)	is	VS
Rule 1-17:	If	FR	is	L	and	LD	is	S	then	P(1)	is	VS
Rule 1-18:	If	FR	is	L	and	LD	is	M	then	P(1)	is	S
Rule 1-19:	If	FR	is	L	and	LD	is	L	then	P(1)	is	M
Rule 1-20:	If	FR	is	L	and	LD	is	VL	then	P(1)	is	L
Rule 1-21:	If	FR	is	VL	and	LD	is	VS	then	P(1)	is	VS
Rule 1-22:	If	FR	is	VL	and	LD	is	S	then	P(1)	is	S
Rule 1-23:	If	FR	is	VL	and	LD	is	M	then	P(1)	is	M
Rule 1-24:	If	FR	is	VL	and	LD	is	L	then	P(1)	is	L
Rule 1-25:	If	FR	is	VL	and	LD	is	VL	then	P(1)	is	VL

Table 7.9. Comparison of optimization algorithm performance. Genetic algorithms result in lower cost solutions when compared to integer programming and simulated annealing. In addition, fuzzy initialization in genetic algorithms improves solution quality, reduces the standard deviation of solutions, and slightly reduces computation time. Relative CPU time is the ratio of CPU time to the average CPU time taken to perform integer programming with random initialization.

Algorithm	Mean TCR ($1000/yr)	Standard Deviation (% of mean)	Relative CPU Time
Hill climbing			
Random Initialization	2,061	4.57	1.00
Fuzzy Initialization	2,039	3.50	1.09
Simulated Annealing			
Random Initialization	1,970	2.09	1.62
Fuzzy Initialization	1,968	2.11	1.60
Genetic Algorithms			
Random Initialization	1,944	2.50	1.51
Fuzzy Initialization	1,915	1.37	1.43

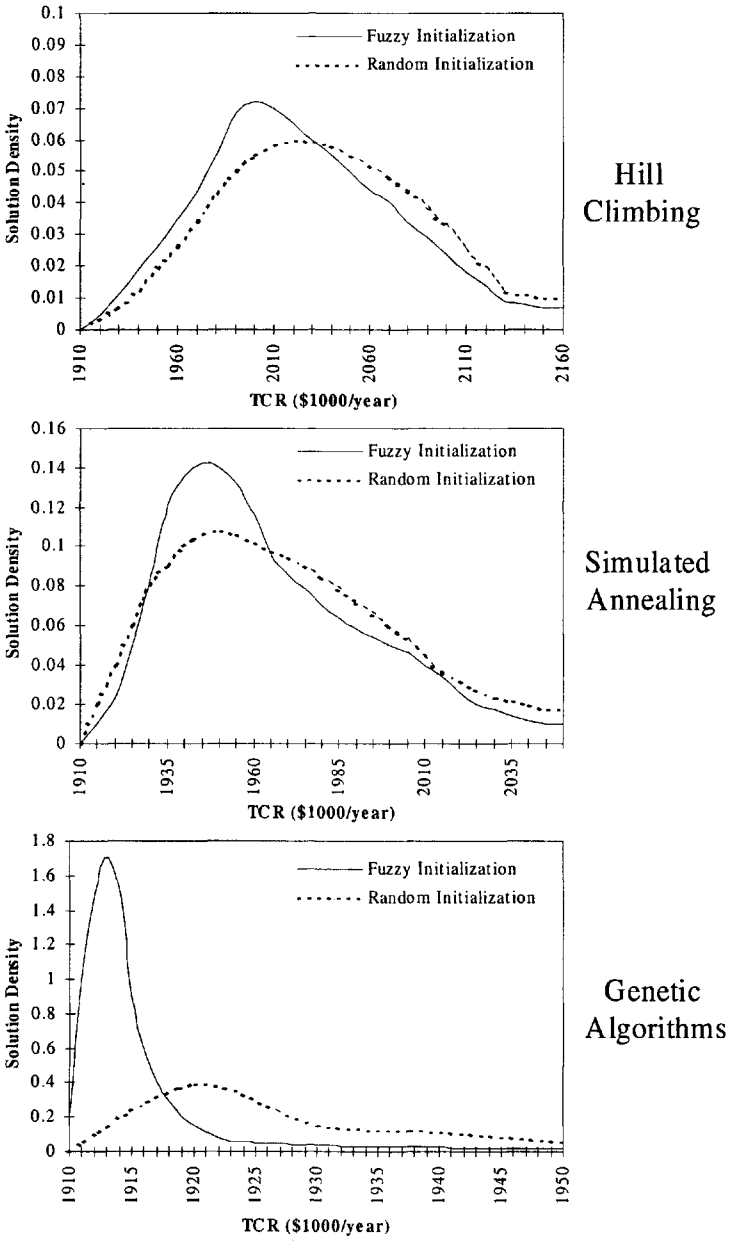


Figure 7.23. Impact of fuzzy initialization on hill climbing, simulated annealing and genetic algorithms. Fuzzy initialization has modest impact on hill climbing and simulated annealing, but substantially improves the performance of genetic algorithms. By providing the genetic algorithm good initial genetic information, lower cost solutions are more likely to be found.

The performance of all three optimization methods improved when random initialization was replaced with fuzzy initialization. This improvement was relatively small for integer programming and simulated annealing, but was substantial for genetic algorithms. Providing the genetic algorithm with good initial genetic material resulted in superior average solutions, reduced solution variance, and improved computation time.

7.5 FINAL THOUGHTS ON OPTIMIZATION

Armed with a basic theoretical knowledge of distribution reliability assessment and optimization, the reader should begin to consider the practical issues associated with applying these concepts on a broad basis. Small applications such as identifying optimal recloser locations on a few feeders may be straightforward. Large applications such as total capital and operational expenditure allocation, however, may prove difficult due to organizational barriers and conflicting agendas that may not be directly related to cost or reliability—at least in a technical sense.

Perhaps the most fundamental barrier to optimization is the unwillingness of distribution companies to define broad-based objectives and problem scopes that span many divisions, departments, budgets and functions. Doing so creates inherent conflict between the affected groups. Each will tend to view reliability in a particular way and will be extremely reluctant to reduce its budget so that others can be increased. Reliability optimization can be addressed separately in each department, but doing so leads to many sub-optimal solutions and is not in the long-term best interest of the distribution company.

Achieving a broad problem definition that encompasses most of the large reliability and cost factors facing a distribution company requires a proponent at the executive level who has formal authority or influence over all affected departments. For investor owned utilities, people at this level in the organizational hierarchy will be interested in objectives that seek to maximize market capitalization, which is ultimately based on the free cash flow associated with reliability decisions. For publicly owned utilities, executives will be interested in objectives that seek to maximize societal welfare, which is generally approached by meeting all customer expectations for the lowest possible rates.

A distribution company maximizes its chances of optimizing reliability when it can continuously reallocate all capital and operational budgets based on well-defined reliability objectives. For example, it should be possible to compare the cost effectiveness of looping an underground distribution lateral through a new subdivision to increasing tree trimming on the main overhead trunk. If tree trimming results in higher reliability for lower cost, the loop should not be built and a part of the construction budget should be transferred to the vegetation management budget. Similarly, the value of substation equipment maintenance

should be able to be directly compared to feeder automation. If feeder automation results in higher marginal value, the substation maintenance budget should be reduced and the feeder automation budget should be increased.

A mental barrier that commonly manifests itself when applying optimization to distribution reliability is a fixation on risk minimization. When a distribution company has a risk-averse culture, the results of an optimization process are often altered to minimize the possibility of reliability ending up worse than promised. For example, an optimization process might conclude that the best size for a transformer is 15 MVA, but a size of 20 MVA is selected "just to make sure." If this increase in capacity is genuinely intended to reduce the risk of an unacceptable reliability outcome, this risk should be explicitly included in the objective function[40-41]. Otherwise, the additional spending undermines the optimization process by increasing cost to satisfy an emotional need.

Other emotional issues that can undermine distribution reliability optimization are personal attachments to the present way of doing business. If an optimization process recommends changes, people responsible for the status quo will tend to become defensive and attack the credibility of the optimization results. This problem can occur at all levels of the organization, from an engineer responsible for equipment ratings to a top executive responsible for setting reliability targets. An engineer who has been recommending loading levels lower than those recommended by the optimization process may warn of impending doom if ratings are adjusted upwards. An executive who has been chanting a mantra of "reliability must improve" for the last three years may entrench and become an opponent of the optimization process when results show that reliability should, in fact, not improve.

As this book is technical in nature, it is not appropriate to address issues concerning leadership, team building, negotiation and influence, which are precisely the skills required to address ego barriers to reliability optimization. For information on these topics, the reader is referred elsewhere[42-44]. It must suffice to say that before implementing reliability optimization, a thorough stakeholder analysis should be performed. Once all people capable of impeding a successful implementation are identified, they can be included in the reliability optimization process from its inception.

Another common argument used against reliability optimization is "These results cannot be trusted due to a lack of good data." It is generally easy to find shortcomings in assumptions relating to cost, failure rates, the impact on failure rates due to reliability improvement options, and so forth. If successful, the insufficient data argument can lead to a "chicken and egg syndrome" where reliability data collection is not justified because there is no use for it and reliability optimization is not justified because there is a lack of required data.

The best way to defend against insufficient data arguments is to carefully select default data, to meticulously calibrate system models based on conservative and justifiable assumptions, and to only use results within the context of

these assumptions. Once a reliability optimization process of limited scope is in place, there will be an incentive to increase reliability data collection efforts to both improve the confidence in results and to gradually increase the optimization scope in both breadth and depth.

Most US distribution systems are operated as regulated monopolies under the governing authority of state regulatory commissions. Regulators approve rate cases and therefore determine the revenue stream of distribution companies. Regulators set performance targets and performance-based rates. Their job is to be an advocate for the people and to insure their best interest, which is to have high levels of reliability for the lowest possible cost without jeopardizing the financial health of the regulated companies. As such, the last reliability optimization difficulties that will be discussed are related to regulation.

The first problem with regulators is their tendency to transfer cost savings to rate payers. Consider a reliability improvement optimization effort that results in a 5% reduction in operational expenditures without impacting reliability. In a fully regulated situation, regulators would consider this a reduction in recoverable costs and would require the distribution company to adjust rates so that these savings are completely passed on to customers. If the optimization effort results in a 5% reduction in the rate base, regulators would force a 5% reduction in earnings so that the ratio of profits to rate base remains the same. Such anticipated regulatory responses pose the question, "If regulators take reliability optimization benefits away as soon as they are realized, why should distribution companies bother to make the effort?" The answer to this question is not simple, but involves a regulatory shift away from guaranteed return-on-assets and towards rate freezes, performance-based rates and reliability guarantees.

There are other difficulties with regulators. Many of them view reliability as something that must always improve, arguing that technology and process efficiencies should allow for higher reliability at a lower cost. Others view electricity as an entitlement and embed cross-subsidies in rate structures. In each case, the regulatory view of optimization may be at odds with the utility view of optimization, potentially resulting in conflict and wasted effort.

Rather than performing a broad-based reliability optimization initiative and having it undermined by regulatory action, distribution companies are encouraged to include regulators in the optimization process. If distribution companies and regulatory commissions work together, the probability of implementing a fair combination of rate structure, reliability incentives, reliability targets and customer choice increases greatly. Once the rules of the game are set, reliability can be optimized knowing it is in the best interest of all parties including distribution regulators, distribution customers and equity investors.

Reliability optimization is imperative to the survival of distribution companies. Although distribution systems are a natural monopoly, there are emerging non-regulated technologies that are able to bypass the distribution system function at increasingly lower costs. If distribution companies are not able to provide

higher reliability for a lower cost than these technologies, they will vanish along with their infrastructure and market valuation. This book has presented all of the major issues related to distribution reliability including system design, system operation, reliability metrics, reliability modeling, reliability analysis and reliability optimization. Using these concepts and techniques, distribution companies can remain competitive by providing increasingly higher value to customers. At the same time, customer satisfaction will increase through rate choices and reliability guarantees, regulatory volatility will stabilize as commissioners see that they are acting in the best public interest, and investor expectations will be exceeded by providing high levels of reliability while improving profitability and minimizing risk.

REFERENCES

1. G. L. Nemhauser and L. A. Wolser, *Integer Programming and Combinatorial Optimization*, Wiley, 1988.
2. F. J. Hill and G. R. Peterson, *Introduction to Switching Theory and Logical Design*, John Wiley and Sons, Inc., 1968.
3. F. Glover, "Tabu Search: Part I," *ORSA Journal on Computing*, Vol. 1, 1990.
4. F. Glover, "Tabu Search: Part II," *ORSA Journal on Computing*, Vol. 1, 1990.
5. E. L. da Silva, J. M. A. Ortiz, G. C. de Oliveira and S. Binato, "Transmission Network Expansion Planning Under a Tabu Search Approach," *IEEE Transactions on Power Systems*, Vol. 16, No. 1, Feb. 2001, pp. 62-68.
6. R. A. Gallego, R. Romero and A. J. Monticelli, "Tabu Search Algorithm for Network Synthesis," *IEEE Transactions on Power Systems*, Vol. 15, No. 2, May 2000, pp. 490-495.
7. K. Nara, Y. Hayashi, Y. Yamafuji, H. Tanaka, J. Hagihara, S. Muto, S. Takaoka and M. Sakuraoka, Á Tabu Search Algorithm for Determining Distribution Tie Lines," *Proceedings of the International Conference on Intelligent Applications to Power Systems*, IEEE Catalog 96TH8152, Jan. 1996.
8. L. Davis, *Genetic Algorithms and Simulated Annealing*, Morgan Kauffmann Publishers, Inc., 1987.
9. V. Miranda, J. V. Ranito and L. M. Proenca, "Genetic Algorithms in Optimal Multistage Distribution Network Planning," *IEEE Transactions on Power Systems*, Vol. 9, No. 4, Nov. 1994, pp. 1927-1933.
10. S. Sundhararajan and A. Pahwa, "Optimal Design of Secondary Distribution Systems Using a Genetic Algorithm," *Proceedings of the 27th Annual North American Power Symposium*, IEEE, October 1995, pp. 681-684.
11. E. C. Yeh, S. S. Venkata and Z. Sumic, "Improved Distribution System Planning Using Computational Evolution," *Proceedings of 1995 IEEE Power Industry Computer Applications Conference*, IEEE, May, 1995.
12. D. Srinivasan, F. Wen, C. S. Chang and A. C. Liew, "A Survey of Applications of Evolutionary Computing to Power Systems," *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, IEEE Catalog 96TH8152, Jan. 1996.
13. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, 1989.
14. William M. Spears, "Crossover or Mutation?" *Foundations of Genetic Algorithms 2*, Morgan Kauffmann Publishers, Inc., 1993.
15. E. Rich and K. Knight, *Artificial Intelligence*, 2nd Edition, McGraw-Hill Int., 1991.
16. G. Brauner and M. Zobel, "Knowledge Based Planning of Distribution Networks," *IEEE Transactions on Power Systems*, Vol. 9, No. 2, May 1994, pp. 942-948.

17. J. Shao, N. D. Rao and Y. Zhang, "An Expert System for Secondary Distribution System Design," *IEEE Transactions on Power Delivery,* Vol. 6, No. 4, Oct. 1991, pp. 1607-1615.
18. *Report of the U.S. Department of Energy's Power Outage Study Team: Findings and Recommendations to Enhance Reliability from the Summer of 1999,* U.S. Department of Energy, March 2000.
19. C. Segneri, "Reversal of Fortunes: ComEd Rebuilds Its System and Its Reputation," *Transmission and Distribution World,* Vol. 53, No. 6, May 2001, pp. 6-13.
20. R. E. Brown, A. P. Hanson, H. L. Willis, F. A. Luedtke and M. F. Born, "Assessing the Reliability of Distribution Systems," *IEEE Computer Applications in Power,* Vol. 14, No. 1, Jan. 2001, pp. 44-49.
21. T. Terano, K. Asai, and M. Sugeno, *Fuzzy Systems Theory and Fuzzy Systems Theory and its Applications,* Academic Press, Inc., 1987.
22. N. Kagan and R. N. Adams, "Electrical Power Systems Planning Using Fuzzy Mathematical Programming," *International Journal of Electrical Power and Energy Systems,* Vol. 16, No. 3, 1994, pp. 191-196.
23. R. E. Brown, "Distribution Reliability Assessment and Reconfiguration Optimization," *2001 IEEE/PES Transmission and Distribution Conference*, Atlanta, GA, Oct. 2001.
24. R. E. Brown, S. S. Gupta, R. D. Christie, and S. S. Venkata, "A Genetic Algorithm for Reliable Distribution System Design," *International Conference on Intelligent Systems Applications to Power Systems*, Orlando, FL, January 1996, pp. 29-33.
25. R. E. Brown and B. Howe, "Optimal Deployment of Reliability Investments," E-Source, Power Quality Series: PQ-6, March 2000, 37 pages.
26. H. L. Willis, H. Tram, M. V. Engel and L. Finley, "Selecting and Applying Distribution Optimization Methods," *IEEE Computer Applications in Power,* Vol. 9, No. 1, IEEE, January 1996, pp. 12-17.
27. W. F. Horton, S. Goldberg and R. A. Hartwell, "A Cost/Benefit Analysis in Feeder Reliability Studies," *IEEE Transactions on Power Delivery,* Vol. 4, No. 1, Jan. 1989, pp. 446-452
28. Y. Tang, "Power Distribution System Planning with Reliability Modeling and Optimization," *IEEE Transactions on Power Systems,* Vol. 11, No. 1, Feb. 1996, pp. 181-189
29. I. J. Ramírez-Rosado and J. L. Bernal-Agustín, "Reliability and Costs Optimization for Distribution Networks Expansion Using an Evolutionary Algorithm," *IEEE Transactions on Power Systems,* Vol. 16, No. 1, Feb. 2001, pp. 111-118.
30. V. Borozan, N. Rajakovic, "Application Assessments of Distribution Network Minimum Loss Reconfiguration," *IEEE Transactions on Power Delivery,* Vol. 12, No. 4, Oct. 1997, pp. 1786-1792.
31. R. Takeski, D. Rajicic, "Distribution Network Reconfiguration for Energy Loss Reduction," *IEEE Transactions on Power Systems,* Vol. 12, No. 1, Feb. 1997, pp. 398-406.
32. Q. Zhou, D. Shirmohammadi, W. H. E. Liu, "Distribution Reconfiguration for Operation Cost Reduction," *IEEE Transactions on Power Systems,* Vol. 12, No. 2, May. 1997, pp. 730-735.
33. H.L. Willis, *Power Distribution Reference Book*, Marcel Dekker, 1997.
34. R. E. Brown, S. Gupta, R. D. Christie, S. S. Venkata, and R. D. Fletcher, "Automated Primary Distribution System Design: Reliability and Cost Optimization," *IEEE Transactions on Power Delivery*, Vol. 12, No. 2, April 1997, pp. 1017-1022.
35. Y. Fukuyama, H. Endo and Y. Nakanishi, "A Hybrid System for Service Restoration Using Expert System and Genetic Algorithm," *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems,* Orlando, FL, IEEE, January 1996.
36. A. H. Mantawy, Y. L. Abdel-Magid and S. Z. Selim, "Integrating Genetic Algorithms, Tabu Search and Simulated Annealing for the Unit Commitment Problem," *IEEE Transactions on Power Systems,* Vol. 14, No. 3, Aug. 1999, pp. 829-836.
37. R. E. Brown, "The Impact of Heuristic Initialization on Distribution System Reliability Optimization," *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, Vol. 8, No. 1, March 2000, pp. 45-52.

38. R. E. Brown, S. S. Venkata, and R. D. Christie, "Hybrid Reliability Optimization Methods for Electric Power Distribution Systems," *International Conference on Intelligent Systems Applications to Power Systems*, Seoul, Korea, IEEE, July 1997.
39. R. E. Brown, *Reliability Assessment and Design Optimization for Electric Power Distribution Systems*, Ph.D. Dissertation, University of Washington, Seattle, WA, June, 1996.
40. V. Miranda and L. M. Proenca, "Probabilistic Choice vs. Risk Analysis–Conflicts and Synthesis in Power System Planning," *IEEE Transactions on Power Systems*, Vol. 13, No. 3, Aug. 1998, pp. 1038-1043.
41. V. Miranda and L. M. Proenca, "Why Risk Analysis Outperforms Probabilistic Choice as the Effective Decision Support Paradigm for Power System Planning," *IEEE Transactions on Power Systems*, Vol. 13, No. 2, May 1998, pp. 643-648.
42. J. P. Kotter, "Leading Change: Why Transformation Efforts Fail," *Harvard Business Review*, March-April 1995.
43. D. Carnegie, *How to Win Friends and Influence People*, Mass Market Paperback, 1936.
44. L. A. Wortman, *Effective Management for Engineers and Scientists*, John Wiley & Sons, 1981.