Chapter 3

Applied Software Techniques

Introduction

Several analytical techniques have been applied in solving the real and reactive power dispatch problems described in chapter two with notable successes. There exists a considerable amount of power system problems that are widely solved by human experts together with either the results from the numerical analysis or decision support. Due to inadequate modeling of the real world, incapability of expressing the solution method employed by humans in an algorithm or mathematical form, decision making of the operators based on fuzzy linguistic description and analysis based on human judgement and experience, the power system community has been motivated to explore alternative solution strategies: Expert systems, artificial neural networks, fuzzy logic systems and evolutionary computation techniques essentially belong to the field of **Artificial Intelligence (AI)**. AI systems basically try to emulate functions of human beings on a machine.

Since this work is concerned with the application of evolutionary techniques and heuristics to real and reactive power dispatch in order to enhance the state of a power system, a brief overview of genetic algorithms and expert systems is presented in this chapter. Finally, a brief survey over relevant existing data setup software packages used in order to supply required power system data for the sub-systems developed in this work are also presented.

3.1 Overview of Genetic Algorithm based Optimization

Genetic Algorithms (GAs), first proposed by John Holland in the 1960's, are numerical optimization algorithms based on principles inspired from the genetic and evolution mechanisms observed in natural systems and populations of living beings [14,15,16]. The method is a general one and finds widespread application as a consequence of two fundamental issues:

- The computational code to implement GAs is quite simple and yet provides a powerful search mechanism.

- GAs are very flexible and robust schemes in that they can be applied to a broad range of optimization problems.

The robust behavior, which is the distinguishing feature of GAs with respect to other optimization methods, implies that GAs must differ in some fundamental ways. These traits are described below:

- GAs search from a population of candidates and do not process only one single solution; thus, they are resistant to being trapped in local optima.
- GAs use probabilistic transition rules and not deterministic rules.
- GAs exploit only the payoff information of the objective function to guide their search towards the **global optimum**. They do not depend on any additional information like the existence of derivatives.
- GAs work with a coding of the problem parameters and not the parameters themselves.
- GAs mimic the natural evolution process.

Some recognized disadvantages inherent in GA search are:

- Large number of function evaluations, with the resulting undesirably long execution time.
- There is no way to definitely know whether the optimal solution has been found.

With the positive qualities highlighted above, the method has been extensively discussed to be used in the field of power system operation and planning [7,42,54]; rather the number of practical implementations is still low.

Binary encoding GAs are dealing with binary strings, where the number of bits of each string simulates the genes of an individual chromosome, and the number of **individuals** constitutes a **population**. Each parameter set is encoded into a series of a fixed length of string symbols, usually from the binary bits which are then concatenated into a complete string called **chromosome** (see figure 3.1). Sub-strings of specified length are extracted successively from the concatenated string and are then decoded and mapped into the value in the corresponding search space. Other encoding mechanisms are real-valued encoding in which real numbers are used to form the chromosomes, and Gray encoding [14,16]. However, the performance depends very much on the problem and the details of the GA being used, and at present there are no rigorous guidelines for predicting which encoding will work best.

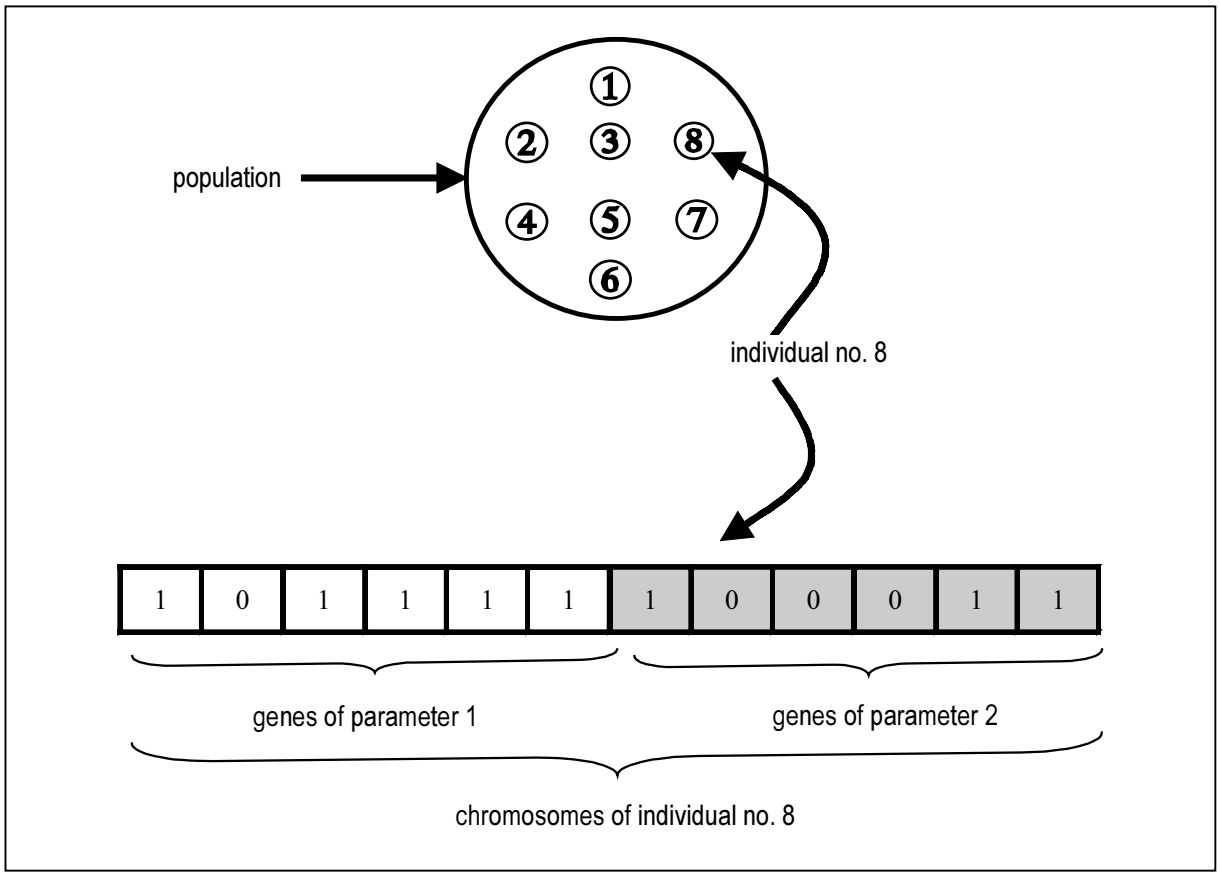

Figure 3.1 Illustration of GA terminology

Generally, GAs implementation comprises three different phases: initial population generation, fitness evaluation and genetic operations. The processes involved in the implementation of conventional GAs are as described below and comprehensively shown in figure 3.5.

3.1.1 Initial Population Generation

The GA control parameters, such as population size, crossover probability and mutation probability (see below) are selected, and an initial population of binary strings of finite length is randomly generated. Each of these individuals, consisting of a number of chromosomes, represents a feasible solution to the search problem. The solution strings are decoded back into their control variables to compute their fitness, see below.

3.1.2 Fitness Evaluation

Each solution must be evaluated by the "fitness function" to produce a value which determines its relative quality. Therefore, the maximum fitness, minimum fitness and average fitness of all individuals within a generation are computed. If a pre-defined convergence criterion is not satisfied, then it is continued with the

following genetic operations of **selection** and **reproduction**, **crossover** and **mutation**.

3.1.3 Selection and Reproduction

This mechanism attempts to apply pressure upon the population in a manner similar to that of natural selection found in biological systems. A new population or generation is created in which poorer performing individuals are weeded out and the most highly fit members in a population are selected to pass on information to the next generation. Figure 3.2a depicts a selection strategy.

This operator can be implemented in a variety of ways, although the most used techniques are “Stochastic Tournament” and “Roulette wheel selection” [14,15] which are discussed below:

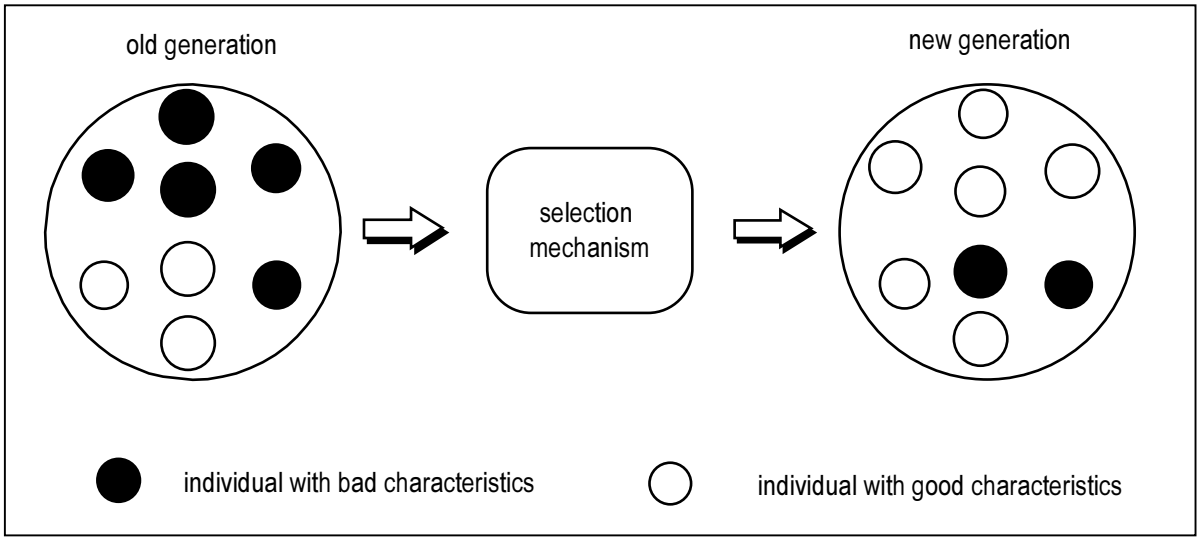

Figure 3.2a Selection strategy

- **Roulette wheel selection**: In this process, the individuals of each generation are selected for survival into the next generation according to a probability value proportional to the ratio of individual fitness over total population fitness. That means the individuals with higher fitness value have higher probability to be selected. It involves spinning a roulette wheel in which each string occupies an area of the wheel equal to the string’s share of the total fitness. This method is illustrated in figure 3.2b with a population consisting of 6 individuals. Here, the individual number 4 has the highest probability to be selected and hence of being represented in the next generation. The selected candidates are gathered in a temporary mating pool and are ready for further processing.

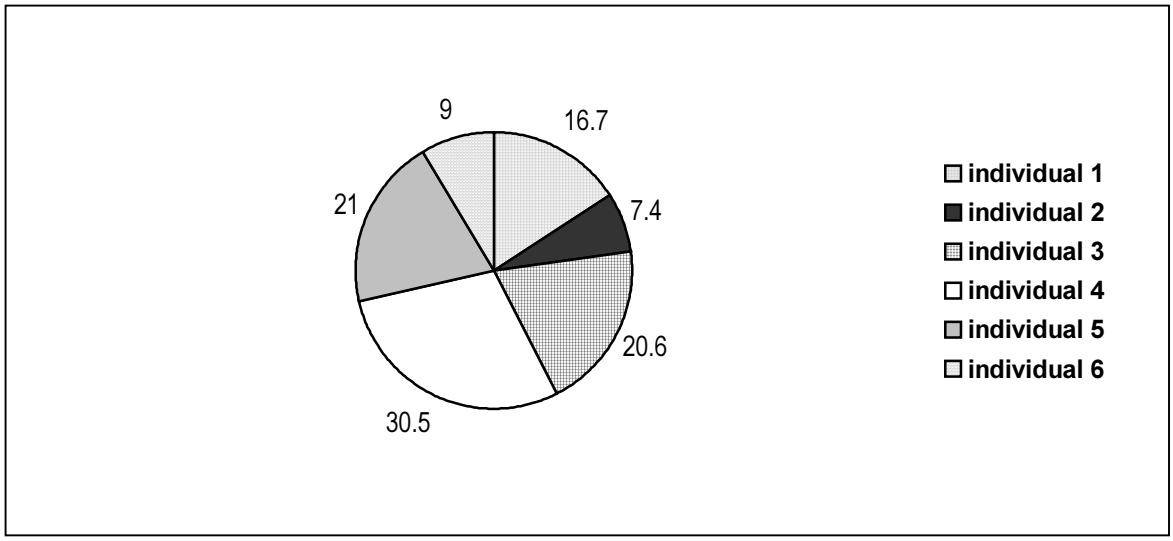

Figure 3.2b Roulette wheel selection strategy

- **Tournament selection**: This is another method, computationally efficient and effective, used to select mates. Pairs of individuals are chosen at random from the population and the most fit of each pair is allowed to mate. Each pair of mates creates a child having some mix of the two parents' characteristics according to the crossover method discussed next. The pair is then returned to the original population and can be selected again. The process of randomly selecting pairs and mating the stronger individuals continues until a new generation of the same number of individuals is reproduced. This approach is illustrated in figure 3.2c for a population of eight individuals.

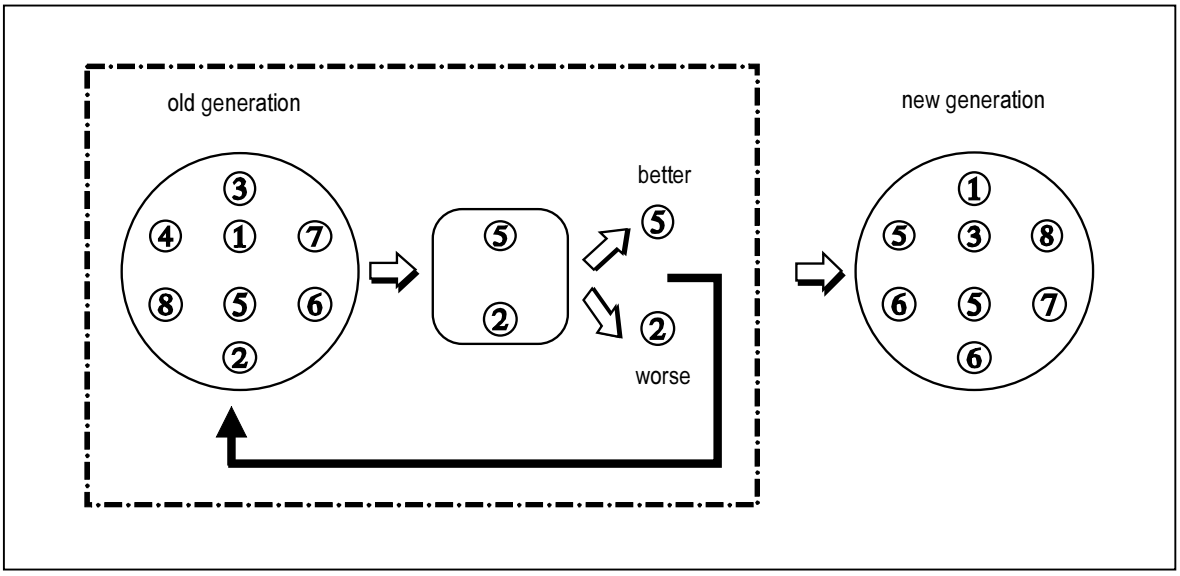

Figure 3.2c Tournament selection strategy

3.1.4 Crossover

This is the primary genetic operator which promotes the exploration of new regions in the search space. It is a randomized mechanism of exchanging information between strings. Two individuals previously placed in the mating pool during reproduction are randomly selected. A crossover point is then randomly selected, and information from one parent up to the crossover point is exchanged with the other parent. Thus, two new offspring which are a mixture of the two parents are created for the next generation. There are three important crossover techniques normally used in GAs which are discussed below:

- **Single point crossover** randomly chooses a single locus (point) along the parents and swaps all the binary bits to the right of this locus between the two parents to form two offspring.

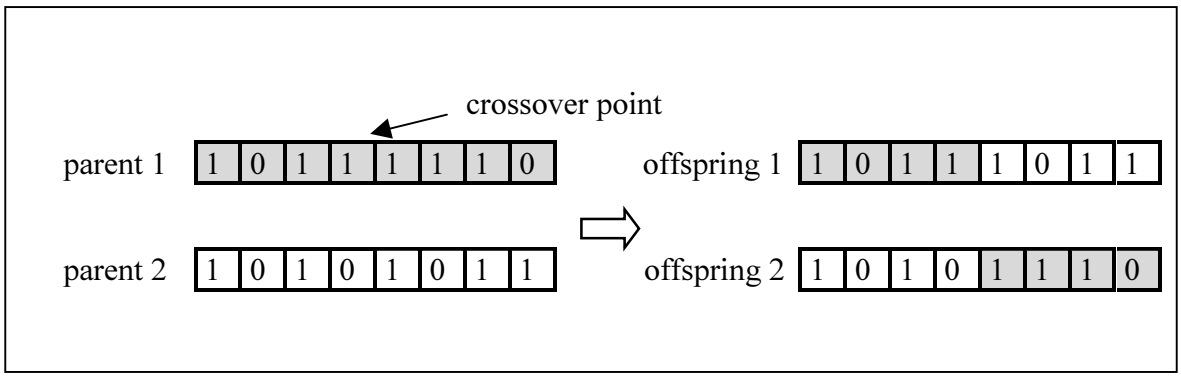

Figure 3.3a Single point crossover

- **Two points crossover** is like single point crossover except that two random crossover positions are selected and binary bits between the two selected points are swapped to form the two offspring.

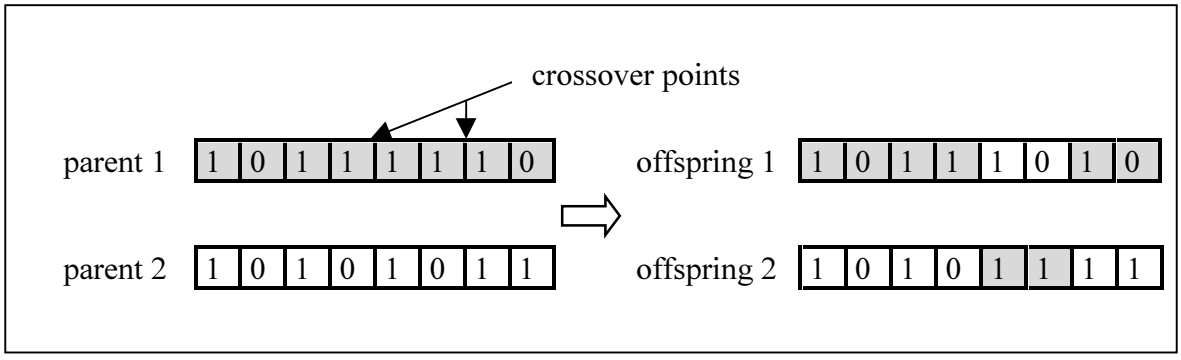

Figure 3.3b Two points crossover

- **Uniform crossover** operates by uniformly selecting positions, and binary bits at each corresponding pair of coordinates are exchanged with the same probability σ_c as shown in figure 3.3c. This can be highly disruptive, especially in early generations. Parameterized uniform crossover [74] moderates this disruption by applying a probability (typically $0.5 \leq \sigma_c \leq 0.8$) to the exchange of bits between strings.

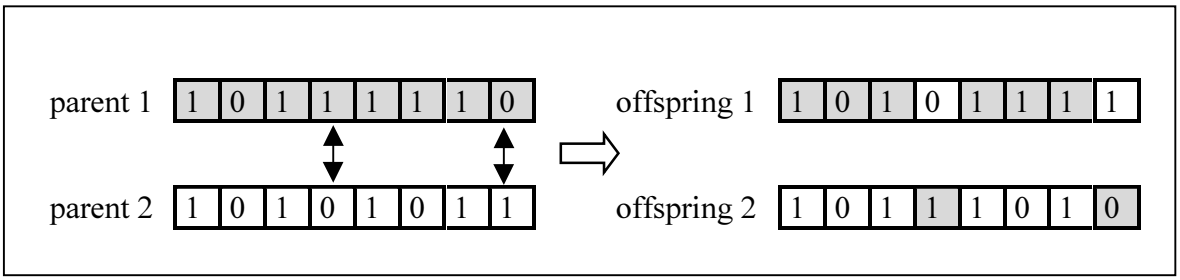

Figure 3.3c Uniform crossover

Empirical comparison of the three techniques shows that uniform crossover performs better than the other two methods as it tends to preserve more alleles than the other methods [14,15].

3.1.5 Mutation

Mutation is the process of randomly changing encoded bit information for a newly created population individual. Mutation is known as an insurance policy to maintain search diversity within the population. It is generally considered as secondary operator extending the search space. It is performed sparingly, and involves randomly selecting a string and bit positions and toggling them from a 1 to 0 or vice-versa. It is used to escape from a local optimum when used sparingly with selection and crossover.

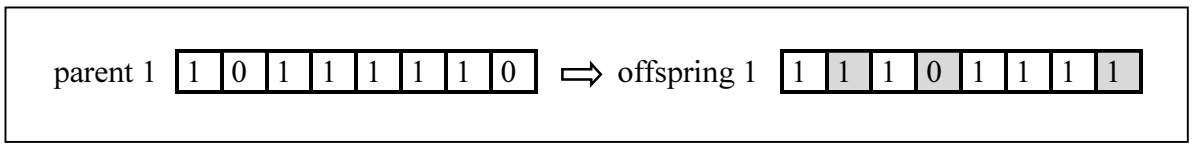

Figure 3.4 Mutation operation

3.1.6 Creep Mutation

This is an operator used to assist the GA search for optimum solution based on an intelligent mechanism. It leaps in a random direction and distance, always within the feasible region of parameter space. The parameter could creep one

increment up or down from one of the parents values, i.e. the creep mutation produces a parameter value that is randomly picked to be larger or smaller, as long as it remains within the range of the appropriate parameter values.

3.1.7 Elitism

The probabilistic nature of the generation process involves the possibility that the highest fit individual may be destroyed by the genetic operator. The elitist strategy ensures that the most fit individual generated actually is reproduced in the subsequent generation. After the population is generated, the GA checks to see if the best parent has been replicated; if not, a random individual is chosen and the chromosome set of the best individual is mapped into that individual. This may increase the speed of domination of the population by a super individual, but on balance it can rapidly increase the GA performance by using the best solution as a seed for further optimization and also the speed of convergence to global optima.

3.1.8 Treatment of Constraints

In many optimization problems like real and reactive power dispatch, there are a number of constraints to be satisfied. The following methods reported in literature [2,8] enable the GAs to be applied to constrained optimization problems are summarized below:

- Infeasible solutions are discarded as soon as they are generated. This method has been used by evolution strategies, evolutionary programming and simulated annealing. The method does not utilize the potential information contained in the infeasible solutions, and a lot of processor time may be consumed searching for a feasible solution.

- Special decoder schemes are used that minimize or eliminate the possibility of producing infeasible solutions through the standard genetic operators.

- Repairs and approximation of invalid solutions: The resulting valid solutions may be substantially different from the originally produced solution. Moreover, in certain problems, finding a feasible approximation of an infeasible solution may be as difficult as the optimization problem.

- Special problem-specific recombination and permutation operators are designed, which are similar to traditional crossover and mutation operators, and produce only feasible solutions. Such operators are sometimes difficult to construct and problem dependent.

- Penalty terms are added to the fitness function. Unfeasible solutions are not removed from the population, but their fitness values are degraded according to the degree of constraints violation. This method is probably the most commonly used method in treatment of constraints, and its implementation variations are reported in [2,8,14,15,16]. However, the problem of this method is the design of an appropriate penalty function that will enable the GA to converge to a feasible sub-optimal or even optimal solution.

3.1.9 Control Parameters

Like other optimization methods, GAs use certain control parameters such as population size, maximum number of generations, parameter resolution, genetic operations probabilities (crossover probability, creep mutation probability and mutation probability) and elitist strategy for their implementation. These parameters must be selected with maximum care, as the performance of GAs depends largely on the values used. Normally, relatively low population size, high crossover and low mutation probabilities are recommended [15]. The population size increases according to the problem difficulty. Typical values for the crossover rate range from 0.5 to 1, and the mutation rate is typically very small.

There are no conclusive results on what is the best parameter setting; most authors report that they used what has worked well in previously reported cases. De Jong [75] found out that the mutation rate σ_m of 0.001 per bit, population size n_p of between 50 to 100 individuals and single-point crossover rate σ_c of 0.6 per pair of parents were optimal for the test suite considered. These settings (along with De Jong's test suite) became widely used in the GA community, even though the efficiency of this parameter setting outside De Jong's test suite was not clear.

Somewhat later, in [60] the mutation rate σ_m of 0.01 per bit, a smaller population size n_p of 30 individuals and higher crossover rate σ_c of 0.95 per pair of parents were found out to be optimal settings. This parameter setting procured a small but significant improvement in performance over De Jong's setting.

It was observed that no general principle about parameter setting can be formulated *a priori* in view of the variety of problem types, encoding and performance criteria that are possible in different applications. Moreover, the optimal population size, crossover rate, and mutation rate likely change over the course of a single run. This prompted the researchers to develop approaches to have the parameter values adapt in real time to the ongoing search. There have

been several approaches to self-adaptation of GA parameters. Two particular mutation rates σ_m given by

$$\sigma_m = \frac{1.0}{l_c} \quad \text{and} \quad \sigma_m = \frac{1.75}{l_c \cdot n_p} \text{ per bit}$$

have been investigated by Bäck in [76] and were found to be satisfactory; where l_c is the total number of chromosome length and n_p is the population size.

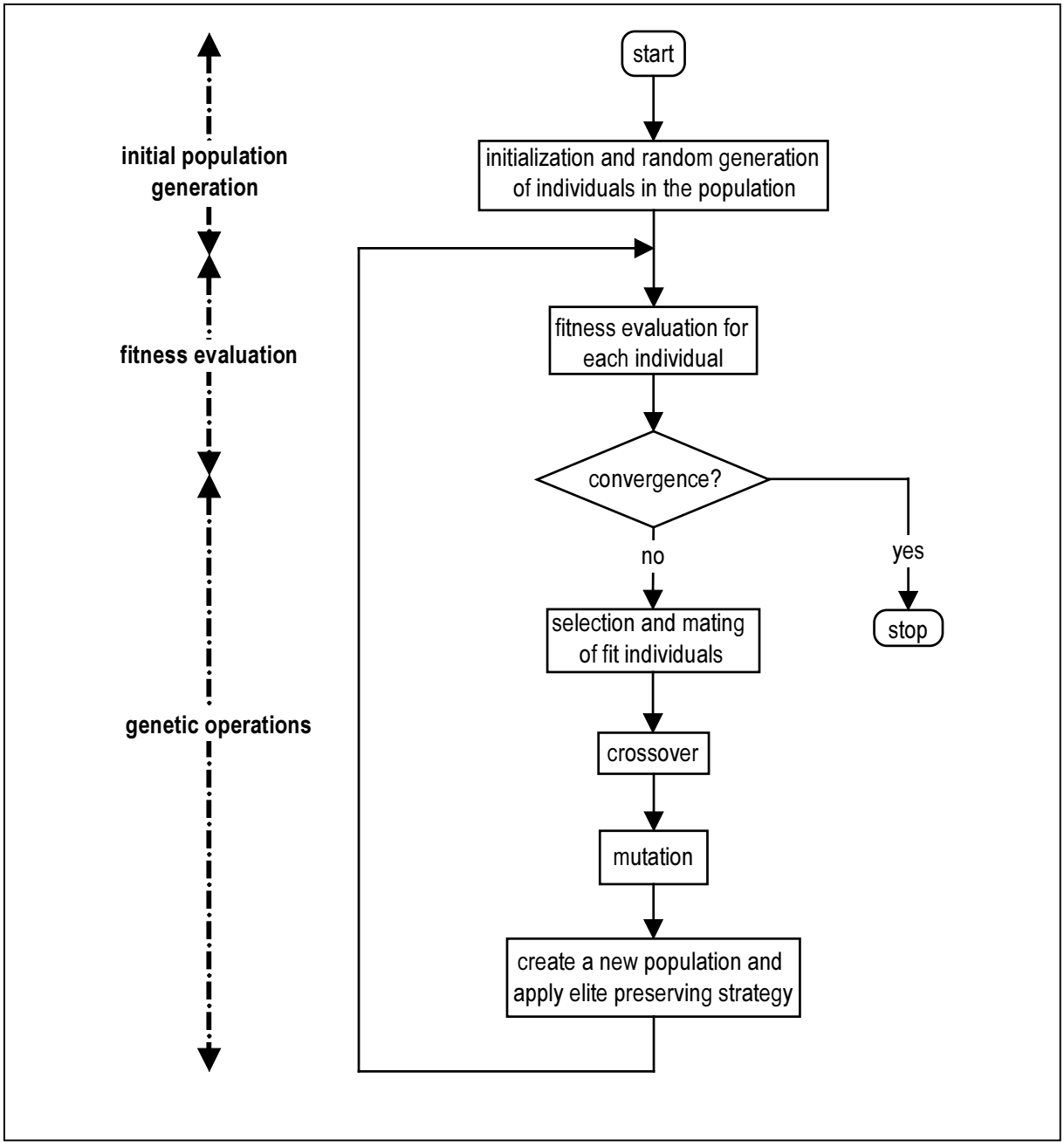

Figure 3.5 Flow chart of conventional GA

3.1.10 Convergence Criteria

The processes stop when the convergence criteria are satisfied; the most commonly used criteria in GAs are:

- Stop the algorithm at some pre-set number of generations.
- Stop whenever the solution does not improve after a specified number of generations.
- Stop if the average fitness of the population exceeds some fraction of the best fit in the population.

The criteria actually applied in this work will be described in sections 4.2.1.3 and 5.2.1.3 respectively.

3.2 Micro Genetic Algorithm

It is expected of GAs to be able to find an acceptable solution within a reasonable time when solving the optimization problem. One of the features that distinguish GAs from other conventional search methods is the characteristics to simultaneously deal with a population of points (solutions), thus leading to the disadvantage of requiring a relatively large number of function evaluations. A survey over existing population studies [15] shows that a larger population (20-200) is generally thought to be able to find a global optimum in few generations. Application of a **small** population size was proposed by Krishnakumar [61], and the effectiveness of such so called **micro GA (μGA)** was tested on stationary and non-stationary functions. The application of this approach has been reported in [62], proving to be conceptually simple and easy to implement as an effective search technique.

The major difference between the micro GA and the conventional GAs lies in the choice of the population size. In the micro GA, an initial very small population, typically of four or five individuals is randomly generated; it is then processed by the three main GA operators except that the mutation rate is fixed at 0.0. The algorithm thus converges quickly within a few function evaluations.

A restart procedure in which new individuals are randomly generated while keeping a copy of the best individual of the previous converged generation ensures the infusion of new genetic information and the retention of the previous best individual. The **genotype convergence** is said to occur when less than 5% of the bits of other individuals differ from the best individual. The flow chart of

figure 3.6 shows a comparison of the conventional GA approach and the micro GA depicting their differences.

In the investigations described later in this work, a comparison is made between both the conventional - and micro - GA approaches.

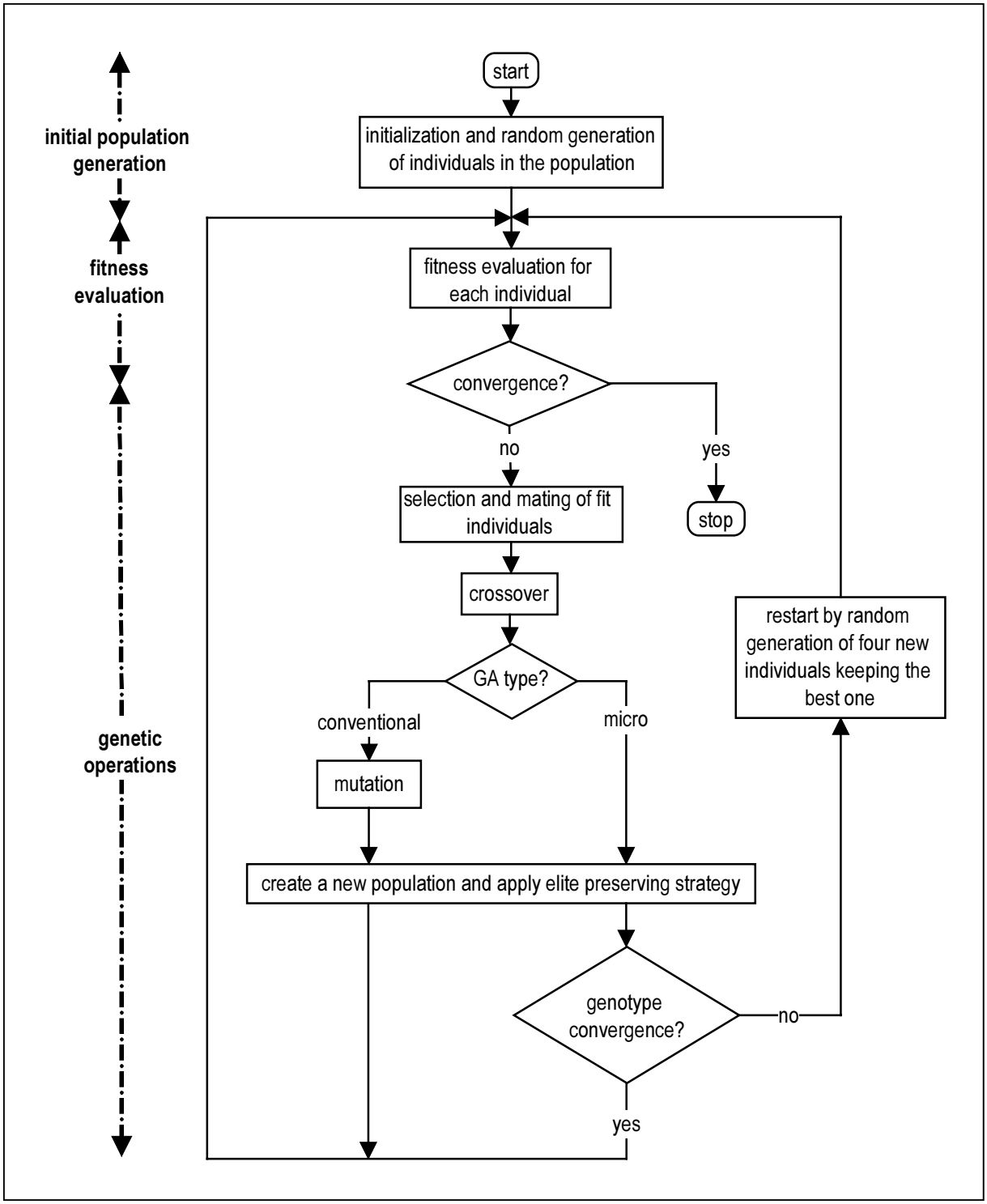

Figure 3.6 Combined flow chart of micro - and conventional - GA depicting their differences

3.3 Expert Systems applied to Power System Control

3.3.1 Nature of Expert Systems

Expert systems (ES), also referred to as knowledge-based systems, are developed to emulate the problem solving behavior of human experts in a specific domain. The term "problem solving behavior" refers to the experience, heuristics, procedural rules and strategies used by human experts in solving a problem. Characteristic for the structure of ES is thus, the consistent separation of the inference mechanism from the specialized domain knowledge stored in the so called knowledge base. This separation of knowledge base and the inference engine enables one to modify or upgrade the knowledge base without influencing the inference engine. Knowledge can be represented in an expert system by using one or more knowledge structures such as production rules, frames, semantic networks, and objects [3,32,40].

3.3.2 Benefits of Expert Systems

Expert systems can bring the following benefits [21,22,32,37]:

- Logical structures can be well expressed due to the correspondence between production rules and logical implications.
- The use of **if - then** rules structure makes the knowledge base easy to set up and to maintain.
- Tracing the application of rules eases to implement an explanation sub-system, thus procuring transparency in reasoning.
- Knowledge of more than one human expert can be accumulated.
- The knowledge is permanently fixed and can well be reproduced.
- Expert systems work free of emotion related factors like stress or time pressure.
- Once set up, an expert system is inexpensive.

Although expert systems have some weakness in comparison with human intelligence such as the limitation of knowledge as well as lacking intuition, creativity, adaptation and learning abilities, the strong points highlighted above make expert systems be useful as decision support tools for many applications.

3.3.3 Knowledge Base

The knowledge about a particular field of interest can be classified into:

- **Structural** or **data-based knowledge** consisting of all relevant information and relations, data or facts about the system under study. This is further classified into **dynamic** and **static** knowledge. With special regard to power systems, a major portion of data-based knowledge can be retrieved from the process database of the control system [50]; this comprises constant data such as the components of the power system, their physical parameters and their topological interconnections as static data in the sense that they are changed only when the model of the power system incorporated in the process database is upgraded or modified to reflect changes in the actual system, as well as the actual information concerning the states of breakers, measurement values, alarms and other events as dynamic data.

- **Methodical** knowledge in the form of algorithms, models or rules. It is basically to differentiate between an **exact** formulation, for instance in the form of a causation model, an algorithm or logical if-then rules, and a **heuristic** formulation which can be provided by if-then rules as well. For power systems, the knowledge about the physical behavior of a certain device in a certain situation or the ability to determine from the network connectivity and switching states the actual topology are examples of methodical knowledge.

Complex expert systems applied in the area of power system control use a combination of both static and dynamic process data, algorithms, physical models and symbolic computations, whereby the logical and heuristic rules are usually applied to derive the higher level decisions.

Knowledge acquisition is a crucial point in the development of expert systems because its usefulness depends strongly on the quality of the knowledge put into the system. Experiences made in the area of power systems revealed that the task of knowledge engineering is best performed by classical power engineers instead of so called “knowledge engineers”; this at the same time eliminates the possible refusal of experts yielding their expertise.

The **correctness** of a knowledge base can be achieved on the level of data, rule syntax and their logical consistency by the application of corresponding auxiliary software; on the level of the technical contents, the correctness can only be checked by comprehensive tests, as for any large conventional program.

3.3.4 Inference Engine

The inference engine of a rule based ES, as a driving force, analyzes the system by applying the rules together with the acquired information about the problem to reason, and provides an expert solution. It therefore deduces conclusions by combining knowledge evaluated from the rule base and case data about the current problem retrieved from outside by either pre-processing external routines or through user dialogue. The rules can be matched among each other using the following inference strategies:

- The **forward chaining** method, also referred to as a data-driven strategy, is an algorithm which is used to match data to the condition of a rule. If a combination of data elements satisfies the conditions, the data combination is stored in a conflict set. Among all data combinations in the conflict set, the one with the highest priority is fired. The procedure of selecting the highest priority rule from the conflict set is referred to as conflict resolution.

- In contrast, the **backward chaining** method also referred to as goal-driven strategy, tries to prove a given goal or conclusion. The advantage of this method is the reduction in the number of external information which makes it attractive for diagnosis rules formulation.

In forward chaining, the names of individual rules under investigation are entered into a dynamic **goal stack**, which controls the processing of further rules by the inference engine. To satisfy the goal, the inference component first tries to satisfy all the conditions of the forward rule under consideration linked together by logical operators ‘and’ and ‘or’; naturally a condition may also consist of a backward chain which then is first evaluated. Having satisfied the condition part of a forward rule, the name(s) of consequence rule(s) is (are) then put forward in the goal stack and its evaluation succeeds. If the rule fails, the inference engine searches for alternatives. The inference process is terminated if the goal stack is empty.

Backward chaining tries to verify a superior condition by checking the sub-posed pre-conditions in any depth. Failing in doing so initiates backtracking to the last node and attempt to pursue an alternative.

3.3.5 Expert Systems in Power System Operation

The origins of ES are outside the electric utility industry. But since the mid 1980s, expert system technology has obtained wide-ranging attention of the

power engineering community for applications in power systems. Specific objectives aimed at supporting the power system operators include [21,22]:

- alarm processing and reduction;
- fault diagnosis;
- contingency selection and security control;
- preventive, emergency and restorative control;
- unit commitment, fuel scheduling;
- maintenance scheduling.

Control center operators benefit from using ES because they not only identify a problem, but they can also provide the underlying reasoning used to define the problem, and a set of recommended actions to ameliorate the problem. For application in power systems, system information can be retrieved from the real time process data base or a simulator. A comprehensive survey of many applications of expert systems in electric power engineering are reported in [21,22].

In the work considered here, expert system technology is used for the implementation of the superior strategy employing the GA based active and reactive power dispatch modules.

3.4 Training Simulator

In order to implement and verify the work developed in the frame of this thesis in a realistic environment it was embedded into an existing operator training simulator which represents the replicated power system including all physical components and its control system [20,29,63,64,65]. This training simulator realistically imitates the control room view on the performance of a power system by the application of the dynamic models for power units, network and loads as well as its SCADA/EMS environment. Therefore, both SCADA/EMS data as well as modeling data are maintained within the process database of the simulator, thus having all available information accessible at one and the same place.

The Grid Data Language (GDL) data system enables the efficient parameterization of the simulator. At the same time, the GDL data format allows open and transparent data exchange between the (simulated) system operation process and the environmental systems including the work described in this thesis.

3.5 The GDL-Data System

The GDL-system enables one to describe the full set of process data of any real power system according to substations, lines, transformers, busbars, switchgear, protection and auxiliary equipment, including their potential topology (connectivity) in format-free alpha-text; this also applies to power units. In the source data, it uses the basic form of syntax given by five hierarchical levels to describe the objects of a network with the format [64,65,66]:

’’’LOK’’NUM’PART[Ś=(A1)*Ś=(A2)] (3.1)

where the elements of these descriptors are:

LOK	Local (consists of a maximum of 6 characters name of, e.g., substation or generating unit)
NUM	Numeral (consists of a maximum of 4 characters name of, e.g., voltage level or transformer)
PART	Partial (a maximum of 8 characters name of, e.g., busbar, line, bay)
A	attributes (such as ‘open’, ‘closed’ or ‘disturbed’ state of switching equipment)
Ś	absolute species (such as switches)
*Ś	relative species (such as remotely controlled).

The first three hierarchical levels “’’’LOK’’NUM’PART” indicate the **location** of the objects, and in actual application the element symbols are replaced by the terms in operators’ terminology. Thus an object is defined by a statement giving its location and species.

The use of quantifier “?” in the sense of an existence quantor on any of the hierarchical levels of GDL allows querying the process database in various manner [66]. As for example an expression of the form

’’’?’’?’?[V.*UL=?]

in query mode will cause the output of lists of location and values of **u**pper **l**imits of the nodal **v**oltages.

The GDL format is used in the simulator where the work described here was implemented on; thus, all the data dealt with are of this format.

3.6 Data Actualization

Three different sources of case data can be identified which have to be provided for expert systems supporting power system operation:

- Immediate process data which can be directly retrieved from the process (SCADA) database; examples would be measurement values, states of switching equipment, or events such as alarms or breaker trips.
- Pre-processed SCADA data which may be evaluated, condensed or pre-computed through application-specific routines, for instance the actual system topology (derived from the system connectivity and the actual states of switches) or the power flow.
- Information from outside the system under regard, such as power plant states or the availability of power from a neighboring utility, which have to be asked for by telephone calls of the operators to the corresponding control rooms and then entered by an appropriate man-machine interface (MMI).

3.6.1 Original Process Data

Retrieving immediate process data requires that the data formats used in the process (SCADA) database are understood by the numeric computation as well as symbolic computation. This can be achieved on principle by appropriate transformation interfaces. In the actual case of the work described here, the training simulator coupled with the expert system is used which is based on the Grid Data Language (GDL) process data system.

Due to the relationship to natural language, GDL descriptors can immediately be embedded in an expert system's rules as construing parts of them. This enables the expert system to conveniently retrieve SCADA information from the GDL process database by a corresponding interface [50,66,67]. In the present work described here, information about the available shunt capacitor banks and reactors and the nodes voltage feasibility range defined in the database are retrieved from the process database by the **database (DB) access** routine existing in the training simulator's environment in query mode using GDL descriptors of equation (3.1).

3.6.2 Pre-Processed SCADA Data

As described in [50,51], several algorithmic routines are available which observe the power system under regard with respect to sources (power units, tie lines), loads and network. These algorithmic routines take SCADA data as input,

and are addressed by the embodiment of their calling names within the corresponding numerical computation program for the actualization of the information. The results of their simulations are mapped into system object files (lists) and are made available in a predefined format to the numerical computation program of **real power dispatch**, **reactive power dispatch**, **reactive power controller pre-selection mechanism** and **state assessment,** developed within the frame of this thesis, through a communication channel. These routines are briefly reviewed as follows:

3.6.2.1 Generation Observer

This routine is a central observer which continuously monitors the operational status of each generating unit and external tie. Besides static information such as the generating unit's name, type and dynamic information such as its current status, it also provides information such as operational identifiers of interconnected region, generating units under consideration and actual house load power demand [50]. External power sources are modeled in lumped fashion. The data dealt with are the minimum and maximum reactive power capability of generating units determined from the generator power diagrams, rated and actual power output, the technical minimum power output of different types of generating units computed as percentage of rated power according to table 3.2b, their real and reactive auxiliary power demands as well as generator actual voltage set-points and their operating limits; furthermore, as information for the real power dispatch problem, the generating units' constant, linear and quadratic cost coefficients. A complete list of data provided by this routine for each of the developed modules is clearly enumerated in the left columns of tables 3.1, 3.2a and 3.3.

3.6.2.2 Load Observer

This routine combines an observer functionality on the current supply status of each load and a heuristic estimation of their recovery behavior. It also provides the static and dynamic information as well as operational identifiers of loads and reactive power compensation devices. The data are in the form of real and reactive power (capacitive or inductive) drawn by each consumer [50]. The daily load curves are also provided in the form of ASCII-files [41]. The corresponding values of active and reactive powers at any particular time of the day are made available thus making the load time dependent. A complete list of data provided by this routine on request through the communication channel for each of the developed modules is clearly enumerated in the middle columns of tables 3.1, 3.2a and 3.3.

3.6.2.3 Network State Assessment

This routine combines topology evaluation, load observer, generation observer, load flow, database access routine, branch (transformer and line) overloads and voltage profile checking functions to provide the information about the state of the network. Therefore, the **topology evaluation** program determines for the power system under consideration the actual topology from the potential topology (connectivity) lodged in the data-model and the state of the switching elements by filtering all topologically non-relevant objects, and establishes lists for nodes and branches [29]. This information includes the numbers and identifiers of islands (if the system is split) with their corresponding number of nodes. A complete list of data provided by the topology evaluation program for each of the developed modules is clearly enumerated in tables 3.1, 3.2a and 3.3, right columns.

3.7 Autonomous Execution of Operating Commands

Provided that remote operation from the control center is installed, actions suggested by a decision support system can be executed autonomously; a corresponding functionality for automatic operation of switching elements and commands for set-point change of power units was developed and presented in [50]. In the work described here, such functionality was used in order to enter commands addressing generation units in the form of computed optimum **real power set-points**, as well as commands for **voltage control set-points** into the process database of the training simulator used, and the corresponding power unit model responds accordingly. In case of **on load tap changing transformers**, the computed optimal tap position settings are executed automatically by an internal message executing utility program which, before the actual execution, checks for the remote controllability of the particular transformer as well as the physical feasibility of the tap change. The commands in the form of integer values of transformer steps are then passed on to the event processing and entered into the process database of the training simulator for further action. With regard to **shunt reactances (capacitors and reactors)**, the same principle applies except that the command to be passed on to the event processor is in the form of switching ‘on’ and ‘off’ the breakers and isolators of the shunt elements. This also presupposes the remote controllability of the switches.

Table 3.1 Summary of data set required by **reactive power dispatch module**

Data Actualization Program		
Generation Observer	**Load Observer**	**Topology Evaluation**
• generating unit's type, name in GDL format • current status • generating unit terminal voltages ▪ nominal (V_{Gi}^{nom}) ▪ minimum (V_{Gi}^{min}) ▪ maximum (V_{Gi}^{max}) • reactive power capability ▪ minimum (Q_{Gi}^{min}) ▪ maximum (Q_{Gi}^{max}) • actual power output ▪ real (P_{Gi}^{act}) ▪ reactive (Q_{Gi}^{act}) • generating unit house load power requirements ▪ real (P_{HDi}) ▪ reactive (Q_{HDi})	• load identifiers • consumer time dependent ▪ real power (P_{Di}) ▪ reactive power (Q_{Di}) demand • loading status • reactive power compensation devices	• total number of islands • number of nodes • nodal voltage levels • node locations specified in GDL format of eqn. (3.1) • branches (transformers and lines) ▪ initial and end nodes ▪ node's location in GDL format ▪ series resistance (r) ▪ series reactance (x) ▪ shunt conductance (g_0) ▪ shunt susceptance (b_0) per island • on-load tap changing transformers ▪ minimum step (TS_i^{min}) ▪ maximum step (TS_i^{max}) ▪ nominal step (TS_i^{nom}) ▪ tap ratio change T_i^{step}) ▪ phase angle (Φ_i) per island

Table 3.2a Summary of data set required by **real power dispatch module**

Data Actualization Program		
Generation Observer	**Load Observer**	**Topology Evaluation**
• generating unit's type, name in GDL format • unit generation cost coefficients ▪ constant (α) ▪ linear (β) ▪ quadratic (γ) • current status • real power capability ▪ minimum (P_{Gi}^{min}) ▪ maximum (P_{Gi}^{max}) • reactive power capabilities ▪ minimum (Q_{Gi}^{min}) ▪ maximum (Q_{Gi}^{max}) • nominal terminal voltage (V_{Gi}^{nom}) • actual power output ▪ real (P_{Gi}^{act}) ▪ reactive (Q_{Gi}^{act}) • generating unit house load power requirements ▪ real (P_{HDi}) ▪ reactive (Q_{HDi})	• load identifiers • consumer time dependent ▪ real power (P_{Di}) ▪ reactive power (Q_{Di}) demand • loading status • reactive power compensation devices	• total number of islands • number of nodes • nodal voltage levels • node locations specified in GDL format of eqn. (3.1) • branches (transformers and lines) ▪ initial and end nodes ▪ locations in GDL format ▪ rated apparent power flow (S^{max}) ▪ series resistance (r) ▪ series reactance (x) ▪ shunt conductance (g_0) ▪ shunt susceptance (b_0) per island

Table 3.2b Technical value of minimum power output computed as a percentage of rated power output

Generating Unit's Type	Type Identifier	Minimum Power Output $P_G^{min}=X\%\cdot P_G^{max}$ *
Thermal unit (TH)	1	30
Hydro unit (HY)	2	10
Gas turbine (GT)	3	10
Pressurized water reactor unit (PWR)	4	30
Boiling water reactor unit (BWR)	5	30

*Values obtained from [12,39]

Table 3.3 Summary of data set required by **reactive power controller pre-Selection mechanism**

Data Actualization Program		
Generation Observer	**Load Observer**	**Topology Evaluation**
• generating unit's type, name in GDL format • current status • generating unit terminal voltages ▪ nominal (V_{Gi}^{nom}) ▪ minimum (V_{Gi}^{min}) ▪ maximum (V_{Gi}^{max}) • reactive power capability ▪ minimum (Q_{Gi}^{min}) ▪ maximum (Q_{Gi}^{max}) • actual power output ▪ real (P_{Gi}^{act}) ▪ reactive (Q_{Gi}^{act}) • generating unit house load power requirements ▪ real (P_{HDi}) ▪ reactive (Q_{HDi})	• load identifiers • consumer time dependent ▪ real power (P_{Di}) ▪ reactive power (Q_{Di}) demand • loading status	• total number of islands • number of nodes • nodal voltage levels • node locations specified in GDL format of eqn. (3.1) • branches (transformers and lines) ▪ initial and end nodes ▪ locations in GDL format ▪ series resistance (r) ▪ series reactance (x) ▪ shunt conductance (g_0) ▪ shunt susceptance (b_0) per island • availability of on-load tap changing transformers