# Power Systems Generation Scheduling and Optimisation Using Evolutionary Computation Techniques

A thesis submitted for the degree of

Doctor of Philosophy

by

**Shadrack Otieno Orero**

Department of Electrical Engineering and Electronics

Brunel University

June 1996

# Power Systems Generation Scheduling and Optimisation Using Evolutionary Computation Techniques

**Shadrack Otieno Orero**

Department of Electrical Engineering and Electronics,
Brunel University, Uxbridge, Middlesex, UB8 3PH, U.K.

Ph.D Thesis, 1996

## Abstract

Optimal generation scheduling attempts to minimise the cost of power production while satisfying the various operation constraints and physical limitations on the power system components. The thermal generation scheduling problem can be considered as a power system control problem acting over different time frames. The unit commitment phase determines the optimum pattern for starting up and shutting down the generating units over the designated scheduling period, while the economic dispatch phase is concerned with allocation of the load demand among the on-line generators. In a hydrothermal system the optimal scheduling of generation involves the allocation of generation among the hydro electric and thermal plants so as to minimise total operation costs of thermal plants while satisfying the various constraints on the hydraulic and power system network.

This thesis reports on the development of genetic algorithm computation techniques for the solution of the short term generation scheduling problem for power systems having both thermal and hydro units. A comprehensive genetic algorithm modelling framework for thermal and hydrothermal scheduling problems using two genetic algorithm models, a canonical genetic algorithm and a deterministic crowding genetic algorithm, is presented. The thermal scheduling modelling framework incorporates unit minimum up and down times, demand and reserve constraints, cooling time dependent start up costs, unit ramp rates, and multiple unit operating states, while constraints such as multiple cascade hydraulic networks, river transport delays and variable head hydro plants, are accounted for in the hydraulic system modelling.

These basic genetic algorithm models have been enhanced, using quasi problem decomposition, and hybridisation techniques, resulting in efficient generation scheduling algorithms. The results of the performance of the algorithms on small, medium and large scale power system problems is presented and compared with other conventional scheduling techniques.

## Acknowledgements

First and foremost I wish to express my profound thanks and gratitude to my supervisor Professor Malcolm Irving for introducing me to the exciting research on evolutionary computation and for allowing me the independence to carry out this work. I am most grateful for his sound guidance, kindness, meticulous supervision and the provision of an enabling research environment within the Brunel Institute of Power Systems.

I would also wish to thank the following:

- All members of the research group at the Brunel Institute of Power Systems, for all their encouragement and fruitful discussions. I have never worked with a more professional and hard working group. Special thanks go to Jeremy Gann for always ensuring that the computer system was up and running and Dr. Chris Macqeen for providing me with the shell scripts for preparing this thesis.
- The members of the wider genetic algorithm and power system research communities who responded to my queries and discussions in one way or the other. To all those who provided their most useful research publications at various sites on the internet, I say thank you, for without some of your publications, this work would not have been completed in good time.
- All my teachers and mentors throughout my academic and professional careers
- My parents, for their belief in me and for teaching me the value of hard work.
- All my friends who have provided the moral support whenever I needed it.
- The Overseas Development Agency for providing me with the financial assistance necessary for this work.

Finally and most importantly, I must thank my wife, Millicent for her love, patience understanding and encouragement through the years and my three wonderful children, Curie, Victoria and Emannuel for keeping me sane while this work was going on.

# DECLARATION

The work described in this thesis has not been previously submitted

for a degree in this or any other university,

and unless otherwise referenced it is the author's own work.

# STATEMENT OF COPYRIGHT

# List of Principal Symbols and Abbreviations

## Chapter 1

| | |
|---|---|
| EA | Evolutionary algorithms |
| EMS | Energy management system |
| EP | Evolutionary programming |
| ES | Evolutionary strategies |
| GA | Genetic algorithms |
| GALESIA | Genetic algorithms in engineering systems: Innovations and applications |
| ICGA | International conference on genetic algorithms |
| NGC | National grid company |
| P(t) | population in generation t |

## Chapter 2

| | |
|---|---|
| H | schemata |
| M(H,t) | number of strings in the population with schema H in generation t |
| M(H,t+s) | expected number of strings with schema H after selection, |
| f(H,t) | observed fitness of the schema H in generation t |
| $f_{avg}$ | average fitness of all the strings in the population |
| L | chromosome string length |
| ε(H,t) | probability of schema disruption by an operator in generation t |

## Chapter 3

| | |
|---|---|
| ξ | formae or equivalence class |
| $f(c_i)$ | fitness of $i^{th}$ child |
| $f(p_i)$ | fitness of $i^{th}$ parent |
| $F(x_i)$ | value objective function, F |
| DCGA | Deterministic crowding genetic algorithm |

## Chapter 4

| | |
|---|---|
| N | number of synchronised units |
| $P_i$ | loading level of the $i^{th}$ unit in MW, |

| | |
|---|---|
| $P_D$ | total load demand in MW |
| $P_L$ | total transmission losses |
| $F_i$ | fuel input-power output cost function for the $i^{th}$ unit |
| $P_i$min(max) | minimum (maximum) generation limits of the $i^{th}$ unit |
| $\lambda$ | the system incremental cost |
| k | iteration count |
| $L_i$ | string length representing the power output of the $i^{th}$ unit |
| L | the total string length representing all the system unit loadings |
| $\Psi$ | penalty function for not satisfying load demand |
| $\Phi$ | penalty function for unit loading in a prohibited operating zone |
| $n_i$ | number of prohibited operating zones of the $i^{th}$ unit |
| $p_{i,k}^{L,U}$ | lower / upper bounds of the $k^{th}$ prohibited zones of unit i |
| ARMA | autoregressive moving average |

## Chapter 5

| | |
|---|---|
| $\alpha_i$, $\beta_i$ $\gamma_i$ | power output-cost coefficients of the $i^{th}$ unit |
| $\sigma_i$ | hot start up cost of the $i^{th}$ unit |
| $\delta_i$ | cold start up cost of the $i^{th}$ unit |
| $\tau_i$ | cooling time constant of the $i^{th}$ unit |
| $u_{i,t}$ | off/on, [0,1], status of the $i^{th}$ unit |
| $F_T$ | total production cost |
| T | scheduling period |
| $FC_i$ | fuel cost of the $i^{th}$ unit |
| $SC_i$ | start up cost of the $i^{th}$ uni |
| $SD_i$ | shut down of the $i^{th}$ unit |
| $T_{off/on,i}$ | time the $i^{th}$ unit has been off/on prior to start up/shut down |
| ANN | Artificial neural networks |
| DP, LP, QP, MIP | Dynamic, linear, quadratic and mixed integer programming |
| EPRI | Electric Power Research Institute |
| LR | Langrangian relaxation |
| NF | Network flow |
| SA | Simulated annealing |
| $SO_2$, $CO_2$, $NO_2$ | sulphur dioxide, carbon dioxide and nitrogen dioxide respectively |

## Chapter 6

| | |
|---|---|
| $F_T$ | sum of fuel, start up and shut down costs |
| $F_G$ | total cost function |
| $a_i$, $b_i$, $c_i$ | power output-cost coefficients of the $i^{th}$ unit |
| PSC(i,t) | premature start up penalty cost of the $i^{th}$ unit during time t |
| PSD(i,t) | premature shut down penalty cost of the $i^{th}$ unit during time t |
| PR(t) | failure to meet reserve penalty cost during time t |
| $f_i$ | fitness of a string |
| $R_t$ | reserve margin at time t |
| pst(i,t) | number of time steps the $i^{th}$ unit has been prematurely started |

| | |
|---|---|
| psd(i,t) | number of time steps the $i^{th}$ unit has been prematurely shut down |
| CPU | central processing unit |
| MIN | minimum |
| P(x) | penalty function |
| PGEN | available generation capacity, MW |
| PDEM | load demand, MW |
| RES | system reserve, MW |
| PSTCOST | premature start up cost |
| PSDCOST | premature shut down cost |
| FTRCOST | failure to meet reserve penalty cost |
| SPR | spare capacity, in MW |

## Chapter 7

| | |
|---|---|
| gen. | generation |
| N | number of units |
| HR | hour |
| PSTCOST | premature start up cost |
| PSDCOST | premature shut down cost |
| FTRCOST | failure to meet reserve penalty cost |
| T | scheduling period |

## Chapter 8

| | |
|---|---|
| F | total fuel cost of thermal system |
| $F_i$ | fuel cost of $i^{th}$ thermal unit |
| $P_{si,t}$ | loading of $i^{th}$ thermal unit at time t |
| $P_{hi,t}$ | generation level of hydro $i^{th}$ unit at time t |
| $V_{hi,t}$ | storage volume of $i^{th}$ reservoir at time t |
| $Q_{hi,t}$ | water discharge rate of $i^{th}$ reservoir at time t |
| $P_{D,t}$ | load demand at time t |
| $P_{L,t}$ | total transmission line losses at time t |
| $S_{hi,t}$ | spillage of of $i^{th}$ reservoir at time t |
| $I_{h,it}$ | inflow rate of $i^{th}$ reservoir at time t |
| $H_{it}$ | net head of $i^{th}$ reservoir at time t |
| $\alpha, \beta, \gamma$ | thermal generation cost coefficients |
| $C_{i1},...,C_{i6}$ | hydro power generation coefficients |
| $\tau_{i,m}$ | water transport delay from reservoir m to i |
| $R_{ui}$ | set of upstream units directly above $i^{th}$ hydro plant |
| $R_h$ | set of hydro plants in the system |
| $R_s$ | set of thermal units in the system |
| i, t, T | unit index, time index and scheduling period respectively |
| $V,i^{begin}$ | initial storage volume of $i^{th}$ reservoir |
| $V,i^{end}$ | final storage volume of $i^{th}$ reservoir |

# Contents

# Figures

## Tables

# Chapter 1. Introduction

## 1.1 Power System Operation and Control

Optimal power system operation requires a blend of engineering experience with the adoption of powerful mathematical optimisation techniques. Power systems the world over are currently undergoing major revolutionary structural changes in the way they are operated and controlled. These changes, coupled with the increased network complexity, pose major challenges to the utility operators. Because of the competitive nature of the power supply industry, the present power systems are operated near their physical limits. In order to satisfy consumers who are now more aware of their rights and the choices available, utilities must find computational techniques that enable them to run an efficient service as economically as possible.

The availability of powerful computers and the accompanying enormous increase in both computation speed and memory storage capabilities have resulted in a re-assessment of power system optimisation methods. Techniques that previously could not be given a consideration are now back in the list of possibilities. For example some problems that could not be solved numerically due to time and memory limitations can now be solved with relative ease. However some problems are still too large to be solved in a reasonable time frame using conventional solution methods, and the search for new and better computational techniques continues. One of the operational control problems which has not been satisfactorily solved is generation scheduling in practical power systems.

## 1.2 Generation Scheduling

The generation scheduling function is one of the core components of a modern power system energy management system (EMS). The EMS helps in the determination of the generation level of each unit by minimising utility wide production costs while meeting system and unit constraints. The generation scheduling function has to satisfy the main objective of economics, which involves an optimisation of cost over a future period of time. The economic dispatch sub function which optimises operation cost over a much shorter time interval is embedded in the generation scheduling function. Figure 1.1 shows how the

generation scheduling functions fit in the overall EMS structure of a modern power system control.

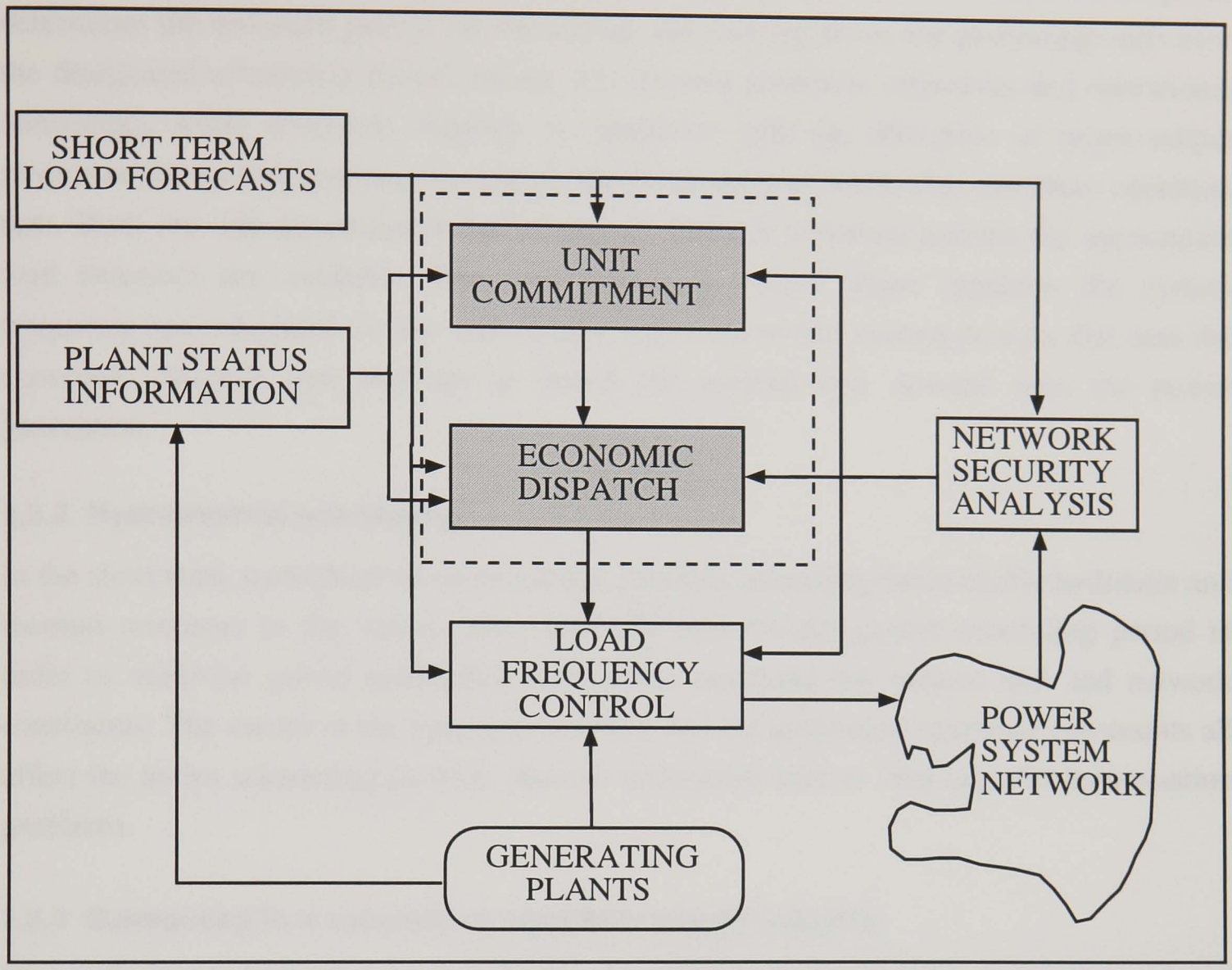



**Figure 1.1 Generation scheduling and control functions in a modern EMS**

The advances in computer hardware and communications technologies makes real time plant and network data available for the generation scheduling and control functions. Most generating units are now equipped with on line performance monitoring equipment that can provide very accurate up to date information about the equipment performance. This means that unit performance characteristics used in generation scheduling can be more accurately determined. The ideal generation scheduling system for the present day EMS system should:

- incorporate real time plant information in modelling plant characteristics,
- take into consideration real and reactive power system constraints,
- satisfy environmental considerations,

These requirements call for reliable and robust optimisation algorithms.

### 1.2.1 Scheduling thermal generators

The main objective of thermal generation scheduling is to minimise the cost of power production while satisfying the physical limitations on the power system and the individual units. An optimal generation schedule can result in substantial economic benefits, especially

for large scale systems. The generation scheduling problem can be considered as a power system control problem acting over different time frames. The unit commitment phase determines the optimum pattern for starting up and shutting down the generating units over the designated scheduling period, taking into account economic objectives and operational constraints, while economic dispatch is concerned with the allocation of target output powers to the on line generators to satisfy the predicted load demand at minimum operation cost. Both the unit commitment and economic dispatch functions assume the appropriate load forecasts are available. The load frequency control phase regulates the system frequency and scheduled tie line interchange, and is an on line control process that uses the economic dispatch unit loadings to match the system load demand with the power generation.

### 1.2.2 Hydrothermal scheduling

In the short term, hydrothermal co-ordination involves allocating the available hydraulic and thermal resources to the various time intervals over the designated scheduling period in order to minimise power production costs while satisfying the various unit and network constraints. The nature of the hydraulic network and the associated operating constraints all affect the hydro scheduling process, and can sometimes lead to very complex optimisation problems.

### 1.2.3 Scheduling in a competitive electricity supply industry

In a deregulated electricity supply industry, the scheduling of generation is usually entrusted to a designated authority, that runs *a pool* trading system. For example in the U.K., the national grid company (NGC) decides which power stations are to be scheduled, taking into account factors such as: the load pattern, amount of power generation capacity offered by the different generators at each time interval over the scheduling period, the price at which the various capacities of generation can be supplied, the constraints in supplying the said capacities, transmission network constraints etc. These factors are all then taken into account in determining the wholesale price at which electricity is traded in the pool. At the moment the price is evaluated at half hourly intervals. This price is dependent on the generation schedules of the various power generators, compiled by the NGC using their chosen scheduling algorithm. The decision to admit any generators' plant in the schedule will depend on their offer prices covering the scheduling intervals over which they want to bid. These offers must be provided to the NGC in advance. Since the the pool price is usually set as the highest offer price by a power station accepted and included in the schedule, the optimality of the scheduling algorithm is critical, since the costs and profits for the separate entities involved in the trade are affected by the performance of the scheduling program.

The quality of solution provided by the scheduling algorithm will affect every one using electricity since the the marginal price for electricity set in the trading pool (the wholesale market price) is determined by the scheduling algorithm. Large industrial consumers who pay pool prices for the commodity can be directly affected by the variation in pool prices. With the freedoms accompanying the deregulated electricity industry some people have even gone further to suggest that the pool operator should be sued, if it can be proved they scheduled power stations unfairly or provided schedules at higher than necessary cost. Thus it is even more critical than before to get the generation scheduling function as near optimal as possible.

## 1.3 Conventional Generation Scheduling Techniques

Before an optimisation method is applied, the particular power system must be adequately modelled, taking into account the various system and unit operational constraints. The choice of a particular solution technique will depend on a number of factors such as:

- the allowed computation time,
- problem size,
- desired accuracy of solution,
- level of modelling detail,

among others. The generation scheduling problem for practical power systems is usually quite complex due to the varied number of operation scenarios that have to be taken into account. It is quite common to combine two or more solution techniques to tackle the problem. Although a number of techniques have been applied in solving the scheduling problem, there is no single technique that has proved to be universally applicable, as each technique has its own shortcomings. These have included methods such as:

- Heuristics, expert systems and artificial intelligence based methods,
- Dynamic Programming
- Linear, Quadratic and Mixed Integer Programming,
- Mathematical Decomposition Techniques,
  - Lagrangian Relaxation,
  - Benders Decomposition,
  - Branch and Bound,
- Simulated Annealing,
- Artificial Neural Networks.

## 1.4 Evolutionary Computation Techniques

### 1.4.1 Introduction

In the last decade interest in computational techniques that have analogies with biological and physical processes have gained widespread use in the solution of complex problems. One such group of techniques is called evolutionary algorithms. Recent studies report on the success of these algorithms in the solution of a number of practical problems.

Evolutionary algorithms are computational techniques bearing a close similarity with Charles Darwin's theories of evolution of biological species. The flexibility and robustness of these algorithms has enabled them to be applied in a wide range of problem solving areas.

As early as the 1950s scientists had began studying evolution theory with a view to applying this to optimisation problems. [Goldberg, 1989] highlights the crucial pieces of work that led to the birth of the field of evolutionary computation. The works of [Box] and [Bremerman] represent some of the early attempts to utilise one form or the other of what can be termed evolutionary computation. There was very little mathematical analysis of the computation process at that stage.

Later, [Rechenberg] working at Berlin University introduced what has become known as *Evolution Strategies* (ES) [Back et. al., 1991] and used it mainly for function optimisation. This work was further developed by [Schwefel] culminating in the book *Numerical Optimisation of Computer Models*, whose title did not bring out the considerable amount of work on evolution strategies contained inside. Schwefel carried out a comprehensive analysis of various standard mathematical test functions and compared the performance of the evolution strategy with other conventional optimisation techniques on these functions. He found out that his evolution strategy did better than the standard techniques on the more difficult test functions.

While Rechenberg and Schwefel were working on evolution strategies, [Fogel et. al.] were also developing *Evolutionary Programming* (EP), a technique, where candidate solutions to a given problem are represented as finite state machines, with selection and mutation as the main search operators.

Genetic algorithms were first described by [Holland, 1975] in the 1960s and further developed by [Holland] and his students at the University of Michigan in the late 1960s and early 1970s. Holland's book, *Adaptation in Artificial and Natural Systems,* first published in 1975, provided the most solid theoretical foundations of the genetic algorithm computational model. Holland's evolutionary model uses a population of binary strings to represent the solutions to a problem, and uses the genetic operators of selection, crossover, inversion and mutation to advance the search for better solutions. Mathematical analysis of the Holland genetic algorithm model is based on the concept of *schema processing* [Holland, 1975], [Goldberg, 1989].

Since the pioneering work of [Holland], research in the evolutionary computation field has flourished, especially after publication of the books *Genetic Algorithms in Search, Optimisation and Machine Learning* by [Goldberg, 1989], *A Handbook of Genetic Algorithms* by [Davis, 1991] and the publications of the bi-annual *International Conferences on Genetic Algorithms* (ICGA), *IEE / IEEE* (GALESIA), *Foundations of Genetic Algorithm Workshops (FOGA)* and the *Parallel Problem Solving in Nature (PPSN)* series of conferences. A brief review of the various evolutionary computational techniques is presented in the next section.

### 1.4.2 Evolutionary computational models

Evolutionary computation uses computational models of biological evolutionary processes as key elements in the design of computer based problem solving systems. All the evolutionary algorithm models are based on a learning process using a population of potential solutions to a problem, each representing a search point in the Euclidean space of all the possible solutions. The initial population is usually randomly created, and evolves towards the optimal solution, according to rules of selection and other operators such as recombination and mutation. Each individual in the population is awarded a fitness measure, based on the problem objective function, which determines its ability to propagate its elements in the search process. The recombination allows the mixing of parental information, while the mutation operator introduces diversity in the population. Figure 1.2 outlines a typical evolutionary algorithm.

```
gen.=0
create initial population P(gen.)
evaluate P(gen.)
Do while (not converged)
    gen.=gen.+1
    select P(gen.)
    recombine P(gen.)
    mutate P(gen.)
    evaluate P(gen.)
    create new P(gen.)
End do
```

**Figure 1.2 A typical evolutionary algorithm**

A population of individual structures is initialised and then allowed to evolve from generation to generation, until a given stopping criterion is satisfied. The selection step decides the number of copies each element contributes to the mating pool, while the child population is created via recombination and mutation. The "*create new*" operation decides which elements form the next population. Although the algorithm appears very simple, it provides a very sophisticated and robust search mechanism. Although the various evolutionary algorithm models are quite similar at the system level, each of the varieties is implemented in a different way.

They differ in ways such as:

- problem representation,
- selection mechanisms,
- genetic operators, their combinations and emphasis,
- performance measures.

The next section provides a summary of the various implementations.

### 1.4.3 Variations of the evolutionary algorithms

#### 1.4.3.1 Genetic Algorithms

Genetic algorithms (GA) [Holland, 1975], [Goldberg, 1989] are one of the most widely studied evolutionary techniques. These techniques use a more domain independent representation. In a typical GA, parent selection is based on a probabilistic function based on the relative fitness of a particular structure. The child population is formed from the selected parent population via crossover and mutation operators, with the mutation operator being applied with a small probability. The crossover operator is emphasised as the main search operator.

#### 1.4.3.2 Evolution Strategies

Evolution strategies (ES) as initially developed by Rechenberg [Back et. al., 1991], used selection and mutation with a population size of one, to create one offspring per generation using Gaussian mutation. [Schwefel] Introduced recombination and population sizes greater than one, a strategy where pairs of parents recombine to produce children, which are further pertubed by mutation, creating more children than parents. Deterministic rules are used to create the new population, where for example the best N children replace N parents. Recent applications have focused on dynamically adapting the mutation rate as the search progresses.

#### 1.4.3.3 Evolutionary Programming

Evolutionary programming (EP) [Fogel et. al.], or Genetic programming (GP) [Koza] uses representations that are more tailored to the problem domain, such as real number coding schemes or ordered lists. No recombination is used, and the method relies mainly on the selection and mutation operators. In an EP, after selection, all parents are mutated producing N children. The new population is created from both the children and parent population using a probabilistic function based on fitness. The mutation operators are defined according to the problem domain and are usually made adaptive as the search sequence progresses.

### 1.4.4 Advantages of evolutionary computation techniques

The work described in this thesis is mainly concerned with the solution of generation scheduling and economic dispatch phases of the generation control function using evolutionary computation techniques.

The main reasons for choosing the evolutionary computation methods in the solution of the generation scheduling problem are:

1. The generation scheduling problem is a complex large scale mixed integer non linear optimisation problem, which has given enormous challenges to the presently available computational techniques. The flexibility, computational simplicity, robustness and recent success of the evolutionary algorithms in other complex problem domains makes such methods attractive.

2. The robust nature of these algorithms can enable them to be applied to a wide range of other difficult power system optimisation problems, apart from the generation scheduling problem.

3. The evolutionary algorithms are not restricted to the availability of the problem function derivatives, and may be used for the solution of continuous, combinatorial, or mixed integer nonlinear optimisation problems without major shortcomings. These properties make such algorithms suitable for the generator scheduling problem, which is subject to a large number of nonlinear objective function and constraints.

## 1.5 Scope of the Thesis

This work is concerned with an investigation of the suitability of the evolutionary computation technique in solving the generation scheduling problem for practical power systems with both thermal and hydro plants.

The effectiveness of the genetic algorithm as a power systems optimisation computational tool is demonstrated by developing a new computation technique for the generation scheduling problem.

The genetic algorithm solution technique is applied by using rigorous empirical testing combined with the application genetic algorithm fundamental theorems. Two fundamentally different genetic algorithm models, the canonical (standard) genetic algorithm and the deterministic crowding genetic algorithm have been developed and their performance evaluated on the economic dispatch, thermal scheduling and hydrothermal scheduling problems for practical power systems.

The unfolding theoretical developments concerning the relationships among the main genetic algorithm control parameter settings, such as population size, selection mechanisms, crossover methods and mutation rates have been fully tested using a number of generation

scheduling test problems. The work also reports on the limitations of the current GA theory on the solution of large scale problems and makes suggestions on future areas of further research.

## 1.6 Contributions of this Thesis

1. This work has reviewed the theoretical developments of genetic algorithms and their applications in the power systems field. A study of the GA fundamental issues such as problem representation, mechanisms for awarding the fitness function, choice of optimal GA parameters and GA convergence characteristics has been carried out.

2. A comprehensive genetic algorithm modelling framework for the thermal scheduling problem that includes the difficult problem constraints such as unit ramping and combined cycle multiple unit operating modes has been developed for practical power systems.

3. A detailed genetic algorithm modelling framework and solution technique for the hydrothermal co-ordination problem that includes a hydraulic sub-system with multiple cascaded reservoirs, water transport delay, variable head reservoirs and a non linear hydro power generation characteristics has been developed.

4. A new genetic algorithm based computational tool for the solution of the thermal generation scheduling problem and the economic dispatch sub problem has been developed.

5. It has been demonstrated that the basic genetic algorithm model must be modified in order to solve practical power system problems, and this has led to the development of both the canonical and the deterministic crowding genetic algorithm models for the solution of the generation scheduling problem.

6. This work has demonstrated that a decomposition approach to problem solving, even when the problem is not completely decomposable, can result in substantial improvements in algorithm performance.

7. A hybrid genetic algorithm, incorporating a priority list unit commitment method, has been developed and shown to be suitable for the solution of large scale thermal scheduling problems.

8. It has been demonstrated that the robustness of the genetic algorithm enables it to be used to solve complex problem scenarios encountered in power systems generation scheduling, provided that an innovative algorithm design is applied.

## 1.7 Thesis Layout

Earlier sections of this chapter introduce the basic generation scheduling problem and the available solution techniques. The scope of the thesis is then summarised, followed by highlights of the major contributions of this work.

In chapter 2, the theoretical basis of the genetic algorithm is presented, with discussion of the general features of the algorithm focusing attention on the various component parts and their implementations. The practical considerations and issues involved in the use of the various algorithm architectures as a practical computational tool are then analysed. The chapter concludes by providing an outline of some of the general areas of recent success of the genetic algorithm method.

In chapter 3, a summary of the theoretical foundations, modelling framework and applications examples of the deterministic crowding genetic algorithm is presented.

Chapter 4 considers the solution of the economic dispatch problem using the deterministic crowding and standard genetic algorithms. We present an analysis of the performance of the GA on a range of economic dispatch problems and discuss the conditions under which it would be useful to apply the GA to the dispatch problem. A comparison of the genetic algorithm methods with other conventional economic dispatch techniques is presented.

Chapter 5 begins with a discussion of the modelling aspects of the thermal generation scheduling problem, highlighting the various operation constraints and the practical difficulties encountered in solving the problem for practical systems. A literature review of the various solution techniques is then presented.

Chapter 6 considers the solution of the thermal scheduling problem on a range of test systems using the deterministic crowding and canonical genetic algorithm models. We present an analysis of the performance of the various GA modifications on the thermal scheduling problems and discuss the necessary steps in implementing a genetic algorithm method.

In chapter 7, a decomposition approach to the thermal scheduling problem is presented. A genetic and a hybrid genetic algorithm solution to the thermal scheduling problem based on the decomposition approach is presented.

Chapter 8 extends the thermal scheduling modelling by including the hydro-sub system models and concludes by providing a literature review of the main hydrothermal scheduling techniques to date.

In chapter 9, the solution of the hydrothermal scheduling problem using a genetic algorithm is presented. The system considered is that of a coupled hydro subsystem with river transport delays taken into account. The case of a parallel cascaded hydro network is also

considered. The genetic algorithm implementation details are given and results of numerical tests presented.

Chapter 10 draws together the main conclusions of the thesis, and makes some suggestions for future research, while Chapter 11 provides a list of references and bibliography for the thesis.

# Chapter 2. Genetic Algorithms

## 2.1 Introduction

Genetic algorithms (GA) are computational search techniques based on *models of genetic change in a population of individuals* bearing a close resemblance with the science of evolution and genetics. These models consist of three basic elements :

- A notion of *fitness* which governs the individuals ability to influence future generations,
- A reproduction operation which produces offspring for the next generation, through a *selection* and mating process and
- Genetic *operators* which determine the genetic make up of the offspring.

The distinguishing feature of a GA (from other function optimisers), is that the search process proceeds NOT by incremental changes to a single structure but by maintaining a *population* of structures from which new structures are created. Each structure in the population has an associated fitness, usually derived from the problem objective function which is used in a competitive environment to decide the structures that propagate their elements in the search for the optimal solution.

The basic power of a GA arises from the concept of *implicit parallelism,* [Holland, 1975], [Goldberg, 1989] the simultaneous allocation of trials to many regions of the search space. This theory suggests that through the repeated process of selection, crossover and mutation, the schemata (building blocks) of competing hyper-planes decrease or increase their presence in the population according to the relative fitness of those strings.

The basic genetic algorithm steps shown in Figure 2.1 are:

1. *create an initial random population,*
2. *assign a fitness to each population member and test for convergence, if converged, return best solution otherwise continue,*
3. *select the individuals that participate in the mating process,*
4. *create new offspring from the mating pool using the recombination operators,*
5. *replace old population with newly created population,*
6. *go to step 2.*

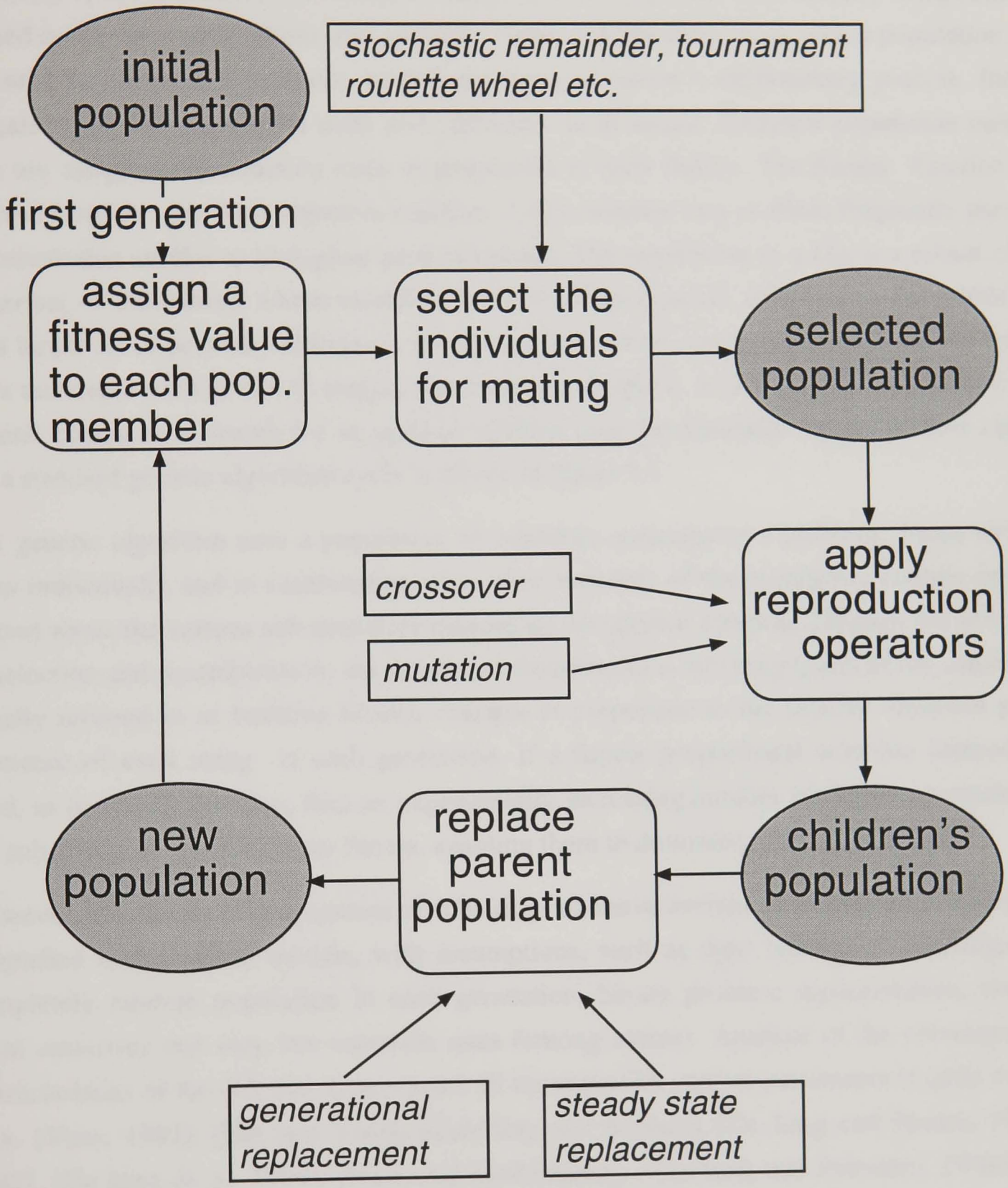



**Figure 2.1 A typical genetic algorithm cycle**

In the remainder of this chapter we provide a brief review of the historical development of genetic algorithms their theoretical foundations and the current state of research in this field. We focus on the theoretical developments and applications in fields related to power systems operations and control.

## 2.2 Theory

The basic theory of genetic algorithms can be found in [Holland, 1975], [Goldberg, 1989] with extensions to the theory in various genetic algorithm conferences [ICGA, 1985, 1987, 1989, 1991, 1993, 1995], [IEE / IEEE GALESIA, 1995] and other evolutionary computation related conference publications and journals. Genetic Algorithms work with a population of potential solutions to a problem, mimicking some of nature's evolutionary process. Individuals in the GA population mate and reproduce as in nature. Different population members are assigned reproduction rates in proportion to their fitness. The fitness function is derived from the problem objective function. A GA solution to a problem frequently uses a representation similar to biological gene structures. The population of a GA is a subset of a larger set of individuals whose members include all the possible solutions to the problem. This larger set of possible solutions is usually too large to be enumerated and hence the need for a technique such as GA to sample this large search space. A GA uses a combined set of genetic operators to search for an optimal solution over the parameter space. A flow chart for a standard genetic algorithm cycle is shown in figure 2.1

The genetic algorithm uses a population of candidate solutions to a problem. These structures individually and in combination with other members of the population contain information about the various sub structures making up the optimal solution. Through the process of selection and recombination, the number of instances of a substring (parts of the solution) usually referred to as building blocks, changes in proportion to the relative observed performance of each string in each generation. If a fitness proportional selection method is used, as is usually the case, then an exponentially increasing number of copies are made of the substrings of above average fitness, enabling them to dominate future populations.

However, this theory of propagation of substrings of above average performance is based on simplified mathematical models, with assumptions, such as tight linkage of substrings, a completely random population in each generation, binary problem representation, single point crossover and very low mutation rates (among others). Analysis of the convergence characteristics of the GA that incorporates all the main GA control parameters is quite complex. [Vose, 1992], [Nix and Vose], [Goldberg and Segrest], [De Jong and Spears, 1991, 1992], [De Jong et. al. 1994], [Back and Hoffmeister], [Xiaofang and Palmieri], [Whitley, 1994]provide a mathematical treatise on models that try to explain these characteristics. The rigorous mathematical models have only been used in the analysis of small population GA, with short string lengths. GA performance on practical problems involves a highly complex dynamic interaction between the operators of selection, crossover, mutation and the fitness functions that do not usually obey the simple substring propagation theory.

### 2.2.1 Schema analysis

The basic explanation of the robust performance of the genetic algorithm is based on the schema theorem, [Holland, 1975], [Goldberg, 1989], [Whitley, 1994], which postulates that while the fitness function is evaluated based on the performance of the whole string, information is gathered about all the component parts that make up the string.

#### 2.2.1.1 Schemata

The essentials of genetic algorithm theory were derived by viewing a standard GA as an algorithm that processes schemata. A schemata (H), $H \in \{0,1*\}^L$ is a description of a similarity template or hyperplane in L-dimensional bit space or a subset of strings in a population with similarities at certain string positions. Instances of a schemata H are all bit strings $a \in \{0,1\}^L$ which are identical to H in all positions where H has a value of 0 or 1. For example, considering binary strings of length 3 over the alphabet {0,1}, the two strings 011 and 111 are similar in the sense that they are identical when the first bit position is ignored. Regarding * as a symbol which may have either [0 or 1] value, the string can be represented as *11, thus *11 is described as a schemata, H.

#### 2.2.1.2 Schema theorem

The schema theorem provides a lower bound for schema growth in a GA population. Consider a schema H existing in a population at a given time t. After selection, schema numbers change according to

$$M(H, t+s) \geq M(H, t)\left[f(H,t) / f_{avg}\right] \qquad (2\text{-}1)$$

and after crossover, mutation and other operators, they change according to

$$M(H, t+1) \geq M(H, t)\left[f(H,t) / f_{avg}\right]\left[1 - \epsilon(H,t)\right] \qquad (2\text{-}2)$$

where:

M($H,t$) is the number of strings in the population with schema H (or are members of schema H) at time t,

M($H,t+s$) the expected number of strings with schema H after selection,

f($H,t$) the observed fitness of the schema H at time t (i.e. the average fitness of all the members of the population at a time t that are instances of the schema H),

$f_{avg}$ the average fitness of all the strings in the population and

$\varepsilon(H,t)$ the probability that a schema has been disrupted by an operator such as crossover or mutation. The schema theorem [Holland, 1975], summarised by equation 2-2 shows that

above average schemata receive exponentially increasing trials in the following generations. The choice of a binary alphabet for encoding maximises the number of schemata processed by a standard GA, and supports the hyperplane sampling process. The combination of shorter schemata (building blocks) to form longer and useful substrings is the most important working mechanism of the GA [Goldberg, 1989]. Using the theorem, the minimum proportion of a particular schema that is expected to be present in the succeeding generation of the trial can be evaluated. According to equation 2-2, the new distribution of points in each hyperplane representing a particular schema, should change according to the average fitness of strings in the previous population that contained the corresponding hyperplane partition. Further analysis [Holland] shows that even though the GA never explicitly evaluates the fitness of any schema, it implicitly changes the distribution of schema in the strings as if it had evaluated them, a process often termed *implicit parallelism*, which is thought to give the GA its robust sampling ability that enables it to obtain global solutions to many problems. Finite GA populations, do not however contain all instances of particular schemata, and sometimes hyperplanes might contain schemata that mislead the GA search, or an insufficient distribution of schemata necessary for finding an optimal solution. Artificial problems that contain objective functions that mislead the GA, so called deceptive problems [Goldberg, 1992], [Forrest and Mitchel], do not obey the schemata theorem and they provide an important research area for the development of a universal GA theory.

#### 2.2.1.3 Limitations of the schema theorem

Although the schema theorem has provided many insights on GA performance and has enabled tremendous advancement of the work on genetic algorithms, it does not give an exact distribution of the schemata in the population. The mathematical expression is an inequality that ignores string gains created by crossover and underestimates the string losses. It is based on a normally distributed population, a condition that is only accurate in the first generation before the population is biased by selection and other GA operators, thus it can only accurately predict the GA behaviour in the first generation. Thus looking at the average fitness of all the strings in a particular hyperplane (or using a random sample to estimate this fitness) is only accurate in the first or second generation [Grefenstatte and Baker, 1989], [Whitley, 1994]. After this, the sampling of strings is is biased and the schema theorem cannot guarantee an accurate prediction of the GA computational behaviour. Markov chain analysis can be used to complement and extend the schema theorem and provide better insights into the GA process.

### 2.2.2 Analysis of GA performance using Markov chains

Since a GA uses stochastic control parameters to guide the search form a random initial population, random process theory can be used to model its behaviour. A Markov process is specified by a matrix of transition probabilities which give the probability of moving from

one state to the next. The GA can be modelled as a Markov process in which the state of the GA in any given generation is given by the contents of the current population [Goldberg and Segrest], [Nix and Vose], [De Jong and Spears, 1991], [De Jong et. al., 1994]. The size of the matrix will depend on the granularity of the modelling required, for example it can be at string level, schema level or class level, where a class is a group of schema or strings sharing a common property. The state space of all possible population members representing any given problem solution provides the total region to be searched by the GA. An analysis of the population trajectories as the GA proceeds should provide some insights into the performance of the GA as it searches for the optimal solution. [Davis and Principe], [Mahfoud and Goldberg, 1992], provide a simulated annealing like convergence analysis theory for a simple GA.

The main limitation of the Markov chain analysis is the high computation burden implied by the treatment of the large matrices used for computing the transition probabilities, that are crucial in determining the population trajectories. Such models become unwieldy with increasing population size and string length, since the size of the transition matrix grows exponentially with increasing string length. Despite these difficulties, Markov chain modelling of GA on small test systems has provided important insights into the fundamental functioning of GA. Using matrix analysis, without manipulating the individual matrix elements [Nix and Vose], a steady state convergence analysis based on the assumptions of infinite population size with solution strings of infinite length can be obtained.

An analysis of the transient behaviour of a finite population GA [De Jong et. al., 1994] that attempts to answer questions pertinent to GA performance such as:

- the probability that a GA will contain a copy of the optimum solution at generation k,
- the probability that a GA will have a fitness value greater than some value at generation k,
- what is the expected best individual at generation k,

involves computing the individual elements of the transition probability matrices, each representing a string solution. To give an idea of the complexity of the computations, assuming a population size n, with strings of binary length L, the total possible number of states when modelling is done at the string level, is :

$$[n+x]! \ / \ [n!\,x!] \qquad (2\text{-}3)$$

where $x=2^L-1$

Raising the transition matrix to the power k, will yield a matrix containing probabilities of each solution after k generations. To put this in perspective, table 1 shows the variations of the number of states with string length and population size.

**Table 2.1 Variation of the number states in transition matrix with population size and string length**

| Population size | String length | | | |
|---|---|---|---|---|
| | 1 | 2 | 5 | 10 |
| 1 | 2 | 4 | 32 | 1024 |
| 2 | 3 | 10 | 528 | 524,800 |
| 5 | 6 | 56 | 376,992 | $9.4744388 \times 10^{12}$ |
| 10 | 11 | 286 | 1,121,099,408 | $3.6497489 \times 10^{23}$ |
| 20 | 21 | 1771 | $7.7535156 \times 10^{13}$ | $7.49423643 \times 10^{41}$ |

This number grows exponentially in n and L rendering an analysis of a transition matrix for a realistic GA implementation impossible. Availability of computational resources limits the ability to deal with large transition matrices as the population size and string length increase, restricting analysis at present to problems of small population size and string lengths. Recent results [De Jong et. al., 1994] show that such small scale models appear to hold up as the models are scaled up. The application of visualisation techniques, where the population trajectories are treated as fractals [Vose and Juliany], is also under exploration for the GA transient analysis. As computational power becomes more readily available there is hope that a full analysis of GA search mechanism will be obtained.

## 2.3 Genetic Algorithm Components

A typical genetic algorithm cycle involves four major processes of fitness evaluation, selection, recombination and new population creation as shown in the basic GA steps in figure 2.1. The choice and setting of the various GA control parameters such as population size, selection mechanism, fitness scaling method, crossover and mutation rates have a strong bearing on the algorithm performance. In order to use the genetic algorithm to solve any problem, the GA itself must be designed. The main GA design parameters to consider include:

- choice of an appropriate genetic algorithm model,
- population size,
- selection method,
- crossover and mutation operators,
- number of generations for evolution,
- parent replacement strategy,

### 2.3.1 Genetic algorithm population

A genetic algorithm is a population based search technique that derives its power from the fact that it advances its search based on feedback obtained from a number of potential solutions to the problem which form what is usually termed the GA population. The initial popu-

lation of solutions is usually generated randomly, although sometimes the search can benefit from inclusion of good previous solutions if available, but this must be done with utmost care, since lack of sufficient diversity in the initial population can easily result in premature convergence. The population size is generally fixed for all the generations of a run, although it is possible to have a variable population GA. The size of the population is one of the major GA control parameters. A lot of theoretical and empirical studies [Goldberg, Deb and Clark], [Grefensttete, 1986] provide guidelines on the choice of appropriate population sizes that guarantee good final optimal solutions. However there is, as yet, no empirical formulae linking the population size to the other GA variables or any problem specific parameters.

The population size affects both the ultimate performance and efficiency of a GA. GA usually do poorly with very small populations, because such populations might not provide a sufficient schemata sample. A large population is more likely to contain more representatives from a large number of hyper-planes, leading to a more informed search, discouraging premature convergence. However, a large population requires more evaluations per iteration (generation) possibly resulting in an unacceptable slow rate of convergence. Also in large populations a good string is more likely to miss the chance of a recombination or selection due to sampling errors. [Goldberg, Deb and Clark] have linked the length of the strings with population size. The population size is usually chosen after a number of trials runs on a given problem domain.

### 2.3.2 Selection

Selection in genetic algorithms is quite similar to *natural selection* in biological systems, where the higher the fitness, the higher the chances of an organism propagating its characteristics to future generations. This phase of the GA chooses the mating pairs from the population set according to objective function values, the better the function value, the higher the chance of selection. Selection provides the main driving force in evolutionary algorithms. Parent selection dynamics are based on the fitness measure which is some figure of merit computed using some domain knowledge such as the objective function of the problem at hand. There are several ways of implementing selection, and [Goldberg, Deb] compare some of the most widely used selection techniques. All of these selection methods allocate the copies in some proportion to the fitness of an individual relative to the population aver-

age fitness. Figure 2.2 shows an example of a general selection process for 10 individuals, where parents are selected randomly in proportion to their fitness.

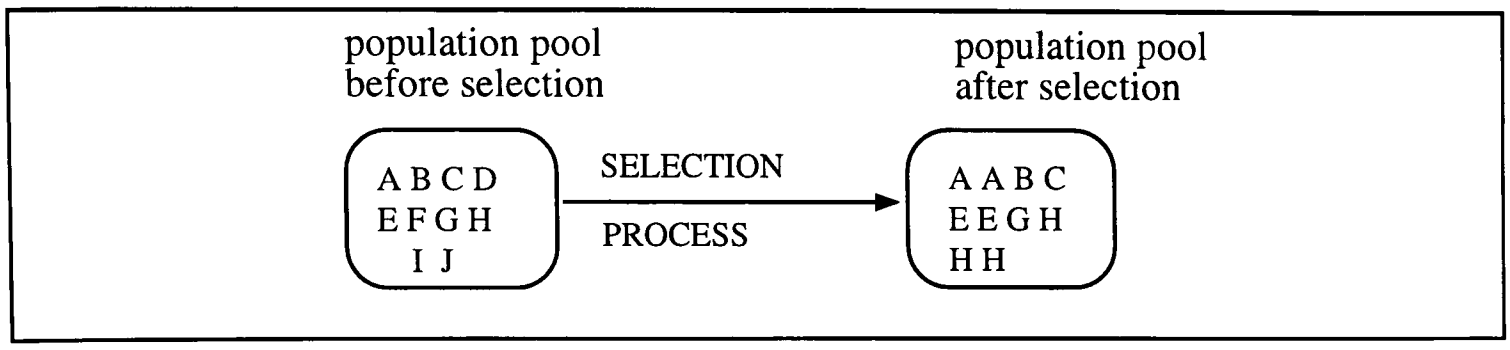



**Figure 2.2 A typical selection process**

#### 2.3.2.1 Roulette wheel selection

This involves a mapping into a *roulette wheel* [Goldberg, 1989], [Davis, 1991] where an individual is represented by a space that proportionally corresponds to its fitness. By repeatedly spinning the wheel, individuals are chosen using a stochastic sampling with replacement until the population is filled.

#### 2.3.2.2 Stochastic remainder selection

This is a proportional selection scheme that allocates copies of individuals according to the relative fitness of each individual, and the number of copies allocated is closest to the expected values. The integer portion of an individual relative fitness determines the number of copies of that string transferred to the mating pool. All strings include those with relative fitness's less than 1, place additional copies in the mating pool with a probability corresponding to the fractional portion of their relative fitness. A more elegant implementation of this method [Baker] is termed stochastic universal selection.

#### 2.3.2.3 Tournament selection

A more popular method of selection in recent years is tournament selection [Goldberg and Deb]. In k tournament selection, k individuals are chosen at random from the population, and the most fit individual is selected for mating, and the process repeated until the mating population is filled. Varying k varies the selection pressure, hence. fitness scaling can be done away with. When k is two, the process is usually referred to as binary tournament selection.

### 2.3.3 Fitness scaling

Fitness scaling is usually applied to the fitness values to prevent a super individual from dominating the population in subsequent generations, a condition that often leads to premature convergence. The premature convergence is caused by lack of diversity in the population due to a decrease in the variance of fitness. Fitness scaling avoids two processes that may hamper the proper convergence of the algorithm. The populations in the initial genera-

tions typically consist of relatively unfit individuals along with a few good ones. Although these good members are by no means optimal, the standard selection operator will provide them with a large number of offspring which increases the danger of premature convergence to a local optimum.

The second convergence problem occurs when most of the population members have nearly identical fitness function values, especially near the end of the GA iteration process. Direct use of the raw fitness function in conjunction with the selection operator leads to a slow convergence, since the selection operator then provides each member with nearly identical number of offspring.

Both these convergence problems can be avoided by *properly scaling or spreading out* the objective function values for all the population members before selection. The scaling basically fixes the relative spread between the highest and the average objective function values occurring in a population. In the early stages of the iteration process, the scaling reduces the relative spread in order to avoid premature convergence while in the later stages it enlarges the spread to speed up convergence. Scaling methods include: Linear scaling, Sigma scaling ,windowing [Goldberg, 1989], [Davis, 1991], ranking [Whitley, 1989].

### 2.3.4 Crossover operator

Reproduction involves creation of new offspring from the mating of two selected parents, or mating pairs. Chief among the mating operators applied to create new offspring is the cross over operator, which simply exchanges randomly selected elements between the mating pairs. There are several methods of implementing cross over such as:

- single point crossover,
- two point crossover,
- multi-point crossover,
- uniform crossover, among others.

The *cross over probability* or *rate* controls the frequency with which the operator is applied. The higher the value, the more quickly new structures are introduced into the population. If crossover rate is too great, high performance structures are discarded faster than selection can produce improvements, while if the rate is too low, the search may stagnate. It is thought that the crossover operator is mainly responsible for the global search property of the GA. The operator basically combines substructures of two parent chromosomes to produce new structures.

#### 2.3.4.1 Single point crossover

A single point crossover takes two individuals and cuts their chromosome strings at some randomly chosen position along the string length and swaps the strings after or before the

cut position among the parents to produce two new structures. A typical single crossover process is shown figure 2.3.

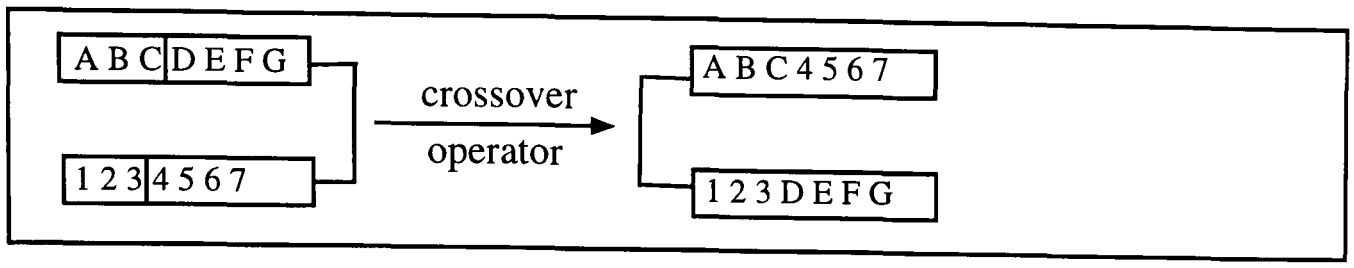



**Figure 2.3 Single point crossover process**

#### 2.3.4.2 Two point and multi-point crossover

For two or multi-point crossover operations, only the string bits lying between the randomly chosen crossover points are exchanged among the two parents to create the offspring. A typical two point crossover method is shown figure 2.4.

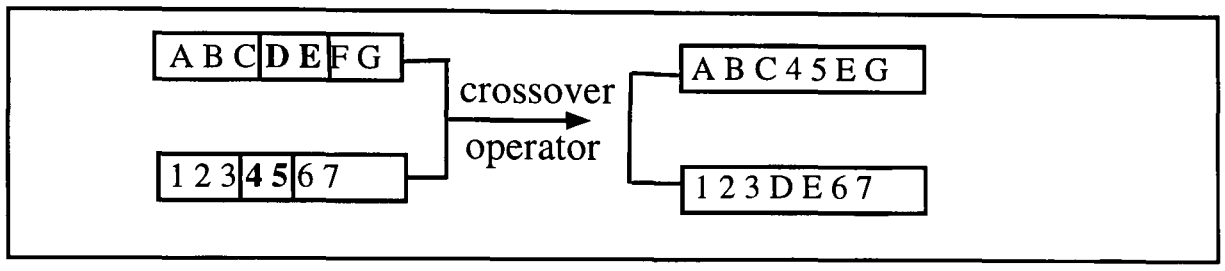



**Figure 2.4 Two point crossover process**

#### 2.3.4.3 Uniform crossover

Uniform crossover [Syswerda, 1989] is radically different to 1-point crossover. Each gene in the offspring is created by copying the corresponding gene from one or the other parent, chosen according to a randomly generated crossover mask. Where there is a 1 in the mask, the gene is copied from the first parent and where there is a 0, the gene is copied from the second parent, as shown in figure 2.5. The process is repeated with the parents exchanged to produce the second offspring. A new crossover mask is generated for each pair of parents. Offspring therefore contain a mixture of genes from each parent.

| | |
|---|---|
| *Crossover mask* | **1 0 0 1 0 1 1 1 0 0** |
| *Parent 1* | 1 0 1 0 0 0 1 1 1 0 |
| *Offspring 1* | 1 1 0 0 0 0 1 1 1 1 |
| *Parent 2* | 0 1 0 1 0 1 0 0 1 1 |

**Figure 2.5 Uniform crossover**

### 2.3.5 Mutation

The main reason for using the mutation operator is to prevent the permanent loss of any particular bit values, as without a mutation there is no possibility of re-introducing a bit value which is missing from the population. The mutation operator is used to inject new genetic material into the population and it is usually applied to each new structure individually. A

given mutation involves randomly altering each gene with a small probability. The mutation rate must be kept low since a high value tends to make the algorithm behave more like a random search strategy. A general mutation process is shown in figure 2.6.

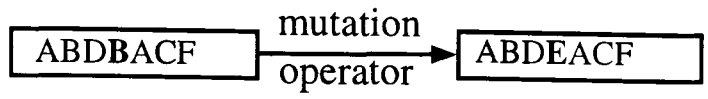



**Figure 2.6 Mutation process**

There are several ways for implementing the mutation operator for a GA with binary representation, [Davis, 1991], [Goldberg, 1989], some of these are:

- Generate a random integer number (m) between 1 and (L) where L is the string length, and make a random change in the $m^{th}$ element of the string,
- Sweep down the list of bits, replacing each bit by a randomly selected bit if a probability test is passed,
- Sweep down the list of bits, replacing each bit by its complement if a probability test is passed, as shown in table 2.2.

**Table 2.2 flip bit mutation operator**

| Initial string | Random numbers ( $P_{mut}$=0.008) | String after mutation |
|---|---|---|
| 1 0 1 0 | 0.84 0.16 0.22 0.34 | 1 0 1 0 |
| 1 1 0 0 | 0.11 0.09 **0.001** 0.85 | 1 1 **1** 0 |
| 0 0 1 0 | 0.25 0.43 0.77 **0.005** | 0 0 1 **1** |

### 2.3.6 Parent replacement method

In moving from one generation to the next, the old population should be replaced by the newly created offspring population in some optimal way that keeps the search for better solutions on the appropriate track. This step is important for the GA because it determines the degree of exploitation of the new search material in the advancement of the search for the optimal solution. The percentage of parents replaced in each generation is measured by a factor called generation gap.

#### 2.3.6.1 Generational replacement

Generational replacement is when the whole parent population is replaced by the child population, while the other extreme case where only one parent is replaced in each generation is termed steady state reproduction. When only a few parents are replaced, it is necessary to choose which parents are to be replaced by the offspring.

#### 2.3.6.2 Elitism

The probabilistic nature of the selection process cannot guarantee that the best population member will always be selected for reproduction. Even if it is, the other operators such as mutation and cross over might destroy it, and hence elitism tries to save it. Elitism *preserves the best population member* by copying it to the next generation intact. Experimental analysis has shown that elitism generally leads to local convergence at the expense of global convergence, but it speeds the search process. Sometimes in optimisation, it is good practise to keep track of the best solution obtained so far as the optimisation progresses. This is achieved in a GA by the *elitist* strategy.

### 2.3.7 Fitness evaluation

Before using a GA, one must find a some means of assigning a measure of quality, (fitness) to each structure in the search space. The fitness function is usually derived from the problem objective function, or through a simulation for a complex learning task. A GA searches for the optimal solution by maximising a given fitness function and therefore for a minimisation process, the problem objective function must be transformed to a non negative fitness value before being used in the GA cycle.

### 2.3.8 Problem representation

One of the main actions that link a GA to the problem it is solving is the way of *translating* the problem to a chromosome-like representation. In order to use a GA, a suitable structure representing the problem solution is necessary. The chromosome is typically a string of bits, although the representation is not restricted to binary, as real number [Antonisse], [Goldberg, 1991], [Davis], [Janikow and Michalewicz], [Radcliffe, 1992] and other higher cardinality alphabets [Koza] have also proved to be suitable. In the GA research community, the term genotype is often used to refer to the genetic make up (representation) of an individual solution, while the phenotype refers to the outward characteristics of an individual and corresponds to the decoded solution.

#### 2.3.8.1 Binary Encoding

Binary representation has been widely used for GA analysis, partly because of the ease of binary number manipulation by the various standard GA operators and the fact that GA theory is based on the binary alphabet. A minimal alphabet maximises the number of hyperplane partitions available in the encoding for schema processing. The next example demonstrates the relationship between the problem encoding and chromosome (string) structure.

##### 2.3.8.1.1 Binary problem encoding example

Consider the economic dispatch problem in power systems, in which the optimum generator loadings that result in minimum system operation costs are to be determined. Using the unit

power output as the main decision variable, each unit's loading range is represented by a binary number. The representation implicitly takes care of the unit minimum and maximum loading limits, $P_{min}$ and $P_{max}$ since the binary representation is made to cover the ratings between the limits. Once the required accuracy, or resolution in unit output is decided, the number of binary bits used to represent each unit's output can be calculated. Since each unit must be loaded within limits $P_{min}$ and $P_{max.}$, the value $P_i$ in the loading interval is represented by a string length, $L_i$

$$L_i = Log_2 \left[ \left[ \left( P_i^{max} - P_i^{min} \right) + \Delta P \right] / \Delta P \right] \tag{2-4}$$

where $\Delta P$ is the resolution in power output, and the chromosome length, L is

$$L = \sum_{i=1}^{N} L_i \tag{2-5}$$

The total string length is obtained by concatenating the binary bits representing each units output. To evaluate the unit power output levels (phenotype), the binary values are decoded to give the decimal equivalent value.

#### 2.3.8.2 Non-binary representations

A number of researchers have used non binary coded genetic algorithms in problems which do not map easily into a binary representation. Others [Antonisse], [Radcliffe, 1992], [Eshelman and Schaffer, 1992] have gone further to challenge the use of binary problem representation as opposed to real number treatment in general problem solving. Experimental evidence suggest that real number coding is also suitable for genetic search. The main argument against the use of higher cardinality alphabets is that these characters and their associated hyper-plane partitions will not be well represented in a finite population. This forces the use of large populations sizes in order to achieve equivalent statistical sampling as compared with binary representations. In order to use non binary representations, suitable mutation and crossover operators must be redefined.

##### 2.3.8.2.1 Non Binary problem encoding example

An example of a problem that can benefit from non binary number representation is the power system network partitioning problem [Taylor et. al.]. A numerical coding scheme is used where each node is allocated to one sub network. Suppose we have a power system having N busbars (Nodes), which we wish to partition into M sub networks. A given partition is represented by an N vector of M symbols. For example symbol x in position y in the N vector means that node y of the network belongs to sub network x. The coding fits quite well into binary representation if the partitioning is only required for two sub networks. This would involve using a binary string of length N, for N nodes. If a digit is a 1, then the corresponding node is in the first sub network, while a 0 puts the node in the second sub-network.

A similar interpretation can be extended to multi area partitioning, but using a integer number coding scheme. For example a possible string solution string for a 10 node network partitioned into 4 sub networks is:

**Table 2.3 An example of an integer number solution encoding**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 3 | 4 | 4 | 3 | 2 | 1 | 2 |

An interpretation of this representation is shown in table 2.4.

**Table 2.4 Interpretation of an an integer number solution encoding, showing partitioned network node distribution**

| Node | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| S-net | 3 | 2 | 2 | 3 | 4 | 4 | 3 | 2 | 1 | 2 |

The table 2.4 shows that node 1 is in sub network 3, node 2 in sub network 2, node 3 in sub network 2 etc. The fitness function for the partitioning problem is a function of the number of branches linking the sub networks, as well as the node balance amongst the sub networks.

### 2.3.9 Optimal GA parameter settings

Genetic algorithms are non linear in their behaviour and it is not always easy to determine the optimum combination of control parameters (population size, cross over rate , mutation rate etc.) that are best for the complete run. The optimum combination of parameters can and does change in the course of a run from generation to generation. Interpolation methods are sometimes applied to track down the variations in operator performance in the course of the run. There is empirical evidence [Davis, 1989], [Back, 1993], [Fogarty] linking the relative operator weights and the degree of convergence of a solution. Thus at any point in time, there is a ratio of operator parameters that would lead to optimal results and this optimal ratio changes throughout the run. At the moment there is a concerted effort to determine algorithm optimum parameter settings. Some of these efforts include; [Grefensttete, 1986, 1995], [De Jong and Spears, 1991], [De Jong et. al., 1994], [Schaffer et. al.], [Goldberg, Deb and Clark] among others.

## 2.4 Variations in Genetic Algorithm Models

### 2.4.1 Parallel Genetic Algorithms

Despite the genetic algorithm's long computation time when used on sequential computers, the algorithms have continued to enjoy a wide attraction because they can easily be implemented on parallel machines. Complex problems usually involve large populations and larger computational costs. In order to reduce the computation time needed to reach an acceptable solution, a number of attempts have been made to implement the genetic algorithms on parallel computers or on networks of loosely coupled workstations. There are dif-

ferent forms of parallel implementation [Muhlenbein, 1989], [Gordon and Whitley] depending on factors such as machine architectures and techniques of implementing the different GA operators, resulting in a number of parallel genetic algorithm structures as outlined in the next section.

#### 2.4.1.1 Global Parallel Genetic Algorithm

This class of parallelisation is characterised by an explicit evaluation and application of the genetic operators. Every individual in the population has a chance to mate with the rest. A single GA population is kept and the evaluation of the individuals is done in parallel, by assigning a subset of individuals to each of the processors available. There is no communication between the processors during the evaluation process, and communication only occurs at the start and end of the evaluation process. On a shared memory multiprocessor, the individuals can be stored in the shared memory, while on a distributed memory computer the population can be stored in one processor, the master processor, to simplify the application of the GA operators. The application of the GA operators can sometimes be done in parallel, for example tournament selection and mutation operators easily fit in a parallel implementation.

#### 2.4.1.2 Course Grained Parallel Genetic Algorithm

In the coarse grained parallel implementation [Cohoon et. al.], [Tanese 1987, 1989], [Belding] the population is divided into a few sub populations, keeping them isolated from each other. The sub populations interact through a migration operator that defines the level and nature of interaction among the sub populations. In the *Island model scheme* the population is partitioned into small sub populations by geographical isolation and individuals can migrate to any sub population, while in the *Stepping stone model* migration is restricted to neighbouring sub populations.

#### 2.4.1.3 Fine Grained Parallel Genetic Algorithm

Fined grained parallel GA [Robertson, 1987], [Spiessens and Manderick] [Muhlenbein, 1989], [Gorges-Schleuter] partition the population into a large number of very small populations, ideally allocating each individual to its own processor. This model requires massively parallel computer architectures for its implementation.

### 2.4.2 Crowding and Sharing Genetic Algorithms

The crowding and sharing genetic algorithms are specialised forms of genetic algorithms designed to deal with multi-modal function optimisation and classification problems that require the maintenance or identification of particular *niches* in the problem landscape. [De Jong, 1975], [Goldberg, 1989] proposed a crowding scheme, where offspring are compared with a few, (typically 2 or 3) randomly chosen individuals from the population, and

the offspring replaces the most similar one, using Hamming distance as a similarity measure.

In sharing [Deb and Goldberg], [Mahfoud, 1993], [Beasley et. al.] several individuals which occupy the same niche are made to share the fitness function among themselves, where the fitness given to an individual is, for example, reduced according to a function of the distance of each neighbour. Various implementations of both the sharing and niching methods is possible depending on the definitions of the niche radius, and the distance metric measures, [Mahfoud, 1995].

### 2.4.3 The *Genitor* Genetic Algorithm

Genitor [Whitley, 1989] is a steady state genetic algorithm implementation where only a few individuals are replaced in every generation and fitness is assigned according to the rank of the individuals in the population. Two parents are selected for reproduction and produce an offspring that is immediately placed back in the population, replacing the least fit, or some relatively less fit member of the population.

### 2.4.4 Genetic Programming

In Genetic Programming (GP) [Koza, 1992] each candidate solution in the population represents a tree structure computer programme. GP uses evolving programmes, with each problem solution represented as a programme, in an explicitly designed language for the particular task, to solve problems. Recombination combines two parent programmes into two syntactically valid children, as the tree structure representation allows a complete sub tree structure to replace another sub structure without disturbing the chromosome syntax.

### 2.4.5 Hybrid Genetic Algorithms

Hybrid genetic algorithms [Davis, 1991], [Grefenstette, 1991], [Kido et. al.], [Orero and Irving] combine a genetic algorithm method with other conventional problem solution strategies in an effort to benefit from the useful performance characteristics of each of the techniques.

## 2.5 Applications of Genetic Algorithms

### 2.5.1 General

The robustness of the genetic algorithm has enabled it to be applied in a wide range of problem solving areas. [Goldberg, Millman and Tidd], [ICGA Conferences, 1983-1995], [IEE / IEEE (GALESIA) Conference, 1995] contain a wide range of applications of genetic algorithms and other evolutionary computation techniques. Broadly classified, the GA has found widespread application in areas including:

- Computer aided design in all engineering branches,

- Pattern recognition and image processing,
- Artificial intelligence, machine learning and robotics,
- Power and telecommunications networks optimisation,
- Biotechnology and medical systems,
- Chemical process optimisation,
- Production planning and scheduling,
- Neural networks optimisation,
- Non linear optimisation.

### 2.5.2 Power System Optimisation

Interconnected power systems represent a large scale complex control problem. The safety and power supply security requirements coupled with the fact that electricity cannot be stored on a large scale, makes optimal power system operation and control very challenging. This control task involves the solution of large scale nonlinear optimisation problems, most of which cannot be solved adequately by conventional optimisation techniques. Evolutionary algorithms have shown tremendous promise in other application areas and recently they have been applied to a number of power system optimisation problems [Jarmo]. Some of these applications include:

- unit commitment [Sheble and Maifeld], [Epri], [DasGupta and McGregor], [Muller and Petrisch], [Karzalis et. al.], [X. Ma et. al.], [Orero and Irving, 1995, 1996], [Wong and Cheung],
- economic dispatch [K.P. Wong and Y.M. Wong], [Bakirtzis et. al.], [Walter and Sheble], [Sheble and Brittig], [F. Li et. al], [M. Ma et. al.], [Furong et. al], [Chen and Chang], [Orero and Irving],
- distribution system planning [Nara et. al], [Yeh et. al.], [Miranda et. al.], [Wen et. al],
- harmonic analysis in distribution networks [Boone and Chiang], [Lee et. al.],
- reactive power optimisation and voltage scheduling [Iba], [Wu and Ma], [Lee and Park],
- load flow solution [Yin and Germay], [Muller],
- network partitioning [Taylor et. al.], [Ding et. al.], [Orero and Irving],
- load forecasting [Maifeld and Sheble], [Yang et. al.],
- power stability and frequency control [Finch and Besmi], [Lansbery et. al.],
- maintenance scheduling [Langdon],
- hydro co-ordination [Hulselman et. al.]

# Chapter 3. Deterministic Crowding Genetic Algorithm

## 3.1 Introduction

Most practical problems have multiple solutions and sometimes it is necessary to find several (or even all) of these solutions. The traditional GA will usually converge to a single solution, and will miss some of the other solutions. In the standard generational GA [Holland, 1975], the entire population is replaced by the offspring formed through crossover and mutation, while in the standard steady state GA models [Whitley, 1989], [Syswerda, 1991] only a few parents are replaced by the offspring in each generation. In their pursuit for the optimal solution, both these models push the population towards a convergence where all individuals in the population become nearly identical, and even when multiple solutions to a problem exist, they tend to locate only one of them. A number of different GA enhancements are aimed at preserving the information available across the diverse population as the search progresses. One of the enhancements, that seeks to prevent convergence to a single solution by maintaining several sub-populations of individuals are *niching* genetic algorithms. These methods reduce competition among population elements when there is sufficient difference (or distance) between them, allowing sub-populations centering on good solutions (niches) to co-exist. Niching methods help to maintain population *diversity* in the population, a factor crucial to the prevention of premature convergence.

Niching methods fall into two broad categories; crowding and sharing. *Crowding* methods restrict the replacement of individuals by discouraging competition among widely differing individuals, while *sharing* methods derate an individual's effective fitness when similar individuals exist. The disadvantage of most traditional niching methods is the computational burden of comparing each individual to many other individuals, in order to determine the similarity measures.

Niching GA models such as crowding and sharing [Goldberg, 1989] are inspired by a corresponding natural ecological phenomena, where similar members of a natural population compete for the same resources. Niched GA models attempt to maintain a population of diverse individuals in the course of their runs. Genetic algorithm models that incorporate nich-

ing are thus capable of locating multiple optimal solutions within a single population. The niching method investigated in this work is based on the crowding concept.

The aim of the crowding genetic algorithm is to maintain population diversity as the run proceeds. In the crowding GA, the new population is created by letting the child population replace the parents that are most similar to them. De Jong [De Jong, 1975] provided a crowding factor model where only a fraction of the population reproduces and dies each generation, each newly created population replacing an existing member, preferably the most similar. Through the analysis and modifications of De Jong's and other crowding methods, Mahfoud [Mahfoud, 1995] has provided a crowding modelling framework, resulting in the design of the deterministic crowding genetic algorithm (DCGA) which seems to exhibit extensive capabilities in solving a wide range of problems such as; multi-modal function optimisation, multi-objective function optimisation, simulation of complex and adaptive systems, machine learning and classification, among others. This crowding GA model is computationally efficient as each offspring is only compared with the two parents, competing with the most similar parent.

Deterministic crowding genetic algorithm, has for example, been recently applied to the optimisation and design of statistical quality control methods [Hatjimihail] and a range of other test problems [Mahfoud, 1995]. It is a generalised niching method that helps in overcoming the time and memory limitation problems faced by many practical GA, which use population sizes that cannot maintain the required diversity as the GA run progresses. It is also capable of forming and maintaining multiple or single solutions to a problem, and basically achieves this by providing selection pressure *within* but not across regions of the search space, leaving the search across the regions to the crossover operator. The combination of its crossover and selection-replacement mechanisms is mainly responsible for its success.

## 3.2 Theoretical Modelling Framework for Deterministic Crowding GA

The theoretical modelling framework for the crowding GA [Mahfoud 1993, 1995] is based on a generalisation of Holland's schema theorem [Holland, 1975], using the more general notion of *formae* [Radcliffe 1991, 1992] or *predicates* [Nix and Vose]. This is not the first time that Holland's schema theorem has been generalised to cover a wider class of problems. For instance, Goldberg and Lingle [Goldberg and Lingle, 1985] extended the schema theorem to cover permutation based problems such as the travelling salesman problem.

The performance of the crowding GA is based on an analysis that partitions the search space into a group of chromosomes termed *equivalence classes or formae*, where the classes are defined as having a one to one correspondence with peaks (troughs) in the fitness landscape, in the case of function optimisation. The forma or predicates framework is a more arbitrary (general) processing scheme than schema processing [Holland, 1975], which is restricted to

binary digit partitioning. An equivalence relation partitions the space of chromosomes, C, into a number of equivalent classes or formae $\xi_1, \xi_2, ..., \xi_n$ as illustrated in figure 3.1.

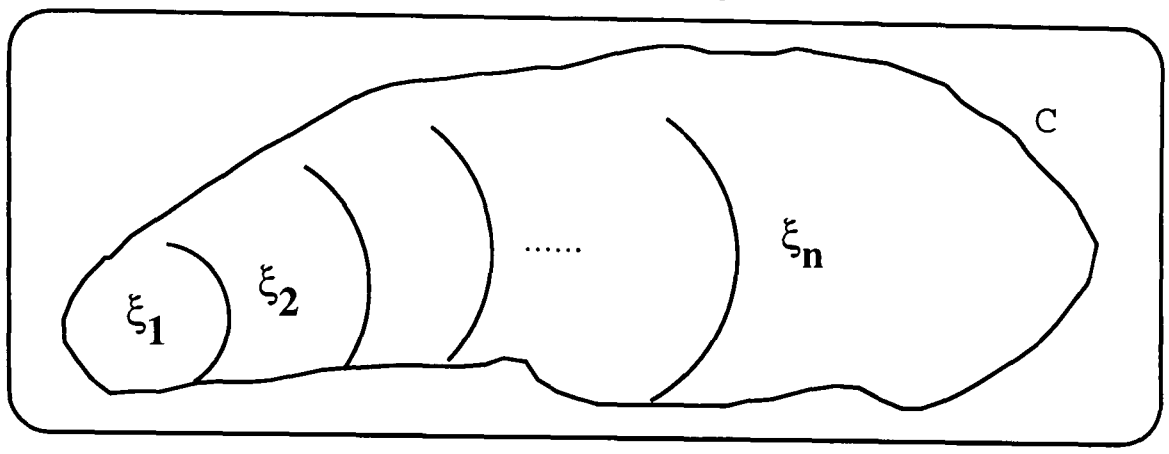



**Figure 3.1 An equivalence relation partitioning of the chromosomes' search space**

The generalisation requires the introduction of the notion of equivalence relations over the search space. Given any formae, (or equivalence class) $\xi$, an equivalence relation connects any pair of chromosomes or population members having the same alleles at the forma's defining positions. Thus a forma can be viewed as a set of chromosomes which are related by some (any) specific characteristic. The approach motivated by forma analysis in function optimisation is to choose relations which induce formae that are most appropriate at grouping together solutions having similar fitness, and it has been shown [Radcliffe, 1992] that such equivalence classes not only obey both the schema and forma theorems, but will also tend to result in relatively accurate fitness estimators.

For example in multimodal function optimisation, the classes can be made to correspond to local optima in the search space. In well defined functions, the local optima will correspond to schema partitions, but the DCGA models are not limited to the treatment of such functions. To assign each point in the search space to a particular class, the notions of *attractors* and *basins of attraction* is introduced, where a local optimal point is considered an attractor, and a basin of attraction is defined to be points in the search space which are within a given distance ($\varepsilon$), from a local optimum in the search space. Each class in the search space is assigned all the points within a corresponding maximum basin of attraction, hence each population element is considered to be a member of the class corresponding to the optimum point to which it is attracted, and all points in the search space are attracted to exactly one optimal point. The models are based on class maintenance rather than class formation, and it is therefore assumed that all the desired classes are available in the initial population. Another simplifying assumption in the analytical framework is the exclusion of the mutation operator. It is assumed that the mutation acts as a hill climbing operator and can be applied at the end of a GA run if necessary, since the DCGA incorporates a hill climber in its framework that puts equivalence classes in a one to one correspondence with local optima.

### 3.3 Deterministic Crowding GA Cycle

The main properties of the deterministic crowding GA that distinguish it from other GA models are: selection and population replacement are combined together, i.e. there is no selection prior to recombination as occurs in a standard GA, random crossover is performed on the whole population, there is no need for a scaling mechanism and the parents are replaced by the most similar children as they are created.

The deterministic crowding GA randomly pairs all population members in each generation to yield N/2 pairs of parents, for a population size N. Each such pair undergoes crossover, possibly followed by mutation to produce two offspring. Each of the two offspring compete with one of the two parents, chosen according to some similarity measure. The fitter among them then forms the population of the next generation. For example, given a pair of parents and their two offspring, two sets of parent-offspring tournaments are possible :

- set 1
  - parent 1 against 1$^{st}$ offspring
  - parent 2 against 2$^{nd}$ offspring
- set 2
  - parent 1 against 2$^{nd}$ offspring
  - parent 2 against 1$^{st}$ offspring

The set of tournaments that forces the closest competition is held, where closeness is the average distance between the parent-offspring couples in a set. Closeness is computed according to some appropriate distance measure, such as the difference in cost function, in the case of function minimisation. In the replacement-selection step, the method of competition between the offspring and the parents is crucial to algorithm performance, as a small change in the replacement strategy can yield an enormous change in the algorithm's capability. The main attractive features of the DCGA include:

- It is very simple to implement.
- There are few GA control parameters to be set.
- It is faster than other crowding methods since it performs replacement selection via binary tournaments and does not sample the whole population for either replacement or selection. Other crowding methods have to perform a comparative evaluation of population members in order to find the most similar ones to replace.
- It can handle both uni-modal and multi-modal function optimisation.
- It can easily be implemented in parallel since all population pairs can proceed simultaneously in each generation, as the selection is done without replacement.

A pseudo code for the deterministic crowding genetic algorithm is shown in figure 3.2.

```
DO WHILE (GEN. ≤ MAX. GENERATION)
   shuffle population
   i=1
   do while (i ≤ population size)
      choose two parents, p1 and p2, randomly,  with out replacement
      perform crossover and mutation to produce offspring c1 and c2
      evaluate fitness, f,  of parents and offspring

      IF [Distance(p1,c1) +Distance(p2,c2) ] ≤ [Distance(p1,c2) +Distance(p2,c1) ] Then
          IF (f(c1) > f(p1) replace p1 with c1
          IF (f(c2) > f(p2) replace p2 with c2
      ELSE
          IF (f(c2) > f(p1) replace p1 with c2
          IF (f(c1) > f(p2) replace p2 with c1
      i=i+2
   end do
END DO
```

**Figure 3.2 Deterministic Crowding GA pseudo code**

For comparison purposes, a flow chart of a standard (canonical genetic algorithm) is reproduced in figure 3.3.

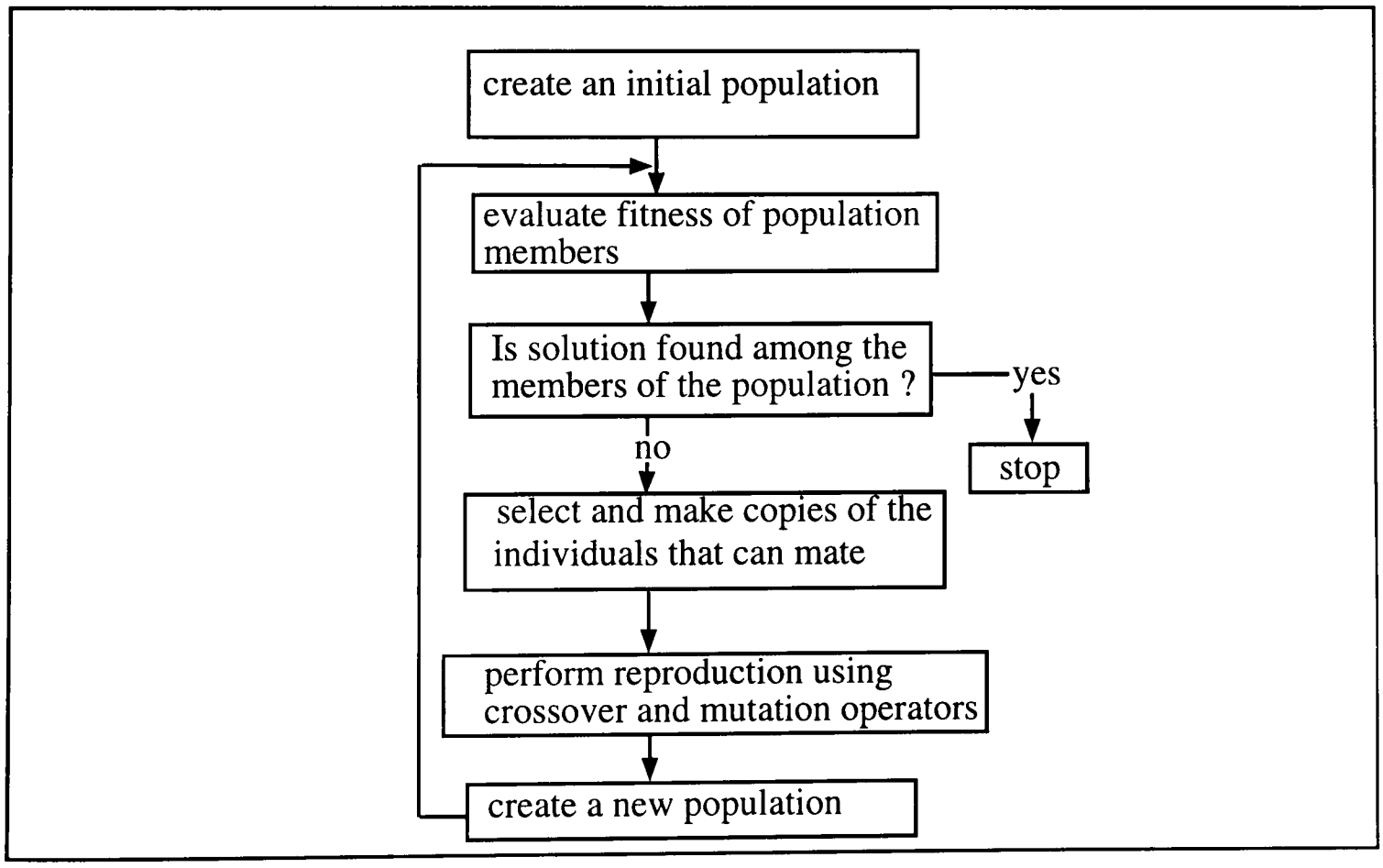



**Figure 3.3 A standard genetic algorithm flow chart**

The performance of the deterministic crowding GA model implemented in this work is compared with that of the canonical GA model whose general flow chart is given in figure 3.3.

## 3.4 Control Parameters of the Deterministic Crowding GA

One of the advantages of the deterministic crowding genetic algorithm is that the numerous standard GA control parameters such as crossover rate, mutation rate, selection method, fitness scaling, generation gap and elitism are inherently incorporated in the DCGA model. There is no need for the usually extensive experimentation with the GA before final choice of optimal control parameters is made. The major components of the DCGA algorithm that can be explicitly varied are the replacement-selection strategy, crossover method, population size and number of generations of the GA run.

### 3.4.1 Crossover

Crossover is performed on the whole population, which is randomly shuffled at the beginning of each generation. It largely determines the behaviour of the DCGA, as without crossover (assuming no mutation, as is the case with the main DCGA model), the algorithm simply advances both parents to the next generation, and hence the population never changes. However, the combination of crossover and the replacement-selection processes produces a powerful interaction among the various problem sub-structures (niches), that advances the resulting selected structures towards the optimal point.

### 3.4.2 Mutation

Mutation is considered as a local search operator in the DCGA model. Although the possibility of including it in the GA run is provided for in the general modelling framework, it must be applied with a small probability, since too much mutation degrades the algorithm performance.

### 3.4.3 Population size

The population size for the DCGA must provide a suitable proportion of good building blocks in order to provide an effective search process. Very small populations tend to become dominated by a single individual and usually lack the diversity required for proper adaptation. For more complicated, high dimensionality and difficult problems, large populations are required, however large populations are accompanied by an almost proportional increase in computation times. Large populations also tend to prevent the tendency for a good individual to propagate its features. Population sizes should be some function of the problem size.

### 3.4.4 Elitism

Elitism is always useful, especially for difficult optimisation problems. In the DCGA, because offspring replace parents only when they are of superior performance, the possibility of losing the best structure in the population is non-existent, and elitism is therefore inherent in the DCGA.

## 3.5 DCGA performance on some general optimisation problems

The performance of the deterministic crowding GA was initially investigated on the optimisation of two simple functions, with the results presented in the following section.

### 3.4.1 Example 1 - *Himmelblau's* Function

This is an example of a two dimensional multimodal function with four peaks of equal values, which has been used in testing the performance of a number of linear and non-linear programming techniques [Himmelblau]. The function is :

$$F(x,y)=\left[2186-\left(x^2+y-11\right)^2-\left(x+y^2-7\right)^2\right]/2186 \qquad (3\text{-}1)$$

The function is to be maximised within the limits, $-6 \le x \le 6$ and $-6 \le y \le 6$. A DCGA run found the [x, y] values that result in the optimal peaks. These values [3, 2], [3.584, -1.848], [-2.805, 3.131] and [-3.779, -3.283] were all well distributed in the final generation, as shown in figure 3.4 which demonstrates the capability of the DCGA in solving a multimodal optimisation problem. In the figure, the horizontal axis represents the population count, while the vertical axis, is the y component of the function. The x component of the objective function is not plotted, for purposes of enhancing the clarity of the population distribution diagram. A standard GA run on the problem only located the peak at [3, 2].

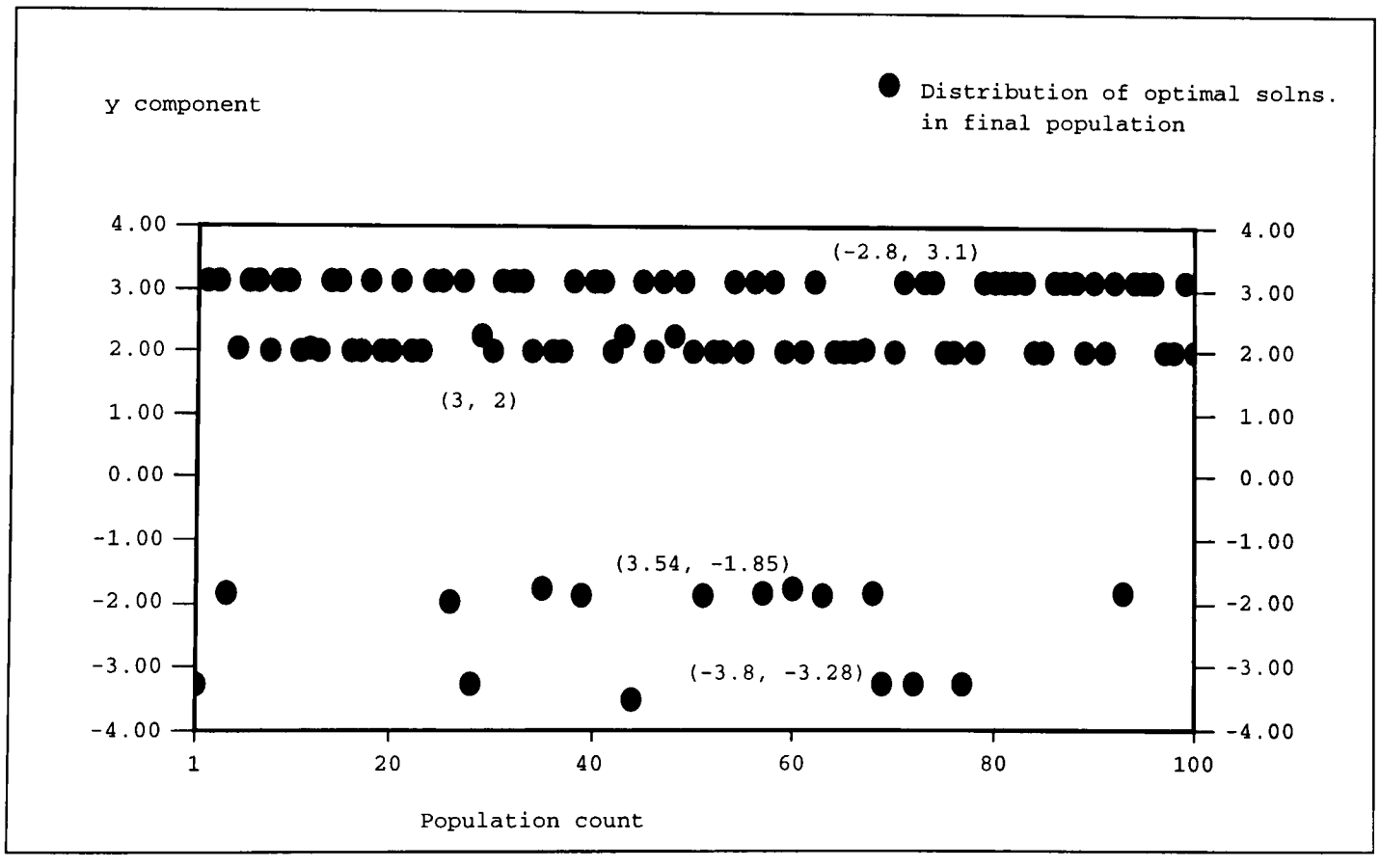



**Figure 3.4 Distribution of population of solutions in the final GA generation (for Himmelblau's function)**

### 3.4.2 Example 2 - *Wood's* Function

This four dimensional function [Schwefel 1981] is to be minimised within the variable limits $-6 \leq x_i \leq 6$, for i=1 to 4. The function has several local minima that can cause many conventional optimisation techniques to prematurely converge, i.e. get trapped at a local minima. The function is:

$$\begin{aligned} F(x_1,x_2,x_3,x_4) = 100\left[\left(x_1 - x_2^2\right)^2 + \left(x_2 - 1\right)^2\right] + \\ 90\left[\left(x_3 - x_4^2\right)^2 + \left(x_4 - 1\right)^2\right] + \\ 10.1\left[\left(x_1 - 1\right)^2 + \left(x_3 - 1\right)^2\right] + \\ 19.8\left[\left(x_1 - 1\right) + \left(x_3 - 1\right)\right] \end{aligned} \tag{3-2}$$

The decision parameters values found from the DCGA run were [0.992, 0.992, 0.992, 0.992] with $F(x_1, x_2, x_3, x_4) = 0.0141$. The expected optimal values of the decision variables are [1, 1, 1, 1], with $F(x_1, x_2, x_3, x_4) = 0$. The values of 0.992 obtained in the experiment are due to the resolution accuracy (number of bits) used for the problem representation. Higher accuracy could be obtained by using more bits in the binary string representation. The results from examples 1 and 2 demonstrate that the DCGA algorithm can be used for both uni-modal and multi-modal function optimisation.

## 3.6 Conclusion

The Deterministic crowding genetic algorithm has a capability of solving both uni-modal and multi-modal optimisation problems and in this work, the algorithm will be evaluated on a number of economic dispatch and thermal scheduling problems. Through a series of design modifications, empirical tests and analysis, the performance of the DCGA model will be compared with that of a canonical (standard) GA implementation on a number of practical power system scheduling problems.

# Chapter 4. Economic Dispatch With Genetic Algorithms

## 4.1 Introduction

Economic dispatch of generating units in a power system is concerned with the allocation of the load demand among the on-line (synchronised) generating units in order to minimise fuel costs, while satisfying the various unit and power system network constraints. Economic dispatch is a sub-problem of both the unit commitment and hydrothermal coordination problems, and assumes that the decision to commit any unit to generation has been made prior to performing economic dispatch. The economic dispatch problem has been the subject of intensive research for a number of years, and a summary of the solution methods is presented in [Wood and Wollenberg], [Sterling, 1978], [Happ, 1977]. In the thermal scheduling problem, an appropriate technique for solving the economic dispatch sub-problem must be used. In this chapter, the possibility of using the genetic algorithm method for solving the dispatch sub-problem is investigated by comparing its performance with other conventional methods.

## 4.2 Economic Dispatch Problem

The economic dispatch problem considered in this work is looked at from the perspective of a sub-problem of unit commitment. It is the dispatch necessary to facilitate generation scheduling and is thus considered to take place at hourly intervals. To solve the standard economic dispatch problem, consider the operation of a power system with $N$ synchronised units, where the $i^{th}$ unit is loaded to $P_i$ MW, to satisfy a total load demand $P_D$ including total transmission losses $P_L$. Let the fuel input-power output cost function of each unit be represented by a function $F_i$. The main objective of optimal economic dispatch is to minimise the total fuel cost:

$$Min \sum_{i=1}^{N} F_i(P_i) \tag{4-1}$$

subject to the power balance and unit loading limits :

$$\begin{aligned} &\sum_{i=1}^{N} P_i - (P_D + P_L) = 0 \\ &P_i^{min} \le P_i \le P_i^{max} \quad i = 1,2,..,N \end{aligned} \tag{4-2}$$

where $i$ is the unit index, $P_i^{min(max)}$ are the unit minimum (maximum) generation limits. Other constraints usually considered in the dispatch problem include:

1. rate of change of generator output limitations (unit ramping rates),
2. collective import/export limitations from groups of generators,
3. restrictions on contributions to reserve capacity and assignments for frequency regulation,
4. the dependence of power system losses on load flow pattern.

Most of these additional constraints are considered in the detailed on-line dispatch process and are handled by an optimal power flow program [Dommel] which also considers reactive power control in the network and is beyond the scope of this work.

## 4.3 Conventional Economic Dispatch Methods

Economic dispatch belongs to the class of non linear optimisation problems composed of a non linear objective function and a number of equality and inequality constraints. It is not, in general, straight forward to compute, by classical calculus, the location of the optimum loading points for all the system units, that would minimise the system operating costs, when problem constraints, such unit minimum and maximum loading limits are considered. A number of linear and non linear programming techniques have been proposed for the solution of the dispatch problem. Happ, [Happ, 1977] provides a comprehensive literature survey on economic dispatch solution techniques.

### 4.3.1 Merit Order Dispatch

This is the simplest dispatch method and it relies on the availability of linear, or piece-wise linear, cost functions. The committed generators are indexed in order of increasing incremental cost and are initialised at their minimum power output levels. The generators are then considered for loading to their maximum capacities, according to their rank in the priority list, until the load demand is satisfied. The system incremental cost is then determined by the partly loaded generator in the system. This method is reliable and fast but the results are only accurate for linear cost functions, which ignore system losses.

### 4.3.2 Linear and Quadratic Programming

If the dispatch problem is reduced to a form where only upper and lower bounds on the unit loading and the load balance condition is considered, together with a set of linear constraints, linear programming techniques can be applied to its solution by using a linear approximation of the non linear objective function at some feasible or near feasible points.

The basic simplex linear programming formulation [Dantzig] linearises the cost function over the various operating points. The main advantages of the linear programming methods are: reliability, speed of solution and freedom from convergence difficulties. The sparse dual

revised simplex formulation [Irving and Sterling], has proved to be reliable for solving large scale problems. Quadratic programming solution methods [Nicholson and Sterling], can be used to solve the dispatch problem, if the cost functions are represented by quadratic functions, while the constraint functions remain linear.

### 4.3.3 Equal Incremental Cost

The basic principle of the equal incremental cost method is that for continuous generator power output-cost functions, the most economic load division between the generators occurs they are operated at equal incremental costs [Kirchmayer].

$$\frac{\partial F(P_1)}{\partial P_1} = \frac{\partial F(P_2)}{\partial P_2} = \cdots \frac{\partial F(P_N)}{\partial P_N} = \lambda \tag{4-3}$$

Equation 4-3 shows that for optimality, individual units should share the total load such that their incremental costs are equal to $\lambda$, the system optimal value of the incremental cost at the operating point.

When the cost function is quadratic, the incremental, costs are linear and the co-ordination equations can be solved by a simple formulae, that expresses the incremental cost as a function of the power output/cost function coefficients, thus,

$$\lambda = \frac{P_D + \sum_{i=1}^{N} \frac{b_i}{2c_i}}{\sum_{i=1}^{N} \frac{1}{2c_i}} \tag{4-4}$$

and the individual unit loadings are obtained by,

$$P_i = \frac{\lambda - b_i}{2c_i} \tag{4-5}$$

where $P_D$ is the total load demand, b and c are coefficients in the cost function, F,

$$F_i = a_i + b_i P_i + c_i P_i^2 \qquad i \in R_s \tag{4-6}$$

This formulation in equation 4-4 however, assumes there are no violations on the unit loading limits. One of the more widely used equal incremental cost techniques is based on an iterative procedure, termed *Lambda* ($\lambda$) *Iteration* where the value of the system incremental cost, $\lambda$, is altered continuously until the load demand is satisfied to within a specified tolerance margin. Extrapolation using the last two successive values of lambda is used to esti-

mate the next value, and the process repeated until convergence is achieved. The new estimate of $\lambda$ is given by:

$$\lambda^{k+1} = \lambda^{k} + \frac{P_D - \left(\sum_{i=1}^{N} P_i\right)^{k}}{\left(\sum_{i=1}^{N} P_i\right)^{k-1} - \left(\sum_{i=1}^{N} P_i\right)^{k}} \bullet \left(\lambda^{k} - \lambda^{k-1}\right) \qquad (4\text{-}7)$$

This value of $\lambda$ is then used in equation 4-3 to determine the unit power output and hence total power generation. If a unit's power limit is violated in the course of solution, its output is set to the violated limit.

Gradient techniques [Wood and Wollenberg] based on the equal system incremental cost have also been applied to solve the economic dispatch problem. In a number of privatised electricity utilities, such as that of England and Wales, the thermal scheduling of generation is based on a merit order rating. Both the Lambda - Iteration and merit order dispatch methods have been implemented in this work and used both for evaluating the performance of the genetic algorithm dispatch technique and as a sub-component of the thermal and hydrothermal scheduling techniques.

## 4.4 Economic Dispatch Using Genetic Algorithms

Before using any of the GA models for economic dispatch, the problem must be transformed to a suitable format that allows the application of the various genetic algorithm operators.

### 4.4.1 Economic dispatch problem encoding

For the economic dispatch problem, the binary problem representation and encoding is described in chapter 2 by equations 2-4 and 2-5.

### 4.4.2 Economic dispatch fitness function

The main objective of economic dispatch is to minimise fuel costs while satisfying the physical limitations of units as well as those on the power system. The system constraint to be satisfied is that of matching the load demand and reserve requirements with power generation. The unit minimum and maximum loading limits are taken care of in the problem encoding, and the only other constraints to be considered are the unit prohibited operating zones. A penalty function approach [Orero and Irving, 1996], [Richardson et. al.], [Siedlecki and Sklanky], [Smith and Tate] is used to handle the explicit constraints. The penalty terms are incorporated in the fitness function, and are set to reduce the fitness of the string according to the magnitude of the violation. Because it is usually very difficult to determine the penalty function coefficients, an infeasible solution is awarded a fitness worse than the

weakest feasible string. Since two infeasible strings are not treated equally, the string further away from the feasibility boundary, is more heavily penalised. Because these strings are not discarded, they are still able to contribute to the search process. The general economic dispatch problem objective function is :

$$Min \sum_{i=1}^{N} F_i(P_i) + \Psi\left[\sum_{i=1}^{N} P_i - (P_D + P_L)\right] + \Phi\left[\sum_{k=1}^{n_i} P_i(zone_k)\right] \qquad (4\text{-}8)$$

where $\Psi$ is the penalty function for not satisfying load demand, and $\Phi$ represents the penalty function for a unit loading falling within a prohibited operating zone, in cases where the unit prohibited operating range is to be taken into account. Quadratic penalty functions are used for both $\Psi$ and $\Phi$, and are made proportional to the distance from the feasibility boundary. The constraints must be satisfied within a set tolerance $\varepsilon$, e.g. it might be decided that the load demand be satisfied within 0.1 MW. The GA works by maximising a single variable, the *fitness* function, and hence the dispatch minimisation function must be transformed into a maximisation.

## 4.5 Simulations and Results

The merit order, equal incremental cost, standard GA and deterministic crowding GA (DCGA) economic dispatch methods were applied to various test systems. The effect of various problem difficulties such as:

- effect of different cost functions,
- dispatch of units with prohibited operating zones,
- dispatch of units with valve point loading,
- effect of inclusion of transmission losses in the dispatch,

were also investigated. Some of the test results are presented in the following sections.

### 4.5.1 Economic dispatch in a 54 unit test system

One of the important GA parameters is the population size. Table 4.1 shows the results of GA trials using two different population sizes for a power system consisting of 54 generators. For both the DCGA and standard GA models an increase in population size from 100 to 500 provides only a marginal improvement in the value of the final solution obtained, although this incurs an almost proportional increase in computation time, since all the tests were carried out for the same number of generations. A population size of 100 seems to be appropriate for this test system. The standard GA and DCGA provide solutions which are within 0.2% and 0.1% of the Lambda-Iteration and merit order dispatch solutions respectively. The sub optimal results provided by the genetic algorithms can be attributed to the

premature convergence of the algorithms, as an increase in discretization (string length) did not provide any improvements in solution quality.

**Table 4.1 Economic dispatch costs ( 54 unit test system)**

| Trial | Dispatch Costs | | | |
|---|---|---|---|---|
| | Standard GA | | DCGA | |
| | pop. size 100 | pop. size 500 | pop. size 100 | pop. size 500 |
| 1 | 145,704 | 143,788 | 145,814 | 143,371 |
| 2 | 148,457 | 143,450 | 146,050 | 143,325 |
| 3 | 147,095 | 143,534 | 146,362 | 143,404 |
| 4 | 146,696 | 143,694 | 143,757 | 143,308 |
| 5 | 147,125 | 144,205 | 144,445 | 143,282 |
| 6 | 147,345 | 144,207 | 146,051 | 144,440 |
| 7 | 144,516 | 144,096 | 146,343 | 143,457 |
| 8 | 145,706 | 146,002 | 145,840 | 143,416 |
| 9 | 145,716 | 144,025 | 143,371 | 143,570 |
| 10 | 145,717 | 144,562 | 146,773 | 143,480 |
| Best | **144, 516** | **143, 450** | **143, 371** | **143,282** |
| Percentage above Lambda - Iteration | 0.18 | 0.14 | 0.081 | 0.019 |
| GA parameters | pcross=1.0, Pmut=0.001, gen. =1000, Trunc. fitness scaling , 1 std. dev., elitism= 10% | | pcross=1.0, Pmut=0.001, gen. =1000 | |
| Merit order solution =**143,255.9**, Equal Lambda - Iteration solution =**143,255.2** | | | | |

### 4.5.2 Economic dispatch results for other test systems

The GA dispatch methods were applied to a number of test systems and the results for some of the test systems are presented in table 4.2.

**Table 4.2 A summary of economic dispatch solutions ( other test systems)**

| Test system | Standard GA | | DCGA | | Merit order | | Lambda - Iteration solution Cost |
|---|---|---|---|---|---|---|---|
| | Cost | % above Lambda - Iteration | Cost | % above Lambda - Iteration | Cost | % above Lambda - Iteration | |
| 3 units | 8194.4 | 0 | 8194.4 | 0 | 8,227.9 | 0.41 | 8194.4 |
| 8 units | 2432.4. | 0.054 | 2431.1 | 0.107 | 2,437.5 | 0.32 | 2,429.8 |
| 28 units | 100, 389 | 0.061 | 100,357 | 0.029 | 100, 688 | 0.36 | 100, 328 |
| 75 units | 210, 735 | 0.30 | 210, 617 | 0.24 | 210, 106 | 0 | 210, 106 |

The standard GA and the Deterministic Crowding GA methods both provide acceptable solutions to the standard economic dispatch problems across a number of problem sizes, with the DCGA model having a slight edge over the standard GA method. They do not appear to have any significant advantage over the Lambda - Iteration method for the simple dispatch problem formulation, when considering solution accuracy and computation time. They however provide better solutions than the merit order dispatch solution method, especially for small test systems. The advantage of the GA methods is that they can be used with any functional representation of the cost function and are thus suitable for the more difficult economic dispatch cases such as those considering valve point loading, prohibited zone loading, effect of losses or cases that treat cost functions of polynomials of order greater than two.

### 4.5.3 Effect of unit output resolution on the performance of the GA

In any optimisation process, in the search for an optimal solution, a step length, (or problem resolution for GA), must be chosen. If the step lengths are too small, the search takes an unnecessarily large number of iterations (or generations). In GA terms, the finer the resolution, the longer the string lengths, which results in longer solution times. If step lengths are too large, the optimum solution can only be crudely approached and the search can get stuck at a local optimum, especially if the route to the optimum passes through a narrow valley. Thus step length, or resolution control, is very closely related to the convergence behaviour of the optimisation model. A multiple resolution GA search mechanism can be implemented, where the problem resolution is increased in steps. This can provide a much faster search mechanism. The population based nature of the GA helps the optimisation process in avoiding convergence to local optima as the search moves from course grained one to fine grained. The slight difficulty with a multi step GA is the mismatch in "converting" decimal quantities back to the integer binary variables for the GA manipulation, which involves some approximation to the nearest integer bit. The speed improvement achieved by the multi step search far outweighs the loss of accuracy in conversion, especially for large scale problems. The effect of problem resolution on solution quality is demonstrated by the sample results in table 4.3, for both a small and a large test system.

**Table 4.3 Effect of problem resolution on economic dispatch GA**

| Trial | Small test system (3 units) | | Large scale system (75 units) | |
|---|---|---|---|---|
| | Total string length | DCGA operation Cost | Total string length | DCGA operation Cost |
| 1 | 8 | 8203 | 225 | 210, 617 |
| 2 | 16 | 8194.5 | 300 | 213,471 |
| 3 | 26 | 8194.4 | 375 | 213,913 |
| 4 | 36 | 8194.4 | 450 | 214, 349 |
| Best solution | **8194.4** | | **210, 617** | |

The resolution, and hence string length is very crucial to the accuracy within which solutions can be obtained. It is important to determine the proper levels of resolution accuracy depending on the problem at hand. For example in the 3 unit economic dispatch problem, using a resolution of 0.01 MW in unit output instead of 1 MW increases the total number of possibilities in the binary problem search space from $2^{26}$ to $2^{36}$, giving the GA a much harder search task. However, the added accuracy of solution may be of little practical value as it might not be feasible to control unit generation to within an accuracy of 0.01 MW in practise.

### 4.5.4 Initial population seeding with a merit order dispatch solution

Inclusion of domain knowledge in an initial GA population is one of the ways in which GA performance can be improved, especially for large scale or difficult problems which would otherwise require long computation times before converging to the required solution. For the economic dispatch problem, the results of the merit order dispatch method were included in the GA initial population. For each of the unit output resolutions in table 4.4, 10 independent GA runs were carried out and the average value obtained is the one shown in the table.

**Table 4.4 Dispatch with merit order initial population seeding ( 54 units)**

| decimal resolution in unit power output (MW) | Total string length (binary bits) | Average standard GA production Cost (with seeding) | Percentage improvement over best GA solution without seeding |
|---|---|---|---|
| 25 | 164 | 143,601 | 0.63 |
| 1 | 398 | 143,383 | 0.78 |
| 0.1 | 579 | 143,303 | 0.84 |
| 0.01 | 752 | 143,255.8 | 0.87 |
| GA parameters | popsize=100, pcross=1.0, Pmut=0.001, gen. =1000, Trunc. fitness scaling , 1 std. dev., elitism= 10% | | |
| merit order dispatch solution =**143,255.9,** Best standard GA solution (without seeding) = **144,516** | | | |

From table 4.4, it can be seen that the use of the merit order dispatch solution in the initial standard GA population improves the performance of the GA method as expected, but does not always result in a performance better than the merit order dispatch solution. In fact in a number of cases, such as those with unit output resolutions of 25, 1 and 0.1 MW, the GA performance is worse than that of the merit order solution. This can be attributed to the fact that the merit order dispatch loading levels are given as a decimal output and for them to be used in a binary representation, the decimal output (phenotype) must be converted to binary representation, usually resulting in some loss of accuracy. The finer the decimal resolution,

the less is the loss of accuracy in conversion form decimal to binary representation, as demonstrated by the performance for a resolution of 0.01 MW, which gives the closest conversion to the merit order solution. However, a finer resolution results in a proportional increase in string length making the search space for the GA much larger. For this test system, the initial population seeding helps the standard GA search, but to gain more from initial population seeding, appropriate resolution levels must be used.

### 4.5.5 Improving the performance of the DCGA using a local hill climber

A conventional local search optimisation method such as that of [Hooke and Jeeves], [Mead and Nelder], [Powell], can be incorporated in the deterministic crowding GA to enhance its performance, by concentrating the search in the local neighbourhoods. An important requirement of the the local hill climber is that it should not require the evaluation of derivatives. Although gradient and second derivative methods generally converge faster than direct search methods, it can be very laborious or impossible to provide the analytical functions required for the application of these derivative based techniques for moderately sized or large problems.

The local search algorithm must be applied in an innovative manner in order to provide a good balance between the global and local search mechanisms, while taking into account the GA speed limitations. When used with the genetic algorithm, once the fitness function has been evaluated, the hill climber can be applied in one of the following ways:

1. to the final best solution at the end of a GA run,
2. to all population members, in the final generation of the GA,
3. to all population members during each generation,
4. to the best individual in each generation,
5. to a certain percentage of the population within each generation.

The choice made must strike a balance between the computation speed and the degree of local convergence that is beneficial.

#### 4.5.5.1 Hooke and Jeeves Hill climber

Hooke and Jeeves algorithm is suitable for the optimisation of a multi variable unconstrained function and is based on the direct search method proposed in [Hooke and Jeeves]. This search technique does not require the evaluation of problem derivatives and can be used in the optimisation of a problem having any functional representation. It is one of the simplest direct search techniques, and hence if it can provide improvements to an optimisation method, then better improvements could possibly be obtained with other more advanced direct search techniques. It has previously been successfully used [Orero, 1988] in the estimation of autoregressive moving average (ARMA) load demand models used for

load forecasting. A flow chart of the Hooke and Jeeves search technique is shown in figure 4.1

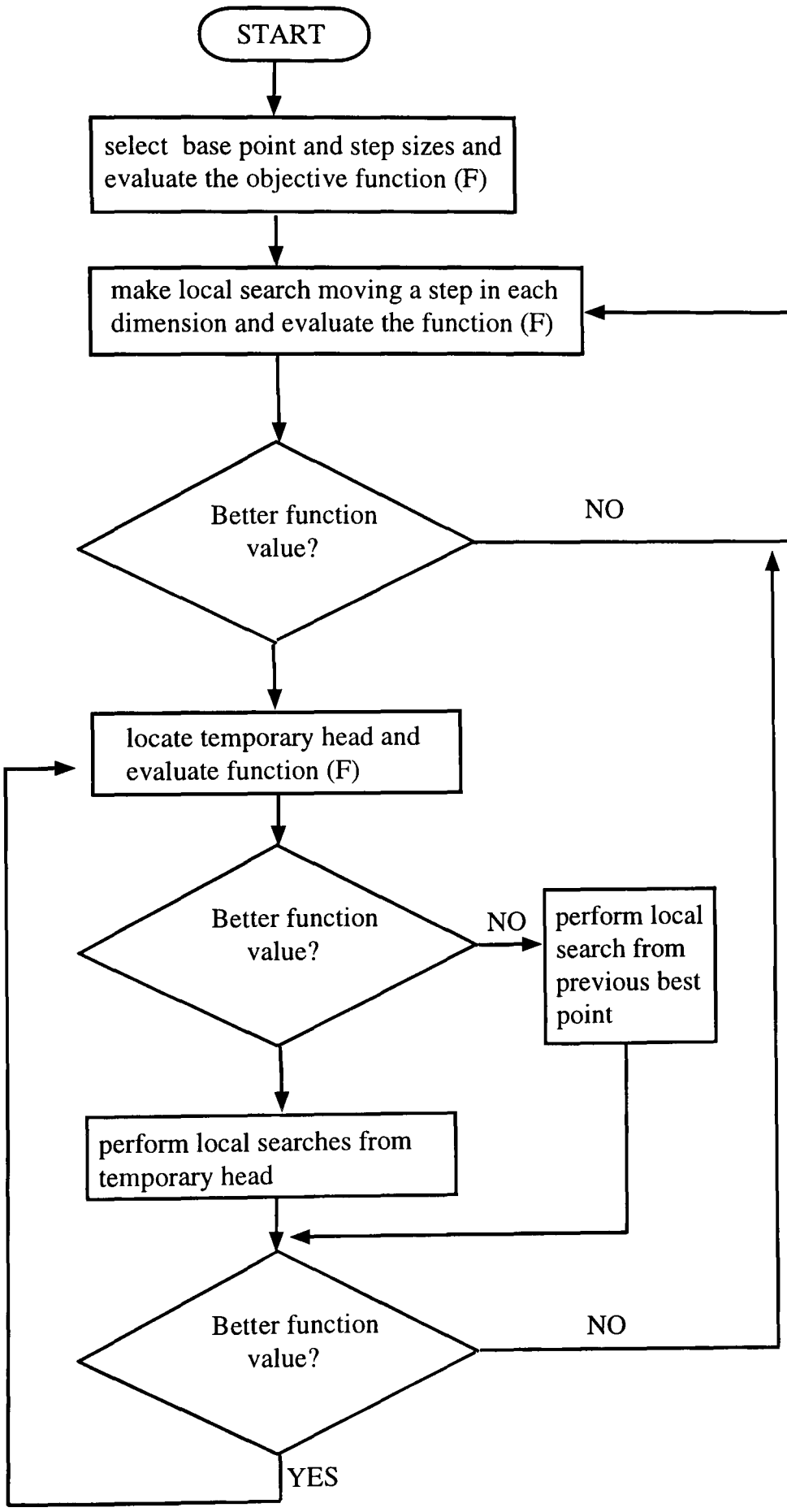



**Figure 4.1 Hooke and Jeeves local hill climber flow chart**

The Hooke and Jeeves hill climber was applied to all the population elements in the final generation of GA runs on two test systems. The best solution obtained in a set of experiments is shown in tables 4.5 and 4.6 respectively.

**Table 4.5 DCGA dispatch with hill climbing ( 54 unit test system )**

| Trial | Operation Cost | | |
|---|---|---|---|
| | DCGA - No Hill climbing | DCGA with Hill climbing | Percentage improvement |
| 1 | 143,371 | 143,326 | 0.031 |
| 2 | 143,325 | 143,300 | 0.017 |
| 3 | 143,404 | 143,380 | 0.017 |
| 4 | 143,308 | 143,307 | 0 |
| 5 | 143,282 | 143,272 | 0.007 |
| 6 | 144,440 | 144,428 | 0.008 |
| 7 | 143,457 | 143,412 | 0.031 |
| 8 | 143,416 | 143,415 | 0 |
| 9 | 143,570 | 143,563 | 0.005 |
| 10 | 143,480 | 143,480 | 0 |
| Mean percentage improvement | **0.012** | | |

**Table 4.6 DCGA dispatch with hill climbing ( 75 unit test system )**

| Trial | Operation Cost | | |
|---|---|---|---|
| | DCGA - No Hill climbing | DCGA with Hill climbing | Percentage improvement |
| 1 | 210,617 | 210,605 | 0.006 |
| 2 | 215,718 | 215,717 | 0 |
| 3 | 213,610 | 213,610 | 0 |
| 4 | 218,525 | 218,521 | 0.002 |
| 5 | 214,065 | 214,064 | 0 |
| 6 | 215,990 | 215,980 | 0.005 |
| 7 | 217,170 | 217,167 | 0.001 |
| 8 | 213,471 | 213,295 | 0.083 |
| 9 | 213,913 | 213,907 | 0.003 |
| 10 | 212,876 | 212,870 | 0.003 |
| Mean percentage improvement | **0.010** | | |

From the results presented in the tables, the hill climber does not appear to significantly improve the final results of the DCGA run. This could possibly be attributed to the problem dimensionality, which may be too high for this hill climbing technique. In order to avoid the hill climber becoming stuck at local optima, more powerful local search techniques, such as Tabu Search [Glover], [Kido et. al.] which uses an adaptive memory concept to recognise local optima, can be used to enhance the GA search mechanism.

## 4.6 Economic Dispatch of Units With Prohibited Operating Zones

The constrained economic dispatch problem in which some of the units have prohibited operating regions is a more difficult dispatch problem encountered in the operation of many power systems. In practice, the whole of the unit operating range is not always available for load allocation due to physical operation limitations. Units can have prohibited operation regions due to faults in the machines themselves or the associated auxiliaries, such as boilers, feed pumps etc. Such faults usually lead to instabilities over certain ranges of the unit loading, rendering them unable to carry that load for any appreciable time. A unit with prohibited operating zones has a discontinuous input - output power generation characteristic shown in figure 4.2.

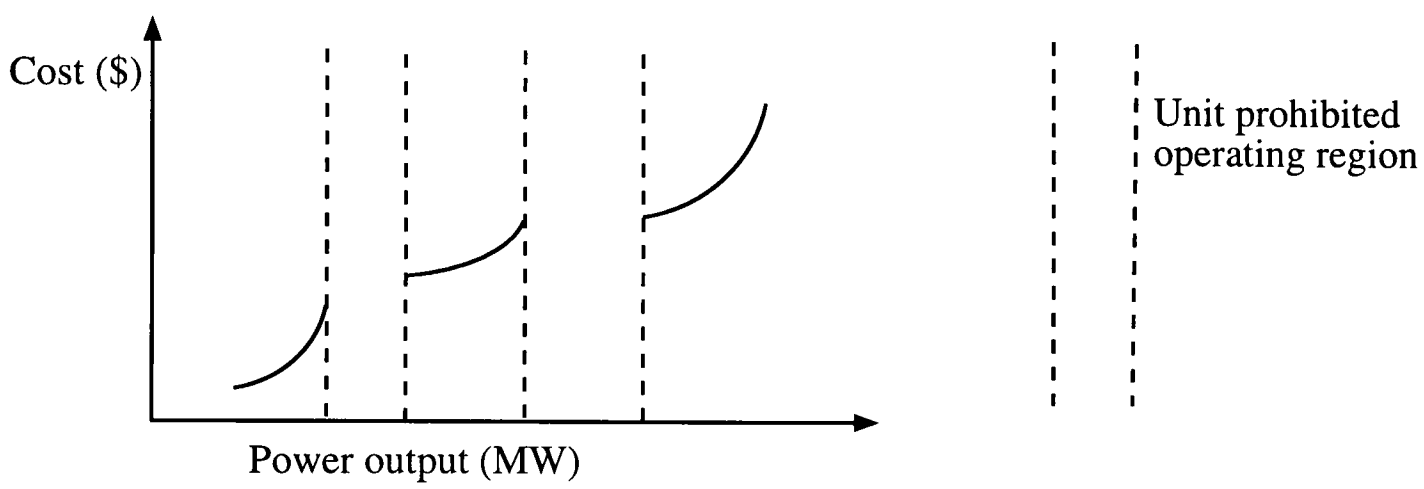



**Figure 4.2 Input / output characteristic for units with prohibited zone**

The discontinuous nature of the problem objective function prevents the use of conventional economic dispatch techniques such as the Lambda - Iteration or gradient based methods. For units with prohibited operating zones, further to the usual dispatch constraints described in equations 4.1 and 4.2, there are additional constraints on the unit operating range

$$\begin{aligned} & P_i^{min} \le P_i \le P_{i,1}^{L} \quad or \\ & P_{i,k-1}^{U} \le P_i \le P_{i,k}^{L} \quad k = 2,..,n_i, \; or \\ & P_{i,n_i}^{U} \le P_i \le P_i^{max} \end{aligned} \tag{4-9}$$

where $n_i$ - number of prohibited zones for unit $i$,

$k$ - index of prohibited zones of a unit,

$p_{i,k}^{L,U}$ - lower / upper bounds of the $k^{th}$ prohibited zones of unit $i$.

### 4.6.1 Results

10 independent runs were carried out for each of the two different GA models and their performance compared. The comparison was based on the same number of function evaluations (population size x number of generations) and the same final resolution in control parameter

variables. The number of function evaluations in each trial was $10^6$. Each pair of solutions is evaluated based on the same initial population. The test problem is based on a 15 unit practical power system [Lee and Breiphol], with 4 of the units having up to three prohibited operating zones. The test system data is given in tables 4.7 and 4.8.

**Table 4.7 Unit characteristics for constrained economic dispatch problem**

| $F(P_i)=\alpha+\beta(P_i)+\gamma(P_i)^2$ (load demand 2650 MW) | | | | | |
|---|---|---|---|---|---|
| Unit | $\alpha$ ($/H) | $\beta$ ($/MWH) | $\gamma$ ($/MWH$^2$) | $P_{min}$ | $P_{max}$ |
| 1 | 671.03 | 10.07 | 0.000299 | 150 | 455 |
| 2 | 574.54 | 10.22 | 0.000183 | 150 | 455 |
| 3 | 374.59 | 8.80 | 0.001126 | 20 | 130 |
| 4 | 374.59 | 8.80 | 0.001126 | 20 | 130 |
| 5 | 461.37 | 10.40 | 0.000205 | 105 | 470 |
| 6 | 630.14 | 10.10 | 0.000301 | 135 | 460 |
| 7 | 548.20 | 9.87 | 0.000364 | 135 | 465 |
| 8 | 227.09 | 11.21 | 0.000338 | 60 | 300 |
| 9 | 173.72 | 11.21 | 0.000807 | 25 | 162 |
| 10 | 175.95 | 10.72 | 0.001203 | 20 | 160 |
| 11 | 186.86 | 10.21 | 0.003586 | 20 | 80 |
| 12 | 230.27 | 9.90 | 0.005513 | 20 | 80 |
| 13 | 225.28 | 13.12 | 0.000371 | 25 | 85 |
| 14 | 309.03 | 12.12 | 0.001929 | 15 | 55 |
| 15 | 323.79 | 12.41 | 0.004447 | 15 | 55 |

**Table 4.8 Unit prohibited zones**

| unit | Prohibited regions | | |
|---|---|---|---|
| | zone 1 | zone 2 | zone 3 |
| 2 | [185, 225] | [305, 335] | [240, 450] |
| 5 | [180, 200] | [260, 335] | [390, 420] |
| 6 | [230, 255] | [365, 395] | [430, 455] |
| 12 | [30, 55] | [65, 75] | |

The GA parameter settings are given in table 4.9. The initial population is generated randomly. Uniform crossover has been used in all the tests and mutation has also been applied.

**Table 4.9 GA parameter settings for prohibited zone dispatch**

| GA control variable | Parameter settings | |
|---|---|---|
| | Standard GA | Deterministic Crowding GA |
| population size | 200 | 200 |
| selection method | stochastic remainder | implicit in parent replacement strategy |
| fitness scaling | truncation, 1 standard deviation | none |
| crossover | uniform, rate 1.0 | uniform, rate 1.0 |
| mutation | rate 0.005 | rate 0.005 (optional) |
| elitism | 10% | none |
| function evaluations | $1x10^6$ | $1x10^6$ |

The results of the dispatch solutions for the two GA models are shown in table 4.10. The cost values shown in the table are those of the best solution obtained at the end of a GA trial.

**Table 4.10 GA dispatch results for system with prohibited operating zones**

| Dispatch solution cost | | | |
|---|---|---|---|
| Trial | Deterministic crowding GA | Standard GA | |
| | | multiple resolution [10 MW, 1 MW] | Single resolution [1 MW] |
| 1 | 32,532 | 32,536 | 32,581 |
| 2 | 32,535 | 32,532 | 32,527 |
| 3 | 32,552 | 32,528 | 32,576 |
| 4 | 32,561 | 32,563 | 32,670 |
| 5 | 32,520 | 32,539 | 32,579 |
| 6 | 32,543 | 32,550 | 32,544 |
| 7 | 32,556 | 32,536 | 32,544 |
| 8 | 32,514 | 32,523 | 32,528 |
| 9 | 32,518 | 32,535 | 32,530 |
| 10 | 32,520 | 32,529 | 32,517 |
| Best | **32,514** | **32,523** | **32,517** |
| Mean | 32,535.1 | 32,537.1 | 32,559.6 |
| SD | 16.3 | 11.4 | 43.0 |

In carrying out GA experiments, it is very important to make the comparisons under similar conditions. This is borne out for example, by looking at trial 4, where all the methods perform quite poorly. The poor performance can be attributed to the initial random population. Different initial populations result in different final solutions and hence it is important to base the comparisons of GA performance on a number of trials.

From table 4.10 it can be seen that the crowding GA model performs better than the standard GA. With enhancements such as multiple resolution search mechanisms added to the standard GA, the performance improves and is almost as good as the crowding method. The point to be noted however, is that the crowding model has been applied in its basic form, without any modifications, as opposed to the standard GA which has had features such as scaling and elitism added to it. Table 4.11 compares the unit loading levels for the lambda -iteration ($\lambda$) dispatch method and a sample result obtained from the GA methods.

**Table 4.11 Sample unit power output levels from dispatch programs**

| Unit | power output (MW) | | |
|---|---|---|---|
| | Lambda - Iteration | Standard GA | Deterministic Crowding GA |
| 1 | 455 | 406.1 | 451.4 |
| 2 | 455 | 453.8 | 455 |
| 3 | 130 | 130 | 130 |
| 4 | 130 | 130 | 129.1 |
| 5 | 295.3* | 355 | 337.1 |
| 6 | 460 | 456.8 | 429.5 |
| 7 | 465 | 459.8 | 464.4 |
| 8 | 60 | 60 | 60 |
| 9 | 25 | 26.6 | 26.6 |
| 10 | 20 | 21.6 | 27.1 |
| 11 | 43.4 | 36.2 | 25.7 |
| 12 | 56.3 | 59 | 59 |
| 13 | 25 | 25 | 25 |
| 14 | 15 | 15 | 15 |
| 15 | 15 | 15 | 15 |
| production cost | 32,503 | 32,515 | 32,517 |
| * unit loading in prohibited zone | | | |

The unit loadings in table 4.11 show why the classical Lambda - Iteration method is not capable of solving this constrained dispatch problem. For example, using the lambda - Iteration technique results in a solution that requires unit 5 to operate in one of the prohibited operation zones. The GA methods on the other hand, provide final optimal loadings which do not fall in any of the *illegal* zones. The advantage of the GA method is that other operation and system constraints such as transmission capacity limitations, regulating spinning reserve requirements, higher order cost functions, etc. can easily be incorporated into the problem formulation.

#### 4.6.1.1 Local hill climbing on Deterministic Crowding GA final solutions

To give a further basis for comparison, the Hooke and Jeeves optimisation technique was also used as a parallel local hill climber by running it alone using an initial population of starting points. Using a population of 500, (i.e. initial start from 500 different random starting solutions) gave the results shown in table 4.12. The results indicate that the algorithm prematurely converges to a local optimum before obtaining the expected optimal solution, even when initialised from 500 different initial staring points.

**Table 4.12 Hooke and Jeeves algorithm as a parallel hill climber (prohibited zones dispatch)**

| Trial | Operation Cost | |
|---|---|---|
| | Best initial solution in population | Final best solution obtained with parallel hill climbing |
| 1 | 75,830 | 33,122 |
| 2 | 34,775 | 33,534 |
| 3 | 139,099 | 33,069 |
| 4 | 123,682 | 75,809 |
| 5 | 86,005 | 33,163 |
| 6 | 78,201 | 32,813 |
| 7 | 76,514 | 33,124 |
| 8 | 98,793 | 75,587 |
| 9 | 78,367 | 33,255 |
| 10 | 114,296 | 33,035 |
| Best solution | **32,813** | |

The Hooke and Jeeves local search algorithm was applied to each of the population elements in the final generation of the DCGA in an effort to improve on the solutions already obtained from the GA run. Since the final population of the DCGA contains several near optimal solutions, applying a local hill climber to all of them instead of just the best should result in an improvement to all the possible solutions. The results are shown in table 4.13.

**Table 4.13 DCGA prohibited zones dispatch results with hill climbing**

| Trial | Operation Cost | | |
|---|---|---|---|
| | DCGA - No Hill climbing | DCGA with Hill climbing | Percentage improvement |
| 1 | 32,515 | 32,510 | 0.015 |
| 2 | 32,526 | 32,520 | 0.018 |
| 3 | 32,519 | 32,511 | 0.025 |
| 4 | 32,511 | 32,508 | 0.009 |
| 5 | 32,510 | 32,509 | 0.003 |
| 6 | 32,521 | 32,517 | 0.009 |
| 7 | 32,518 | 32,513 | 0.015 |
| 8 | 32,509 | 32,509 | 0 |
| 9 | 32,518 | 32,512 | 0.018 |
| 10 | 32,519 | 32,510 | 0.028 |
| Mean percentage improvement | 0.014 | | |

The results in table 4.13 show that the local hill climber provides a slight improvement on the final results of the DCGA run.

#### 4.6.1.2 Role of the mutation operator in a Deterministic Crowding GA

The theoretical foundations of DCGA [Mahfoud, 1995] does not account for the effect of mutation in an explicit manner and allows it to be included only as a local search operator. It has also been shown that using local parallel hill climbing on the converged GA solution can provide improved results. The effect of disabling the mutation operator was investigated for the deterministic crowding algorithm for the economic dispatch problem. The results are presented in table 4.14. The results in this table were obtained with a population size of 500.

**Table 4.14 Effect of mutation on the performance of the DCGA**

| Trial | Dispatch cost | |
|---|---|---|
| | With mutation | Without mutation |
| 1 | 32,513 | 32,515 |
| 2 | 32,518 | 32,526 |
| 3 | 32,520 | 32,519 |
| 4 | 32,517 | 32,511 |
| 5 | 32,517 | 32,510 |
| 6 | 32,534 | 32,521 |
| 7 | 32,525 | 32,518 |
| 8 | 32,516 | 32,509 |
| 9 | 32,524 | 32,518 |
| 10 | 32,525 | 32,519 |
| Best | 32,513 | 32,509 |
| Mean | 32,520.9 | 32,516.5 |
| SD | 5.84 | 5.23 |

A sample result showing the generation dispatch levels and the simulation conditions for a typical GA run is presented in tables 4.15 and 4.16.

**Table 4.15 Sample results for unit dispatch levels for DCGA with no mutation**

```
   Cross over performed
   uniform cross over
   NO mutation performed
   The function is being minimised
   genotype resolution  7 bits per parameter
   Random number seed =    746463
   population size =       500
   number of generations of run =      5000
   number of function variables (parameters) =        15
   pcross=  1.00   pmut=0.0000
Tolerance in satisfying demand  9.9999998E-03
unit          1 substring length          7Decimal resolution   2.401575
unit          2 substring length          7Decimal resolution   2.401575
unit          3 substring length          7Decimal resolution  0.8661417
unit          4 substring length          7Decimal resolution  0.8661417
unit          5 substring length          7Decimal resolution   2.874016
unit          6 substring length          7Decimal resolution   2.559055
unit          7 substring length          7Decimal resolution   2.598425
unit          8 substring length          7Decimal resolution   1.889764
unit          9 substring length          7Decimal resolution   1.078740
unit         10 substring length          7Decimal resolution   1.102362
unit         11 substring length          7Decimal resolution  0.4724410
unit         12 substring length          7Decimal resolution  0.4724410
unit         13 substring length          7Decimal resolution  0.4724410
unit         14 substring length          7Decimal resolution  0.3149606
unit         15 substring length          7Decimal resolution  0.3149606
Total chromosome length,no of parameters        105          15
Best member number  185  fitness    32508.94
```

**Table 4.16 Sample unit dispatch levels for DCGA with no mutation**

| Unit | power generation |
|---|---|
| 1 | 455.0000 |
| 2 | 450.1968 |
| 3 | 130.0000 |
| 4 | 130.0000 |
| 5 | 257.3228 |
| 6 | 460.0000 |
| 7 | 465.0000 |
| 8 | 60.00000 |
| 9 | 25.00000 |
| 10 | 20.00000 |
| 11 | 77.63779 |
| 12 | 64.88189 |
| 13 | 25.00000 |
| 14 | 15.00000 |
| 15 | 15.00000 |

One of the interesting observations which can be made regarding the DCGA model is its good performance, even without the application of mutation. The mutation operator appears to introduce some noise that slightly degrades the DCGA performance. This is contrast to experience with the standard GA which was found to perform poorly without the mutation operator. The effect of the mutation operator on the crowding model is not unexpected since the success of the crowding GA is mainly attributed to the crossover and the selection-replacement mechanism it employs.

#### 4.6.1.3 Comparing the performance of the genetic algorithm dispatch methods with other techniques

The performance of the two genetic algorithm models on the prohibited zone dispatch problem was compared with other solution techniques which have been proposed for this problem [Fan and McDonald], [Lee and Breiphol], and an optimal Dynamic Programming solution method with a discretization of 1 MW. These results are shown in table 4.17. The desired optimal solution, provided by the Lambda - Iteration technique, without considering unit prohibited operating zones indicates that one of the unit's loading, (unit 5), falls within a prohibited operating zone, (zone 2). Both the GA methods, provide solutions of almost the same quality as the best solution obtained by [Fan and McDonald] and [Lee and Breiphol], but the main advantage of the GA approach is its simplicity. The GA methods provide better solutions than the second set of solutions, R2, given by [Fan and McDonald], further confirming the robustness of the GA technique.

**Table 4.17 Unit power outputs and dispatch costs (prohibited zones dispatch)**

| unit | power output (MW) | | | | | | |
|---|---|---|---|---|---|---|---|
| | Lambda - Iteration | Standard GA | Deterministic Crowding GA | Fan and McDonald (R1) | Fan and McDonald (R2) | Lee and Breiphol | Dynamic Programming |
| 1 | 455 | 406.1 | 455 | 455 | 450 | 450 | 455 |
| 2 | 455 | 453.8 | 450.2 | 455 | 450 | 450 | 455 |
| 3 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| 4 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| 5 | 295.3* | 355 | 257.3 | 260 | 335 | 335 | 260 |
| 6 | 460 | 456.8 | 460 | 460 | 455 | 455 | 460 |
| 7 | 465 | 459.8 | 465 | 465 | 465 | 465 | 465 |
| 8 | 60 | 60 | 60 | 60 | 60 | 60 | 60 |
| 9 | 25 | 26.6 | 25 | 25 | 25 | 25 | 25 |
| 10 | 20 | 21.6 | 20 | 70 | 20 | 20 | 20 |
| 11 | 43.4 | 36.2 | 77.6 | 20 | 20 | 20 | 60 |
| 12 | 56.3 | 59 | 64.9 | 65 | 55 | 55 | 75 |
| 13 | 25 | 25 | 25 | 25 | 25 | 25 | 25 |
| 14 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| Cost | 32,503 | 32,515 | 32,509 | 32,521 | 32,508 | 32,508 | 32,506 |
| * unit loading in prohibited zone | | | | | | | |

## 4.7 Conclusions

The genetic algorithm method is capable of solving the constrained economic dispatch problem for practical power systems. The proper choice of the appropriate GA model, is however important as has been demonstrated in this work. The deterministic crowding genetic algorithm has shown its superiority over the standard GA in solving this problem. The method is attractive because there are few GA parameters to be adjusted, resulting in less prior experimentation before application of the model. Using a parallel local hill climbing algorithm on the final population of the crowding algorithm has been found to provide improved final solutions.

# Chapter 5. The Thermal Generation Scheduling Problem

## 5.1 Introduction

A number of estimates have shown that a 1% reduction in power production costs can result in annual savings of up to one million dollars for each 1000 MW of installed capacity. This economic incentive has led to a concerted effort in the search for algorithms that can provide any improvements in system operation costs. The scheduling of thermal generators in a power system is the act of determining the optimum combination of the available units to supply a given load profile at minimum cost. Scheduling power system operation involves two basic economic decisions:

1. a unit commitment decision that determines which units should be brought on-line to meet the expected load demand and reserve requirements, and
2. an embedded economic dispatch decision that determines the most economic generation level for each of the committed (synchronised) units.

In the unit commitment phase, the start up and shut down times of the units over the whole scheduling period must be specified. Once units are commited, an economic dispatch phase allocates the load among the on-line units to satisfy the load demand at a given time interval, as described in chapter 4. The solution of the thermal generation scheduling problem involves a non-linear optimisation problem, consisting of both integer and continuous variables, with a large number of equality and inequality constraints.

## 5.2 Modelling The Thermal Scheduling Problem

The thermal scheduling problem involves the determination of the start up and shut down times as well as the power output levels of all the system generating units at each time step, over a specified scheduling period T, so that the total start up, shutdown and running costs are minimised subject to a number of system and unit constraints. Obtaining an optimal schedule of generation involves the solution of a mixed integer non linear optimisation problem with a large number of constraints.

### 5.2.1 Problem objective function

The main objective of scheduling in thermal systems is to minimise system operation costs. The total production cost, $F_T$ for the scheduling period is the sum of the running cost, start up cost and shut down cost for all the units and is given by,

$$F_T = \sum_{t=1}^{T} \sum_{i=1}^{N} FC_{i,t} + SC_{i,t} + +SD_{i,t} \tag{5-1}$$

where $FC_i$, $SC_i$ and $SD_i$ are running costs, start up costs and shut down costs respectively.

### 5.2.2 Fuel costs

Fuel costs of thermal units are usually represented by a quadratic heat rate curve as a function of power output multiplied by the price of the selected fuel. If $FC_i$ is the function that relates the generator power output, $P_i$ to fuel cost, an example of a frequently used function is:

$$FC_i = \alpha_i + \beta_i P_i + \gamma_i P_i^2 \tag{5-2}$$

where $\alpha_i$, $\beta_i$,$\gamma_i$ represent unit cost coefficients.

### 5.2.3 Transition costs

The start up and shut down costs of a unit is a mixture of fixed and variable down time dependent costs. The costs will depend on, for instance unit cooling constant, number of boilers and other plant components involved in the unit start up or shut down process. The generator start up cost, $SC_i$ depends on the time the unit has been off prior to start up and can be represented by an exponential cost function,

$$SC_{i,t} = \sigma_i + \delta_i \left[ 1 - exp\left( \frac{-T_{off,i}}{\tau_i} \right) \right] \tag{5-3}$$

$\sigma_i$ is the hot start up cost, $\delta_i$ the cold start up cost, $\tau_i$ the unit cooling time constant and $T_{off,i}$ is the time a unit has been off prior to start up. The shut down cost, $SD_i$, is usually given a constant value for each unit. The Objective function $F_T$ is minimised subject to a number of system and unit constraints. These constraints are described in the following sections.

### 5.2.4 Active power balance

The load demand variation over the scheduling interval provides one of the biggest challenges for the scheduling algorithm. At each time step, the power generated must supply

the load demand plus system losses. The total power generated by the on line units must supply the load demand, $P_D$ and system losses, $P_L$,

$$\sum_{i=1}^{N} P_i u_{i,t} = P_{D,t} + P_{L,t} \qquad t \in T \tag{5-4}$$

### 5.2.5 System reserve requirements

An operating reserve is usually required to cover any shortfalls in generation. The on-line units must be able to satisfy a given system reserve policy within the stipulated time frames. The hourly spinning reserve requirements R must be met over the whole scheduling period

$$\sum_{i=1}^{N} P_{i,t}^{max} u_{i,t} \geq P_{D,t} + P_{L,t} + R_t \qquad t \in T \tag{5-5}$$

### 5.2.6 Unit minimum up and down times

Unit minimum up and down times limit the thermal stresses that the machines would be subjected to if they were started or shut down at will. Thus a unit can only be shut down (started) once it has been staying on (off) for a minimum period of time, called minimum up (down) time. Thus the minimum up (down) (MUT/MDT) time limits of units must not be violated,

$$\begin{aligned} \left(T_{t-1,i}^{on} - MUT_i\right)\left(u_{i,t-1} - u_{i,t}\right) &\geq 0 \\ \left(T_{t-1,i}^{off} - MDT_i\right)\left(u_{i,t} - u_{i,t-1}\right) &\geq 0 \end{aligned} \tag{5-6}$$

$T_{off}$ / $T_{on}$ *is* the unit off / on time, while $u_{t,i}$ denotes the unit *off / on* [0,1] status.

### 5.2.7 Unit constraints

The generating unit can be subjected to a number of constraints that must be accounted for in the scheduling process: These include

1) Unit rated minimum and maximum capacities must not be violated,

$$P_{i,t}^{min} \leq P_{i,t} \leq P_{i,t}^{max} \qquad t \in T \tag{5-7}$$

2) The initial unit states at the start of the scheduling period must be taken into account,

3) unit ramp rate limits which restrict the loading of generators between adjacent hours,

4) crew constraints which limit the number of units that can be started at the same time in a given plant,

5) unit operating status or mode restrictions.

### 5.2.8 Transmission network constraints

The transmission line loading limits must not be violated by the power flow resulting from a given schedule. These constraints are especially crucial for systems which are loaded closer to their thermal limits.

### 5.2.9 Emission constraints

With the increased demand for cleaner environments, the thermal schedules must not violate allowable emission limits, and if possible, the Nitrogen dioxide ($NO_2$), Sulphur dioxide ($SO_2$) and Carbon dioxide ($CO_2$) emissions should be minimised.

## 5.3 Review of Thermal Scheduling Techniques

Once utility operators realised the benefit of optimal loading of power system generating units, the search for algorithms that can provide economic benefits to system operation has never ceased. Generation scheduling in practical power systems poses difficult analytical challenges, as it involves the solution of a large scale mixed integer non-linear, non-differentiable, optimisation problem, with a large number of constraints. The ideal method of solving the generator scheduling problem involves an exhaustive trial of all the possible solutions and then choosing the best amongst these. This straightforward method would test all combinations of units that can supply the load and reserve requirements and the combination with the least operating cost is taken as the optimal schedule. Given enough time this enumerative process is guaranteed to find the optimal solution, however, the presently available computation power limits the use of this technique to trivial scheduling problems.

Most utility companies have traditionally relied on the use of heuristic rules and their operating engineer's judgement in the solution of this challenging task. Although these techniques have provided reasonable low cost operation schedules, the proximity of the solution to the optimum could not be determine for realistic power systems. The deregulation of the electricity supply industry has further complicated the scheduling function as there are many more different energy sources, each with its own peculiar operating characteristics resulting in a diverse range of operation constraints which have to be satisfied. For example, the scheduling function must now deal with combined cycle units, with different fuel mixtures and operating zones, energy storage plants, cogeneration facilities and tighter emission limits. Thus as the utility business becomes more complex and competitive, the drive for finding least cost schedules becomes even greater. In the past a number of attempts using both empirical analysis and rigorous mathematical programming techniques has been made at solving the generation scheduling problem.

In recent comprehensive literature surveys [Fahd and Sheble], [Cohen and Sherkat] provide up to date comparisons of the main scheduling solution methods. A closer review of some of these methods is given in the next section.

### 5.3.1 Heuristics and Expert Systems

Heuristic methods are based around rules that are derived from the system characteristics. For example in the merit order unit commitment method [Shoults and Chang] [Lee, 1991] [Happ and Johnson], [Baldwin et. al.], [Wood and Wollenberg], the units are committed according to a priority list based on full load average production and transition costs. Heuristic rules and expert systems are usually built around this scheme. This method will provide reasonably accurate results only if linear cost functions are used. If a more accurate modelling of the unit performance is required, then the merit order scheme fails to provide an acceptable solution. Its simplicity, speed and ability to guarantee feasible solutions makes it one of the most widely used method by electricity utilities.

A number of expert system based techniques [Sheble, 1990] [Ouyang and Shahidehpor, 1991] [Tong, Shahidehpour and Ouyang], with specific rules for particular systems have also found widespread use in utility scheduling. The expert systems incorporate the knowledge of unit commitment specialists and power system operators in a numerical algorithm to create an expert system rule base. The difficulty with most of these expert system techniques is that they are very system specific and they are not guaranteed to provide near optimal solutions.

### 5.3.2 Dynamic Programming

Dynamic programming, originally formulated by [Bellman] is one of the techniques that has been most widely researched since its inception because of its ability to handle any problem which can be formulated as a set of of separable state transitions. Its disadvantage, however is that the number of independent discrete variables is restricted to very low number, restricting its use for only very small problems. Dynamic programming works by determining the optimal set of state transitions which will bring the system from an initial state to a final state. However, since the number of system states is determined by the full enumeration of the values of all the discrete variables that satisfy the system constraints, only small systems can be optimised, as the number of states to be stored and evaluated increases exponentially with the increase in the number of variables.

To avoid this *curse of dimensionality problem* in unit commitment, dynamic programming is used to solve the commitment of a single unit, in combination with other techniques such as Lagrangian relaxation. When used on its own, for the unit commitment problem, the DP decomposes the problem in time, beginning at the first hour of the scheduling period, committing the units progressively one hour at a time and storing the unit combinations at each hour and their associated costs. At the end of the schedule, the steps are traced

backwards to obtain the optimal combinations, that results in the least cost schedule. Storing all possible unit combination at every hour is impossible, and heuristic techniques and approximations [Lowery], [Snyder et. al.] [Hobbs and Hermon] are used to reduce the computational and storage burden. These heuristics and approximations generally lead to sub optimal solutions.

The truncated or variable window DP [Pang and Chen] [Pang et. al.] method is another attempt at reducing the DP storage difficulties. As has already been mentioned, once the unit commitment is broken down into smaller sub problems, the DP method can be easily used to solve the sub problem, and the sub problems are then co-ordinated through a successive approximation approach or through a Lagrangian dual co-ordinator [Van De Bosch and Honderdard]. Expert systems [Ouyang and Shahidepour, 1992] have also been used to improve the solutions provided by the truncated, variable window or approximated DP methods.

### 5.3.3 Lagrangian Relaxation

Lagrangian relaxation (LR) is a mathematical decomposition technique which is rapidly growing in importance as a solution method for the generation scheduling problem, especially for large power systems. The LR decomposition technique, based on duality theory [Bertsekas et. al.], [Geoffrion] generates a separable problem by integrating some constraints into the objective function, through penalty factors, which are functions of the constraint violation. The penalty factors, called Lagrangian multipliers, are determined iteratively and they determine the solution quality. The Lagrangian relaxation method involves two optimisation processes; one for solving the individual sub problems and the other for estimating the values of the Lagrangian multipliers for the iterative co-ordinator. Based on the duality theory, the LR method tries to obtain those values of the Lagrangian multipliers that maximise the dual objective function. Even though the solution to the dual problem can easily be found, feasibility of the original problem is not guaranteed due to the non convexity of the primal function.

In the thermal unit commitment problem, the LR generates single unit sub problems (dual) which are optimised by any suitable technique such as dynamic or mixed integer programming, while the co-ordinator is usually solved by a sub gradient method or heuristics. The dual problem always has a lower dimension than the primal problem and is easier to solve. The difference between the two functions yields the duality gap which provides a measure of the optimality of the solution. Most of the LR research [Zhuang and Galiana] [Bertsekas and Lauer], [Lauer et. al], [Muckstadt and Koenig] [Merlin and Sandrin], [Cohen and Wan] [Aoki and Sato], [Bard] [Tong and Shahidepour] [Virmani et. al.], [ Ruzic and Rajacovic], has therefore concentrated on finding an appropriate co-ordination technique for generating feasible primal solutions, while minimising the

duality gap. Most of the studies update the Lagrangian multipliers using sub gradient algorithms with scaling factors and tuning constants determined heuristically.

Despite the difficulty encountered in obtaining feasible solutions with the LR method, the LR method is so far one of the most promising scheduling techniques, especially for large scale power systems [Wollenberg, see personal communications in appendix]

### 5.3.4 Branch and Bound

The branch and bound method [Lauer et. al.], [Cohen and Yoshimura], [Chen and Wang] is a powerful enumeration strategy that helps to reduce the number of combinations of integer variables considered in a mixed integer non linear programming problem. The advantage of the branch and bound technique is that it can provide a sequence of solutions with estimates of their sub optimality. The branch and bound method also suffers from the curse of dimensionality problem like the dynamic programming method.

### 5.3.5 Mixed Integer Programming

Mixed integer programming method [Garver] [Mucksdat and Wilson] [Dillon et. al.], [Turgeon, 1978], is based on at least two different algorithms, the first of which determines the integer variables, the second (usually a linear program) solves the remaining continuous problem and provides for a new run of the the first algorithm. The integer variables are usually determined by some enumeration algorithm such as branch and bound, which searches for the feasible states. Though theoretically guaranteed to find the optimal solution, the MIP program usually cannot be allowed to complete the full search because of computation time limitations, hence this technique is only limited to small systems.

### 5.3.6 Linear and Quadratic Programming

In solving the thermal scheduling the linear programming method [Van Meteren] is usually combined with other techniques such as DP or MIP, with the linear program solving the continuous economic dispatch sub problem. The main disadvantage of this method is that all the objective functions and constraints must be linearised, which often leads to sub optimal solutions. The quadratic programming method can deal with a quadratic objective function with linearised constraints and can be used as a sub-process of the mixed integer programming method.

### 5.3.7 Simulated Annealing

[Kirkpatrick et. al.] developed an intelligent search strategy based on an analogy with the physical process of thermal annealing of liquids, termed simulated annealing. The search process is based on statistical mechanics, and was originally used to obtain global solutions for combinatorial or discrete optimisation problems. Simulated annealing generates feasible solutions of a minimisation problem which corresponds to the states of a metal undergoing a

cooling process, with the value of the problem objective function corresponding to the energy of the metal in a given state. A more detailed exposition of the method can be found in [Aarts and Korts]. [Fuang and Galiana] have applied the simulated annealing method to thermal unit commitment. [Sasaki and Watanable] also used the simulated annealing method in combination with artificial neural networks to solve the unit commitment problem, while [Annakkage et. al.] use a parallel simulated annealing for the unit commitment problem. The main draw back of the simulated annealing method is the usually long computation as well as the inability to guarantee feasible solutions.

### 5.3.8 Artificial Neural Networks

Artificial neural networks (ANN), are computational techniques derived from brain neurone models. A Hopfield neural network model [Hopfield], [Hopfield and Tank] has been applied to the thermal scheduling problem in a number of studies. The Hopfield and Tank model mimics the computational capability of biological organisms by utilising simple computing units (neurones) with high interconnectivity and parallelism.

[Sasaki and Watanabe] have applied the Hopfield Tank ANN model to a linearised unit commitment problem, incorporating simulated annealing method to escape local minima. The solutions obtained, however were far from optimal. Similar results on the Hopfield Tank model were reported in an EPRI study, [EPRI, 1994], where the ANN was used to relate past schedules, through a learning process, to predict future schedules. [Musoke et. al.] also apply the Hopfield Tank neural network model to the unit commitment problem.

[Ouyang and Shahidehpour] used a combination of the back propagation trained neural network combined with dynamic programming to solve the unit commitment problem. The training data set used for back propagation contains past unit schedules which are assumed to be optimal. Once trained, the network is presented with a new load profile for which a schedule is desired. [Liu et. al.] formulate a coupled artificial neural network for the unit commitment problem where a Boltzman machine is used for the unit states while a Hopfield Tank model solves the economic dispatch sub problem.

In all the above studies, the artificial neural networks were not able to provide satisfactory solutions if the system configuration or the the typical load profile changed. Although, the ANN have the ability to produce fast schedules, their main draw back is the difficulty in providing feasible and near optimal solutions.

### 5.3.9 Evolutionary Algorithms

In the power systems scheduling area, a number of studies using evolutionary algorithms has been reported. Evolutionary algorithms have shown good performance on a number of test problems, but the results so far presented have mainly been limited to small test systems.

[DasGupta and McGregor], [X. Ma et. al] have used the genetic algorithm for thermal unit commitment, with test results based on a 10 unit system. [Sheble and Maifeld,] have also based their genetic algorithm results on a 10 unit test system. [EPRI, 1994] provide an analysis of the performance of the genetic algorithm method compared with Lagrangian relation and artificial neural network methods for a 10 unit unit test system. [Muller and Petrisch] combine dynamic programming, simulated annealing and evolution strategies for the unit commitment problem. [Karzalis et. al.] provide an enhanced genetic algorithm solution model to the unit commitment problem, based on a swap window mutation and hill climbing operators and report test results on systems of up to 100 units. The results are however far from optimal. [Orero and Irving 1995, 1996] report on the successful use of a GA for scheduling in medium and large scale test systems.

None of the basic evolutionary computational methods can guarantee an optimal solution to the scheduling problem. They only provide near optimal solutions, and the quality of each solution provided is affected by either solution time limitations, or by premature convergence. Various enhancements and modifications are required to enable the basic evolutionary computational models to be applied successfully in the solution of the scheduling problem. It is the design of these appropriate computational models that form the basis of this study.

# Chapter 6. Thermal Scheduling With Genetic Algorithms

In this chapter the design and implementation of *canonical* (standard) genetic algorithm and *deterministic crowding* genetic algorithm models for the solution of the thermal scheduling problem is investigated and the experimental results on various power system test networks presented.

## 6.1 Problem Representation

The two main actions that link the genetic algorithm to the problem it is solving are; the means of translating the problem solution to a chromosome representation and an evaluation function that returns a measurement of worth for any chromosome in the context of the problem.

The success of the GA approach greatly depends on the problem encoding and the choice of the evaluation function. In order to use a genetic algorithm to solve any problem, a mapping function that translates the problem into a suitable format for the application of the GA operators is necessary. This encoding is problem specific and varies from one application domain to another. In most applications, binary strings have been used to represent the problem solution, since they allow the genetic operators to be applied with relative ease. [Holland, 1975] provides an analysis which suggests that binary representation provides a minimalist optimal search space. [Antonisse], [Davis], [Eshelman et. al.], [Goldberg, 1989], [Radcliffe, 1991], among others have argued that real number representation is equally valid. Other higher cardinality representations can also be used, as has been demonstrated by Koza [Koza, 1992] where symbols are used to represent high level computer structures.

### 6.1.1 Basic problem representation

The thermal scheduling problem is a mixed integer non linear optimisation problem that lends itself to a suitable binary representation. The basic unit commitment problem suggests a convenient binary representation in which {0, 1} denotes the unit *off / on* status. A candidate solution is then a string whose length is the product of the number of generators and the scheduling period. The {0, 1} representation facilitates the determination of the unit on / off times and the start up / shut down decision variables. A typical binary unit commitment problem representation for scheduling $n$ units over $T$ hours is shown in figure 6.1.

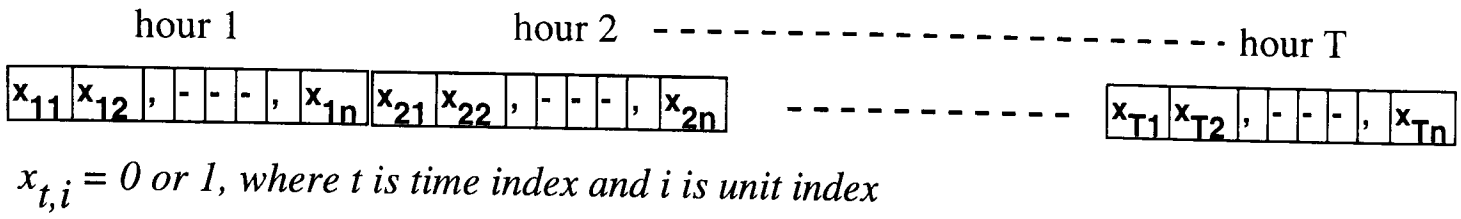



**Figure 6.1 An example of binary thermal scheduling problem encoding**

### 6.1.2 Problem representation incorporating multiple unit operating modes

Modern generating units are increasingly having to cope with complex operation regimes in order to increase efficiency and reduce costs. For example, a combined cycle unit can combine several sub-units, such as gas turbine and steam turbine sub-units, resulting in a number of different operating modes. Thus, the unit commitment decision is no longer limited to running or not running a unit. Modelling constraints such as multiple unit operating modes provide enormous challenges to conventional thermal scheduling techniques. In the GA approach, these different unit operating modes can easily be taken into account by modifying the basic unit commitment representation in figure 6.1. Each unit will be represented by a string length proportional to the number of operating modes required for the particular unit. For example, if a unit is a combined cycle unit, with both steam and gas turbines, the bit representations are as shown in table 6.1, increasing the number of possible bit representations for each unit by the number of operating modes of the given plant. The resultant total chromosome length for all the units to schedule is then given by the sum of the binary bits representing each unit.

**Table 6.1 Encoding of combined cycle units**

| possible unit states | Interpretation |
|---|---|
| 0 0 | Both Steam and Gas turbine OFF |
| 0 1 | Steam turbine ON, Gas turbine OFF |
| 1 0 | Steam turbine OFF , Gas turbine ON |
| 1 1 | Both Steam and Gas turbine ON |

### 6.1.3 GA representation of other unit constraints

Other unit constraints such as maintenance schedules, ramp rates, must run, *must not run*, unit derated capacities can be properly accounted for in the genetic algorithm encoding scheme.

#### 6.1.3.1 Unit maintenance

If a unit is on planned maintenance that begins and ends within the scheduling period, it is important to avail the unit to the scheduling process as soon as it is available, instead of waiting for the next scheduling period. This can be easily incorporated into the GA formulation by modelling the unit status changes as shown in figure 6.2.

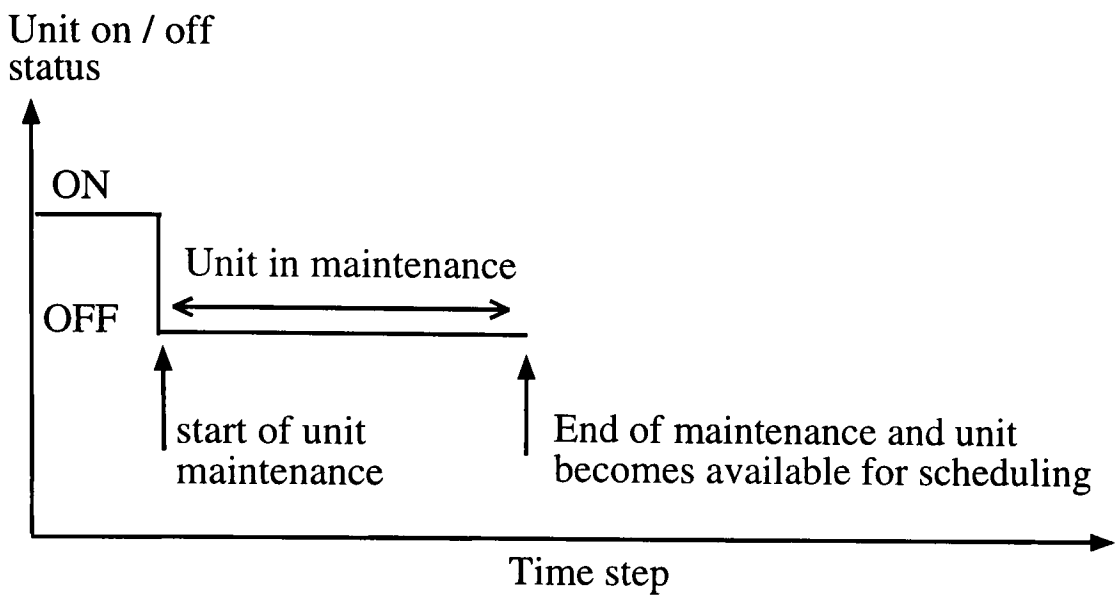



**Figure 6.2 Unit maintenance modelling**

In the binary string representation, the unit status is given a value of {0} during the time intervals over which the unit is undergoing maintenance, but is free to participate in the schedules in the other time intervals.

#### 6.1.3.2 Ramp rates

The ramp rate constraints limit the rate at which units can pick up or drop their loads. In the GA representation, the ramp rate limits constrain the maximum possible unit loading as a function of the number of hours a unit has been on and the number of hours to the next shut down interval, both quantities are easily computed from the unit *on / off* status in the binary string representation.

#### 6.1.3.3 Must run / *must not run* units

Units with must run status are assigned a status of {1}, while the *must not run* units are given a status of {0} in the GA binary coding scheme.

#### 6.1.3.4 Fixed generation / derated capacity units

Units which are pre-scheduled to run over certain time intervals, or to carry a fixed amount of generation for certain intervals of the scheduling time can be accounted for in the fitness function by including them through a logic function.

### 6.1.4 Economic dispatch sub-problem

In the scheduling process, once a decision has been made on whether a unit is to be *on* or *off*, a further optimisation process must be carried out to optimally allocate the load to the on-line units, a process usually referred to as *economic dispatch*. In solving the thermal scheduling problem with a genetic algorithm, one is tempted to solve the economic dispatch sub problem with the GA as well. However, the complete solution would take unreasonable computation time as a full GA run would be required to solve the economic dispatch problem for each population member. The extremely large string sizes encountered for large systems would result in enormous computation times, with possible degradation in the solution accuracy.

#### 6.1.4.1 Economic dispatch sub-problem with a GA

For the economic dispatch sub problem using a GA, once the required accuracy, or resolution in unit output is decided, the number of binary bits used to represent each unit's output can be calculated using equations 2-4 and 2-5.

For example, consider the thermal scheduling problem involving 10 units over a 24 hour period. Let us make the following additional assumptions; each unit output can be varied to an accuracy of 1 MW and that all the units are of the same ratings with minimum and maximum loading limits of 40 and 168 respectively.

The number of bits required to represent each unit will be 7. Thus the sub string length per hour will be 70, resulting in a total string length of 1680, for the whole scheduling period. An increase in unit output resolution, number of units or scheduling period will result in a proportional increase in string length and problem complexity. Using a GA for the thermal scheduling dispatch sub-problem requires two levels of a genetic algorithm process, the upper for determining the unit status (unit commitment), and the lower for the solving the dispatch sub-problem, for each population string. While such an approach is theoretically possible, it is not computationally feasible for any realistic power system problem. A fast economic dispatch method is required in order to speed up the GA search process.

#### 6.1.4.2 Economic dispatch sub-problem using conventional dispatch methods

Tests were carried out to determine the most suitable economic dispatch method to use with the GA thermal scheduling technique. This was done by comparing the performance of two widely used economic dispatch methods, the Lambda-iteration and merit order dispatch methods [Wood and Wollenberg], with the genetic algorithm method. Table 6.2 shows the comparative performance of the economic dispatch methods on the various test systems. The genetic algorithm results shown are an average of 10 independent trials.

**Table 6.2 Comparing the performance of genetic algorithm and classical economic dispatch methods**

| Test system | Economic Dispatch production costs | | |
|---|---|---|---|
| | Genetic algorithm | Merit order | Lambda-iteration |
| 28 units | 100,465 | 100,688 | 100,328 |
| 43 units | 149,888 | 150,971 | 149,681 |
| 54 units | 143,623 | 143,256 | 143,255 |
| 66 units | 193,931 | 193,604 | 193,486 |
| 75 units | 210,888 | 210,105 | 210,105 |
| Average CPU time (sec.) (based on 54 unit system) | 200 | 0.05 | 0.12 |

Table 6.3 compares the performance of the merit order dispatch and genetic algorithm methods relative to the lambda-iteration method.

**Table 6.3 Comparing the merit dispatch and the genetic algorithm**

| Test system | Percentage cost above the lambda-Iteration method | |
|---|---|---|
| | Genetic algorithm | Merit order |
| 28 units | 0.136 | 0.359 |
| 43 units | 0.138 | 0.861 |
| 54 units | 0.257 | 0.001 |
| 66 units | 0.229 | 0.061 |
| 75 units | 0.373 | 0 |

From tables 6.2 and 6.3 it can be seen that the merit order dispatch method provides a fairly good approximation of the dispatch production costs and can be used as a sub-process of the GA in the unit commitment scheme. For the second order polynomial cost function used to represent the unit characteristics in this work, the genetic algorithm dispatch method does not perform as well as the merit order dispatch method for the problem sizes considered. Although the lambda-iteration dispatch method provides better solutions than the merit order scheme, the merit order method is preferable for solving the economic dispatch sub-problem because it is faster and is not prone to convergence difficulties sometimes experienced by the lambda-iteration method.

## 6.2 Thermal Scheduling Fitness Function

The fitness function is one of the key elements of a genetic algorithm. It is the fitness function that determines whether a given potential solution will contribute its elements to the future generations through the selection process. The fitness function should be able to provide a good measure of the quality of the solution, and should be able to differentiate between the performance of two different strings. In the thermal scheduling problem, the GA is used to search for the best combination of units over the whole scheduling period that satisfies both the system and unit constraints. The economic dispatch sub-problem is then solved by using a merit order dispatch routine based on the average incremental fuel costs

for each member of the population that satisfies all the above constraints. The total cost function, $F_G$ , for each string to be minimised by the GA is :

$$F_G = \sum_{t=1}^{T} F_T(t) + P_{SC}(t) + P_{SD}(t) + P_R(t) \tag{6-1}$$

where

$$F_T(i,t) = F_C(i,t) + S_C(i,t) + S_D(i,t) \tag{6-2}$$

where *SC(i,t) / SD(i,t)* are the start up / shut down costs, *PSC(i,t), / PSD(i,t)* are the premature start / shut down penalty costs, *FC(i,t)* is the running cost and *PR(i,t)* is the failure to meet reserve penalty cost. Since the basic framework for the genetic algorithm maximises a fitness function, the minimisation cost function must be transformed to a maximisation function. The fitness of a string (a particular solution), $f_i$, can for instance, be obtained by subtracting its overall cost function, $C_i$ value from the maximum cost, $C_{max}$ in the population, of size, P, in any given generation.

$$f_i = C_{max}|_{i \in P} - C_i \tag{6-3}$$

### 6.2.1 Running costs

The running cost, $F_C$(i,t) is composed of the costs of fuel and maintenance. In this work, a quadratic cost function is used to represent the running costs, although the GA can equally handle non convex functions without any modifications to the solution methodology. The cost function used is of the form:

$$FC_i = a_i + b_i P_i + c_i P_i^2 \tag{6-4}$$

### 6.2.2 Transition costs

Transition costs incurred when a unit changes its *on* or *off* status during a shut down or start up are described in section 5.2.3.

### 6.2.3 Spinning reserve

Spinning reserve is the excess capacity of synchronised generation above the load demand and is usually provided to cover for a shortfall in demand either due to the loss of a generating unit or inaccuracies in load prediction. Spinning reserve is costly as it requires some units to be partially loaded, at which point fuel efficiency may be less than at higher loading points. For fuel economy, a minimum amount of spinning reserve subject to an acceptable risk level should be maintained. The spinning reserve is usually set according to some of the following rules:

- as a fixed percentage of predicted load demand,
- to cover the loss of largest generation within a predetermined time,
- based on loss of load probabilities and accepted levels of risk.

The spinning reserve contribution from a generating unit, *SPR*, is the spare capacity available from the unit, or the ramping capability of the unit within a prescribed time period, t, whichever is smaller.

$$SPR = MIN\left\{\left(P^{max} - P_{load}\right), Ramp\,rate * t\right\} \tag{6-5}$$

The available spinning reserve from the committed units can easily be calculated from the GA binary representation.

### 6.2.4 Load balance

The total generation of all the committed units must satisfy the load demand plus losses, for all the time intervals over the whole scheduling period. The load balance is guaranteed to be satisfied once the spinning reserve requirements are met. For optimum system operation, the most economical loading levels that satisfy the reserve requirements is sought, and this is achieved by incorporating the economic dispatch sub-problem into the overall thermal scheduling process, as earlier discussed.

### 6.2.5 Penalty function method for constraint handling

In order to apply a genetic algorithm to a constrained problem, one of three main approaches can be adopted, to deal with the problem constraints, namely:

1. generate only feasible candidate solutions by testing each proposed solution for feasibility, a process similar to an enumerative method, which can be very time consuming,
2. modify the genetic operators to suit the constraints, an operation which is only possible for few problem constraints, otherwise the representation could become too complex,
3. penalise solutions, or parts of the solution space, that violate the constraints.

The third approach is the most attractive and can allow the constraints to be treated as penalty functions in a composite objective function. The constrained optimisation problem is converted to an unconstrained problem, where the new objective function is formed by adding a penalty term or group of penalty terms that force the solution to satisfy the constraints.

Several varieties of penalty function methods [Fiacco and McCormick] exist, but the essence of all the methods is to transform a constrained optimisation problem into an unconstrained problem or a sequence of unconstrained problems. While the cost function can be well defined, choosing a penalty function that combines well with the cost function is a difficult task. For the unit commitment task, the penalty function should take care of the

premature start up and shut down constraints as well as the failure to meet demand and reserve requirements. Since the penalty functions and their coefficients affect the performance of the algorithm, the careful selection and grading of these parameters is necessary. Well chosen, graded penalties which differentiate the relative performance of all strings should provide a better performance than harsh penalty functions. The setting of the penalty terms determines the cost values assigned to the constraint violations, as well as how the different constraints relate to each other. Because it is very difficult to calculate an optimal value of the penalty coefficient an empirical choice of a penalty factor that provides an upper bound on the cost to satisfy the violated constraints is adopted [Richardson et. al].

#### 6.2.5.1 Premature start up and shut down penalty functions

The premature start up / shut down, $P_{SC} / P_{SD}$, penalties for each solution string is :

$$P_{SC} / P_{SD} = \sum_{t=1}^{T} \sum_{i=1}^{N} \mu_{i,t}\, r \left[ pst(i,t)^2 / psd(i,t)^2 \right] \qquad (6\text{-}6)$$

$\mu_{i,t} = 1$ for a constraint violation and zero otherwise, while $r$ is a constant term. The quadratic penalty costs are a function of the number of hours a unit has been prematurely started up, *pst(i,t)* or prematurely shut down, *psd(i,t)* as shown by the unit *on / off* time characteristics in figure 6.3.

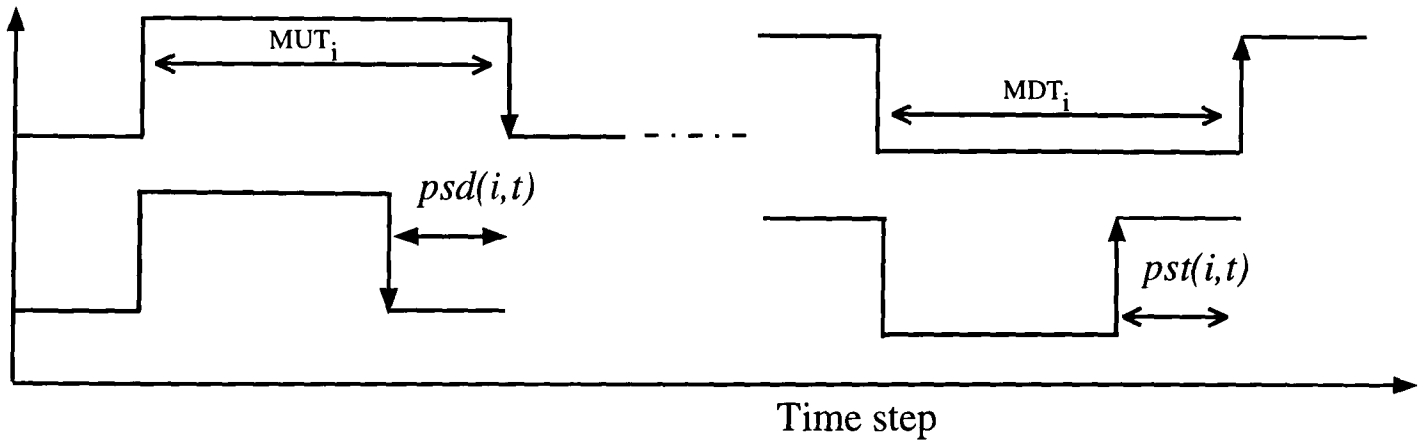



**Figure 6.3 Unit on / off time characteristics**

Since the genetic algorithm works by searching both the feasible and infeasible problem solution search space, it is necessary, because of its population based nature and its selection mechanism, that it differentiates between the performance of two infeasible strings. Thus a string that violates a constraint more than another should receive a higher penalty and expect a corresponding diminished chance of selection.

#### 6.2.5.2 Penalty functions for failure to meet demand and reserve requirements

For the reserve and load demand violations, the penalty term, $P_R$, is a function of the number of megawatts below the sum of the reserve margin, $R_t$, and the load demand, $P_D$. The constant, ε, represents the acceptable tolerance for the satisfaction of the equality constraint.

$$IF \quad \left( R_t + P_{D,t} - \sum_{i=1}^{N} P_{i,t} > \epsilon \right) \quad Then$$

$$P_R = \sum_{t=1}^{T} r \left[ R_t + P_{D,t} - \sum_{i=1}^{N} P_{i,t} \right]^2 \qquad (6\text{-}7)$$

$$else \; P_R = 0$$

A typical load demand and power generation variation and constraint satisfaction boundary over the scheduling interval is shown in figure 6.4.

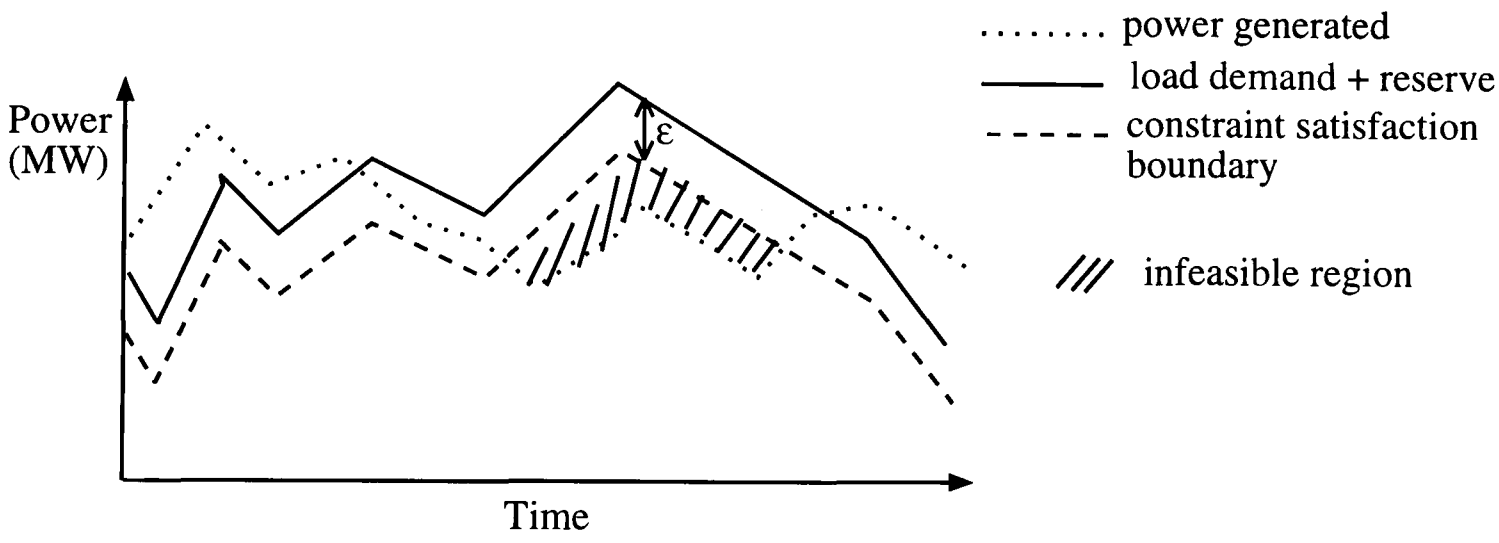



**Figure 6.4 Load demand and power generation curve**

#### 6.2.5.3 Penalty function grading

The discrimination between feasible and non feasible solutions is achieved by making sure that any non feasible solution is assigned a fitness lower that of a feasible solution through the application of the penalty functions. The violations which are far from the feasibility boundary are awarded higher penalties than those near the constraint boundary. The penalty costs are made directly proportional to the distance from the feasibility boundary. The penalty grading is crucial to the GA performance since the genetic algorithm advances its search for the optimal solution by awarding better survival chances to the overall structure with a higher fitness value. If the penalty function grading is not done properly, the genetic algorithm can be deceived into awarding fitness erroneously, and this leads to a GA *deceptive* [ Goldberg, 1987) problem.

## 6.3 Design of the Thermal Scheduling Canonical GA

The basic design of a genetic algorithm solution method involves making an optimal choice of the various genetic algorithm parameters, once the appropriate genetic algorithm model has been chosen. The *canonical* (standard) genetic algorithm model implemented in this work is based on Holland's genetic plan [Holland, 1975].

This genetic algorithm model is composed of the basic sequential processes of fitness evaluation, selection, recombination and mutation. A binary representation is used to map the problem into a GA framework. An initial population of possible problem solutions is created, and then allowed to repeatedly undergo the process of fitness evaluation and possibly scaling, selection, recombination and mutation for a pre-set number of generations or until a given stopping criterion is satisfied. The theoretical foundations of this model is based on the schema theorem [Holland, 1975], [Goldberg, 1989] which has been summarised in chapter 2.

### 6.3.1 Choice of Genetic Algorithm Parameters

It has long been established from numerous studies that the parameters that control the genetic algorithm can have a significant influence on its performance, although the main GA theory does not provide sufficient guidance on the optimal setting of these parameters. Factors such as the population size, crossover probability, mutation probability, selection pressure, generation gap are among the major control parameters of the genetic algorithm that can be varied by the user. Other criteria such as selection method, crossover method and parent replacement mechanisms are equally important and must be decided, using prior experimentation and theoretical insights before applying a GA to the problem.

A number of empirical studies, [Grefensttete, 1986], [Schaffer et. al.], [De Jong, 1975], and numerous others have shown that parameter settings which are optimal for a particular set of test functions may not necessarily be useful for functions outside that set, leading to the general conclusion that optimal parameter settings depend on the GA model being used as well as the function being optimised, and can only be chosen after a number of trials in the particular problem domain. Most of the studies have been restricted to certain problem suites, which sometimes are not very representative of problems usually encountered in practise. Grefensttete [Grefensttete, 1986] used a meta GA to locate parameter sets which, were themselves optimised using a GA and in an extended analysis of meta GA approach, he proposes the concept of a virtual GA, [Grefensttete, 1995] to set the control parameters. The virtual genetic algorithm tool operates like a genetic algorithm, except that evaluations of individuals are based on derived statistical fitness models, and because it by-passes the evaluation process, the virtual GA can be executed in a fraction of the time of the GA that it models, allowing it to provide cost effective predictions on the performance of alternative GA representations and operators.

In the recent past, a lot of research has concentrated on establishing a number of theoretical and empirical relationships such as :

- linking GA population size with problem size and noise [Goldberg and Deb],
- establishing the link between selection methods and pressure with genetic algorithm convergence characteristics, [Miller and Goldberg], [Goldberg, 1989] [Muhlenbein, Schlierkamp-Voosen, 1993], [Thierens and Goldberg 1994], [Back, 1994],
- choice of optimum crossover methods and rates, [Spears and De Jong], [Spears, 1995],
- role of mutation on GA convergence characteristics [Back], [Muhlenbein], [Spears, 1993]
- relationship between selection, string length and population size on GA convergence [Spears and De Jong], [Nix and Vose],
- relationships between multi-modal functions and GA models [Mahfoud, 1993], [Horn], [Goldberg, 1989], [De Jong], [Beasley].

In this work, computer programmes have been developed to implement a genetic algorithm for thermal scheduling based on the canonical GA model. It has been modularized into a set of independent modules that perform various functions such as; data input, creation of an initial population, fitness evaluation, selection, reproduction, mutation, parent replacement, performance analysis statistics and output of results among, others.

Several experiments have been carried out to determine the optimum genetic algorithm control parameter settings for the thermal scheduling problem. The experiments have been carefully designed taking into account the evidence of numerous empirical studies and the already available and unfolding theoretical foundations of the GA process. These tests have included:

1. determination of optimum selection, crossover, mutation, fitness scaling, parent replacement methods,
2. determination of a reasonable population size, crossover rate and mutation rate,
3. evaluating the effect of randomisation process and *seeding* of the initial population,
4. determination of the algorithm convergence characteristics for various control parameter settings,
5. analysing the effect of the thermal scheduling problem difficulty such as variation of unit minimum *on / off* times, ramp rates effects, load demand profile, number of units. unit characteristics on the algorithm performance,
6. determination of optimum combination of the GA parameters.

A summary of the findings of these experiments is presented in the following sections.

In most of the GA studies, the population size is chosen from empirical studies, and population sizes between 30 and 100 have been suggested to be optimal, but these settings must be treated with caution as they were generally derived for small problem sets.

The population size and overall GA performance is also linked to other control parameters, for example, it has been found that the uniform crossover operator leads to improved performance in relatively small population GA [Syswerda, 1989]. In this study, various population size settings have been investigated, and good GA performance was obtained when the population size was made proportional to the problem string length.

### 6.3.4 Crossover operation

Crossover involves creation of new offspring from the mating of two selected parents. There are several methods of performing crossover such as single point crossover, two point crossover and uniform crossover among, others. [Spears], [Spears and De Jong], [Whitley, 1994], [Syswerda, 1989] provide a theoretical analysis and comparison of the different crossover mechanisms and their effects on the GA convergence process. The single point crossover, two point crossover and uniform crossover methods have all been investigated in this work. The uniform crossover was found to offer a superior performance for the population sizes considered for the various scheduling tests. This is corroborated by the evidence of a number of researchers. [Spears and De Jong, 1991] and [Syswerda, 1989] provide the theoretical analysis that shows that the uniform crossover operator has a better recombination potential than the single point or two point crossover operators, especially for small population GA.

### 6.3.5 Mutation operation

Mutation operator offers the opportunity for new genetic material to be introduced into the population, and has been applied alongside the crossover operator in this study. Mutation was found to be beneficial, but only when applied at very low rates. A mutation rate setting equal to the reciprocal of the string length was found to give adequate performance.

### 6.3.6 Fitness scaling

Fitness scaling avoids two processes that may hamper the proper convergence of the algorithm. These are:

1. The early populations typically consist of relatively unfit individuals along with a few good ones. Although these good members are by no means optimal, the standard selection operator will provide them with a large number of offspring which increases the danger of premature convergence to a local optimum.

2. A second convergence occurs when most of the population members have nearly identical fitness function values, especially near the end of a series of iterations. Direct use of the raw fitness function in conjunction with the selection operator leads to a slow con-

In most of the GA studies, the population size is chosen from empirical studies, and population sizes between 30 and 100 have been suggested to be optimal, but these settings must be treated with caution as they were generally derived for small problem sets.

The population size and overall GA performance is also linked to other control parameters, for example, it has been found that the uniform crossover operator leads to improved performance in relatively small population GA [Syswerda, 1989]. In this study, various population size settings have been investigated and good GA performance were obtained when the population size was made proportional to the problem string length.

### 6.3.4 Crossover operation

Crossover involves creation of new offspring from the mating of two selected parents. There are several methods of performing crossover such as single point crossover, two point crossover and uniform crossover among, others. [Spears], [Spears and De Jong], [Whitley, 1994], [Syswerda, 1989] provide a theoretical analysis and comparison of the different crossover mechanisms and their effects on the GA convergence process. The single point crossover, two point crossover and uniform crossover methods have all been investigated in this work. The uniform crossover was found to offer a superior performance for the population sizes considered for the various scheduling tests. This is corroborated by the evidence of a number of researchers. [Spears and De Jong, 1991] and [Syswerda, 1989] provide the theoretical analysis that shows that the uniform crossover operator has a better recombination potential than the single point or two point crossover operators, especially for small population GA.

### 6.3.5 Mutation operation

Mutation operator offers the opportunity for new genetic material to be introduced into the population, and has been applied alongside the crossover operator in this study. Mutation was found to beneficial, but only when applied at very low rates. A mutation rate setting equal to the reciprocal of the string length was found to give adequate performance.

### 6.3.6 Fitness scaling

fitness scaling avoids two processes that may hamper the proper convergence of the algorithm. These are:

1. The early populations typically consist of relatively unfit individuals along with a few good ones. Although these good members are by no means optimal, the standard selection operator will provide them with a large number of offspring which increases the danger of premature convergence to a local optimum.
2. A second convergence occurs when most of the population members have nearly identical fitness function values, especially near the end of a series of iterations. Direct use of the raw fitness function in conjunction with the selection operator leads to a slow con-

vergence, since the selection operator then provides each member with nearly identical number of offspring.

Both these convergence problems can be avoided by properly scaling or spreading out the objective function values for all the population members before selection. The scaling basically fixes the relative spread between the highest and the average objective function values occurring in a population. In the early stages of the iteration process, the scaling reduces the relative spread in order to avoid premature convergence while in the later stages it enlarges the spread to speed up convergence. The effects of linear ranking, exponential ranking and truncation based scaling methods have been investigated, and the truncation scaling method based on the normal population distribution statistics provided a suitable and effective problem scaling mechanism.

### 6.3.7 Elitism

The probabilistic nature of the selection, recombination and mutation process does not guarantee that the best population member will always be selected for reproduction. Elitism basically avoids the loss of the best member in a given generation by preserving and copying it to the next generation intact. The elitist parent replacement strategy in its various forms has been applied in this study, and its effects evaluated.

### 6.3.8 Dynamic variation of GA control parameters

Genetic algorithms are non linear in their behaviour and it is not always easy to determine the optimum combination of GA control parameters (population size, cross over rate, mutation rate etc.) that are best for the complete run. The optimum combination of parameters can and does change in the course of a run from generation to generation. Interpolation methods were applied to track down the variations in operator performance in the course of the test runs. There emerged empirical evidence of some relationship between the relative operator weights and the degree of convergence of a solution. Thus at any point in time, during the execution of a GA, there is a ratio of operator parameters that lead to optimal results and this optimal ratio changes throughout the run.

### 6.3.9 Initial population seeding

The possibility of using previous load schedules, generated in earlier experiments, using GA or other methods was also investigated. In these experiments, the initial GA starting population is composed of past solution strings mixed with randomly generated strings in appropriate proportions. The presence of *old* solutions in the population means that a good proportion of useful building blocks or *schema* are made available to the GA initial population, hence the desired solution can be found in a much reduced time.

### 6.3.10 Use of a local search mechanism on final solution

A local search (hill climbing) mechanism attempts to improve on the solution by moving to a better neighbouring solution. By augmenting the GA approach with local optimisation, impressive results can be achieved. In a GA, one way of implementing a local search is to use a final GA solution, mixed with randomly generated strings as the initial population, of a re-trial run. A conventional hill climbing algorithm can also be used on the final GA solution in an attempt to improve on the results.

## 6.4 Design of The Thermal Scheduling Deterministic Crowding GA

In the deterministic crowding genetic algorithm (DCGA) selection and population replacement are combined together. The main GA control parameters of this approach are the parent replacement-selection strategy and the population size. In this work, computer programmes have been developed to implement a crowding GA for thermal scheduling based on the deterministic crowding GA model described in chapter 3. This GA model was adopted mainly because of its simplicity and robustness, and since there are relatively few GA parameters to be varied.

### 6.4.1 Population size

The population size for the deterministic crowding GA must offer a suitable proportion of good building blocks in order to provide an effective search process. Very small populations tend to become dominated by a single individual and usually lack the diversity required for proper adaptation. For more complicated, large and difficult problems, large populations are required, however large populations are accompanied by an almost proportional increase in computation times. Large populations also tend to prevent the performance of a good individual from propagating its features to future generations. Population sizes should be some function of the problem size.

### 6.4.2 Crossover and mutation

Crossover is performed on the whole population, which is randomly shuffled at the beginning of each generation. It largely determines the behaviour of the DCGA, as without crossover (assuming no mutation as is the case with the main DCGA model), the algorithm would simply advance both parents to the next generation, and hence the population would never change. The uniform crossover operator has been used, since it has been found to be the most suitable crossover operator for the relatively small population sizes adopted in this work.

Mutation is considered as a local search operator in the DCGA model and it is applied with a small probability, usually set to the reciprocal of the string length.

### 6.4.3 Selection and population replacement mechanism

A parent replacement - selection strategy that involves a tournament between the parent and child combinations where, each of the two offspring compete with one of the two parents, chosen according to a phenotype distance metric, is adopted. The fitter individual in a tournament then enters the population for the next generation. The parents are replaced by the most similar children as they are created, as described in chapter 3, and because offspring replace parents only when they are of superior performance, the possibility of losing the best structure in the population is zero and hence there is no need to explicitly apply elitism.

## 6.5 Hybrid GA Solution of The Thermal Scheduling Problem

Hybrid genetic algorithms incorporate other solution methods into the genetic algorithm solution technique. By including the domain knowledge of a conventional solution method into the GA process, an algorithm that out performs both solution techniques can be obtained. The central and original goal of the fundamental research in genetic algorithms was to create robust systems that would perform well across a range of problem domains and therefore little consideration was given to problem solution times. However, in the solution of most engineering problems, the performance of the GA is subject to computing and time resource constraints. The main limitations of the genetic algorithm solution method in solving large scale problems are the long computation times and the possibility of premature convergence. Both of these problems can be greatly alleviated by the use of a hybrid technique. Some researchers [Whitley, 1994] are against hybridisation or any form of domain knowledge inclusion in a GA, arguing that it goes against Holland's [Holland, 1975] fundamental schema processing theorem. However, a number of studies, [Davis, 1991], [Kido and Kitano] have found that hybrid genetic algorithms are able to solve real world problems within acceptable computation times.

The thermal scheduling hybrid GA proposed in this work incorporates the solution produced by a priority list unit commitment scheme as part of its initial population. The choice of the priority list unit commitment method for the hybridisation process was based on the following factors:

- its extreme simplicity makes it very easy to implement,
- it is very fast, hence is able to deal with large scale problems,
- its results are predictable,
- it is one of the most widely used techniques in industry.

The priority list commitment schedule helps in the creation of useful building blocks that are necessary for further GA search. In other words, it injects domain knowledge into the GA search space and acts as a coarse search mechanism. This allows the GA to concentrate its search efforts on the more difficult areas of the search domain. Since the priority list unit

commitment forms part of the initial solution, if elitism is applied, then the hybrid algorithm is guaranteed to to do no worse than the priority list commitment method. In general crossing over the solutions of the priority list method with other population members will often lead to improved solutions.

The hybrid GA flow chart is shown in figure 6.5.

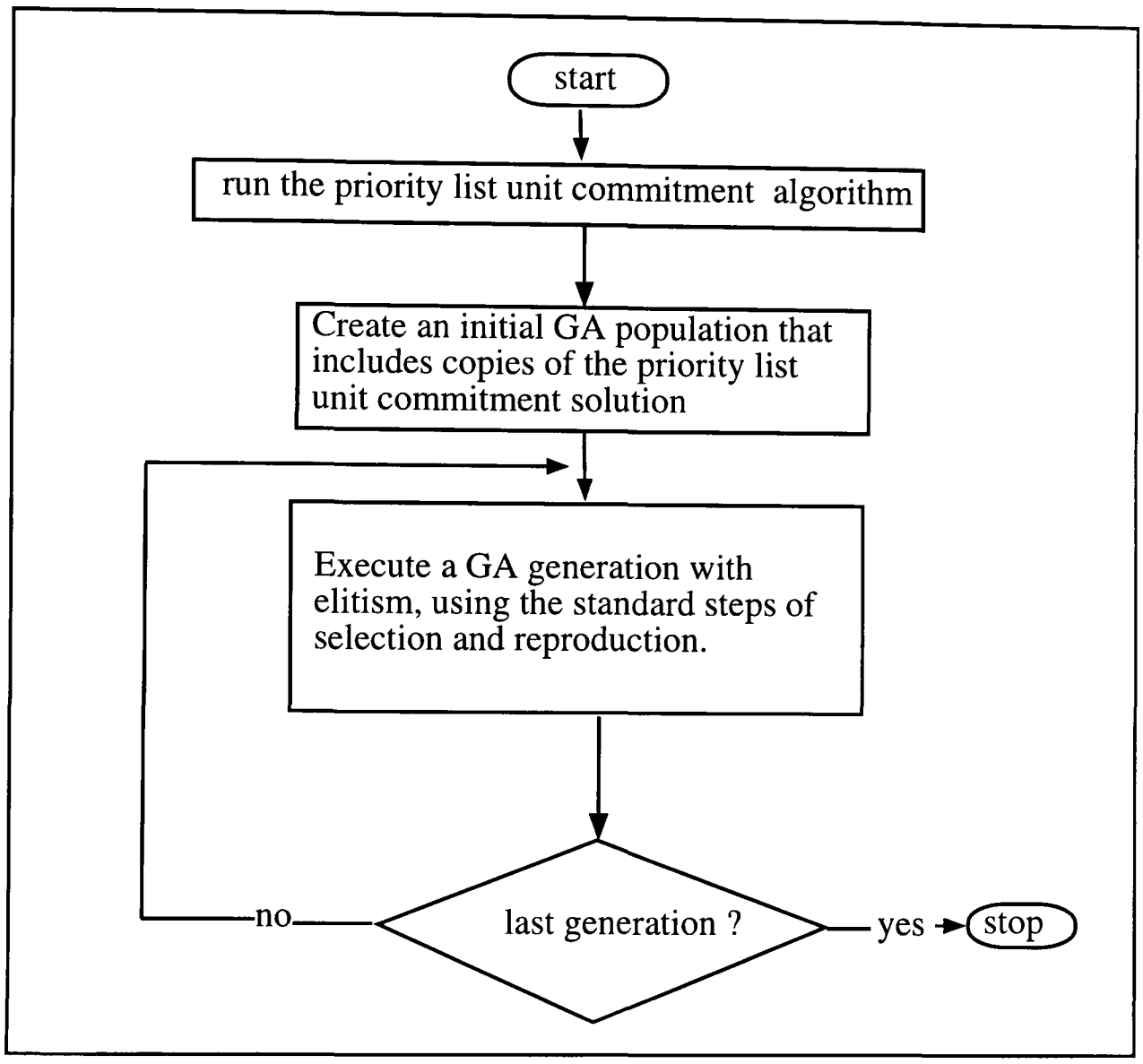



**Figure 6.5 Hybrid unit commitment GA cycle**

### 6.5.1 Priority list unit commitment component

In the priority list unit commitment scheme, units are committed in ascending order of average full load production costs. At each time interval, the next unit in the priority list that satisfies all the unit constraints such as minimum up / down times is committed to supply the additional load demand. Similarly the units are shut down in descending order of average full load production cost. A few rules are added to enhance the basic algorithm, for example, before shutting down a unit, it must be ensured that it will be available for the next time it is required to meet the rising load demand. This method is very fast but its accuracy is limited by the approximate linearisation of the cost function. The basic steps of the priority list unit commitment method are those described in [Wood and Wollenberg].

## 6.6 Genetic Algorithm Performance Measures and Test Systems

### 6.6.1 Performance measures

The main performance measures used to judge the solutions provided by the genetic algorithms in this work are:

1. Quality of final solution as compared with the priority list based solution or other conventional techniques.
2. Efficiency of the GA,
   a) computation time,
   b) GA convergence characteristics.
3. Robustness of the algorithm,
   a) performance on systems with different characteristics,
   b) performance on systems of different sizes, (scalability).

The performance of the GA was judged by the following statistical measures:

- *Off_line performance measure*, This performance measure selects the population member with the best fitness at the end of the run as the optimal solution.
- *On_line performance measure* Provides a measure of the algorithm performance in the course of a test run and can be measured as a the average fitness of the population up to a particular time (generation)

The algorithm was terminated using one or more of the following criteria:

- stopping after a given number of generations or function evaluations, which is obtained from empirical evidence,
- based on the ratio of the average population fitness to the best fitness and when this ratio approaches unity, only little improvement is expected and the algorithm is stopped.
- when all the solutions in a generation are nearly identical, i.e. when the difference between the fitness measure of all the members is less than some small tolerance value.

All the three termination criteria provided satisfactory GA results, that did not significantly differ from each other. On termination the candidate solution with the best performance in the trial was taken as the optimal solution.

### 6.6.2 Test systems

Test systems representing small, medium and large scale power systems of various system characteristics were considered for validating the performance of the developed genetic algorithm models. Making comparisons on a range of problems from very easy to very hard can provide much information about the merits of an algorithm and the scalability of its performance. The factors that varied among the test systems considered included: the load demand pattern, system generator characteristics, cost functions (relative efficiencies), system reserve requirements and number of units to schedule. The smallest test system considered included 5 generating units, scheduled over a 12 hour period. This system was mainly used for developing the GA scheduling algorithm and for implementing a number of problem variations. For example, the systematic incremental variations in the load demand patterns shown in load profiles in figures 6.6 and 6.7 were used to test the ability of the developed algorithm to deal with a scheduling problem with fluctuating load demand profiles. The load profiles show the variations in load demand patterns that represent varying seasonal load representations. The developed GA codes were then used on 10, 26, 100 and 110 unit test systems. In general it was assumed that, the larger the problem, the harder it would be for the GA to solve it.

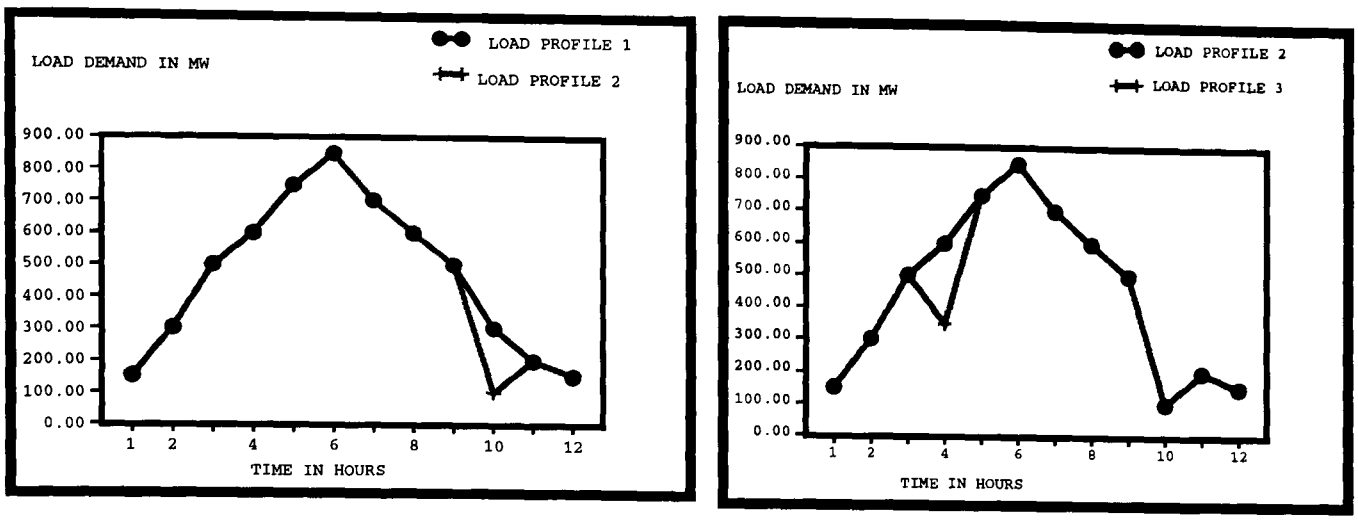
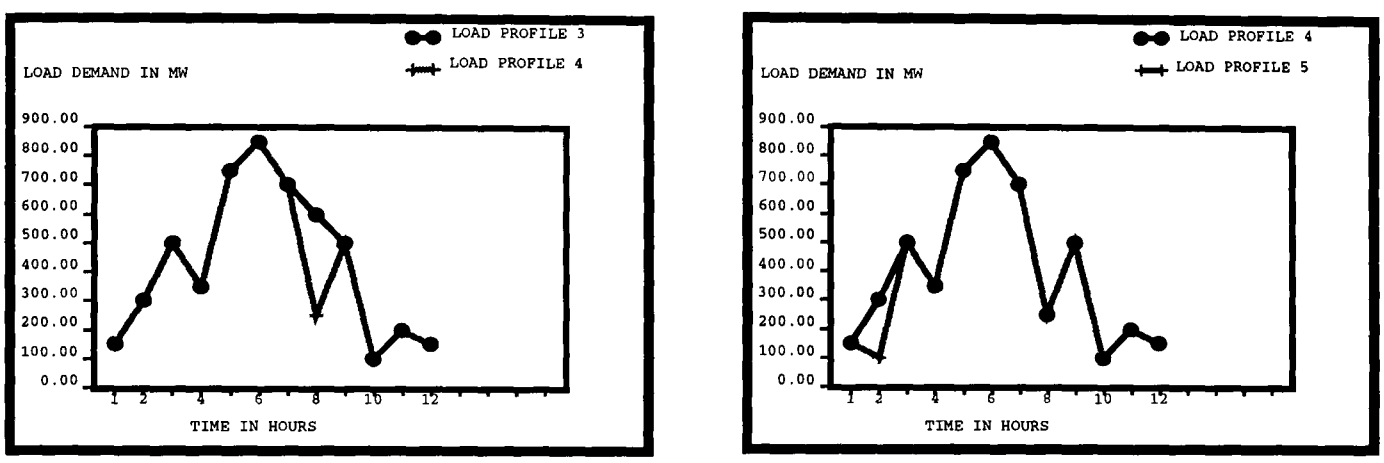



**Figure 6.6 Load profiles 1 to 3 (5 unit test system)**



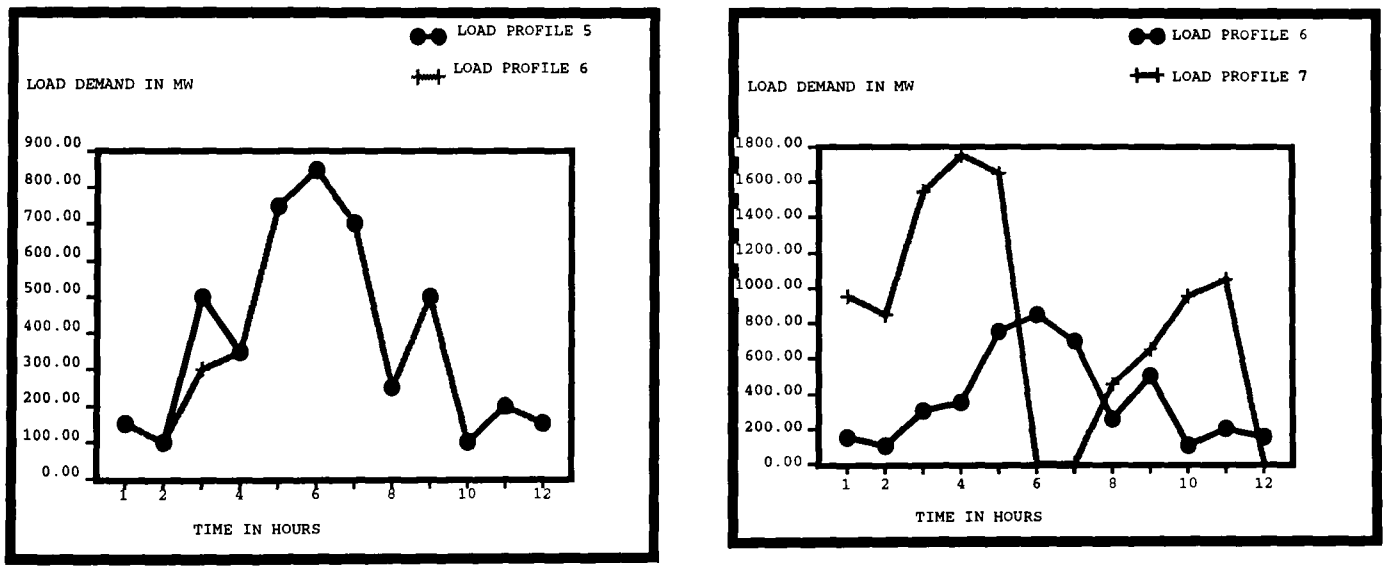




**Figure 6.7 Load profiles 4 to 7 (5 unit test system)**

## 6.7 Simulations and Results

Investigations were carried out to find out the suitability of GA methods in solving the thermal scheduling problem. The deterministic crowding and standard (canonical) GA models were applied to a number of thermal scheduling problems. All the thermal scheduling modelling, problem representations and fitness functions used for the genetic algorithms are based on the binary representation. Several experiments, that consumed several months of

cpu time, were carried out to determine the performance of the genetic algorithm methods for thermal scheduling on a number of power system test problems including 10, 26 100 and 110 unit tests systems. For clarity, only a sample of the test results that demonstrate some of the salient and fundamental features of performance are reported in the following sections. The programmes were written in FORTRAN 77 and all the tests were done on a DEC Alpha computer system (AXP 4610).

### 6.7.1 Results for 10 unit test system

A summary of the performance of the GA methods on a 10 unit test system [EPRI, 1994] is given in table 6.4.

**Table 6.4 Standard GA and DCGA scheduling costs (10 unit test system)**

| Trial | Production cost | | | |
|---|---|---|---|---|
| | DCGA | Hybrid DCGA | Standard GA | Hybrid Standard GA |
| 1 | 49,140 | 48,210 | 48,946 | 48,211 |
| 2 | 49,376 | 48,273 | 51,846 | 47,872 |
| 3 | 49,046 | 48,322 | 49,747 | 47,881 |
| 4 | 48,746 | 48,294 | 49,162 | 48,188 |
| 5 | 48,688 | 48,285 | 48,736 | 47,877 |
| Best | **48,688** | **48,210** | **48,736** | **47,872** |
| Average | 48,999 | 48,277 | 49,686 | 48006 |
| Standard deviation | 255 | 37 | 1129 | 158 |
| Average cpu time | 22 minutes | 12 minutes | 17 minutes | 9 minutes |
| % improvement by best over priority solution | 0.8% | 1.77% | 0.7% | 2.46% |
| % improvement by mean over priority solution | 0.16% | 1.63% | -1.24% | 2.19% |
| GA parameters | popsize=100, Pmut=0.004, generation =2500 ( 1,250 for hybrid method) | | popsize=100, pcross=1.0, Pmut=0.004, , Trunc. fitness scaling , 1 std. dev., elitism= 10%, generation =2500 ( 1,250 for hybrid method) | |
| Priority solution= **49, 079** | | | | |

Out of the trials reported in table 6.4, both the two GA methods returned a best solution which is slightly better than the priority list solution. On average, the DCGA method also provides better solutions than the priority list method while the mean standard GA solutions are worse than the priority list solution.

Both the hybrid standard GA and the hybrid DCGA methods provided significant improvements in performance over the priority list solution, with the standard GA providing better performance than the crowding GA, which is somewhat unexpected, since without hybridisation the DCGA usually out performs the standard GA method. It seems that the standard GA is better suited for performance improvement if a local search search or hybrid tech-

nique is to be used with the GA. The time to convergence in using the hybrid GA was also reduced by almost half, as half the number of generations were needed by the hybrid GA to reach convergence.

For a population size of 100, a run of more than 2500 generations did not provide any significant improvement in performance for either GA model. When the population size was doubled, the GA methods produced slightly poorer results than with the small population, as shown in table 6.5. This can be attributed to the fact that the larger population converges much more slowly and required more than 2500 generations to fully converge. Bearing in mind that the genetic algorithm can be used for all types of cost function representation, the results obtained for the 10 unit system by the GA methods are quite favourable for networks of this size.

**Table 6.5 Standard GA and DCGA results using a population size of 200 (10 unit test system)**

| Trial | Production cost | |
|---|---|---|
| | Crowding GA | Standard GA |
| 1 | 49,727 | 53,812 |
| 2 | 49,328 | 51,765 |
| 3 | 49,398 | 49,192 |
| 4 | 49,614 | 49,527 |
| 5 | 49,722 | 48,914 |
| Best | **49,328** | **48,914** |
| Average | 49,558 | 50,588 |
| Standard deviation | 166 | 1,913 |
| Average cpu time | 42 minutes | 33 minutes |
| % improvement of best over priority solution | -0.51% | +0.34% |
| % improvement of mean over priority solution | -0.98% | -3.07% |
| GA parameters | Pmut=0.004, generations =2500 | Pcross=1.0, Pmut=0.004, gen. =2500, Trunc. fitness scaling , 1 standard deviation, elitism= 10% |
| Priority solution= **49, 079** | | |

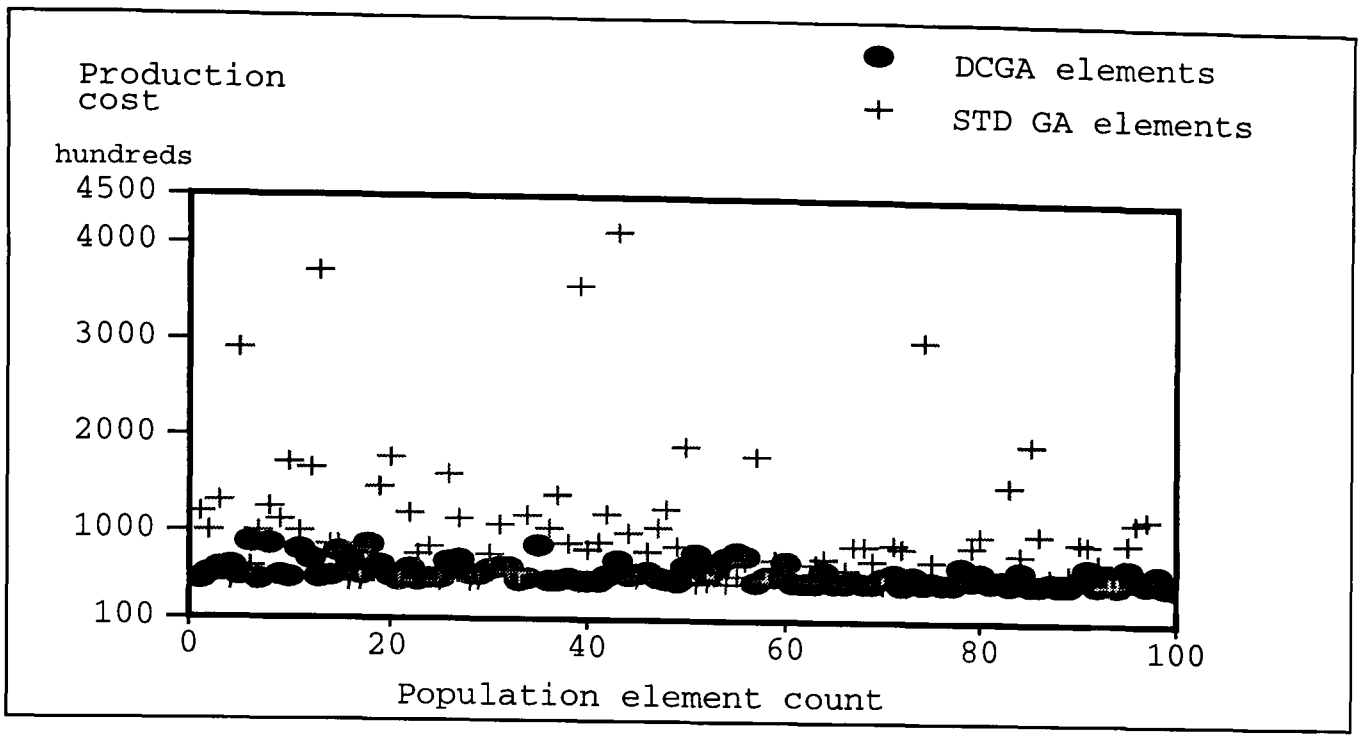



**Figure 6.8 The distribution of population elements in the final GA generations**

Figure 6.8 and 6.9 show the distribution of the population elements after a sample run of each GA algorithm on the 10 unit test system. The DCGA shows more elements having solutions closer to the optimal value than those of the standard GA, thus offering a wider choice of near optimal solutions that can be selected for final use, depending on the decision makers preferences.

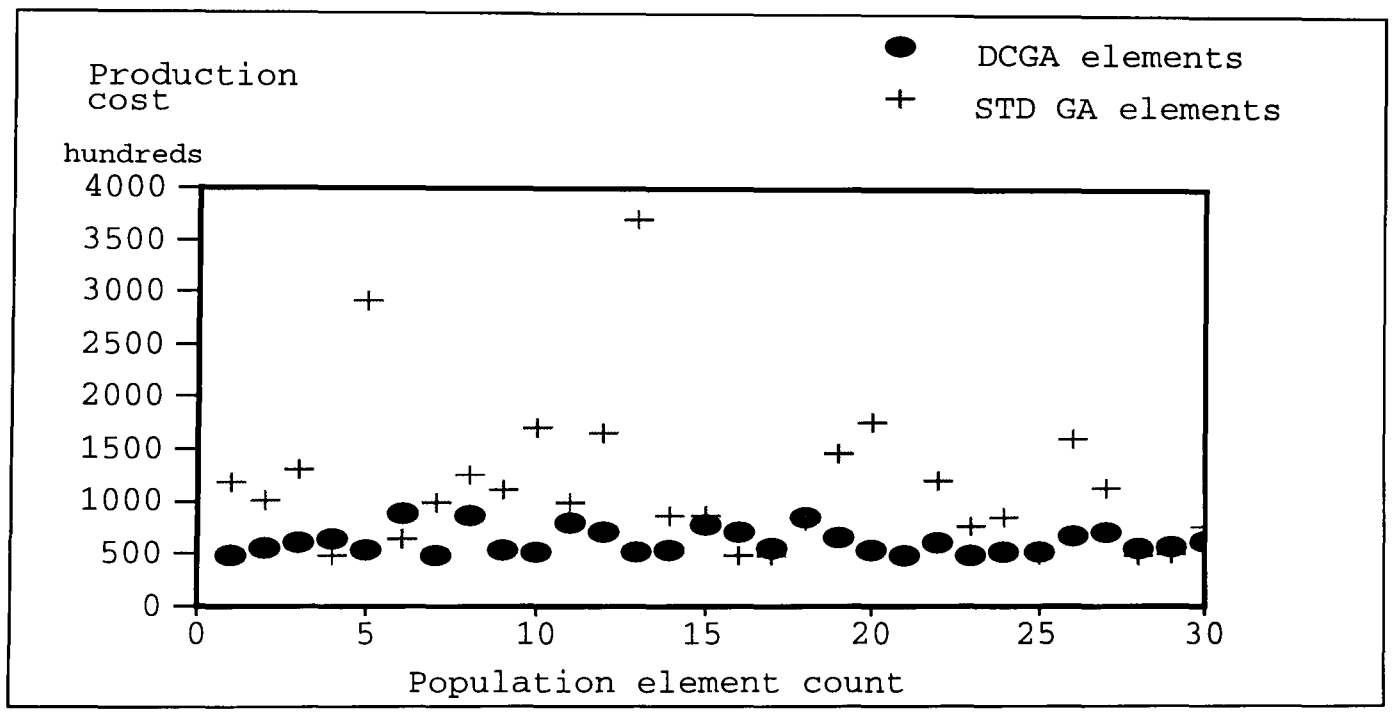



**Figure 6.9 The random distribution of 30 (out of 100) population elements in the final GA generations**

### 6.7.2 Results for 26 unit test system

A test system consisting of 26 units [Wang and Shahidepour] was considered a good representative of medium sized power systems. This system has a complete data suite necessary

for testing the performance of a scheduling method. It provides unit ramp rate limits as well as two widely differing load demand profiles which are quite appropriate for test purposes.

**Table 6.6 Standard GA and DCGA results using a population size of 100, 2500 generations (26 unit test system, load profile_1)**

| Trial | Production cost | | | |
|---|---|---|---|---|
| | DCGA | Hybrid DCGA | Standard GA | Hybrid Standard GA |
| 1 | 733,487 | 724,043 | 768,986 | 723,884 |
| 2 | 734,247 | 724,355 | 749,133 | 723,790 |
| 3 | 728,816 | 724,035 | 773,923 | 723,906 |
| 4 | 734,003 | 724,355 | 751,498 | 724,133 |
| 5 | 731,287 | 724,121 | 753,163 | 724,034 |
| Best | **728,816** | **724,035** | **749,133** | **723,949** |
| Mean | 732,368 | 724,182 | 759,341 | 723,884 |
| Standard deviation | 2061 | 145 | 10,095 | 120 |
| Average cpu time | 80 minutes | 16 minutes | 51 minutes | 10 minutes |
| % improvement of best over priority solution | -0.22% | 0.44% | -3.02% | 0.46% |
| % improvement of mean over priority solution | -0.71% | 0.42% | -4.42% | 0.46% |
| GA parameters | popsize=100, Pmut=0.0016, generations =2500 ( 500 for hybrid method) | | popsize=100, pcross=1.0, Pmut=0.0016, Trunc. fitness scaling , 1 standard deviation, elitism= 10%, generations=2500 ( 500 for hybrid method) | |
| Priority solution = **727,200** | | | | |

From the results indicated in table 6.6, it can be seen that without applying a hybrid technique, both the GA models perform less well than the simple priority list unit commitment method. However with the hybrid technique, both models produce significant performance improvements over the priority list solution with much reduced computation times relative to the standard GA and DCGA models.

**Table 6.7 Standard GA and DCGA results using a population size of 100, 5000 generations, load profile_1**

| Trial | Production cost | |
|---|---|---|
| | DCGA | Standard GA |
| 1 | 731,853 | 753,120 |
| 2 | 725,480 | 786,260 |
| 3 | 728,850 | 776,317 |
| 4 | 728,523 | 768,521 |
| 5 | 729,949 | 735,768 |
| Best | **725,480** | **735,768** |
| Mean | 728,931 | 761,997 |
| % improvement of best over priority solution | 0.24% | -1.18% |
| % improvement of mean over priority solution | -0.23% | -4.79% |
| Average cpu time | 2 hrs. 30 min. | 1 hr. 45 min. |

Increasing the number of generations of the DCGA to 5000 provides a small improvement in the solution quality with a doubling in computation time. An increase in the allowed number of generations does not, in general, improve the performance of the standard GA method which seems to prematurely converge by generation 2500, as indicated by the results in table 6.7. Thus increasing the number of generations of the GA runs beyond a certain threshold does not result in any substantial improvements in solution quality.

**Table 6.8 Standard GA and DCGA results using a population size of 624, for various final generations of run (load profile_1)**

| Trial | generations of GA run | Crowding GA | | | Standard GA | | |
|---|---|---|---|---|---|---|---|
| | | Avg. cpu time (hrs: min.) | cost | % improvement over priority soln. | Avg. cpu time (hrs: min.) | cost | % improvement over priority soln. |
| 1 | 1000 | 3:20 | 747,226 | -2.75 | 2:13 | 733,588 | -0.88 |
| 2 | 2500 | 9:05 | 734,131 | -0.95 | 5:20 | 733,374 | -0.85 |
| 3 | 5000 | 16:08 | 725,998 | 0.17 | 10:58 | 732,859 | -0.78 |
| Priority list solution **727,200** | | | | | | | |

Each run in table 6.8 had a different random initial starting population. The population size was set equal to the binary string length and each run terminated after a set number of generations and the final solution obtained noted. For the higher population size, the deterministic crowding model provides some improvement in solution but the computation time used is much higher. For example, a population size of 624 run for 5000 generations takes 16 hours of cpu time to provide a 0.17% in improvement over the priority list solution, while a population size of 100 run for 5000 generations provides a 0.24% in improvement in 2.5 hours of cpu time, showing that an increase in population size does not necessarily result

in improved performance, although it should be noted that small percentage improvements in production cost can be very significant in financial terms, since the total production cost may be very large. The increase in population size for the standard GA provides some improvements in the performance as indicated by the results in tables 6.7 and 6.8. The convergence characteristics for the genetic algorithms using a population size of 624 is shown in figures 6.10 and 6.11.

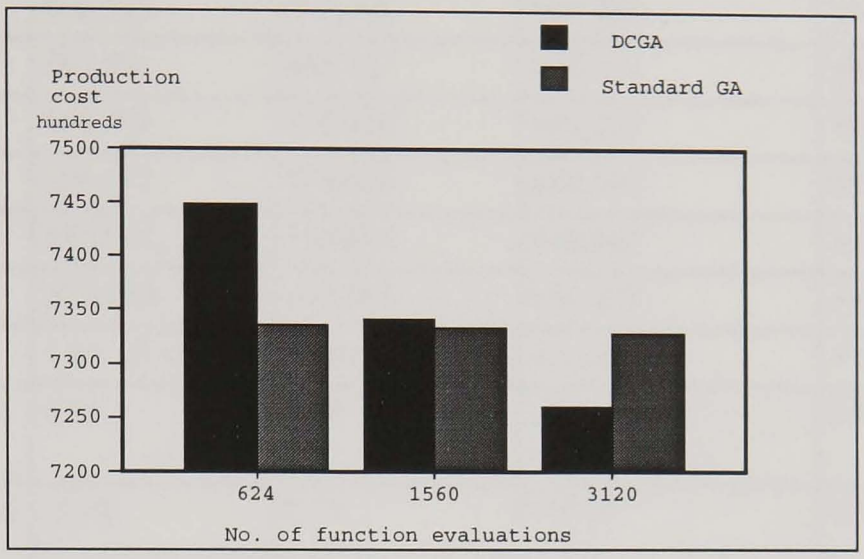



**Figure 6.10 Variation of scheduling cost with function evaluations (for a population size of 624).**

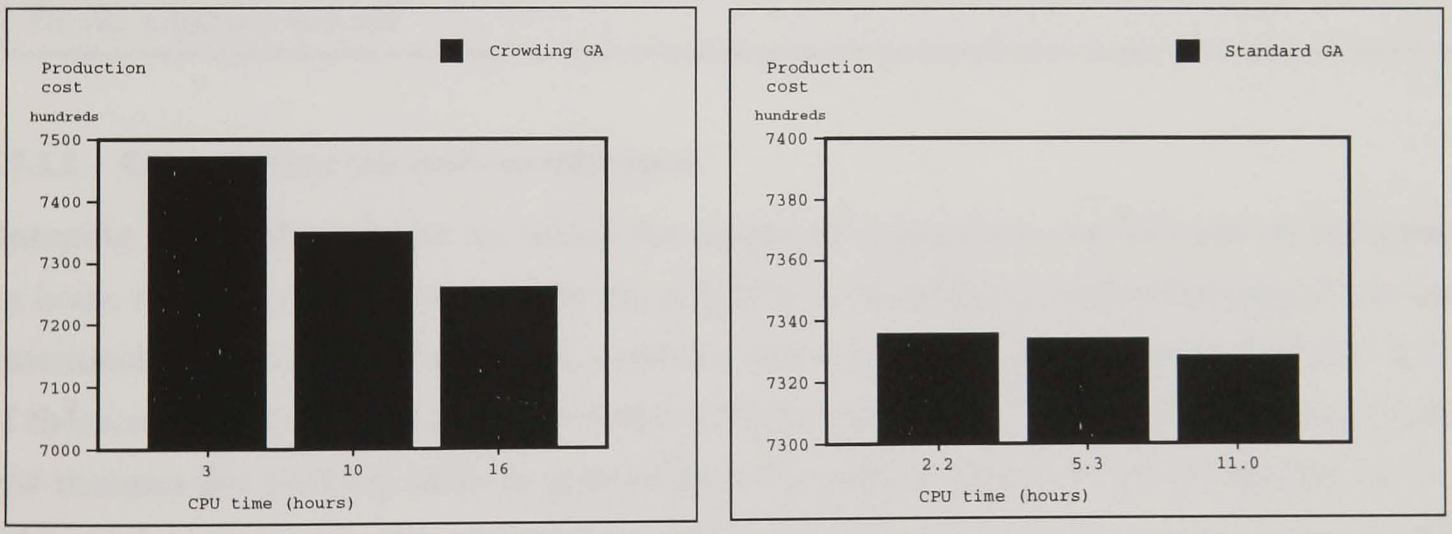



**Figure 6.11 Variation of scheduling costs with computation time**

### 6.7.2.1 Effect of load variation on scheduling results

Using a different load demand pattern can have a great influence on the scheduling results. The effect a variation in the load demand pattern on the scheduling results was investigated for the 26 unit test system. The results are presented in table 6.9.

**Table 6.9 Standard GA and DCGA results using a population size of 100, 2500 generations (26 unit test system, load profile_2)**

| Trial | Production cost | | | |
|---|---|---|---|---|
| | DCGA | Hybrid DCGA | Standard GA | Hybrid Standard GA |
| 1 | 616,712 | 592,969 | 672,118 | 592,152 |
| 2 | 600,515 | 592,614 | 661,246 | 594,094 |
| 3 | 593,591 | 592,982 | 646,104 | 593,017 |
| 4 | 608,088 | 592,890 | 648,257 | 591,666 |
| 5 | 606,242 | 593,050 | 669,290 | 592,440 |
| Best | **593,591** | **592,614** | **646,104** | **591,666** |
| Average | 605,030 | 592,901 | 659,403 | 592,674 |
| Average cpu time | 1 hr: 10 min. | 13 min. | 43 min. | 8 min. |
| % improvement of best over priority solution | 0.61 | 0.78 | -8.18 | 0.94 |
| % improvement of mean over priority solution | -1.30 | 0.73 | -10.41 | 0.77 |
| GA parameters | popsize=100, Pmut=0.0016, generations =2500 ( 500 for hybrid GA) | | popsize=100, pcross=1.0, Pmut=0.0016, Trunc. fitness scaling , 1 std. dev., elitism= 10%, generations =2500 (500 for hybrid GA) | |
| Priority solution = **597,254** | | | | |

#### 6.7.2.2 Effect of ramp rate limit considerations

Ramping limits, the amount by which the output of a generator can increase or decrease in an hour, is usually not considered in the majority of scheduling studies because of the computational difficulties it imposes on conventional techniques. Many factors limit the ability of thermal generators to ramp up or down, and the time to ramp up or down can vary from a few minutes for peaking units to several days for nuclear plants. If ramp rates are not considered, thus assuming the system generators can instantly pick up or drop loads, a utility may not be able to meet its generation requirements.

In a GA using full binary string representation, ramp rate limits can easily be taken into consideration. The binary coding represents the possible solution over the whole scheduling period, and there is therefore no need for *a look ahead* logic usually adopted by a number of methods to guarantee future ramp rate limits are not violated. In this work, the effect of ramp rate limits on the scheduling solution has been taken into account by including ramping logic into the GA problem formulation. The ramp up logic calculates the maximum generation that a unit can achieve as a function of time after start up, while the ramp down logic uses the number of hours to the next shutdown to determine the units maximum load-

ing capability. Both the resulting loading levels are used to calculate the maximum amount any unit can contribute to generation and reserve requirements.

#### 6.7.2.3 Demonstrating the importance of ramp limit consideration in generation scheduling

The scheduling results of the priority list method (without considering ramp limits) shown in table 6.10 was analysed for the effect of ramp considerations. For example at hour 8 and 9, units 22 and 23 are limited by their ramp up rate limits and cannot be loaded to their full capacities. Assuming that all units are initially loaded at their minimum allowable loading levels when synchronised, the maximum loading capability of the two units during hours 8 and 9 are 123.95 MW and 178.95 MW respectively, given their minimum loading and ramp up limits to be as shown on table 6.11. These maximum loading capability limitations result in a schedule that does not satisfy the 400 MW reserve requirements as shown in table 6.12.

**Table 6.10 Effect of ramp limits on priority list unit commitment solution**

THE SCHEDULE CREATED BY THE GENETIC ALGORITHM

| HOUR | UNITS (1 ~ 26) |
|---|---|
| 1 | 00000000011111001111000111 |
| 2 | 00000000011111001111000111 |
| 3 | 00000000011101001111000111 |
| 4 | 00000000011101101111000111 |
| 5 | 00000000011101101111000111 |
| 6 | 00000000011111101111000111 |
| 7 | 00000000011111111111100111 |
| 8 | 00000000011111111111111111 |
| 9 | 00000000011111111111111111 |
| 10 | 11100000011111111111111111 |
| 11 | 11111111011111111111111111 |
| 12 | 11100000011111111111111111 |
| 13 | 11100000011111111111111111 |
| 14 | 00000000011111111111111111 |
| 15 | 11111000011111111111111111 |
| 16 | 11111110011111111111111111 |
| 17 | 00000000011111111111111111 |
| 18 | 00000000011111111111111111 |
| 19 | 00000000011111111111111111 |
| 20 | 00000000011111111111111111 |
| 21 | 11100000011111111111111111 |
| 22 | 00000000011101111111111111 |
| 23 | 00000000011001001111111111 |
| 24 | 00000000011001001111111111 |

**Table 6.11 Plant ramp rates**

Plant parameters and initial conditions

| Unit | Pgen (MW) | Pmin (MW) | Ramp Up (MW / hour) | Ramp Down (MW / hour) |
|---|---|---|---|---|
| . | . | . | . | . |
| . | . | . | . | . |
| 22 | 197 | 68.95 | 55.0 | 99.0 |
| 23 | 197 | 68.95 | 55.0 | 99.0 |
| . | . | . | . | . |
| 26 | 400 | 100.0 | 50.5 | 100.0 |

**Table 6.12 Power generation and capability limits for ramp constrained scheduling**

| HR | PGEN | PDEM | RES | PSTCOST | PSDCOST | FTRCOST |
|---|---|---|---|---|---|---|
| 1 | 2150.0 | 1700.0 | 450.0 | 0.0 | 0.0 | 0.0 |
| 2 | 2174.0 | 1730.0 | 444.0 | 0.0 | 0.0 | 0.0 |
| 3 | 2098.0 | 1690.0 | 408.0 | 0.0 | 0.0 | 0.0 |
| 4 | 2174.0 | 1700.0 | 474.0 | 0.0 | 0.0 | 0.0 |
| 5 | 2198.0 | 1750.0 | 448.0 | 0.0 | 0.0 | 0.0 |
| 6 | 2251.7 | 1850.0 | 401.7 | 0.0 | 0.0 | 0.0 |
| 7 | 2473.9 | 2000.0 | 473.9 | 0.0 | 0.0 | 0.0 |
| 8 | 2800.8 | 2430.0 | **370.8** | 0.0 | 0.0 | **50741.9** |
| 9 | 2928.9 | 2540.0 | **388.9** | 0.0 | 0.0 | **50015.4** |
| 10 | 3013.0 | 2600.0 | 413.0 | 0.0 | 0.0 | 0.0 |
| 11 | 3097.0 | 2670.0 | 427.0 | 0.0 | 0.0 | 0.0 |
| 12 | 3013.0 | 2590.0 | 423.0 | 0.0 | 0.0 | 0.0 |
| 13 | 3013.0 | 2590.0 | 423.0 | 0.0 | 0.0 | 0.0 |
| 14 | 2965.0 | 2550.0 | 415.0 | 0.0 | 0.0 | 0.0 |
| 15 | 3037.0 | 2620.0 | 417.0 | 0.0 | 0.0 | 0.0 |
| 16 | 3077.0 | 2650.0 | 427.0 | 0.0 | 0.0 | 0.0 |
| 17 | 2965.0 | 2550.0 | 415.0 | 0.0 | 0.0 | 0.0 |
| 18 | 2965.0 | 2530.0 | 435.0 | 0.0 | 0.0 | 0.0 |
| 19 | 2965.0 | 2500.0 | 465.0 | 0.0 | 0.0 | 0.0 |
| 20 | 2965.0 | 2550.0 | 415.0 | 0.0 | 0.0 | 0.0 |
| 21 | 3013.0 | 2600.0 | 413.0 | 0.0 | 0.0 | 0.0 |
| 22 | 2887.0 | 2480.0 | 407.0 | 0.0 | 0.0 | 0.0 |
| 23 | 2613.0 | 2200.0 | 413.0 | 0.0 | 0.0 | 0.0 |
| 24 | 2613.0 | 1840.0 | 773.0 | 0.0 | 0.0 | 0.0 |

Where PGEN, PDEM, RES, PSTCOST, PSDCOST, FTRCOST are the maximum system generation capacity, load demand, system reserve, premature startup / shut down costs and failure to meet reserve penalties respectively.

The scheduling results that include the effect of ramp rate limitations for the 26 unit test system for the two load demand profiles are presented in tables 6.12 and 6.13 and are compared with those obtained in the study by [Wang and Shahidepour].

**Table 6.13 Standard GA and DCGA results with ramp rate limits considered, (26 unit test system, load profile_1)**

| Trial | Production cost | | | |
|---|---|---|---|---|
| | DCGA | Hybrid DCGA | Standard GA | Hybrid Standard GA |
| 1 | 733,506 | 725,100 | 777,435 | 725,333 |
| 2 | 732,003 | 725,021 | 736,807 | 725,651 |
| 3 | 729,146 | 724,459 | 765,417 | 724,287 |
| 4 | 728,868 | 725,137 | 748,002 | 725,230 |
| 5 | 732,177 | 724,720 | 761,344 | 724,851 |
| Best | **728,868** | **724,459** | **736,807** | **724,287** |
| mean | 731,140 | 724,887 | 757,801 | 725,070 |
| Standard deviation | 1820 | 260 | 14,091 | 468 |
| Average cpu time | 1 hr 35 min. | 20 min. | 57 min. | 11 min. |
| % improvement of best over ANN/DP method | 0.46 | 1.06 | -0.62 | 1.09 |
| % improvement of mean over ANN/DP method | 0.15 | 1.01 | -3.49 | 0.98 |
| GA parameters | popsize=100, Pmut=0.0016, generations =2500 (500 for hybrid method) | | popsize=100, pcross=1.0, Pmut=0.0016, generations =2500 (500 for hybrid method), Truncation fitness scaling , 1 std. deviation, elitism= 10% | |
| ANN/DP [Wang /Shahidepour] solution = **732,251** | | | | |

**Table 6.14 DCGA results with ramp rate limits considered, ( 26 unit test system, load profile_2)**

| Hybrid DCGA | |
|---|---|
| **Trial** | **Production cost** |
| 1 | 594,398 |
| 2 | 593,674 |
| 3 | 594,381 |
| 4 | 594,660 |
| 5 | 594,785 |
| Best | **593,674** |
| mean | 594,380 |
| Average cpu time | 15 minutes |
| % Improv. of best over ANN/DP method | 3.75 |
| % Improv. of mean over ANN/DP method | 3.63 |
| GA parameters | popsize=100, Pmut=0.0016, generations =500 |
| ANN/DP [Wang /Shahidepour] solution = **616,793** | |

### 6.7.3 Results for 110 unit test system

A test system consisting of 110 units [Orero and Irving, 1995] was considered a good representation of large scale power systems. Both the standard GA and the crowding GA scheduling methods were run until a nearly converged solution was obtained. The results in table 6.14 show the average solutions of 10 trials for each set number of generations of a GA. The convergence characteristics are shown in figure 6.12. It can be seen that for this large scale system, both the GA methods converge to solutions which are very far from optimal. The deterministic crowding GA converges to a mean solution which is 22.4% worse than the simple priority list solution while the standard GA converges to a solution which is an incredible 379% worse than the priority list solution. These results demonstrate the inability of the GA models in solving the thermal scheduling problem in its basic raw formulation for large scale systems.

**Table 6.15 Standard GA and DCGA results for 110 unit test system**

| Generations of GA run | Average production cost x $10^7$ | |
|---|---|---|
| | Standard GA | Crowding GA |
| 200 | 6.933 | 9.534 |
| 400 | 3.071 | 3.217 |
| 600 | 2.404 | 1.524 |
| 800 | 2.107 | 1.026 |
| 1000 | 1.998 | 0.811 |
| 1500 | 1.902 | 0.547 |
| 2000 | 1.887 | 0.501 |
| 2500 | 1.886 | 0.492 |
| 3000 | 1.886 | 0.482 |
| 3500 | 1.851 | 0.480 |
| 4000 | 1.850 | 0.476 |
| 4500 | 1.850 | 0.475 |
| 5000 | 1.847 | 0.472 |
| % improvement of mean final solution over priority solution | -379% | -22.4% |
| Total cpu time (hr: min.) | 6:55 | 31:45 |
| GA parameters | population=110, pcross=1.0, Pmut.=0.0004, Elite copies=10, stochastic remainder selection, uniform crossover | population=110, Pmut.=0.0004, uniform crossover |

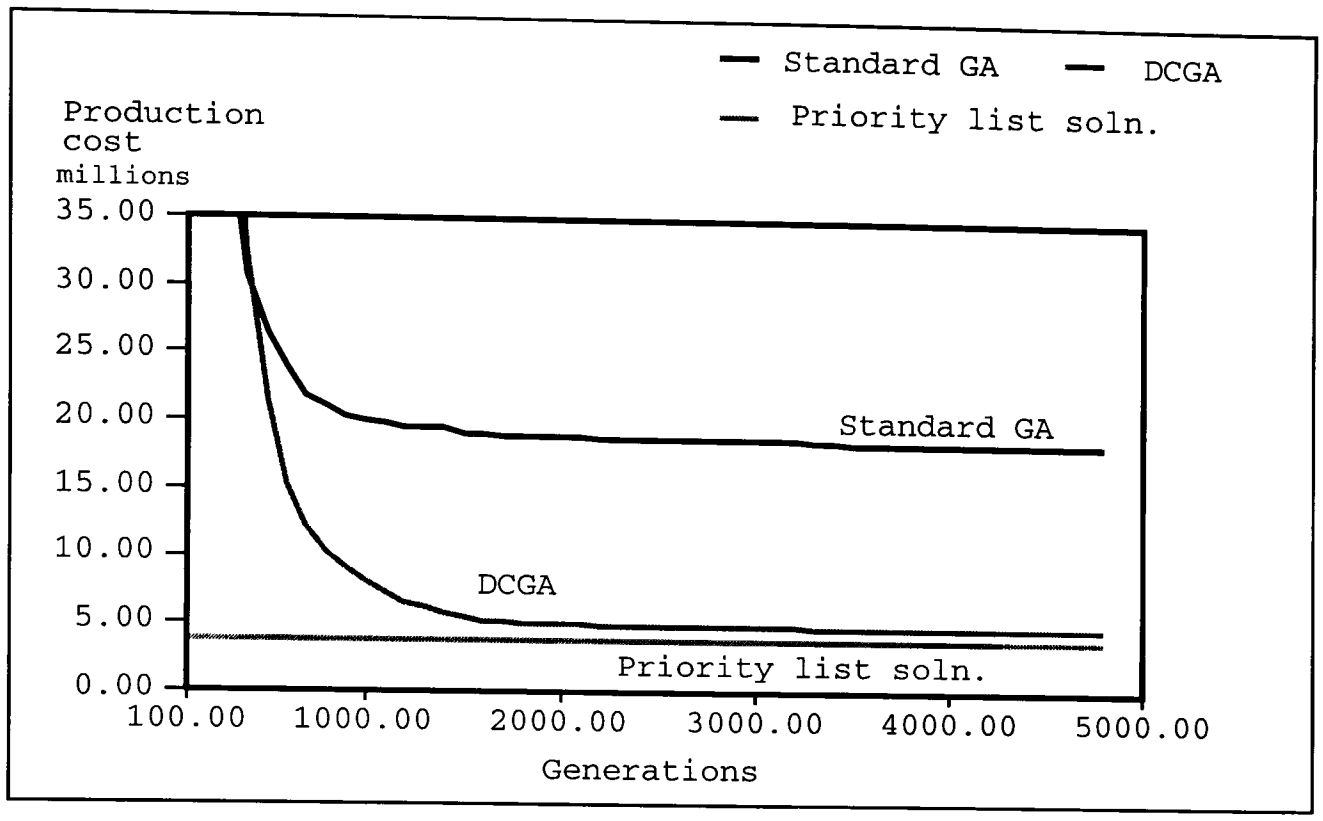



**Figure 6.12 Convergence characteristics of standard GA and DCGA (110 unit test system).**

**Table 6.16 Hybrid standard GA and DCGA results for 110 unit test system**

| Generations of GA run | Standard GA | | | Crowding GA | | |
|---|---|---|---|---|---|---|
| | Average Production cost | % Improv. on priority list soln. | Cpu time (hr: min.) | Average Production cost | % Improv. on priority list soln. | Cpu time (hr: min.) |
| 100 | 3,846,127 | 0.226 | 0:20 | 3,851,908 | 0.076 | 0:25 |
| 200 | 3,840,681 | 0.367 | 0:40 | 3,844,144 | 0.277 | 1:03 |
| 300 | 3,838,877 | 0.414 | 0:59 | 3,840,010 | 0.384 | 1:40 |
| 400 | 3,837,168 | 0.455 | 1:19 | 3,837,614 | 0.446 | 2:20 |
| 500 | 3,836,327 | 0.480 | 1:38 | 3,836,618 | 0.472 | 2:59 |
| 600 | 3,835,930 | 0.490 | 2:01 | 3,836,159 | 0.484 | 3:38 |
| 700 | 3,835,691 | 0.496 | 2:20 | 3,835,991 | 0.488 | 4:17 |
| 800 | 3,835,128 | 0.510 | 2:40 | 3,835,885 | 0.491 | 4.56 |
| 900 | 3,835,105 | 0.511 | 3:00 | 3,835,818 | 0.493 | 5:35 |
| 1000 | 3,834,986 | 0.512 | 3:20 | 3,835,507 | 0.50 | 6:15 |
| GA parameters: | population=110, pcross=1.0, Pmut=0.0004, Elite copies=10, stochastic remainder selection, uniform crossover | | | population=110, Pmut=0.0004, uniform crossover | | |

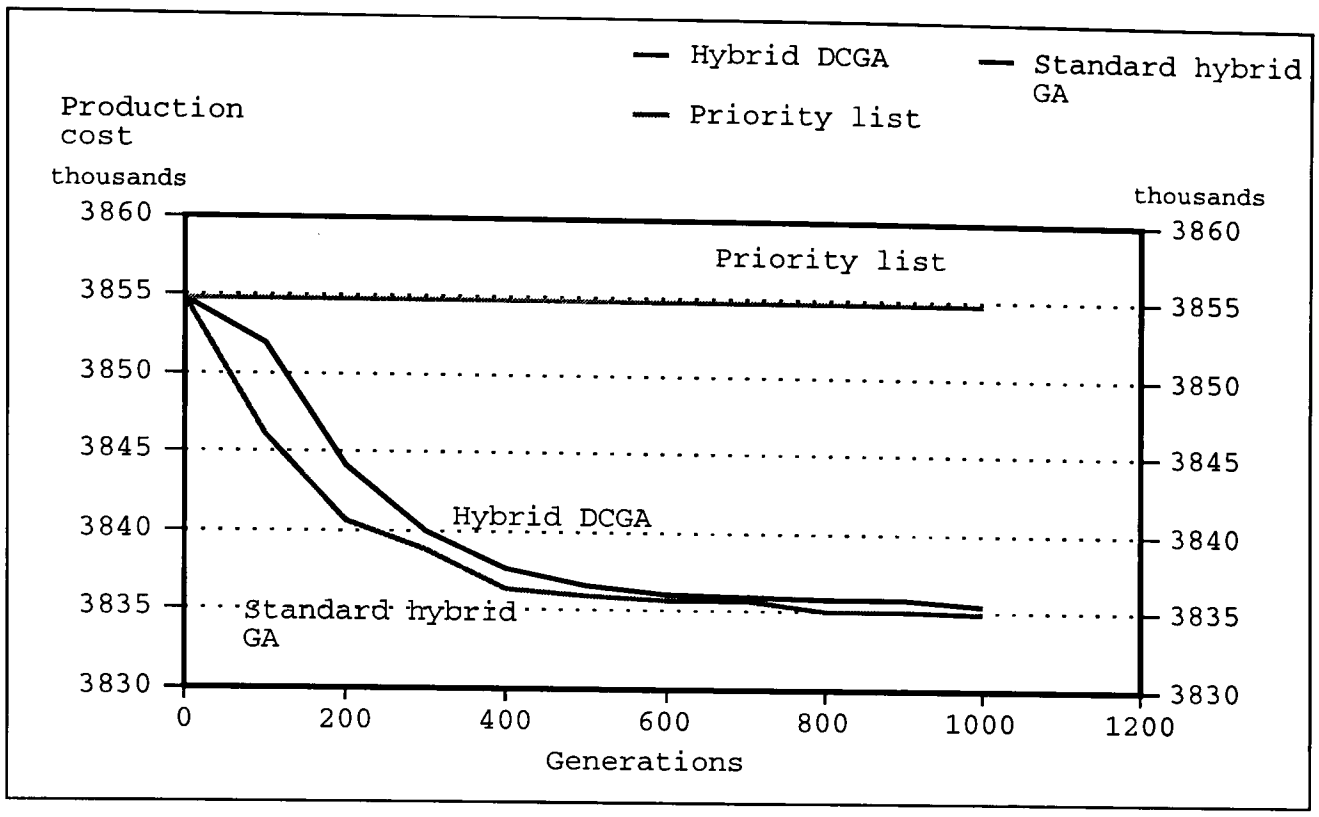



**Figure 6.13 Convergence characteristics of hybrid DCGA and hybrid standard GA (110 unit test system).**

#### 6.7.3.1 Sample hybrid standard GA scheduling results for the 110 unit test system

The simulation conditions were: uniform cross over, random number seed = 27362121, population size = 110, number of generations of run = 1000, Pcross = 1.00 Pmut = 0.0004, number of scheduling hours = 24 number of units =110, chromosome length = 2640, lambda - iterative dispatch method.

**Figure 6.14 A standard hybrid GA unit commitment list (110 unit system)**

```
HOUR                          UNITS ( 1 ~ 110)
 1  00000000011111001111000111101001100111011010110000000001111000111100011100011010011101110110101100001111010100
 2  00000000011111001111000111101000100111011010110000000000100000111100011100111010001101000110101100001110010000
 3  00000000000000100100000011110100010110001101111000000000000000001111000111001110100001110001101111000011100000
 4  00000000000001001000000111101000101100011011110000000000000000001111000111001110100001110001101111000000000000
 5  00000000000001001000000111101000111000111011110000000000000000001111000111011110100001110001101111000000000000
 6  00000000000001001111000111111000111100011011110000000000000000001111000111011110100001110001101111000010000000
 7  00000010000001001111000111111000111101011011110000000000000000001111000111111111100001110001101111000010000000
 8  00000000000001001111000111111000111101111011110000000000000000001111000111111111100001110001101111000010000000
 9  00000000000001001111000111111000111101111111110000000000000000001111000111111111100001110001101111000000000000
10  00000000011111001111000111111100111101111111110000000001000000111100011111111110000111001110111100000000000000
11  00000000011111001111000111111100111101111111110000000001100000111100011111111110000111001110111100000000000000
12  00000000011110001111000111111000011101111111110000000001100000111100011111111110000111000010111100000000000000
13  00000000000000001111000111111000011101111111000000000001000001111000111111111100001110000101111000000000000000
14  00000000000000001111000111111000011101001111100000000000000001111000111111111100001110000001111000000000000000
15  00000000011110001111000111111100111111011111110000000001011000111100011111111110000111001100111100000000000000
16  00000000011110001111000111111100111111111111110000000001011000111100011111111110000111001100111100000000000000
17  00000000011110001111000111111100111101111111110000000001011000111100011111011110000111001110111100000000000000
18  00000000011110001111000111111100111111111111110000000001111000111100011111011110000111001110111100000000000000
19  00000000011111111111000111111100111111111111110000000001111000111100011111111110111111111111111100001100000000
20  00000000011111111111000111111101111111111111110000000001111111111100011111111110111111111111111100001111000000
21  00000000011111111111101111111001111111111111100000000001111111111100011111111110111111111111111100001111000000
22  00000000011111111111101111111000111111111111100000000000001111111000111111111100001110011101111000000110000000
23  00000000011110001111101111110000111111111111100000000000001111111000111111111100001110000101111000000110000000
24  00000000010100001111101111110000111011111111100000000000000001111000111111101100001110000101111000000000000000
PRODUCTION COST = 3,771,638
```

This sample result obtained from the standard hybrid GA run was the best result obtained for the scheduling of the 110 unit test system. It provides solutions which are 2.16% better than the priority list solution (of 3,854, 821). The GA can sometimes produce some very

good solutions as demonstrated by this sample result, a fact that might not be clearly brought out by the tabulated results, most of which are average values.

**Table 6.17 Total hourly capacity output, load demand and penalty coefficients for the schedules in table 6.15**

| HR | PGEN | PDEM | RES | PSTCOST | PSDCOST | FTRCOST |
|---|---|---|---|---|---|---|
| 1 | 12316.0 | 11600.0 | 716.0 | 0.0 | 0.0 | 0.0 |
| 2 | 11690.0 | 10900.0 | 790.0 | 0.0 | 0.0 | 0.0 |
| 3 | 12100.0 | 9500.0 | 2600.0 | 0.0 | 0.0 | 0.0 |
| 4 | 11860.0 | 9300.0 | 2560.0 | 0.0 | 0.0 | 0.0 |
| 5 | 12360.0 | 10500.0 | 1860.0 | 0.0 | 0.0 | 0.0 |
| 6 | 13385.0 | 11200.0 | 2185.0 | 0.0 | 0.0 | 0.0 |
| 7 | 14375.0 | 12500.0 | 1875.0 | 0.0 | 0.0 | 0.0 |
| 8 | 14455.0 | 12900.0 | 1555.0 | 0.0 | 0.0 | 0.0 |
| 9 | 14615.0 | 13500.0 | 1115.0 | 0.0 | 0.0 | 0.0 |
| 10 | 15215.0 | 14500.0 | 715.0 | 0.0 | 0.0 | 0.0 |
| 11 | 15311.0 | 14600.0 | 711.0 | 0.0 | 0.0 | 0.0 |
| 12 | 14711.0 | 14000.0 | 711.0 | 0.0 | 0.0 | 0.0 |
| 13 | 14191.0 | 13200.0 | 991.0 | 0.0 | 0.0 | 0.0 |
| 14 | 13855.0 | 13000.0 | 855.0 | 0.0 | 0.0 | 0.0 |
| 15 | 15235.0 | 14500.0 | 735.0 | 0.0 | 0.0 | 0.0 |
| 16 | 15335.0 | 14600.0 | 735.0 | 0.0 | 0.0 | 0.0 |
| 17 | 15345.0 | 14000.0 | 1345.0 | 0.0 | 0.0 | 0.0 |
| 18 | 15481.0 | 14700.0 | 781.0 | 0.0 | 0.0 | 0.0 |
| 19 | 16321.0 | 15600.0 | 721.0 | 0.0 | 0.0 | 0.0 |
| 20 | 16921.0 | 16200.0 | 721.0 | 0.0 | 0.0 | 0.0 |
| 21 | 17315.0 | 16500.0 | 815.0 | 0.0 | 0.0 | 0.0 |
| 22 | 16253.0 | 15000.0 | 1253.0 | 0.0 | 0.0 | 0.0 |
| 23 | 15533.0 | 14300.0 | 1233.0 | 0.0 | 0.0 | 0.0 |
| 24 | 14311.0 | 13500.0 | 811.0 | 0.0 | 0.0 | 0.0 |

## 6.8 Conclusions

Table 6.18 and 6.19 give a general summary of the performance the two GA thermal scheduling methods on the various power system test problems.

**Table 6.18 Comparative performance of solution quality across test systems**

| Test system | Percentage improvement over priority list method | | | |
|---|---|---|---|---|
| | Without Hybrid | | Hybrid | |
| | Standard GA | DCGA | Standard GA | DCGA |
| 10 unit | 0.70 | 0.80 | 2.46 | 1.77 |
| 26 unit (load profile_1) | -3.02 | -0.22 | 0.46 | 0.44 |
| 26 unit (load profile_2) | -8.18 | 0.61 | 0.94 | 0.78 |
| 110 unit | -379 | -22.4 | 0.51 | 0.50 |

**Table 6.19 Comparison of average computation times across test systems and solution techniques**

| Test system | (hour: minutes) | | | |
|---|---|---|---|---|
| | No hybrid | | Hybrid | |
| | Standard GA | Crowding GA | Standard GA | Crowding GA |
| 10 unit | 0:17 | 0:22 | 0:09 | 0:12 |
| 26 unit | 0:46 | 1:15 | 0:10 | 0:14 |
| 110 unit | 6:55 | 31:45 | 2:40 | 6:15 |

From table 6.18, it can be seen that the larger the test system, the greater the challenge offered to the GA method. Apart from the system size, scheduling problem difficulty is also dependent on the system load curve as well as the unit input / output characteristics. One of the early conclusions reached from the empirical analysis of the standard GA performance was that convergence to a good solution was difficult to achieve, especially as the system sizes were scaled up. This was mostly due to premature convergence. Even basic modifications to the algorithm such as elitism, multiple restarts and fitness scaling did not provide substantial improvements in performance. This situation motivated the search for a better GA algorithm structure, resulting in the development of the deterministic crowding genetic algorithm model. The deterministic crowding model provided a superior performance to the standard genetic algorithm, but also suffered from the problem of long computation time and premature convergence as the problem size was scaled up.

The performance of both the GA models was greatly enhanced by the use of a hybrid GA that uses the priority list unit commitment scheme. However as the system sizes increased, this hybrid solution technique too suffered from the problem of long computation times and premature convergence. Two main possibilities for realising further improvements in performance included:

1. decomposing the thermal scheduling problem in some innovative manner, and
2. fine tuning the GA parameter settings.

The second approach had been tried in a number of experiments and although it resulted in some improvements in performance, it was not considered to be the ultimate solution to the problem of the long computation times or premature convergence, hence the more promising approach of thermal scheduling problem decomposition is investigated in the next chapter.

# Chapter 7. A Genetic Algorithm Solution of the Decomposed Thermal Scheduling Problem

## 7.1 Introduction

Often a difficult problem can be decomposed into several simpler sub problems, each of which may be more easily solved than the complete problem. Even when the problem cannot be completely partitioned into smaller sub components, advantage can still be taken of the known sub components to aid in the overall solution of the main problem. Goldberg, [Goldberg, 1993] likens the design of genetic algorithms for practical applications to one of the well known examples of the benefits of problem decomposition in the design of the first aeroplane to fly successfully by the Wright brothers [Bradshaw and Lienert]. The Wright brothers basically decomposed the large complex problem of aeroplane design, by intuitively breaking it into quasi-separate sub problems and then investigating each sub problem separately using simpler models, before recombining them to form the whole entity. A similar approach is proposed to decompose the thermal scheduling problem. One of the major problems with generator scheduling is the large size of the search space and the accompanying constraints. For scheduling a total of N units over time period T, solving N separate problems, one for each unit would be much simpler, but this is not possible because of the unit and system constraints linking the generators. In order to alleviate this difficulty, even with a GA, a method must be sought for decomposing the problem.

## 7.2 Decomposition Method

In the basic thermal scheduling binary problem representation, the total solution string length is a product of the scheduling period, T and the number of generating units, N. The string length increases in proportion to the increase in the number of units, and for large systems, the resulting problem solution space cannot be effectively searched by a GA. Performing selection, mutation and crossover on this whole string can be quite time consuming and the chances for convergence to a sub optimal solution are quite high. For example 110 units scheduled over 24 hours would result in a chromosome of length 2640 bits, giving the GA the task of searching a total search space of $2^{2640}$ or $4.1 \times 10^{795}$, an enormously large search space, in which even a GA would take a very long time to produce any useful solutions.

### 7.2.1 Time partitioning

A method of decomposition that limits the GA search space to $2^N$ instead of $2^{NxT}$ is proposed where the selection, mutation and cross over are restricted to a single time interval. The time intervals are then considered in sequence starting from the first. As the sequence progresses all other variables such as minimum up / down times, ramp rates and spinning reserve requirements are checked for constraint violation and penalised accordingly. During each search interval, the number of generations for the search is randomly chosen with uniform probability within a set interval. This random selection acts in a similar manner to the saving of a number of solutions within each interval in the dynamic programming method. The GA method, by using a random number of generations per interval, picks a near optimal solution of the interval in the process.

The flow chart for the decomposed genetic algorithm cycle is shown in figure 7.1. An elitist GA search is used which guarantees that the best solution so far obtained in the search is retained and used in the succeeding generation, and thereby ensuring no good solution already found can be lost in the search process.

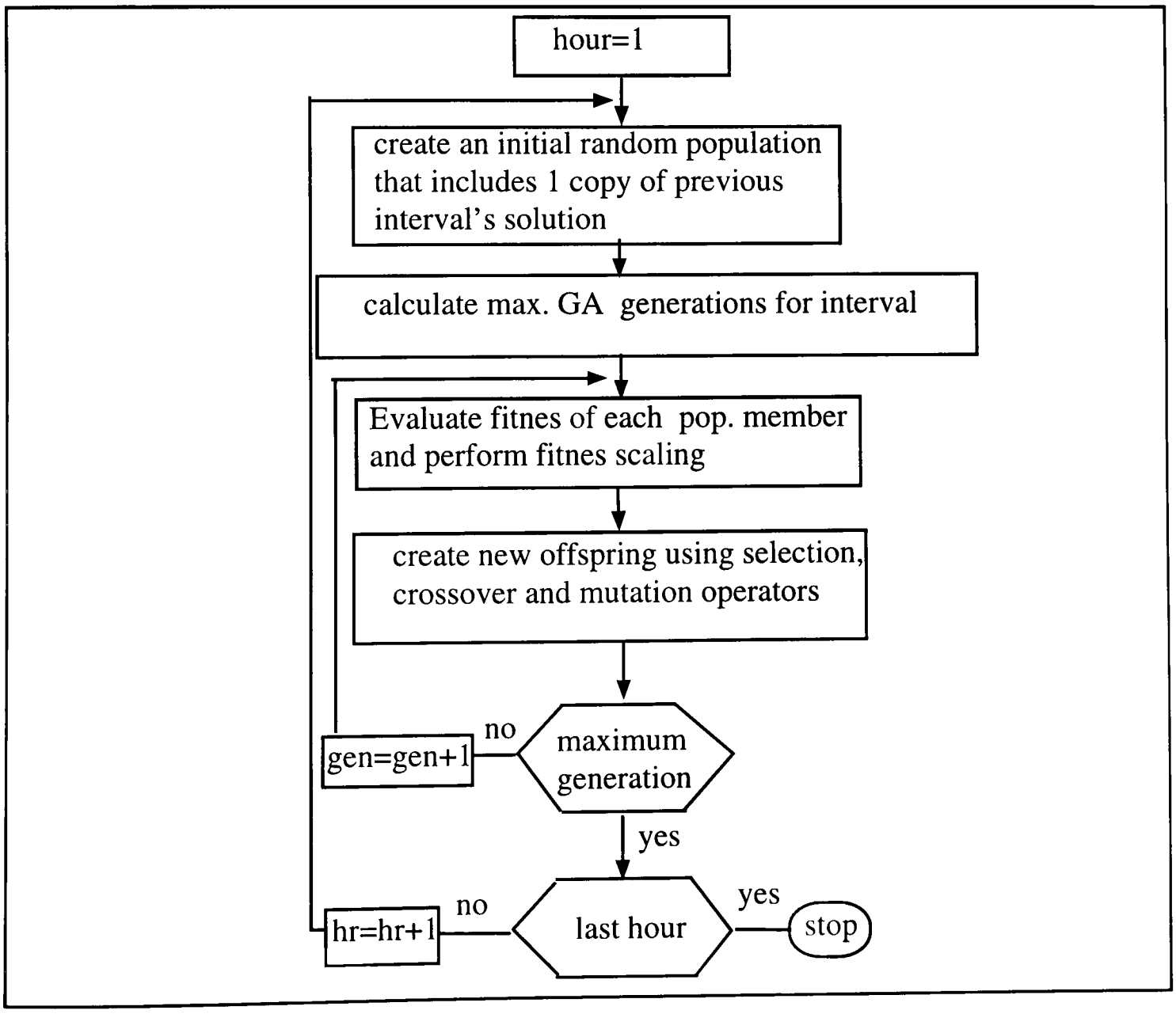



**Figure 7.1 Sequential decomposed GA flow chart**

The decomposition process does not strictly partition the problem into single time spans, but involves a cumulative time span partitioning, in that the linking constraint parameters such

as unit minimum up/down times and ramp rate limits are continuously up dated and re-evaluated as the time steps increase. It is only the search for the running and start up / shut down costs that are limited to an individual interval through the selection, cross over and mutation operators. For example, in the second time interval, all the unit minimum up/down times and ramp rates are considered from the initial condition up to and including hour two, effectively analysing a string solution of length 2n where n is the no. of generators. The only restriction to the search, is that the crossover and mutation operators are only done on solution string bits between lengths n+1 and 2n, while keeping the solution string bits in interval 1 to n, at their already established values in the first time interval. This process is then repeated until the end of the scheduling period is reached. An example of the decomposed GA solution sequence is shown in figure 7.2.

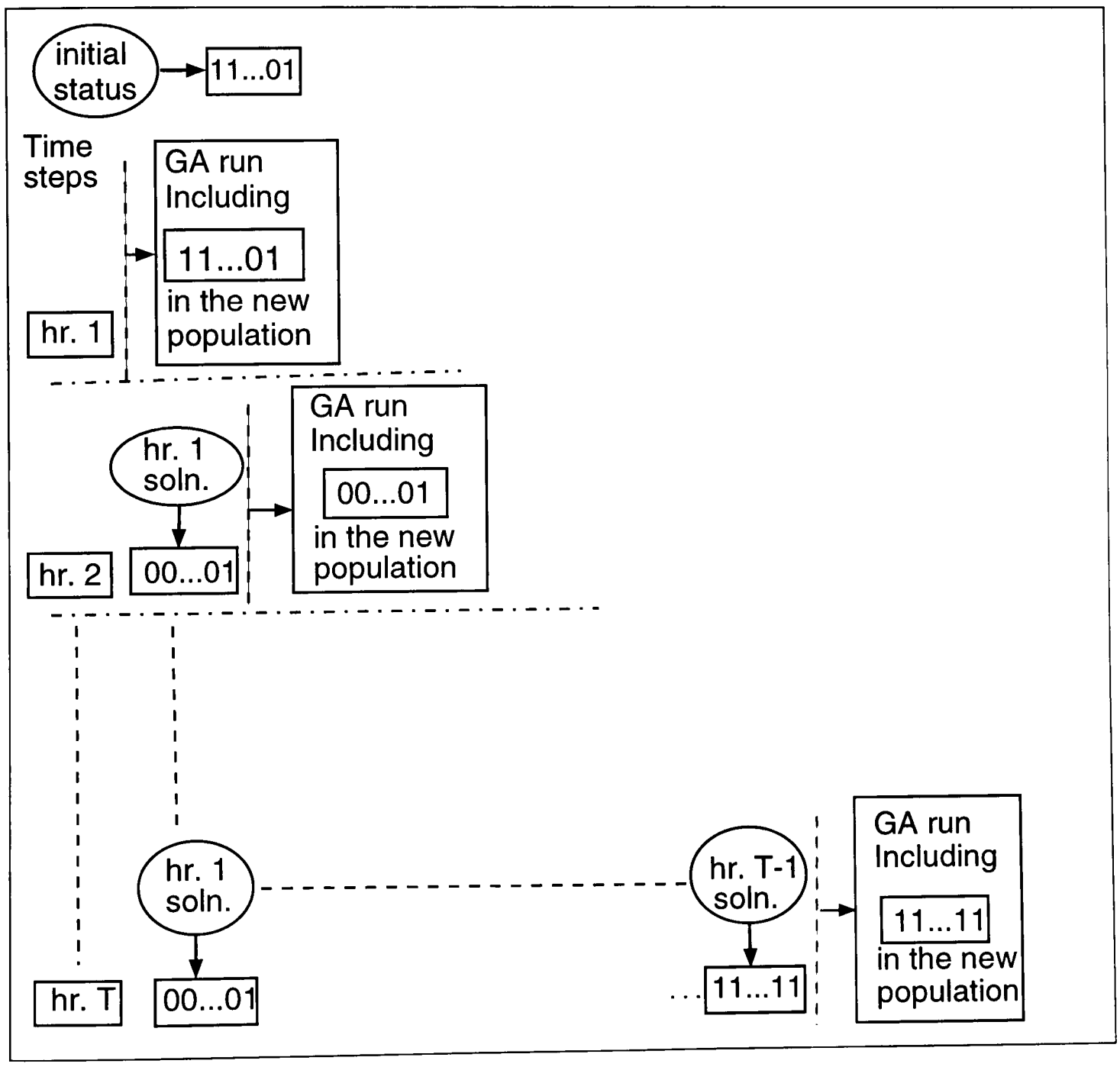



**Figure 7.2 Example of decomposed thermal scheduling GA sequence**

The GA composite objective function includes penalty functions, that are carefully graded to differentiate between feasible and non feasible solutions, by penalising the solutions that violate the linking constraints according to the magnitude of the violation. The only condition that can lead to a final non feasible solution in the sequential search method, is one that

manifests itself as a premature start up condition of a given unit, although this is a very rare occurrence. When this condition appears, the algorithm takes care of this condition through a heuristic that allows the violating unit to run throughout the previous shut down period, so that the failure to meet demand condition is avoided. In the hybrid decomposed GA, this condition is avoided by the inclusion of the priority list solution in the initial GA population, always ensuring a feasible solution better than or equal to that of the priority list method.

The proposed method uses a solution from the previous interval as a member of the starting population in a current time step, to help the GA search in maintaining useful building blocks. An advantage of this approach is that any constraints which are already satisfied cannot be violated later in the sequence. Once a GA operation has taken place in any time interval, the bit strings for that interval are not disturbed, thus preserving any useful schema already obtained in the solution space. The decomposition will lead to some loss of optimality but the loss is negligible in comparison with that which would occur if the whole search space is considered by the GA simultaneously. To further improve the algorithm performance, the results from the decomposed method can be included in a starting population in a final local GA search that uses the whole string in the search process.

### 7.2.2 Decomposed GA performance on a 10 unit test system

The performance of the decomposed genetic algorithm method on a 10 generator system, [EPRI, 1994] ) is described. The algorithm run was repeated 10 times for each set number of generations and the scheduling costs and cpu run times were recorded. Table 7.1 shows the average percentage improvement of the GA method over the priority list method as the number of generations in a run is varied.

**Table 7.1 Comparison of decomposed GA method with priority list method (10 unit system)**

| Average generations of GA run per time step | Total cpu time (sec.) | Average percentage improvement over Priority list |
|---|---|---|
| 5 | 0.5 | 2.81 |
| 10 | 0.6 | 2.86 |
| 20 | 0.9 | 2.89 |
| 40 | 1.7 | 3.01 |
| 60 | 2.4 | 3.01 |
| 80 | 3.1 | 3.01 |
| 100 | 3.8 | 3.01 |
| GA parameters: population size = 10, $p_{cross}$ = 1.0, $p_{mutation}$= 0.05, Elite copies=1, stochastic remainder selection, uniform crossover | | |

In table 7.2, the best genetic algorithm scheduling results obtained for the 10 unit system are compared with those obtained using the Lagrangian relaxation and artificial neural network methods as implemented in [EPRI, 1994].

**Table 7.2 Comparison of best decomposed GA solution with other solution techniques (10 unit system)**

| Method | Schedule cost | Average percentage improvement over Priority list solution |
|---|---|---|
| Lagrangian Relaxation | 47,511 | 3.19 |
| Artificial Neural Network | 48,293 | 1.60 |
| Decomposed GA | 47,596 | 3.02 |
| Priority list | 49,079 | - |

For this 10 generator system, the decomposed GA method provides better schedules than the priority list method with a final improvement of 3% above that given by the priority list method. It also provides schedules which are within 0.17% of a Lagrangian relaxation implementation reported in the EPRI study [EPRI, 1994]. As regards the GA control parameters, it can be seen that a population size as small as 10 is able to provide some very good solutions, since the effective search space at each time interval is limited to $2^{10}$ or 1024, and with the application of the full uniform crossover operator (effective for small populations) and the given mutation rates, an effective search is possible as shown by the results in table 7.1. Investigations with larger population sizes did not significantly improve the GA performance on this test system.

### 7.2.3 Decomposed GA performance on a 110 unit test system

Test results of the performance of the decomposed GA method on a large scale test system (110 generators) is described in this section. The algorithm was run 10 times for each set number of generations, with each trial starting with a different initial population. Table 7.3 shows the average percentage improvement of the decomposed GA method over the priority list method as the GA run time (number of generations) is varied. The best solution, out of 10 trials, obtained from the GA runs for each given number of generations are included in the results shown on table 7.3. The results indicate that if the decomposed GA is left to run for long enough, it provides better solutions than the priority list scheduling method. The GA population size is set to 110 which is the same as the number of units. The GA performance initially starts off worse than the priority list method, but as the number of generations, (cpu time) increases, it overtakes the priority list method. The decomposed GA provides better solutions than the priority list scheme after about 28 minutes of cpu run time (80 generations) and achieves an average of 0.5% improvement in solution within about 12 hours of cpu time (2000 generations). It can be seen that a decomposed GA method will finally arrive at a good solution, but the time it takes to reach the solution might not be within an acceptable limit for the scheduling requirements especially in large scale systems. Thus even with decomposition, the GA method cannot guarantee a significant advantage in performance as compared with the simple priority list based method on large scale systems.

**Table 7.3 Comparison of decomposed GA method with priority list method (110 unit test system)**

| Generations of GA run per step interval | Total CPU time (hr:min.) | Production cost for best solution | percentage cost improvement over the priority list method | |
|---|---|---|---|---|
| | | | Mean | Best |
| 5 | 0:1 | 4,023,409 | -5.74 | -4.37 |
| 10 | 0:2 | 3,938,241 | -3.27 | -2.16 |
| 20 | 0:5 | 3,899,933 | -1.66 | -1.17 |
| 40 | 0:12 | 3,865,835 | -0.75 | -0.29 |
| 60 | 0:20 | 3,858,627 | -0.30 | -0.10 |
| 80 | 0:28 | 3,850,205 | 0.10 | 0.12 |
| 100 | 0:35 | 3,843,275 | 0.08 | 0.30 |
| 200 | 1:13 | 3,840,238 | 0.35 | 0.38 |
| 500 | 3:08 | 3,839,215 | 0.40 | 0.40 |
| 1000 | 6:10 | 3,836,487 | 0.43 | 0.48 |
| 2000 | 12:35 | 3,834,467 | 0.49 | 0.53 |
| Priority list solution | 3,854,821 | | | |
| GA parameters: population=110, $p_{cross}$=1.0, $p_{mutation}$=0.005, Elite copies=10, stochastic remainder selection, uniform crossover | | | | |

## 7.3 Hybrid Decomposed Genetic Algorithm Solution

The convergence and long computation time difficulties of the decomposed thermal scheduling method on large scale systems can be alleviated by the use of a decomposed hybrid GA technique. The decomposed hybrid GA cycle is depicted in figure 7.2.

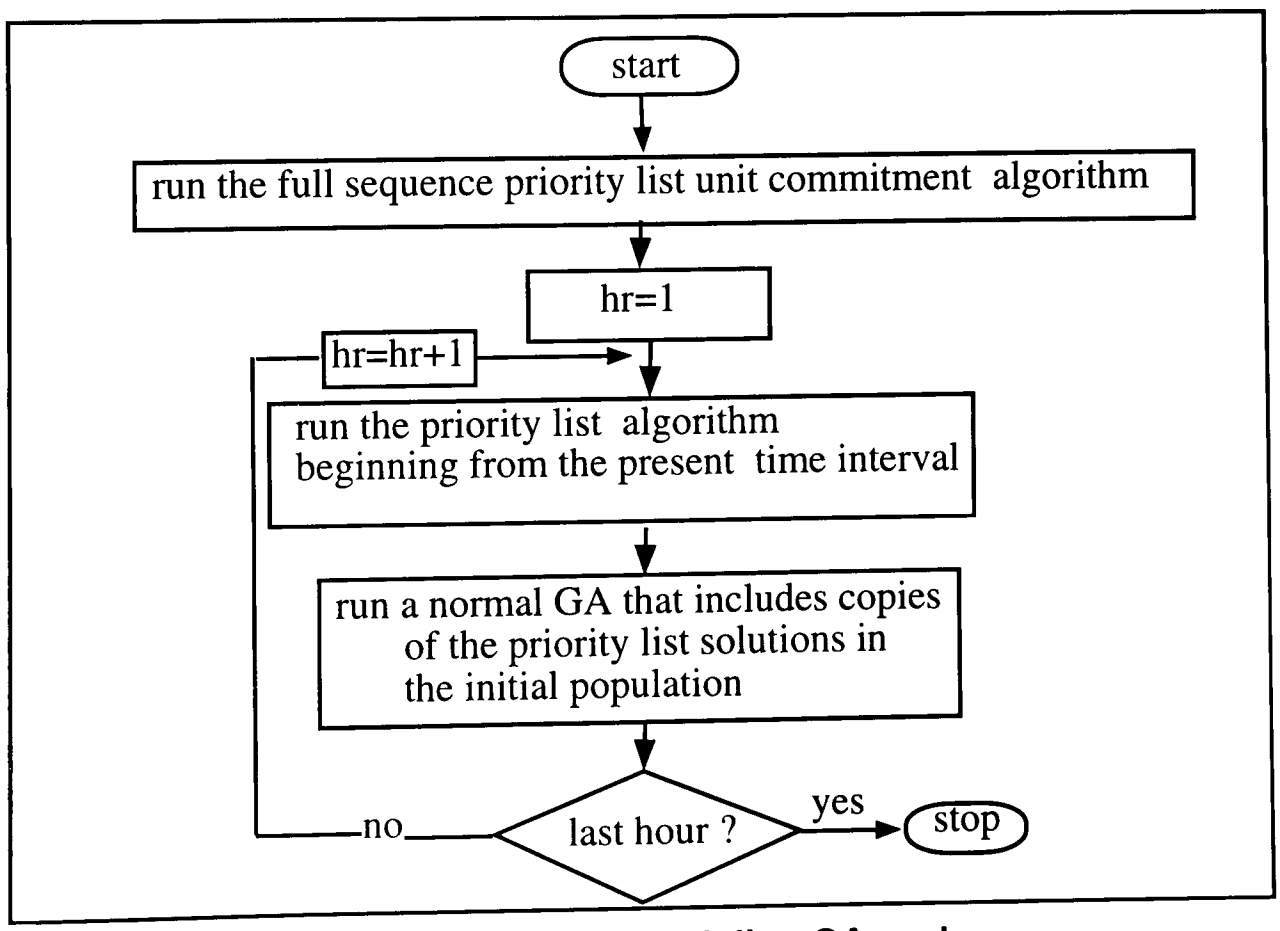



**Figure 7.3 Decomposed hybrid thermal scheduling GA cycle**

The decomposed hybrid thermal scheduling GA implemented in this work incorporates a priority list unit commitment method as a part of the GA process. In the decomposed hybrid GA, the priority list unit commitment algorithm is executed at the beginning of each time interval, using the results of the decomposed GA solution up to the beginning of the interval as the initial scheduling conditions. The priority list unit commitment solution over the whole scheduling period is also kept in the population of solutions throughout the decomposed GA scheduling process.

The decomposed hybrid genetic algorithm thermal scheduling method has been applied to a number of test systems and some of the results compared with those of the decomposed GA, priority list and other methods previously described. The test system results are presented in the following sections.

### 7.3.1 Decomposed hybrid GA performance on a 10 unit test system

The performance of the hybrid decomposed GA solution method on the 10 generator system, [EPRI, 1994] is described. The GA algorithm run was repeated 10 times for each set number of generations and the scheduling costs and cpu run times were recorded. Due to the stochastic nature of the GA method, the solutions obtained will not necessarily be the same because of the random initial starting populations. The best result obtained among the trials is taken as the optimal schedule. In table 7.4 the performance of the decomposed hybrid GA is compared with the priority list and decomposed GA methods for the 10 unit test system.

**Table 7.4 Comparison of decomposed hybrid GA and the decomposed GA method (10 unit system)**

| Average generations of GA run per interval | Total CPU time (sec.) | Average percentage cost improvement of the GA methods over the Priority list | |
|---|---|---|---|
| | | Decomposed GA | Hybrid decomposed GA |
| 5 | 0.5 | 2.81 | 2.99 |
| 10 | 0.6 | 2.86 | 2.98 |
| 20 | 0.9 | 2.89 | 3.01 |
| 40 | 1.7 | 3.01 | 2.98 |
| 60 | 2.4 | 3.01 | 3.01 |
| 80 | 3.1 | 3.01 | 3.01 |
| 100 | 3.8 | 3.01 | 3.01 |
| GA parameters: population size = 10, Pcross = 1.0, Pmut.= 0.05, Elite copies=1, stochastic remainder selection, uniform crossover | | | |

**Table 7.5 Comparison of decomposed hybrid GA method with other solution techniques (10 unit system)**

| Solution method | Schedule cost | % improvement of the methods over the Priority list solution |
|---|---|---|
| Lagrangian Relaxation | 47,511 | 3.19% |
| Artificial Neural Network | 48,293 | 1.60% |
| Decomposed Hybrid GA | 47,576 | 3.06% |
| Decomposed GA | 47,596 | 3.02% |
| Priority list | 49,079 | |

The performance of the two decomposed GA methods do not show any significant difference, in terms of computation time or solution accuracy for this test system, although the hybrid method converges to the final solution in only 20 generations, while the decomposed GA method converges after 40 generations. Both the GA methods provide better schedules than the priority list method with a final improvement of 3% above that given by the priority list method.

In table 7.5, the best decomposed hybrid genetic algorithm scheduling results are also compared with those from other studies [EPRI, 1994] obtained using the Lagrangian relaxation and artificial neural network methods. The decomposed hybrid GA method provides solutions that are within 0.13 percent of the Lagrangian relaxation solution method implemented in [EPRI], and much better than the artificial neural network implementation.

#### 7.3.1.1 Sample decomposed hybrid GA results (10 unit test system)

The results of a decomposed hybrid GA sample test run on the 10 unit sample network are presented in this section. Table 7.6 gives the unit commitment list, table 7.7 the total power generation and available reserve levels, while table 7.8 provides a listing of the unit loading levels.

**Table 7.6 A Hybrid GA unit commitment schedule (10 unit test system)**

| UNIT | HOURS |
|---|---|
| 1 | 111111111111111111111111 |
| 2 | 111111111111111111111111 |
| 3 | 111111111111100000000000 |
| 4 | 111111111111111111110001 |
| 5 | 000000000000000000000000 |
| 6 | 000000000000000000000001 |
| 7 | 111111111111111111111111 |
| 8 | 110000000000000000000000 |
| 9 | 111111111111111111111111 |
| 10 | 111111111111111111111111 |

PRODUCTION COST = **47576.05**

**Table 7.7 Total hourly capacity output, load demand, and penalty coefficients for the hybrid GA schedule of table 7.6**

| HR | capacity | load | reserve | PSTCOST | PSDCOST | FTRCOST |
|---|---|---|---|---|---|---|
| 1 | 1550.0 | 1167.0 | 383.0 | 0.0 | 0.0 | 0.0 |
| 2 | 1550.0 | 1097.0 | 453.0 | 0.0 | 0.0 | 0.0 |
| 3 | 1400.0 | 1039.0 | 361.0 | 0.0 | 0.0 | 0.0 |
| 4 | 1400.0 | 1028.0 | 372.0 | 0.0 | 0.0 | 0.0 |
| 5 | 1400.0 | 1017.0 | 383.0 | 0.0 | 0.0 | 0.0 |
| 6 | 1400.0 | 1051.0 | 349.0 | 0.0 | 0.0 | 0.0 |
| 7 | 1400.0 | 1098.0 | 302.0 | 0.0 | 0.0 | 0.0 |
| 8 | 1400.0 | 1051.0 | 349.0 | 0.0 | 0.0 | 0.0 |
| 9 | 1400.0 | 1017.0 | 383.0 | 0.0 | 0.0 | 0.0 |
| 10 | 1400.0 | 993.0 | 407.0 | 0.0 | 0.0 | 0.0 |
| 11 | 1400.0 | 958.0 | 442.0 | 0.0 | 0.0 | 0.0 |
| 12 | 1400.0 | 946.0 | 454.0 | 0.0 | 0.0 | 0.0 |
| 13 | 1400.0 | 923.0 | 477.0 | 0.0 | 0.0 | 0.0 |
| 14 | 1300.0 | 910.0 | 390.0 | 0.0 | 0.0 | 0.0 |
| 15 | 1300.0 | 900.0 | 400.0 | 0.0 | 0.0 | 0.0 |
| 16 | 1300.0 | 876.0 | 424.0 | 0.0 | 0.0 | 0.0 |
| 17 | 1300.0 | 853.0 | 447.0 | 0.0 | 0.0 | 0.0 |
| 18 | 1300.0 | 829.0 | 471.0 | 0.0 | 0.0 | 0.0 |
| 19 | 1300.0 | 794.0 | 506.0 | 0.0 | 0.0 | 0.0 |
| 20 | 1300.0 | 782.0 | 518.0 | 0.0 | 0.0 | 0.0 |
| 21 | 1180.0 | 770.0 | 410.0 | 0.0 | 0.0 | 0.0 |
| 22 | 1180.0 | 818.0 | 362.0 | 0.0 | 0.0 | 0.0 |
| 23 | 1180.0 | 864.0 | 316.0 | 0.0 | 0.0 | 0.0 |
| 24 | 1580.0 | 1167.0 | 413.0 | 0.0 | 0.0 | 0.0 |

**Table 7.8 Economic dispatch levels for schedule on table 7.6**

| HR | UNIT LOADING LEVELS (1~10) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 60 | 80 | 100 | 105 | 0 | 0 | 310 | 150 | 161 | 200 |
| 2 | 60 | 76 | 97 | 93 | 0 | 0 | 274 | 150 | 145 | 200 |
| 3 | 60 | 80 | 100 | 109 | 0 | 0 | 322 | 0 | 167 | 200 |
| 4 | 60 | 80 | 100 | 107 | 0 | 0 | 316 | 0 | 164 | 200 |
| 5 | 60 | 80 | 100 | 105 | 0 | 0 | 310 | 0 | 161 | 200 |
| 6 | 60 | 80 | 100 | 111 | 0 | 0 | 329 | 0 | 170 | 200 |
| 7 | 60 | 80 | 100 | 120 | 0 | 0 | 355 | 0 | 182 | 200 |
| 8 | 60 | 80 | 100 | 111 | 0 | 0 | 329 | 0 | 170 | 200 |
| 9 | 60 | 80 | 100 | 105 | 0 | 0 | 310 | 0 | 161 | 200 |
| 10 | 60 | 80 | 100 | 100 | 0 | 0 | 296 | 0 | 155 | 200 |
| 11 | 60 | 77 | 98 | 94 | 0 | 0 | 279 | 0 | 147 | 200 |
| 12 | 60 | 75 | 97 | 93 | 0 | 0 | 274 | 0 | 145 | 200 |
| 13 | 60 | 72 | 94 | 89 | 0 | 0 | 264 | 0 | 141 | 200 |
| 14 | 60 | 80 | 0 | 104 | 0 | 0 | 306 | 0 | 159 | 200 |
| 15 | 60 | 80 | 0 | 102 | 0 | 0 | 300 | 0 | 157 | 200 |
| 16 | 60 | 80 | 0 | 97 | 0 | 0 | 287 | 0 | 151 | 200 |
| 17 | 60 | 76 | 0 | 93 | 0 | 0 | 276 | 0 | 146 | 200 |
| 18 | 60 | 72 | 0 | 89 | 0 | 0 | 264 | 0 | 141 | 200 |
| 19 | 60 | 67 | 0 | 83 | 0 | 0 | 250 | 0 | 133 | 200 |
| 20 | 59 | 63 | 0 | 80 | 0 | 0 | 250 | 0 | 128 | 200 |
| 21 | 60 | 78 | 0 | 0 | 0 | 0 | 282 | 0 | 149 | 200 |
| 22 | 60 | 80 | 0 | 0 | 0 | 0 | 314 | 0 | 163 | 200 |
| 23 | 60 | 80 | 0 | 0 | 0 | 0 | 346 | 0 | 177 | 200 |
| 24 | 60 | 80 | 0 | 118 | 0 | 182 | 347 | 0 | 178 | 200 |

### 7.3.2 Decomposed hybrid GA performance on a 26 unit test system

Table 7.9 shows the performance of both the decomposed GA and hybrid GA methods for a relatively small population (size 26), as the GA run time (number of generations) is varied, while Table 7.10 compares the performance of the two GA methods for a population size of 100, for a fixed time of run, (100 generations). In each test case, the GA run is repeated 10 times, each test done with a different initial population.

**Table 7.9 Comparison of the decomposed hybrid GA and decomposed GA methods ( 26 unit test system, load profile_1)**

| Average generations of GA run per step interval | Decomposed GA cost | Decomposed hybrid GA cost |
|---|---|---|
| 10 | 735,304 | 724,497 |
| 25 | 729,096 | 724,177 |
| 50 | 725,611 | 724,630 |
| 100 | 725,062 | 723,905 |
| 250 | 723,714 | 723,174 |
| 500 | 722,847 | 723,705 |
| 1000 | 722,536 | 722,455 |
| 2000 | 722,446 | 722,378 |
| Best | **722,446** | **722,378** |
| Percentage improv. of best over priority list solution | 0.65% | 0.66% |
| GA parameters: popsize=26, Pcross=1.0, Pmut.=0.005, Elite copies=10, stochastic remainder selection, uniform crossover | | |

From table 7.9, it can be seen that with a decomposed hybrid GA technique, even with a modest population of only 26 individuals, some fairly good solutions can be obtained. The hybrid decomposed GA converges after about 1000 generations, while the decomposed GA takes about 2000 generation to converge. For this test system, the decomposed GA method surpasses the priority list solution (of 727, 200), after 50 generations, while the decomposed hybrid GA always provides solutions which are better than the priority list solution as expected from the algorithm design. Both methods perform equally well, as they both converge to similar solutions within more or less similar computation times. When the population size is increased to 100, as shown by the results in table 7.10, convergence can be obtained after only 100 generations, but a larger population size requires more computation time per generation.

**Table 7.10 Effect of population size on the performance of the decomposed hybrid GA and decomposed GA methods ( 26 unit test system, load profile_1)**

| Trial | Decomposed GA cost | Decomposed Hybrid GA cost |
|---|---|---|
| 1 | 722,312 | 722,312 |
| 2 | 722,584 | 722,589 |
| 3 | 722,759 | 722,450 |
| 4 | 722,759 | 722,448 |
| 5 | 722,581 | 722,105 |
| 6 | 722,870 | 722,759 |
| 7 | 722,462 | 722,450 |
| 8 | 722,464 | 722,326 |
| 9 | 722,760 | 722,608 |
| 10 | 722,891 | 722,743 |
| Best | **722,312** | **722,105** |
| Average Cpu time | 1 min.: 20 sec. | 1 min. : 20 sec. |
| Percentage improv. of best over priority list solution | 0.67% | 0.70% |
| GA parameters: popsize=100, Pcross=1.0, Pmut.=0.005, Elite copies=10, Avg. generations=100, stochastic remainder selection, uniform crossover | | |

**Table 7.11 Summary of best scheduling results (26 unit system, load profile_1).**

| Solution method | Production cost | % Improv. over the priority list solution |
|---|---|---|
| Decomposed hybrid GA | 722, 105 | 0.70 |
| Decomposed GA | 722,312 | 0.67 |
| ANN /DP method | 724,077 | 0.43 |
| Priority List | 727,200 | - |

From the results summary provided in table 7.11, it can be seen that for the 26 unit test system, the decomposed hybrid GA method provides a 0.7% improvement on the priority list solution and a 0.03% improvement on the decomposed GA solution. The decomposed hybrid GA solutions are also 0.3% better than those of a technique that uses a combination of artificial neural networks and dynamic programming [Wang and Shahidepour].

#### 7.3.2.1 Effect of variation of load profile on decomposed hybrid GA performance

The effect of a varying demand pattern on the performance of the hybrid GA was investigated by using load profile_2 , instead of load profile_1 used to generate the results in the previous section. Load profile_2 has a lower load demand profile and hence allows a greater choice in the number of units to schedule, and hence should provide a greater challenge to the scheduling algorithms.

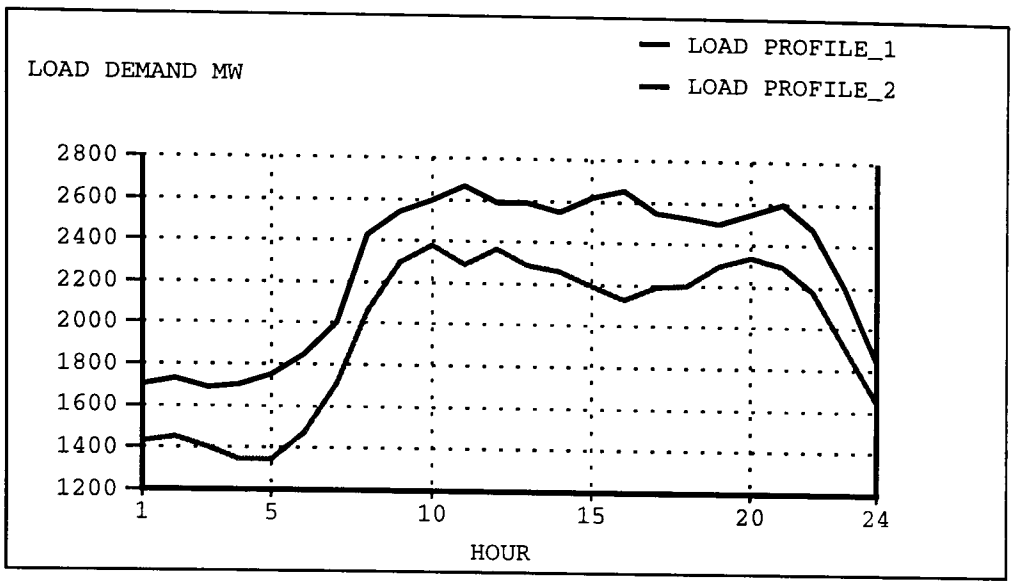



**Figure 7.4 Load demand profiles for the 26 unit test system**

**Table 7.12 Comparative performance of the decomposed hybrid GA and decomposed GA methods ( 26 unit test system, load profile_2)**

| Trial | Decomposed GA cost | Decomposed Hybrid GA cost |
|---|---|---|
| 1 | 582,479 | 581,822 |
| 2 | 581,161 | 581,972 |
| 3 | 581,983 | 581,111 |
| 4 | 581,397 | 581,305 |
| 5 | 581,986 | 581,155 |
| 6 | 581,165 | 581,151 |
| 7 | 581,432 | 581,868 |
| 8 | 582,152 | 581,155 |
| 9 | 581,169 | 580,759 |
| 10 | 581,187 | 581,165 |
| Best | **581,161** | **580,759** |
| % improv. of best over priority list | 2.69% | 2.76% |
| GA parameters: popsize=100, Pcross=1.0, Pmut.=0.001, Elite copies=10, Average no. of generations=100, stochastic remainder selection, uniform crossover | | |

**Table 7.13 Summary of scheduling results for the 26 unit system, load profile_2**

| Solution method | Production cost | % Improv. over the priority list soln. |
|---|---|---|
| Hybrid GA | 580,759 | 2.76 |
| Decomposed GA | 581,161 | 2.69 |
| ANN/DP [Wang and Shahidepour] | 607,051 | -1.64 |
| Priority List | 597,254 | - |

For the lower load demand pattern, load profile_2, the decomposed hybrid GA provides a marked improvement over the priority list method. It also provides solutions which are

0.07% better than the decomposed solution, 2.76% better than the priority list solution and 4.4% better than a technique that uses a combination of artificial neural networks and dynamic programming [Wang and Shahidepour]. These results demonstrate the abilities of the GA methods to cope with thermal scheduling cases where more unit commitment decisions (start / stop) are to be made in order to accommodate a more fluctuating load demand pattern.

#### 7.3.2.2 Sample decomposed hybrid GA results on 26 unit test system (load profile_1)

The Simulation conditions and GA parameter settings were: reserve covers loss of largest committed unit, truncated fitnes scaling, standard deviation scale factor = 1.00, elitist reproduction 10 copies per generation, uniform cross over, cross over rate = 1.0, mutation rate = 0.005, generational replacement, generation gap = 1.00, completely random initial start, merit order economic dispatch, population size = 100, average no. of generations per time step = 100, scheduling period = 24 hours, number of units = 26, random number seed = 7347473.

**Table 7.14 A Hybrid GA unit commitment schedule for 26 test system**

| HOUR | UNITS ( 1 ~ 26 ) |
|---|---|
| 1 | 11100000011110001111000111 |
| 2 | 11111000011110001111000111 |
| 3 | 11000000011110001111000111 |
| 4 | 11100000011110001111000111 |
| 5 | 00000000011111001111000111 |
| 6 | 00000000011111101111000111 |
| 7 | 11100000011111111111000111 |
| 8 | 11110100011111111111110111 |
| 9 | 00000000011111111111111111 |
| 10 | 11100000011111111111111111 |
| 11 | 11110111011111111111111111 |
| 12 | 11100000011111111111111111 |
| 13 | 11100000011111111111111111 |
| 14 | 00000000011111111111111111 |
| 15 | 11111000011111111111111111 |
| 16 | 11110110011111111111111111 |
| 17 | 00000000011111111111111111 |
| 18 | 00000000011111111111111111 |
| 19 | 11100000011111101111111111 |
| 20 | 11110110011111101111111111 |
| 21 | 11100000011111111111111111 |
| 22 | 11000000011111011111111111 |
| 23 | 11100000011110011111110111 |
| 24 | 00000000011110111111000111 |

PRODUCTION COST = **722311.6**

**Table 7.15 Total hourly capacity output, load demand, and penalty coefficients for the 26 unit system schedules**

| HR | capacity | load | reserve | PSTCOST | PSDCOST | FTRCOST |
|---|---|---|---|---|---|---|
| 1 | 2110.0 | 1700.0 | 410.0 | 0.0 | 0.0 | 0.0 |
| 2 | 2134.0 | 1730.0 | 404.0 | 0.0 | 0.0 | 0.0 |
| 3 | 2098.0 | 1690.0 | 408.0 | 0.0 | 0.0 | 0.0 |
| 4 | 2110.0 | 1700.0 | 410.0 | 0.0 | 0.0 | 0.0 |
| 5 | 2174.0 | 1750.0 | 424.0 | 0.0 | 0.0 | 0.0 |
| 6 | 2274.0 | 1850.0 | 424.0 | 0.0 | 0.0 | 0.0 |
| 7 | 2410.0 | 2000.0 | 410.0 | 0.0 | 0.0 | 0.0 |
| 8 | 2836.0 | 2430.0 | 406.0 | 0.0 | 0.0 | 0.0 |
| 9 | 2965.0 | 2540.0 | 425.0 | 0.0 | 0.0 | 0.0 |
| 10 | 3001.0 | 2600.0 | 401.0 | 0.0 | 0.0 | 0.0 |
| 11 | 3073.0 | 2670.0 | 403.0 | 0.0 | 0.0 | 0.0 |
| 12 | 3001.0 | 2590.0 | 411.0 | 0.0 | 0.0 | 0.0 |
| 13 | 3001.0 | 2590.0 | 411.0 | 0.0 | 0.0 | 0.0 |
| 14 | 2965.0 | 2550.0 | 415.0 | 0.0 | 0.0 | 0.0 |
| 15 | 3025.0 | 2620.0 | 405.0 | 0.0 | 0.0 | 0.0 |
| 16 | 3053.0 | 2650.0 | 403.0 | 0.0 | 0.0 | 0.0 |
| 17 | 2965.0 | 2550.0 | 415.0 | 0.0 | 0.0 | 0.0 |
| 18 | 2965.0 | 2530.0 | 435.0 | 0.0 | 0.0 | 0.0 |
| 19 | 2901.0 | 2500.0 | 401.0 | 0.0 | 0.0 | 0.0 |
| 20 | 2953.0 | 2550.0 | 403.0 | 0.0 | 0.0 | 0.0 |
| 21 | 3001.0 | 2600.0 | 401.0 | 0.0 | 0.0 | 0.0 |
| 22 | 2889.0 | 2480.0 | 409.0 | 0.0 | 0.0 | 0.0 |
| 23 | 2604.0 | 2200.0 | 404.0 | 0.0 | 0.0 | 0.0 |
| 24 | 2274.0 | 1840.0 | 434.0 | 0.0 | 0.0 | 0.0 |

Where PSTCOST / PSDCOST, FTRCOST - premature startup / shut down penalty cost and failure to meet reserve requirement penalties respectively.

**Table 7.16 Hourly Production costs for the Hybrid GA schedule on table 7.15**

| HR | RUN COST | START COST | SHUT DOWN COST | SUB TOTAL | CUMULATIVE COST |
|---|---|---|---|---|---|
| 1 | 18858.7 | 0.0 | 0.0 | 18858.7 | 18858.7 |
| 2 | 19335.6 | 0.0 | 0.0 | 19335.6 | 38194.3 |
| 3 | 18682.1 | 0.0 | 0.0 | 18682.1 | 56876.4 |
| 4 | 18858.7 | 0.0 | 0.0 | 18858.7 | 75735.1 |
| 5 | 19658.9 | 127.8 | 0.0 | 19786.8 | 95521.9 |
| 6 | 21208.5 | 130.5 | 0.0 | 21339.1 | 116860.9 |
| 7 | 23740.1 | 132.6 | 0.0 | 23872.7 | 140733.6 |
| 8 | 32476.4 | 738.5 | 0.0 | 33214.9 | 173948.5 |
| 9 | 34766.9 | 355.4 | 0.0 | 35122.3 | 209070.8 |
| 10 | 36087.5 | 0.0 | 0.0 | 36087.5 | 245158.2 |
| 11 | 38291.7 | 112.5 | 0.0 | 38404.1 | 283562.4 |
| 12 | 35853.6 | 0.0 | 0.0 | 35853.6 | 319415.9 |
| 13 | 35853.6 | 0.0 | 0.0 | 35853.6 | 355269.5 |
| 14 | 34956.6 | 0.0 | 0.0 | 34956.6 | 390226.1 |
| 15 | 36618.8 | 0.0 | 0.0 | 36618.8 | 426844.9 |
| 16 | 37643.2 | 74.6 | 0.0 | 37717.8 | 464562.7 |
| 17 | 34956.6 | 0.0 | 0.0 | 34956.6 | 499519.3 |
| 18 | 34578.4 | 0.0 | 0.0 | 34578.4 | 534097.7 |
| 19 | 33988.9 | 0.0 | 0.0 | 33988.9 | 568086.6 |
| 20 | 35544.6 | 71.1 | 0.0 | 35615.7 | 603702.3 |
| 21 | 36087.5 | 97.5 | 0.0 | 36185.0 | 639887.3 |
| 22 | 33522.3 | 0.0 | 0.0 | 33522.3 | 673409.6 |
| 23 | 27709.3 | 0.0 | 0.0 | 27709.3 | 701118.9 |
| 24 | 21095.1 | 97.5 | 0.0 | 21192.6 | 722311.6 |

### 7.3.3 Decomposed hybrid GA performance a 110 unit test system

The test results of the performance of the hybrid GA solution method on a 110 generator test system is presented in this section. The GA algorithm run was repeated 10 times for each set number of generations and the scheduling costs and cpu run times were recorded. Table 7.17

shows the average percentage improvement of the decomposed GA and the hybrid GA methods over the priority list method as the number of generations is varied. For each set number of generations of the GA, the experiment was repeated 10 times to obtain the average readings included in table 7.17. The GA experiments took several months of CPU time to carry out. For example for 2000 generations of run, each trial took about 13 CPU hours, so it took about 130 hours of CPU time to generate the results reported for a 2000 generations run. A summary of the best solution, out of 10 trials, obtained from the GA runs are included in table 7.18. Also included in table 7.18 are results obtained from a different GA implementation [Karzalis et. al.]

**Table 7.17 Comparison of the decomposed hybrid GA with other solution techniques (110 test system)**

| Average generations of GA run per step interval | Total CPU time ( hrs: min.) | Production cost for *best* solution out of 10 runs | | *Average* percentage cost improvement over the priority list method | |
|---|---|---|---|---|---|
| | | Decomposed GA | Decomposed hybrid GA | Decomposed GA | Decomposed Hybrid GA |
| 5 | 0:1 | 4,023,409 | 3,848,123 | -5.74 | 0.04 |
| 10 | 0:2 | 3,938,241 | 3,836,662 | -3.27 | 0.33 |
| 20 | 0:5 | 3,899,933 | 3,832,547 | -1.66 | 0.50 |
| 40 | 0:12 | 3,865,835 | 3,831,502 | -0.75 | 0.40 |
| 60 | 0:20 | 3,858,627 | 3,826,775 | -0.30 | 0.56 |
| 80 | 0:28 | 3,850,205 | 3,833,315 | 0.10 | 0.50 |
| 100 | 0:35 | 3,843,275 | 3,829,520 | 0.08 | 0.63 |
| 200 | 1:13 | 3,840,238 | 3,830,328 | 0.35 | 0.54 |
| 500 | 3:08 | 3,839,215 | 3,832,023 | 0.40 | 0.52 |
| 1000 | 6:10 | 3,836,487 | 3,832,006 | 0.43 | 0.56 |
| 2000 | 12:35 | 3,834,467 | 3,834,747 | 0.49 | 0.52 |
| GA parameters: population=110, Pcross=1.0, Pmut.=0.005, Elite copies=10, stochastic remainder selection, uniform crossover | | | | | |

With 110 units the performance of the decomposed GA initially starts off worse than the priority list method, but as the number of generations, (cpu time) increases, it overtakes the priority list method. The hybrid GA, being a hybrid of both the decomposed GA and the priority list methods, always provides better solutions than the priority list method. The hybrid GA achieves a 0.5 percentage improvement in solution quality within the first 20 minutes of cpu time (60 generations) while the decomposed GA does the same after a whole 12 hours of cpu time (2000 generations). The decomposed hybrid GA method provides a better solution than a genetic algorithm implemented by [Karzalis et. al.] (see personal communications in appendix).

**Table 7.18 Summary of best scheduling results for the 110 unit system**

| Solution method | Production cost | Cpu time | Percentage improvement over the priority list soln. |
|---|---|---|---|
| Decomposed hybrid GA | 3,826,775 | 20 min. | 0.73 |
| Decomposed GA | 3,834,467 | 12 hrs. | 0.53 |
| Other GA [Karzalis et. al.] | 3,866,092 | not given | -0.29 |
| Priority List | 3,854,821 | 1 sec. | - |

The simple priority list scheduling method is very competitive with the GA methods, especially for large systems, but the slight percentage improvement in scheduling costs provided by the GA methods over those of the simple priority list method are quite significant in monetary terms and can result in substantial annual savings in operation costs.

It has been shown that by including the priority list scheme in a hybrid GA and exploiting the separable properties of the scheduling problem, good solutions can be obtained within reasonable times. If the decomposed GA is left to run for long enough, in the end it could provide better solutions than the hybrid method, as shown by the convergence characteristics in figure 7.5, which also reveals that the hybrid GA can sometimes be prone to local convergence, a potential drawback of the inclusion of domain knowledge in a hybrid system.

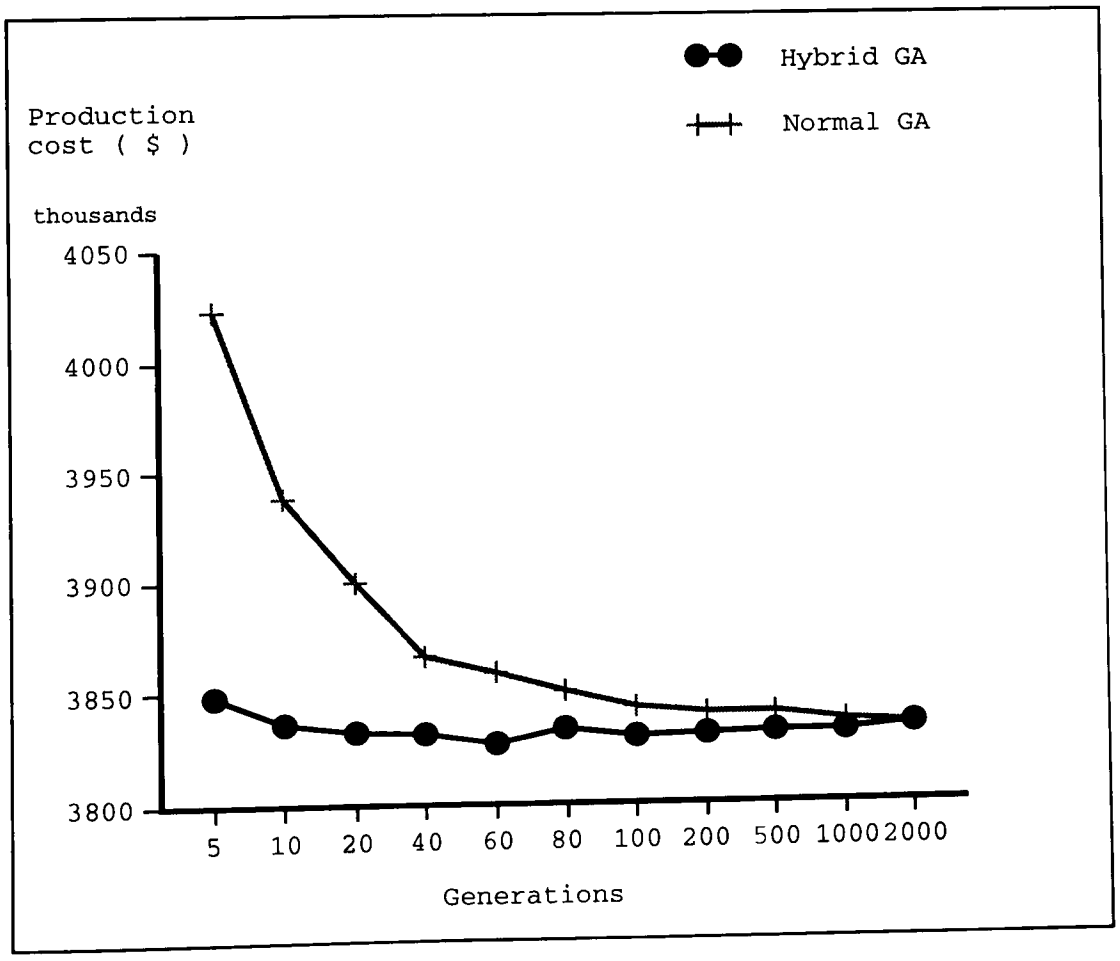



**Figure 7.5 Decomposed hybrid GA and Decomposed GA convergence characteristics (110 unit system)**

### 7.3.3.1 Sample decomposed hybrid GA scheduling results for the 110 unit test system

A sample result from a decomposed hybrid GA program is included in this section. The experimental conditions were: reserve set to cover the loss of the largest committed unit, truncated fitnes scaling, elitist reproduction 10 copies per generation, uniform cross over, cross over rate = 1.0, mutation rate = 0.005, generational replacement, generation gap = 1.00, completely random initial start, merit order economic dispatch, population size = 110, average no. of generations per time step = 100, scheduling period = 24 hours, no. of units = 110, random number seed = 945453334.

**Table 7.19 Hybrid GA unit commitment schedule for 110 test system**

| HOUR | UNITS ( 1 ~ 110 ) |
|---|---|
| 1 | 00001000011110001111000111101001100111011010110000000001111000111100011100011010011101100110101100001110010110 |
| 2 | 00000000001110001111000111101000100111011010110000000000000000011110001110011101000110100011010110000111001001 0 |
| 3 | 00000000000010001111000111101000101100011011110000000000000000011000000110011001000010100011011110000111000001 0 |
| 4 | 00000000000000001111000111101000001100011011110000000000000000010000000110011000000001000000111100000000000010 |
| 5 | 00000000000000001111000111101000011100011011110000000000100000010000000111111000000001100000011110000000000010 |
| 6 | 10001000000000001111000111111100011110011011110000000001000000100000001111110000000011000000111100000000000010 |
| 7 | 00000000000000001111000111111100011110011111110000000001000000101100001111110100000011000010111100000000000010 |
| 8 | 00000000000100001111000111111100011100011111110000000000000000011110001111111010000011100001011110000010000000 |
| 9 | 00000000011110001111000111111100011100111111110000000000000000011110001111111010000011100001011110000101000000 0 |
| 10 | 10000000011110001111000111111101011111111111110000000001100000111100011111111110000111100110111110000010000000 0 |
| 11 | 00000000011110001111000111111101011111111111110000000001100000111100011111111110000111101110111110000010000000 0 |
| 12 | 00000000010000001111000111111100001101111111111000000000011000001111000111111111100001110011101111000000000000 0 |
| 13 | 00000000000000001111000111111100001110101111111000000000000000011110001111111111000011100000011110000000000000 0 |
| 14 | 00000000000000001111000111111100001110101111111000000000000000011110001111111111000011100000011110000000000000 0 |
| 15 | 00000000011110001111000111111101111111111111110000000001111000111100011111111101000111010100111110000000000000 0 |
| 16 | 00000000011110001111000111111101111011111111110000000001111000111100011111111101000111010110111110000000000000 0 |
| 17 | 00000000011110001111000111111100111011111111110000000001111000111100011111111101000111000110111110000000000000 0 |
| 18 | 00000000011110001111000111111100011101111111110000000001111000111100011111111110001110011101111000000000000000 |
| 19 | 01000000011110011111000111111100111111011111111100100000011110001111000111111111110111110011101111000011000100 01 |
| 20 | 01000000011111011111000111111101111111111111111001000000111110011110001111111111101111111111111111000011000100 01 |
| 21 | 00000000011111111111000111111101111111111111111001000000111110011110001111111111101111111111111111000011110100 01 |
| 22 | 00000000011111111111000111111100011111111111111000000000000010011110001111111111000011100111011111000000011010001 |
| 23 | 00000000001111101111000111111101011111111111110000000000010011110001111111101100001110010101111100000011010001 |
| 24 | 00000000000111101111000111111101011101111111110000000000000000011110001111111010000011100100011110000000011000000 |

PRODUCTION COST = **3,826,690**

**Table 7.20 Total hourly capacity output, load demand, and penalty coefficients for a 110 unit system hybrid GA schedule**

| HR | capacity | load | reserve | PSTCOST | PSDCOST | FTRCOST |
|---|---|---|---|---|---|---|
| 1 | 12373.0 | 11600.0 | 773.0 | 0.0 | 0.0 | 0.0 |
| 2 | 11738.0 | 10900.0 | 838.0 | 0.0 | 0.0 | 0.0 |
| 3 | 11356.0 | 9500.0 | 1856.0 | 0.0 | 0.0 | 0.0 |
| 4 | 10015.0 | 9300.0 | 715.0 | 0.0 | 0.0 | 0.0 |
| 5 | 11236.0 | 10500.0 | 736.0 | 0.0 | 0.0 | 0.0 |
| 6 | 11900.0 | 11200.0 | 700.0 | 0.0 | 0.0 | 0.0 |
| 7 | 13206.0 | 12500.0 | 706.0 | 0.0 | 0.0 | 0.0 |
| 8 | 13651.0 | 12900.0 | 751.0 | 0.0 | 0.0 | 0.0 |
| 9 | 14201.0 | 13500.0 | 701.0 | 0.0 | 0.0 | 0.0 |
| 10 | 15218.0 | 14500.0 | 718.0 | 0.0 | 0.0 | 0.0 |
| 11 | 15306.0 | 14600.0 | 706.0 | 0.0 | 0.0 | 0.0 |
| 12 | 14803.0 | 14000.0 | 803.0 | 0.0 | 0.0 | 0.0 |
| 13 | 13975.0 | 13200.0 | 775.0 | 0.0 | 0.0 | 0.0 |
| 14 | 13775.0 | 13000.0 | 775.0 | 0.0 | 0.0 | 0.0 |
| 15 | 15206.0 | 14500.0 | 706.0 | 0.0 | 0.0 | 0.0 |
| 16 | 15306.0 | 14600.0 | 706.0 | 0.0 | 0.0 | 0.0 |
| 17 | 15131.0 | 14000.0 | 1131.0 | 0.0 | 0.0 | 0.0 |
| 18 | 15431.0 | 14700.0 | 731.0 | 0.0 | 0.0 | 0.0 |
| 19 | 16325.0 | 15600.0 | 725.0 | 0.0 | 0.0 | 0.0 |
| 20 | 16905.0 | 16200.0 | 705.0 | 0.0 | 0.0 | 0.0 |
| 21 | 17213.0 | 16500.0 | 713.0 | 0.0 | 0.0 | 0.0 |
| 22 | 16099.0 | 15000.0 | 1099.0 | 0.0 | 0.0 | 0.0 |
| 23 | 15273.0 | 14300.0 | 973.0 | 0.0 | 0.0 | 0.0 |
| 24 | 14217.0 | 13500.0 | 717.0 | 0.0 | 0.0 | 0.0 |

Where PSTCOST / PSDCOST, FTRCOST - premature startup / shut down penalty cost and failure to meet reserve requirement penalty costs respectively.

The sample results included in table 7.19 and 7.20 indicate the hourly unit status as well as the available capacities over the scheduling period. All the units satisfy the unit minimum unit on /off times, and the reserve can be seen to cover the capacity of the largest committed unit.

### 7.3.4 Decomposed hybrid GA performance a 100 unit test system

Based on the general results of the 110 unit test system in the previous section, the decomposed hybrid GA solution method was applied to a test system with 100 units [Karzalis et. al.]. This system, though of approximately the same size as the 110 system, had a completely different set of load profile and unit characteristics. This test system had a number of units with very similar input / output characteristics which can slow the convergence of the optimisation process. Table 7.21 show the results of the decomposed hybrid GA method on the 100 unit test system.

**Table 7.21 Summary of best scheduling results for the 100 unit system**

| Solution method | Production cost | % improvement over the priority list |
|---|---|---|
| Decomposed hybrid GA | 5,542,611 | 1.53 |
| Other GA [Karzalis and Barkitzis] | 5,599,124 | 0.53 |
| Priority List | 5,628,957 | - |

The decomposed hybrid GA solution once again shows a marked improvement on the priority list solution. The decomposed hybrid GA solutions were 1.53% better than the priority list solution and 1% better than another GA implementation [Karzalis et. al.]. In their GA implementation, [Karzalis, Barkitzis and Petridis] found that their GA performed better than a Lagrangian relaxation based method which they had implemented. Their results show that, the performance of a number of scheduling algorithms is very dependent upon the implementation. Even the performance of Lagrangian relaxation, which is thought to be one of the most promising methods for large scale scheduling is dependent on the way it is implemented. It is because of the difficulties in getting a scheduling algorithm with a uniform performance, that the simple but powerful priority list based scheduling method was preferred for making the comparisons. The results of a typical run are presented in the next section.

#### 7.3.4.1 Sample decomposed hybrid GA results on 100 unit system

The experimental simulation conditions were: reserve set at 10% of load demand, truncation fitnes scaling, elitist reproduction 10 copies per generation, uniform cross over, cross over rate = 1.0, mutation rate = 0.005, generational replacement, generation gap = 1.00, completely random initial start, merit order economic dispatch, population size = 100,

average no. of generations per time step = 100, scheduling period = 24 hours, no. of units = 100.

**Table 7.22 A decomposed hybrid GA unit commitment list (100 test system).**

| HOUR | UNITS ( 1 ~ 100 ) |
|---|---|
| 1 | 1100000000110000000011000000001100000000110000000011000000001100000000110000000011000000001100000000 |
| 2 | 1100000000110000000011000000001100000000110000000011000000001100000000110000000011000000001100000000 |
| 3 | 1101000000110100000011000000001100000000110000000011000000001100000000110000000011000000001100000000 |
| 4 | 1111000000110100000011010000001101000000110100000011010000001101000000110100000011010000001101000000 |
| 5 | 1111000000111100000011110000001111000000111100000011010000001101000000110100000011010000001101000000 |
| 6 | 1111100000111110000011111000001111000000111100000011110000001111000000111100000011110000001111000000 |
| 7 | 1111100000111110000011111000001111000000111100000011110000001111000000111100000011110000001111000000 |
| 8 | 1111100000111110000011111000001111000000111100000011110000001111000000111100000011110000001111000000 |
| 9 | 1111111000111111100011111110001111110000111111000011111000011111000011111000001111100000111110000 |
| 10 | 1111111100111111110011111111001111111100111111110011111100111111100111111110011111100001111110000 |
| 11 | 1111111101111111110111111111011111111101111111110111111101111111101111111100111111100011111111000 |
| 12 | 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111100 |
| 13 | 1111111100111111110011111111001111111100111111110011111100111111100111111110011111100111111111110 |
| 14 | 1111111000111111100011111110001111110000111111000011111000011111000011111000011111000001111111000 |
| 15 | 1111100000111110000011111000001111100000111110000011111000001111100000111110000011111000001111100000 |
| 16 | 1111100000111110000011111000001111100000111110000011111000001111100000111110000011111000001101100000 |
| 17 | 1111100000111110000011111000001111100000111110000011111000001111100000111110000011111000001101100000 |
| 18 | 1111100000111110000011111000001111100000111110000011111000001111100000111110000011111000001101100000 |
| 19 | 1111100000111110000011111000001111100000111110000011111000001111100000111110000011111000001101100000 |
| 20 | 1111111101111111110011111111001111111100111111110111111110011111101001111111100111111110011011111100 |
| 21 | 1111111000111111100011111110001111111000111111100011111110001101110000111111100011110110001101011000 |
| 22 | 1111011000110001100011000110001100011000111101100011010110001101010000111111100011100110001101011000 |
| 23 | 1110001000110001000011000000001100010000110000000011000000001101000000110010000011000110001100000000 |
| 24 | 1100000000110001000011000000001100000000110000000011000000001100000000110000000011000100001000000000 |

PRODUCTION COST = **5,542,611**

**Table 7.23 Total hourly capacity output, load demand, and penalty coefficients for the 100 unit system schedules of table 7.22**

| HR | capacity | load | reserve | PSTCOST | PSDCOST | FTRCOST |
|---|---|---|---|---|---|---|
| 1 | 9100.0 | 7000.0 | 2100.0 | 0.0 | 0.0 | 0.0 |
| 2 | 9100.0 | 7500.0 | 1600.0 | 0.0 | 0.0 | 0.0 |
| 3 | 9360.0 | 8500.0 | 860.0 | 0.0 | 0.0 | 0.0 |
| 4 | 10530.0 | 9500.0 | 1030.0 | 0.0 | 0.0 | 0.0 |
| 5 | 11050.0 | 10000.0 | 1050.0 | 0.0 | 0.0 | 0.0 |
| 6 | 12186.0 | 11000.0 | 1186.0 | 0.0 | 0.0 | 0.0 |
| 7 | 12672.0 | 11500.0 | 1172.0 | 0.0 | 0.0 | 0.0 |
| 8 | 13320.0 | 12000.0 | 1320.0 | 0.0 | 0.0 | 0.0 |
| 9 | 14375.0 | 13000.0 | 1375.0 | 0.0 | 0.0 | 0.0 |
| 10 | 15410.0 | 14000.0 | 1410.0 | 0.0 | 0.0 | 0.0 |
| 11 | 15960.0 | 14500.0 | 1460.0 | 0.0 | 0.0 | 0.0 |
| 12 | 16510.0 | 15000.0 | 1510.0 | 0.0 | 0.0 | 0.0 |
| 13 | 15410.0 | 14000.0 | 1410.0 | 0.0 | 0.0 | 0.0 |
| 14 | 14375.0 | 13000.0 | 1375.0 | 0.0 | 0.0 | 0.0 |
| 15 | 13320.0 | 12000.0 | 1320.0 | 0.0 | 0.0 | 0.0 |
| 16 | 13190.0 | 10500.0 | 2690.0 | 0.0 | 0.0 | 0.0 |
| 17 | 13190.0 | 10000.0 | 3190.0 | 0.0 | 0.0 | 0.0 |
| 18 | 13190.0 | 11000.0 | 2190.0 | 0.0 | 0.0 | 0.0 |
| 19 | 13270.0 | 12000.0 | 1270.0 | 0.0 | 0.0 | 0.0 |
| 20 | 15415.0 | 14000.0 | 1415.0 | 0.0 | 0.0 | 0.0 |
| 21 | 14301.0 | 13000.0 | 1301.0 | 0.0 | 0.0 | 0.0 |
| 22 | 12127.0 | 11000.0 | 1127.0 | 0.0 | 0.0 | 0.0 |
| 23 | 9932.0 | 9000.0 | 932.0 | 0.0 | 0.0 | 0.0 |
| 24 | 8805.0 | 8000.0 | 805.0 | 0.0 | 0.0 | 0.0 |

Where PSTCOST / PSDCOST, FTRCOST - premature startup / shut down penalty cost and failure to meet l costs respectively

## 7.4 Overview of performance of the various GA methods across the test systems

An approach combining theoretical foundations of genetic algorithms, careful design of experiments and rigorous empirical testing was adopted in the overall design process of the developed thermal scheduling genetic algorithms. Genetic algorithms are very simple algorithmically, but their operation mechanism is highly complex due to the non linear relationships among its control parameters, which makes it impossible to adopt an ad hoc incremental approach in the design of a successful GA. The basic thermal scheduling genetic algorithmic design steps used in this work are shown in figure 7.6.

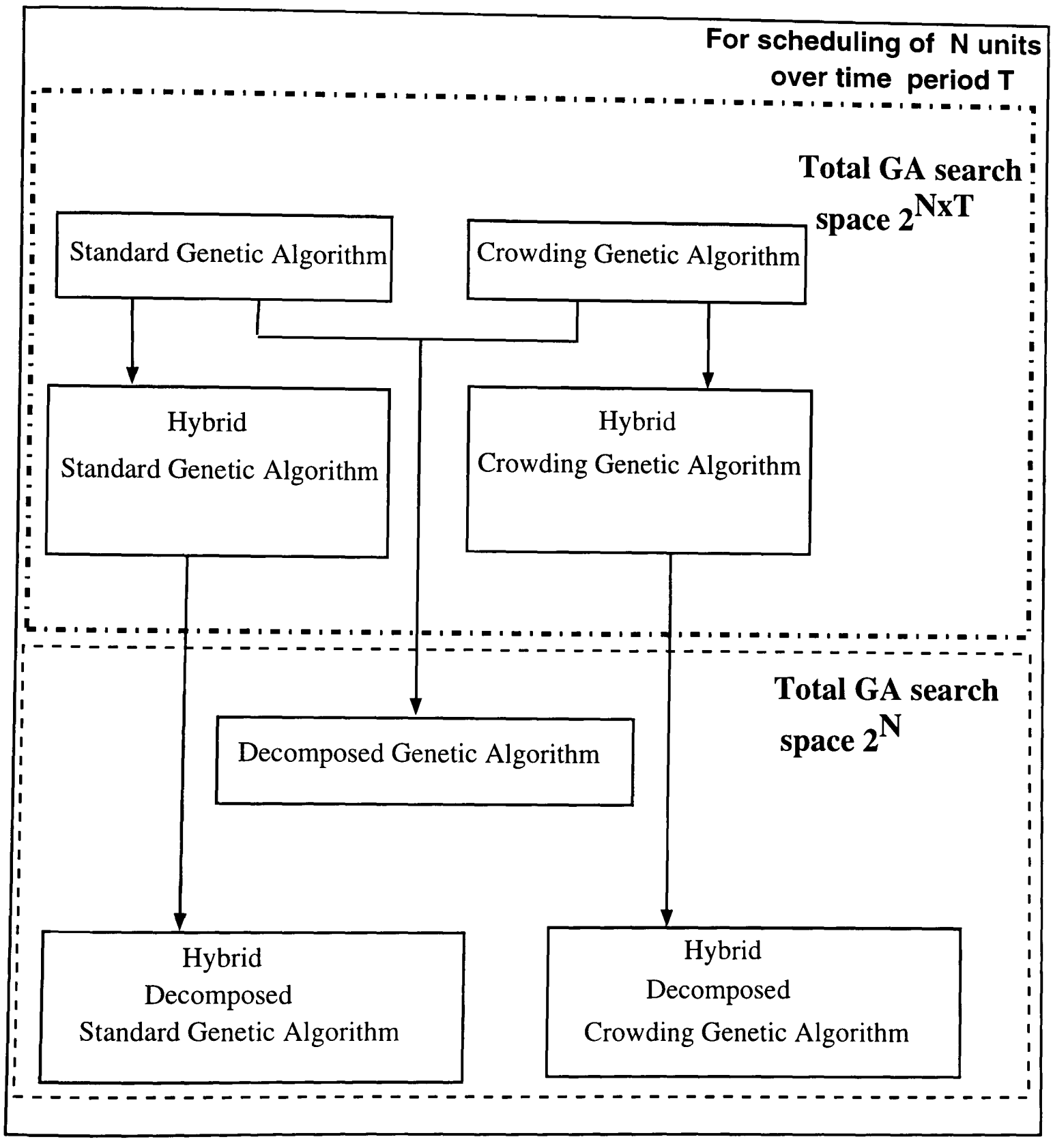



**Figure 7.6 Thermal scheduling GA algorithm design sequence**

Table 7.24 provides a *rough* guide line for comparing the performance of the developed GA thermal scheduling methods across the range of problem test systems. The results shown in the table are not necessarily the best that could be obtained by the GA methods, but are those on which reasonable comparisons could be made, based for instance on same number of function evaluations. If the parameters are fined tuned or the algorithm left to run for a longer time, it is possible to obtain better results than those shown in the table. More detailed comparisons are provided in previous sections.

**Table 7.24 Summary of the various GA scheduling results for the test systems**

| Test system | Percentage improvement over priority list | | | | | |
|---|---|---|---|---|---|---|
| | No decomposition | | | | Decomposed problem | |
| | No hybrid | | With hybrid | | Decomposed GA (No hybrid) | Hybrid GA |
| | Standard GA | Crowding GA | Standard GA | Crowding GA | | |
| 10 unit | 0.7 | 0.8 | 2.46 | 1.77 | 3.02 | 3.06 |
| 26 unit (load profile_1) | -3.02 | -0.22 | 0.46 | 0.44 | 0.67 | 0.70 |
| 26 unit (load profile_2) | -8.18 | 0.61 | 0.94 | 0.78 | 2.69 | 2.76 |
| 110 unit | -379 | -22.4 | 0.51 | 0.50 | 0.53 | 0.73 |

From table 7.24 it can be seen that larger test systems generally offer a greater challenge to the GA method than small tests systems. Apart from the system size, scheduling problem difficulty is also dependent on the system load curve as well as the unit input / output characteristics.

**Table 7.25 Average CPU times used to obtain the scheduling results for the various test systems shown in table 7.24.**

| Test system | (Hour : minutes . seconds) | | | | | |
|---|---|---|---|---|---|---|
| | No decomposition | | | | Decomposed problem | |
| | No hybrid | | With hybrid | | Decomposed standard GA ( No hybrid) | Hybrid standard GA |
| | Standard GA | Crowding GA | Standard GA | Crowding GA | | |
| 10 unit | 0:17 | 0:22 | 0:09 | 0:12 | 0:0.02 | 0:0.02 |
| 26 unit ) | 0:46 | 1:15 | 0:10 | 0:14 | 0:01 | 0:01 |
| 110 unit | 6:55 | 31:45 | 2:40 | 6:15 | 12:35 | 0:20 |

## 7.5 Conclusions

The computational speed limitation of a genetic algorithm solution method that uses a full string representation for generator scheduling becomes apparent as the system size is increased. The decomposed GA and hybrid decomposed GA methods are attempts to enable the GA to produce useful solutions within reasonable time limits. In this chapter it has been demonstrated that adding a fast heuristic scheduling method such as the priority list scheme to the genetic algorithm results in a hybrid GA capable of solving large scale thermal scheduling problems within acceptable computational time limits.

# Chapter 8. The Hydrothermal Scheduling Problem

## 8.1 Introduction

The optimal scheduling of generation in a hydrothermal system involves the allocation of generation among the hydro electric and thermal plants so as to minimise total operation costs of thermal plants while satisfying the various constraints on the hydraulic and power system network. In short term scheduling, the total volume of water or power expected to be generated by each hydro plant over the scheduling period is fixed. It is assumed that the target dam levels at the end of the scheduling period have been set by a medium term scheduling process that takes into account longer term river inflow modelling and load predictions. The short term scheduler then allocates this water (power) to the various time intervals in an effort to minimise thermal generation costs while attempting to satisfy the various unit and reservoir constraints.

The main constraints include:

- time coupling effect of the hydro sub problem, where the water flow in an earlier time interval affects the discharge at a later period of time,
- the varying system load demand,
- the cascade nature of the hydraulic network,
- system hourly load demand,
- hourly reservoir inflows,
- reservoir storage and turbine flow rate limits,
- dynamic hydraulic flow continuity equations,
- minimum and maximum loading limits of both thermal and hydro plants.

Further constraints could be imposed depending on the particular requirements of a given power system, such as the need to satisfy activities such as; flood control, irrigation, fishing, water supply etc. In a hydrothermal power system, apart from replacing the thermal generation which would have incurred a given fuel consumption, the hydro electric power

generation is usually responsible for providing frequency regulation, by taking advantage of its fast load pick up characteristic.

## 8.2 Modelling The Hydrothermal Scheduling Problem

In a hydrothermal power system, the thermal generation is used to supply that part of the load demand that cannot be supplied by the hydro generation. A mathematical formulation of the hydrothermal scheduling problem in a multi-reservoir cascaded hydro electric system with a non linear relationship between water discharge rate, net head and power generation, and water transport delay is presented in the next section.

### 8.2.1 Problem objective function

The basic optimal hydrothermal scheduling in the short term, involves minimising the thermal cost function, F,

$$Min\ F = \sum_{t \in T} \sum_{i \in R_s} F_i\left(P_{s\,i,t}\right) \qquad i \in R_s \quad t \in T \tag{8-1}$$

subject to a number of unit and power system network equality and inequality constraints. More advance models account for the power loss in the transmission networks, mostly through D.C load flow models. The thermal unit commitment is assumed known, and only the unit generation levels are to be determined.

### 8.2.2 System constraints

#### 8.2.2.1 System active load balance

The total active power generation must balance the predicted power demand plus losses, at each time interval over the scheduling horizon

$$\sum_{i \in R_s} P_{s\,i,t} + \sum_{i \in R_h} P_{h\,i,t} = P_{D,t} + P_{L},t \qquad t \in T \tag{8-2}$$

#### 8.2.2.2 Transmission line constraints

The power transported by the transmission lines must not violate their maximum loading limits. Transmission limits constraints are particularly important in systems with major hydro components, as the hydro generation stations are usually located far from load centres.

### 8.2.3 Unit constraints

In the hydrothermal scheduling problem, both the hydro and thermal units loading levels are limited by the physical limitations on the generating units. Thus

1) the thermal plant loading limits must be satisfied,

$$P_{si,t}^{min} \leq P_{si,t} \leq P_{si,t}^{max} \qquad t \in T \tag{8-3}$$

2) the hydro plant loading limits must be satisfied,

$$P_{hi,t}^{min} \leq P_{hi,t} \leq P_{hi,t}^{max} \qquad t \in T \tag{8-4}$$

Other unit constraints such as ramp rates, prohibited unit operating regions, plant crew limitations and unit must run and must out status additionally constrain the scheduling process.

### 8.2.4 Hydraulic network constraints

The hydraulic operational constraints comprise the water balance (continuity) equations for each hydro unit (system) as well as the bounds on reservoir storage and release targets. These bounds are determined by the physical reservoir and plant limitations as well as the multipurpose requirements of the hydro system. These constraints include;

1) physical limitations on reservoir storage volumes and discharge rates,

$$\begin{aligned} V_{hi,t}^{min} &\leq V_{hi,t} \leq Vh_{i,t}^{max} \qquad t \in T \\ Q_{hi,t}^{min} &\leq Q_{hi,t} \leq Q_{hi,t}^{max} \qquad t \in T \end{aligned} \tag{8-5}$$

2) the desired volume of water to be discharged by each reservoir over the scheduling period,

$$\begin{aligned} V_{hi,t}\big|^{t=0} &= V_{h,i}^{begin} \qquad i \in R_h \\ V_{hi,t}\big|^{t=T} &= V_{h,i}^{end} \qquad i \in R_h \end{aligned} \tag{8-6}$$

3) the continuity equation for the hydro reservoir network

$$\begin{aligned} &V_h(i,t)=V_h(i,t-1)+I_h(i,t)-Q_h(i,t)-S_h(i,t) \\ &+\sum_{m=1}^{R_u}\Big[Q_h\big(m,t-\tau(i,m)\big)+S_h\big(m,t-\tau(i,m)\big)\Big] \\ &\qquad i\in R_h \quad and \quad t\in T \end{aligned} \tag{8-7}$$

The hydraulic network constraints provide the biggest challenge to the hydrothermal co-ordination problem. An example of a hydraulic network for a practical power system is shown in figure 9.1, showing the Kenya hydro power network. For example, in that system, apart from providing power generation, the reservoir water release strategy must satisfy irrigation, fishing, navigation and river flood control requirements.

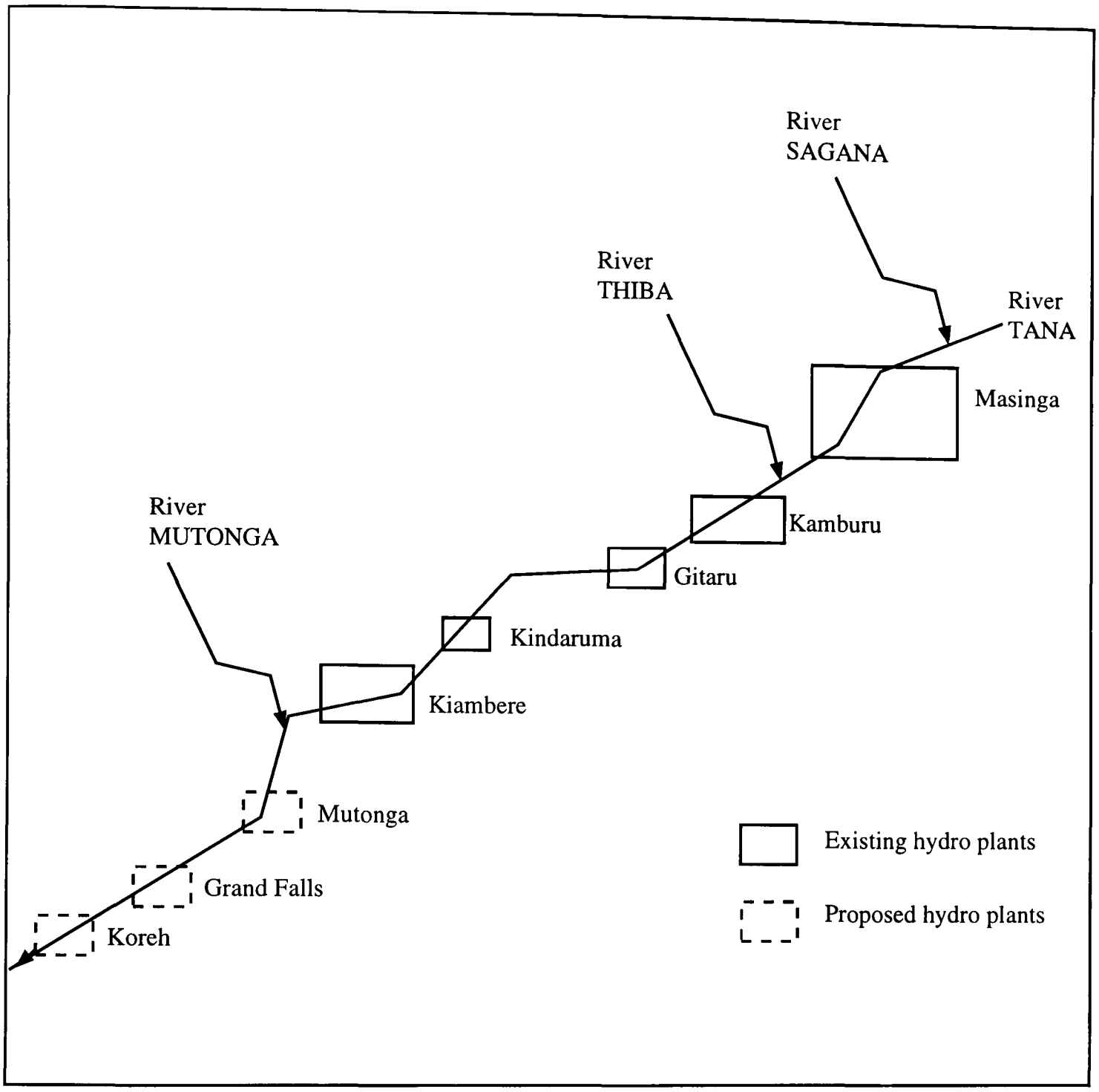



**Figure 8.1 Example of a hydro network - The Kenya power system hydraulic network**

### 8.2.5 Hydro plant power generation characteristics

The power generated from a hydro plant is related to the reservoir characteristics as well as the water discharge rate. A number of models [El-Hawary and Christensen] have been used to represent this relationship. In general, the hydro generator power output is a function of the net hydraulic head, H, reservoir volume, V, and the rate of water discharge, Q,

$$P_{hi,t} = f\left(Q_{hi,t},\ V_{hi,t}\right) \text{ and } V_{hi,t} = f\left(H_{i,t}\right) \qquad (8\text{-}8)$$

and when a variable head and efficiency is taken into account, a double quadratic function can be used to relate the hydro power generation to the net head and discharge,

$$P_{hi,t} = \left[\alpha_{i,0} + \alpha_{i,1} H_i + \alpha_{i,2} H_i^2\right]\left[\beta_{i,0} + \beta_{i,1} Q_i + \beta_{i,2} Q_i^2\right] \qquad (8\text{-}9)$$

where $\alpha_i$ and $\beta_i$ are constants, representing reservoir and turbine characteristics. The model can also be written in terms of reservoir volume instead of using the reservoir net head, and a frequently used functional is

$$P_{hi,t} = C_{1,i} V_{hi,t}^2 + C_{2,i} Q_{hi,t}^2 + C_{3,i}\left(V_{hi,t}\right)\cdot\left(Q_{hi,t}\right) + C_{4,i} V_{hi,t} + C_{5,i} Q_{hi,t} + C_{6,i} \qquad i \in R_h \qquad (8\text{-}10)$$

Net head variation can only be ignored for relatively large reservoirs, in which case power generation is solely dependent on the water discharge.

### 8.2.6 Thermal cost function

In setting the generation levels of the thermal plants, a quadratic cost function is frequently used to model the fuel input / power output characteristic of thermal units,

$$F\left(P_{si,t}\right) = \alpha_i + \beta_i P_{si,t} + \gamma_i P_{si,t}^2 \qquad i \in R_s \qquad (8\text{-}11)$$

## 8.3 Review of Hydrothermal Generation Scheduling Techniques

Hydrothermal scheduling involves the optimisation of a problem with a non linear objective function, with a mixture of linear, non linear and dynamic network flow constraints. The problem difficulty is compounded by the following practical constraints:

1. non linear relationship between turbine discharge and net head, which is itself a function of reservoir storage,
2. non linear relationship between hydro power output and turbine discharge,
3. the dynamic time linkage in the cascaded flow continuity equations,
4. the variable reservoir inflow rates,
5. the variable load demand,
6. non linear cost function representing the thermal unit characteristics.

Unless several simplifying assumptions are made, this problem is difficult to solve for practical power systems using conventional optimisation techniques.

The hydrothermal scheduling problem has been the subject of investigation for several decades now. Most of the methods that have been used to solve the hydro thermal co-ordination problem make several simplifying assumptions in order to make the optimisation problem more tractable. Most of the models make one or more of the following assumptions:

- linear relationship between discharge, reservoir storage and power output,
- ignore cascaded networks,
- ignore water transport delay between series reservoirs,
- assume constant reservoir water heads,
- ignore transmission network constraints.

[El-Hawary and Christensen], [Christensen and Soliman] provide a detailed evaluation of the major hydrothermal scheduling methods to date. Some of these solution methods are reviewed in the next section.

### 8.3.1 Variational Calculus Based Techniques

Most of the initial attempts to solve the hydrothermal scheduling problem [Chandler et. al.] [Drake et. al.] relied on classical variational calculus based methods. Most of these techniques made several simplifying assumptions in the treatment of the numerous problem constraints.

### 8.3.2 Dynamic Programming

Dynamic programming is in principle, the optimisation technique most suitable for this type of problem, in the sense that it can theoretically handle, all the constraints. It is however limited by storage and computation time requirements or the so called *curse of dimensionality.* Various approximations have have been proposed to deal with the dimensionality problem and these include;

- Aggregation of a multi reservoir into a single equivalent reservoir, and then solving the resulting problem using dynamic programming [Arvanitidis and Rosing].
- Dynamic programming with successive approximation [Wood and Wollenberg], [Chang et. al.].
- Aggregation - decomposition approach [Duran et. al. ], [Turgeon, 1980].
- Successive linear programming combined with dynamic programming [ Yang and Chen]

### 8.3.3 Functional Analysis

Pontryagin's maximum principle [Popargogiou], [Bubenko and Waern] [Dahlin and Shen] has been applied to the hydrothermal scheduling problem, however its main limitation is the numerical difficulties encountered in solving the resulting two point boundary value problems, especially when the problem constraints are taken into account. [Bubenko and Waern] try to simplify the analysis by maximising the hydro energy output. [Soliman and Chritenesen, 1986], [Lee, B.Y and K.Y. Lee] have also used the functional analysis in modelling the hydrothermal co-ordination problem.

### 8.3.4 Network Flow and Linear Programming

Network flow (NF) programming [Brannund et. al.], [Habibollahzadeh et. al.] is the basic technique mostly applied to large scale hydrothermal scheduling problems. The network flow technique is able to handle a non linear objective function and a set of linear constraints. [Wakamori et. al.], [Xia et. al.] use a hydrothermal co-ordination solution method that combines linear programming and network flow programming, while [Liang and Hsu] combine fuzzy logic with linear programming to solve the scheduling problem. By formulating the optimal hydro scheduling problem as a large scale linear programming problem, [Piekutowski et. al.] use a commercial linear programming package to solve the problem for a large scale cascaded hydro system.

### 8.3.5 Non Linear Programming

[Saha and Khapade] apply the direct method of feasible directions to the scheduling problem. The fundamental basis of their method is, starting from a feasible solution point, find a direction of movement towards the next feasible point, and in the process hopefully reach the optimal point. This method is prone to convergence difficulties and the probability of finding the next feasible point is quite small when many problem constraints are considered. [Sokkapa et. al.] employ the steepest descent method using a discrete time model to provide solutions for the scheduling problem. Their method was only able to deal with small test networks, otherwise this gradient technique would equally run into convergence difficulties as the system is scaled up. [Brainbridge et. al.] use a a combination of the gradient method and dynamic programming technique in the scheduling of a system that contains pump storage plants. The gradient procedure is used to to model the hydro subsystem, while DP is used to solve the thermal subproblem.

### 8.3.6 Mathematical Decomposition

[Habibblolahzadeh and Bubenko], apply a Lagrangian decomposition co-ordination techniques to the hydrothermal scheduling problem. They decompose the problem into both the hydro and thermal sub problems, and solve the hydro component using network flow techniques, while they employ the branch and bound method to solve the thermal unit sub

problems. They then apply a sub gradient algorithm to provide estimates of the Lagrangian multipliers used in obtaining the primal solution. [Wang and Shahidepour, 1993] use a mathematical decomposition technique to treat the problem of multi-area hydrothermal scheduling, incorporating cascaded reservoir and uncertain load data. They decompose the problem into hydro and thermal sub systems and use a system Lagrangian multiplier to co-ordinate the two solutions. They solve the hydro subproblem using a network flow concept that includes a gradient technique to handle the problem constraints. [Pereira and Pinto], [Soares and Lyra] apply a decomposition co-ordination approach to the hydrothermal scheduling problem.

### 8.3.7 Heuristics, Expert Systems and Artificial Neural Networks

[Liang and Hsu], use an artificial neural network to solve the hydro thermal scheduling problem. Recently [Bornaert et. al.], [Luo and Habibollahzadeh] have provided detailed mathematical models, including such factors as transmission line power flow modelling in the hydrothermal optimisation process. [Soares and Ohishi] use a hybrid simulation optimisation approach for the solution of a hydro dominated power system, while [Wong K.P. and Wong Y.N.] use a simulated annealing approach for the hydrothermal scheduling problem. The main difficulty with most of these artificial intelligence based solution techniques is the balance between modelling accuracy, solution time and accuracy of the final solution.

### 8.3.8 Evolutionary Computation Techniques

[Hulsemann et. al.] combine linear programming, genetic algorithms and simulation techniques in the scheduling of a cascaded *run of river* hydro system. They use the linear programming method to provide a coarse solution to the hydro optimisation problem, before applying a genetic algorithm for optimisation with the non linear variables taken into account, and finally using simulation to remove any constraint violations.

# Chapter 9. Hydrothermal Co-ordination Using Genetic Algorithms

## 9.1 Introduction

The short range hydrothermal scheduling problem (usually 1 day to one week) involves the hour by hour scheduling of all the generation on a system to achieve minimum operation costs for the given time period. The solution to this problem can be defined by specifying the actual load allocated to various hydro and thermal plants at each time step, over the scheduling period. In hydro generation, the basic performance curve is expressed in terms of the water input versus power output hence the turbine water discharge can be used as the problem decision variable. In this work, it is assumed that the thermal unit commitment decision is known *a priori* and that the thermal generation provides the generation that cannot be supplied by the hydro sub-system. The basic optimal hydrothermal scheduling sequence is: assuming a given thermal unit commitment, load demand, and hydraulic inflows, allocate load to the various hydro and thermal units, while satisfying the individual unit loading limits, hydraulic constraints and power network constraints so that the total operation cost is minimised. Included with hydraulic constraints is the desire to satisfy *end point* conditions for the scheduling period in order to conform to medium term water release targets. A genetic algorithm (GA) approach is used to solve this scheduling problem by searching for the optimal dispatch of thermal and hydro units which satisfies the various problem constraints.

## 9.2 GA Representation of the Hydrothermal Scheduling Problem

For the GA solution method, the water discharge through the turbines during each optimisation interval is used as the main control variable. Other possible decision parameters are the hydro plant power outputs and the changes in reservoir volumes. Knowing the water discharge at each plant, the reservoir inflows and the unit characteristic equations, the change in reservoir storage and the hydroelectric power outputs can easily be evaluated. In the GA binary problem representation, the various water discharge rates at each reservoir for each time interval are represented by a given number of binary strings. The number of bits representing each reservoir depends on the required (resolution) accuracy within which the turbine discharge level can be varied. The total solution string length is obtained by concate-

nating all the sub-strings that represent the individual reservoirs in the various time intervals. A typical problem solution shown in figure 9.1 is for a system consisting of N hydro plants over a scheduling period T. The discharge from each plant varies over the scheduling period, for example, in the first time interval, the discharge of the second plant is represented by the string 1 1 0 0, while in the second interval it is represented by 1 0 0 0.

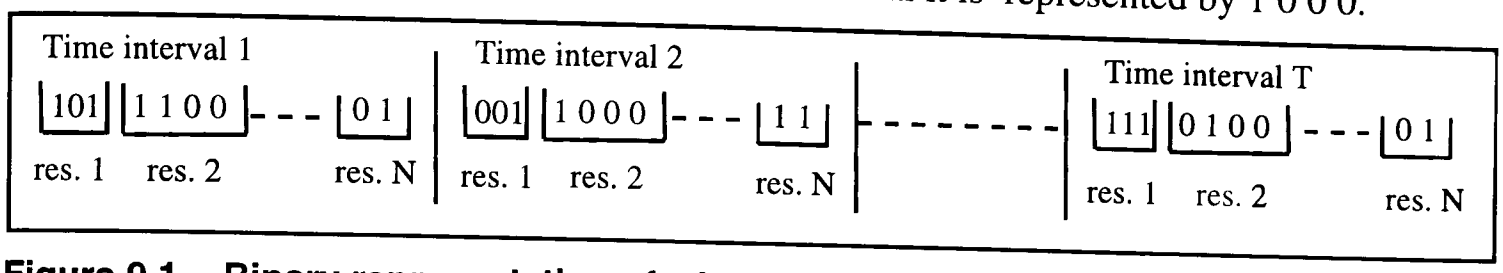



**Figure 9.1 Binary representation of a hydrothermal scheduling solution**

In a GA optimisation process using binary encoding, the solution accuracy depends on the number of bits used to represent the decimal equivalent of the control parameter. The higher the number of bits used, the finer the resolution. The control variable precision, $\xi$, is given by ;

$$\xi = \left(U^{max} - U^{min}\right) / \left(2^L - 1\right) \qquad (9\text{-}1)$$

where $U^{max./min.}$ are the maximum (minimum) values of the turbine discharges, and $L$ is the number of bits used for encoding each plant's discharge. The precision required is chosen according to the solution accuracy desired. The resolution can be given either as a decimal number or as the equivalent number of bits required for each control parameter. The equivalent decimal number representation, $D$, is given by;

$$D = U^{min} + \xi \times (INT) \qquad (9\text{-}2)$$

where $INT$ is the integer value of the binary representation. To illustrate the effect of binary representation on solution accuracy, consider a scheduling problem with 4 reservoirs, where plant water discharge is used as the control variable, with the specified number of binary bits for representing the various dam discharge rates given in column 4 of table 9.1. The corresponding decimal resolution for each variable is given in column 5 of the table.

**Table 9.1 Relationship between binary representation and solution accuracy**

| Dam | $Q_{min}$ | $Q_{max}$ | number of binary bits | Actual decimal resolution |
|---|---|---|---|---|
| 1 | 5 | 15 | 3 | 1.429 |
| 2 | 6 | 15 | 3 | 1.286 |
| 3 | 10 | 30 | 4 | 1.333 |
| 4 | 13 | 25 | 3 | 1.714 |

**Table 9.2 Relationship between binary representation and plant discharge**

| Dam | $Q_{min}$ | $Q_{max}$ | binary string | discharge |
|---|---|---|---|---|
| 1 | 5 | 15 | 0 1 1 | 9.29 |
| 2 | 6 | 15 | 1 1 0 | 13.71 |
| 3 | 10 | 30 | 0 1 1 1 | 19.33 |
| 4 | 13 | 25 | 1 0 1 | 21.57 |

The actual decimal resolution is fixed by the number of bits used for binary representation. The sub string length in each time interval is the sum of the number of bits used to represent each plant's discharge. The total solution string length over the whole scheduling period is given by concatenating the bits representing each plant over the whole scheduling period. For example, the binary solution sub-string,

| 0 1 1 1 1 0 0 1 1 1 1 1 0 1 |
|---|

also shown on table 9.2 represents the discharge of 4 reservoirs in one time interval. The substring length is 13, which for a 24 hour scheduling period will result in a solution string of length 312 bits, giving a total GA search space of $2^{312}$. The binary solution string must represent the whole scheduling period, to take into account the river flow dynamics resulting from the hydraulic coupling effects between the hydro plants on the same stream.

## 9.3 Fitness function

A genetic algorithm conventionally searches for the optimal solution by maximising a given fitness function and therefore an evaluation function which provides a measure of the quality of the problem solution must be provided. For the hydrothermal co-ordination problem, the evaluation function is a combination of the thermal cost function and the penalty function terms that take into account the various system, unit and hydraulic network constraint violations. The evaluation function should differentiate between good and poor solutions, both in the feasible and infeasible search domains. The fitness value is critical to the functioning of the genetic algorithm, since it is this function that determines an individual's ability (chance) to undergo selection hence propagate its features to future generations. The objective of hydrothermal scheduling is the minimisation of the thermal cost, while satisfying the various hydraulic and system constraints. Since a GA maximises the fitness function, the minimisation objective function must be transformed into a maximisation problem. Solution of the scheduling problem involves a minimisation of the composite function, F;

$$F = \sum_{i \in R_s} F_i\left(P_{s,i}\right) + \sum_{i \in R_h} \Phi\left(V_{h,i}\right) + \sum_{i \in R_h} \Psi\left(V_{h,i}(end)\right) + \sum_{i \in R_h} \Omega\left(P_{h,i}\right) + \sum_{i \in R_s} \theta\left(P_{s,i}\right) \tag{9-3}$$

where:

$F_i(P_{s,i})$ is the optimal dispatch cost of the thermal plants,

$\Phi$, the penalty function for reservoir storage capacity limit violation,

$\Psi$, the penalty function for final (end condition) reservoir level violation,

$\Omega$, the penalty function for hydro unit loading limit violation, and

$\theta$, the penalty function for thermal unit loading limit violation.

The thermal fuel cost is implicitly related to the sum of the hydro power generation (hence discharge) and the load demand, $P_D$, according to the following load balance equation,

$$\sum_{i \in R_s} \left(P_{s,i}\right) = P_D - \sum_{i \in R_h} \left(P_{h,i}\right) \qquad (9\text{-}4)$$

The following equation is used to represent the fuel cost / power output characteristics of the equivalent composite thermal plant,

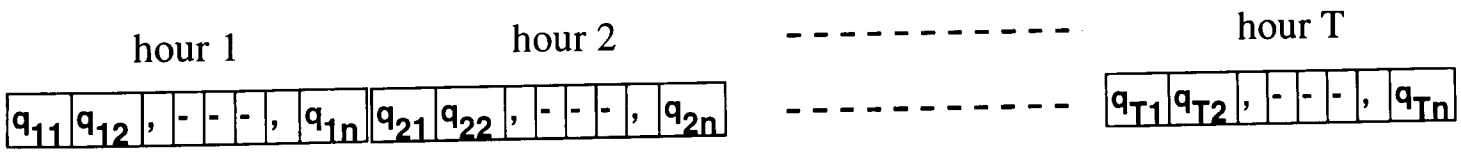
$$F_i\left(P_{s,i}\right) = 5000 + 19.2 P_{s,i} + 0.002 {P_{s,i}}^2 \quad and \quad 500 \le P_{s,i} \le 2500 \qquad (9\text{-}5)$$

while the power output of the $i^{th}$ hydro unit during the time interval t, $P_{h,i}(t)$ as a function of reservoir volume $V_{h,i}(t)$ and discharge $Q_{h,i}(t)$ is given by :

$$P_{hi,t} = C_{1,i} {V_{hi,t}}^2 + C_{2,i} {Q_{hi,t}}^2 + C_{3,i}\left(V_{hi,t}\right) \cdot \left(Q_{hi,t}\right) + C_{4,i} V_{hi,t} + C_{5,i} Q_{hi,t} + C_{6,i} \qquad i \in R_h \qquad (9\text{-}6)$$

where $C_1, \ldots, C_6$, are the hydro plant generation coefficients.

The fitness function is obtained by decoding the binary solution string. The decoded solution gives the actual decimal values of the plant discharge over the whole scheduling period as shown in figure 9.2 for N hydro plants over a scheduling period, T.




*where* $q_{t,i}$ is the discharge of the $i^{th}$ plant during time step t

**Figure 9.2 An example of a decoded binary solution string**

The fitness function is obtained through a sequence of events as shown in figure 9.3.

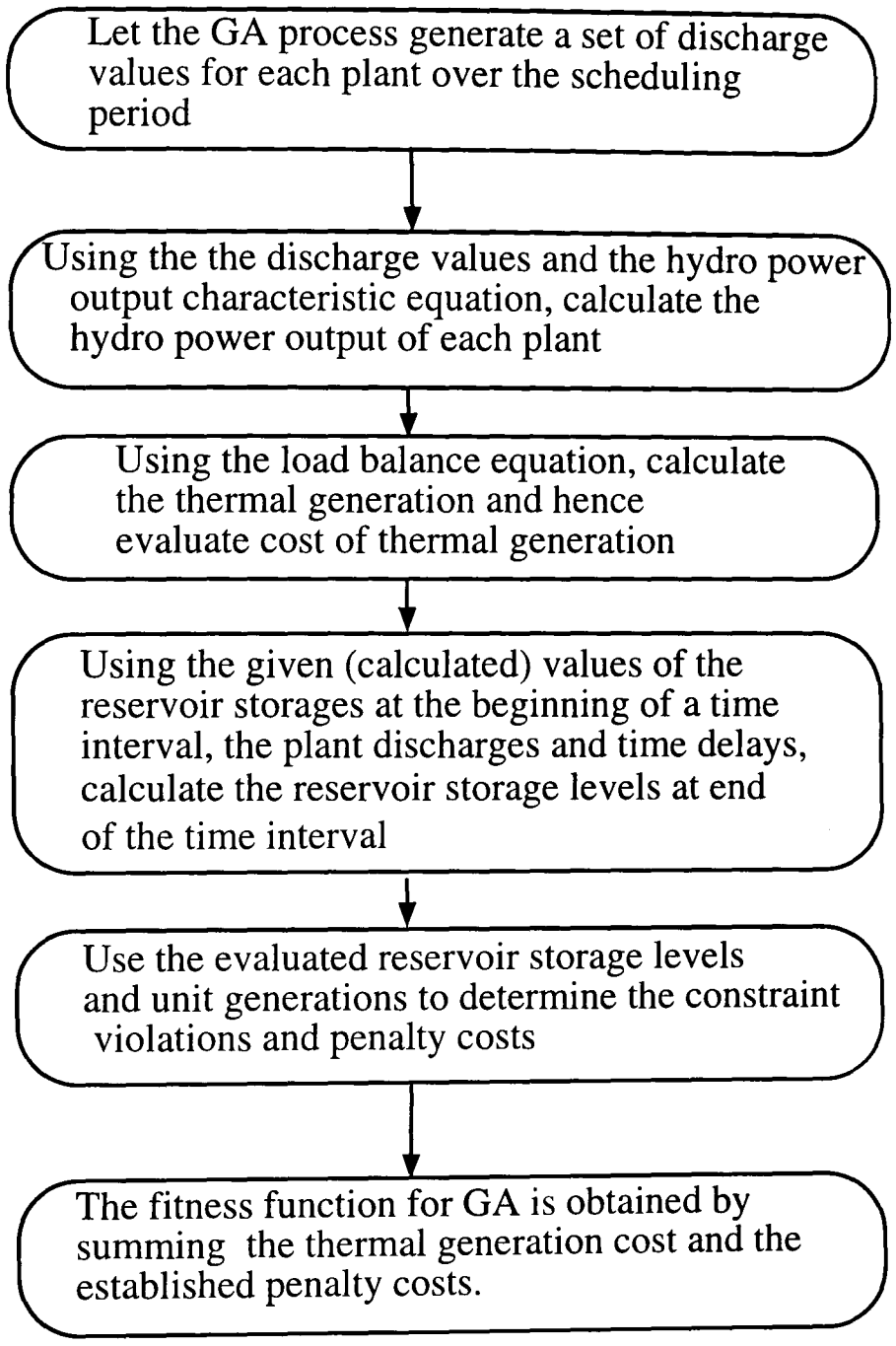



**Figure 9.3 The hydrothermal scheduling GA fitness evaluation sequence**

### 9.3.1 Penalty function grading for the hydrothermal scheduling GA

Finding a solution that satisfies all the hydrothermal scheduling problem constraints is quite difficult. A penalty function approach that takes into account the violation of the problem constraints is adopted in this work. The penalty functions try to force the unconstrained optimum towards the feasibility boundary by incorporating penalty terms into the fitness function to penalise strings that violate the constraints. With the penalty function, the evaluation function, $f$, can be written in the generic form,

$$f = c(x) + \sum_{i=1}^{m} \lambda_i \phi_i(x) \qquad (9\text{-}7)$$

where $c(x)$ is the cost function $\lambda_i$ and $\Phi_i$ the $i^{th}$ penalty coefficient and penalty functions respectively, for the $m$ constraints. The choice of the penalty term can be significant. If the penalty term is too harsh, infeasible strings that carry useful information for the GA search,

but lie outside the feasible region will largely be ignored and their information lost. If the penalty term is not strong enough, the GA may search only among infeasible strings, and miss the feasible solutions.

In this work, a quadratic penalty function is adopted and the penalty coefficients, $\lambda_i$, are set to provide an upper bound on the cost to satisfy the violated constraints. This means that the penalty terms are set so that all feasible strings are always awarded a higher fitnes than infeasible ones, an approach that avoids the difficulties usually encountered in choosing appropriate penalty coefficients, $\lambda_i$, while allowing infeasible solutions into the population.

The penalty boundaries for the hydraulic reservoir are shown in figure 9.4. For example, the same violation of either the reservoir maximum or minimum storage levels must be awarded the same penalty costs, otherwise if one side is penalised more, the population will tend to drift towards the less penalised side. The GA treats the desired final reservoir levels (end volume) as soft constraints which can be violated or relaxed, while the maximum (minimum) allowable reservoir levels must not be violated. Similarly the hydro and thermal unit loading limits must not be violated. The plant discharge limits are never violated because they are implicitly made to vary within their allowable limits when used as the encoded GA decision variable.

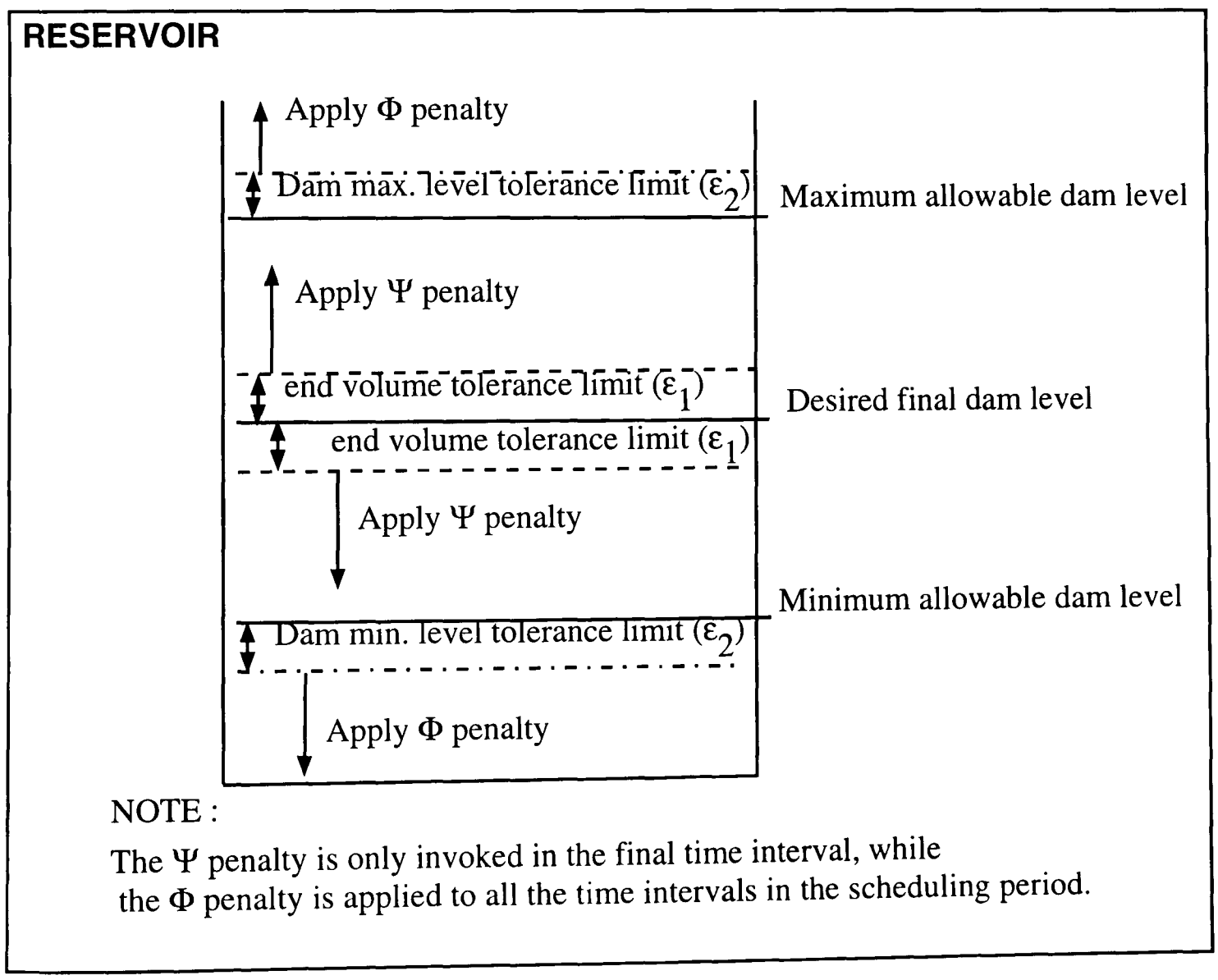



**Figure 9.4 Reservoir constraints penalty boundaries**

## 9.4 Implementation of the Hydrothermal Scheduling GA

Once a method of awarding a fitness to each member of the population has been determined, the standard GA search sequence of population creation, selection, crossover, mutation and parent replacement can be applied to the chosen GA model. A canonical GA model, similar to the one described in chapter 6, has been implemented for the hydrothermal co-ordination problem. The deterministic crowding GA model could equally have been implemented since both GA models have been shown to perform well if a proper problem structure is used. In addition to the usual GA techniques of GA control parameter tuning, a number of enhancements, mainly derived from hydrothermal scheduling problem structure, have been incorporated in the standard GA search mechanism to enable it to solve a wide range of hydrothermal scheduling problems.

### 9.4.1 Modifications to the basic hydrothermal GA search process

In an optimisation process, an appropriate step length for changing each parameter variable at each stage of the optimisation sequence, (or problem resolution for GA), must be chosen. If the step lengths are too small, it can take a large number of iterations to reach the optimal solution, while if step lengths are too large, the optimum solution can only be crudely approached and the optimisation can easily get stuck at a local optimum. For a GA search, the smaller the resolution, the longer the string lengths, which results in longer solution times. Thus the string length of the GA affects the quality of solutions provided by the GA. The GA can perform a multiple step search by using different string lengths for different stages of the optimisation. This allows a coarse grained search in the initial stages of the GA process, which are used as starting points for later runs with finer resolutions. These enhancements are described in the next section.

#### 9.4.1.1 Multiple step GA search process using variable time-steps

Exploiting the relationships between the hydrothermal scheduling problem structure over successive time intervals can reduce the problem size, by providing approximate solutions based on time decomposition. This basically involves varying the step time interval for the scheduling process. For example two hour time intervals can be used instead of one hour, reducing the problem size by half. The solutions obtained from the longer time interval are then used as the starting optimisation point for the desired final time step, a process that should result in a shorter overall solution time, with possible improvements in the solution quality. Using longer time intervals would for example, consider using average load demands and water inflows over these intervals.

The effect of using longer time intervals to solve the hydrothermal scheduling problem was investigated. It is to be expected, that the longer the time interval, the less accurate the results, due to the loss of accuracy in modelling of the load demand and river inflows. The

searches based on longer time intervals however are able to reach the more promising areas of the search space much faster than shorter time intervals, because of the smaller problem dimension, and their results can be used as starting points for the shorter time intervals resulting in a speed up of the optimisation process. Figure 9.5 shows the effect of varying the magnitude of the scheduling time intervals on the convergence of the GA. The longer the time interval, the more rapid the GA convergence towards the promising areas of the search space.

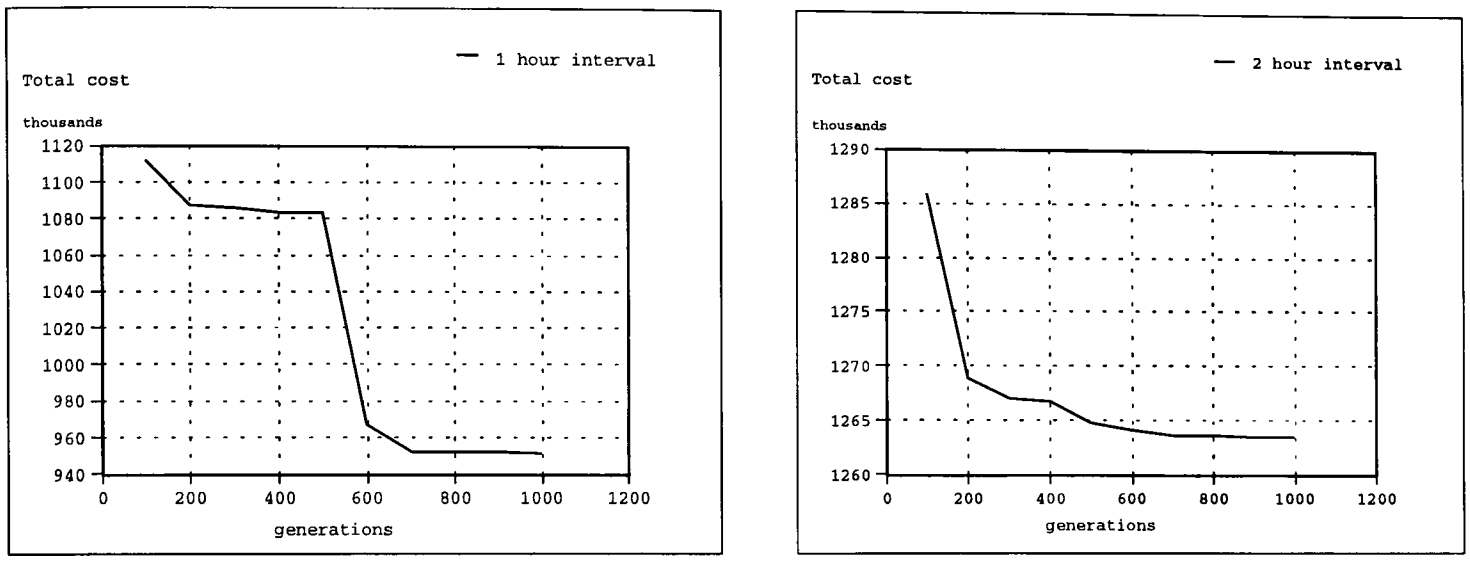



**Figure 9.5 Effect of varying the magnitude of the scheduling time intervals on GA convergence**

#### 9.4.1.2 Multiple step GA search process using variable control parameter resolution

The multiple step search uses different string lengths (parameter resolutions) for the various stages of the GA run. This involves a change in discharge resolution, in which the search starts off with short string lengths which are progressively increased in the course of the optimisation. If a binary representation is used, then at the end of each step, the best solution already obtained is converted to the equivalent binary representation required for the next step of the GA run.

The performance of the GA depends on the resolution chosen for the control variable. If, the resolution is too large, the GA will tend to converge prematurely, while if it is too fine, the convergence might take too long, and therefore a reasonable balance must be made between the resolution accuracy and the convergence time. Figure 9.6 illustrates the variation of the GA convergence with parameter resolution.

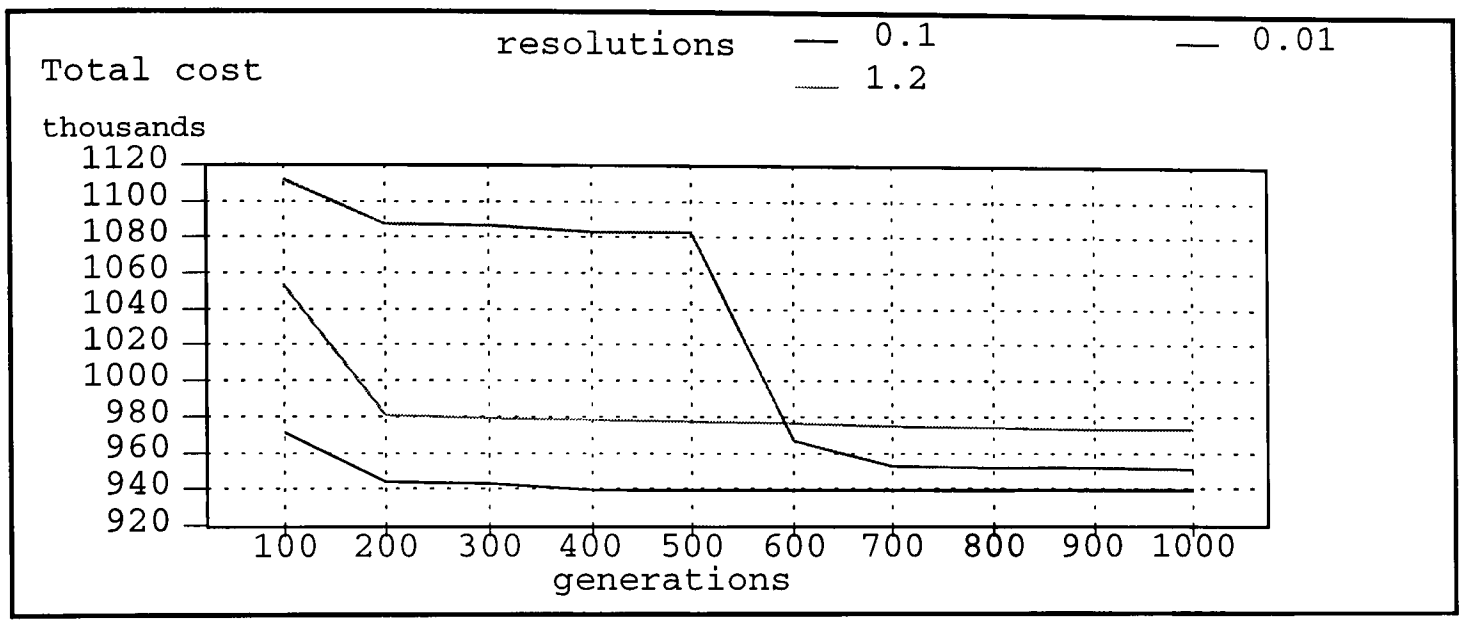



**Figure 9.6 Convergence characteristics for the hydrothermal GA as a function of control parameter resolution**

From figure 9.6, it can be seen that a resolution of 0.1 performs better than that of 1.2 and 0.01 respectively. The higher resolution provides a steeper convergence characteristic, which can be useful for a multiple step resolution search, where the process starts off with a large resolution and as the search proceeds the resolution is reduced. A GA convergence characteristic with multiple resolution is shown in figure 9.7.

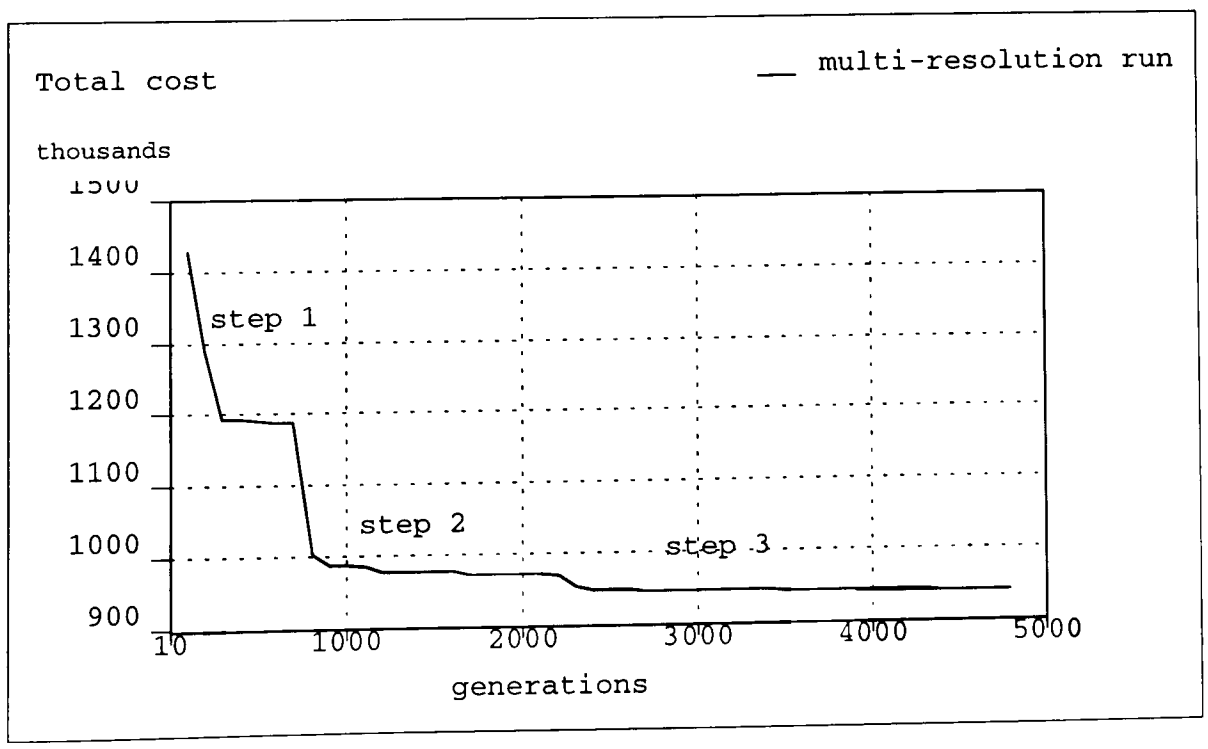



**Figure 9.7 Convergence characteristics of the hydrothermal GA with multiple step control parameter resolution**

Using multiple steps in the search reduces the computation time, and often leads to improved solutions over single resolution runs, but the appropriate resolution steps and number of generations for each step must be carefully chosen. The variation of the size of the scheduling time intervals and parameter resolution can be combined in an optimal manner to produce an efficient GA search mechanism that can solve a wide range of hydrothermal scheduling problems.

### 9.4.2 Genetic algorithm parameter settings

While the fitness function is derived from the problem objective function, as has been described in the previous section, the other GA control parameters are chosen based on the following: theoretical foundations of GA, guidance from previous experience of the application of GA in other problem domains, such as thermal scheduling and performance of empirical trial runs on the hydrothermal scheduling problem, prior to choosing the final parameter values. The genetic algorithm operators used for the hydrothermal scheduling are:

a) stochastic remainder selection, with truncation scaling,

b) uniform crossover and flip bit mutation operators,

c) generational parent replacement mechanism with elitism,

d) a stopping criterion based on either a set maximum number of generations of GA or stopping when no further significant improvements in results are obtained,

e) A GA local search mechanism, that involves re-initialisation (re-start), in which the best solution from the final population is placed into a new random population for a re-trial.

A summary of the genetic algorithm control parameter settings used for the hydrothermal scheduling simulations are listed in table 9.3. These settings were based on a combination of empirical trials and the available GA theory.

**Table 9.3 GA parameter list for the hydrothermal scheduling problem**

| Parameter | Type / Method | Value |
|---|---|---|
| population size | random initialisation | variable (up to size of string length) |
| selection | stochastic remainder | |
| fitness scaling | sigma (truncation) | std. dev.=1.0 |
| crossover | uniform | rate =1.0 |
| mutation | flip bit | set to 1/string length |
| parent replacement | generational replacement | replace all parents |
| elitism | replace randomly chosen children | 10% of children population |
| generations | variable | maximum 5,000 |

## 9.5 Hydrothermal Scheduling Test System

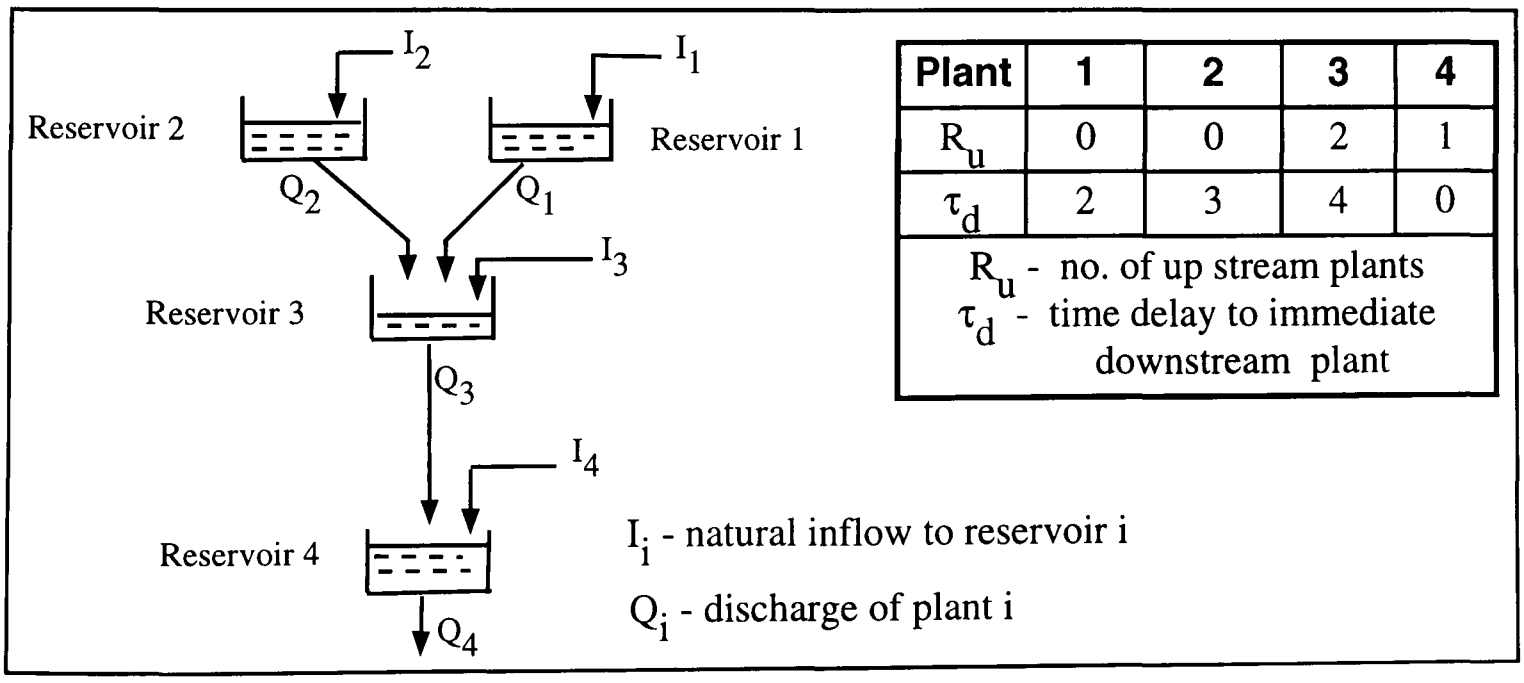
The test system [Wang and Shahidehpour, 1993], [Soares et. al.] used to evaluate the performance of the GA consists of a multi-chain cascade of 4 hydro units, and a number of thermal units represented by an equivalent thermal plant. The scheduling period is 24 hours. with one hour time intervals. The cost of thermal generation can be obtained in two ways:

1. by using a standard economic dispatch technique, such as those described in chapter 4 to find the optimal operation cost of the on-line thermal generators, or
2. by assuming the thermal generation is represented by an equivalent plant, whose characteristics can be determined as shown in [Wood and Wollenberg, 1984], [El_Hawary and Christensen].

The hydraulic sub system is characterised by the following:

- a multi-chain cascade flow network, with all the plants on one stream,
- river transport delay between successive reservoirs,
- variable head hydro plants,
- variable natural inflow rates into each reservoir,
- variable load demand over scheduling period.

The hydro sub-system configuration and network matrix including the water time delays are shown in figure 9.7.




| Plant | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $R_u$ | 0 | 0 | 2 | 1 |
| $\tau_d$ | 2 | 3 | 4 | 0 |

$R_u$ - no. of up stream plants
$\tau_d$ - time delay to immediate downstream plant

**Figure 9.8 Hydraulic system test network**

This hydraulic test network models most of the complexities encountered in practical hydro networks. The load demand, hydro unit power generation coefficients, river inflows and reservoir limits for the test network are given in in tables 9.4, 9.5 and 9.6 respectively.

**Table 9.4 Load demand and hydro power generation coefficients**

| *Load demand ( MW )* | | | | | |
|---|---|---|---|---|---|
| hour | load | hour | load | hour | load |
| **1** | 1370 | **9** | 2240 | **17** | 2130 |
| **2** | 1390 | **10** | 2320 | **18** | 2140 |
| **3** | 1360 | **11** | 2230 | **19** | 2240 |
| **4** | 1290 | **12** | 2310 | **20** | 2280 |
| **5** | 1290 | **13** | 2230 | **21** | 2240 |
| **6** | 1410 | **14** | 2200 | **22** | 2120 |
| **7** | 1650 | **15** | 2130 | **23** | 1850 |
| **8** | 2000 | **16** | 2070 | **24** | 1590 |

| *Hydro power generation coefficients* | | | | | | |
|---|---|---|---|---|---|---|
| plant | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ |
| **1** | -0.0042 | -0.42 | 0.030 | 0.90 | 10.0 | -50 |
| **2** | -0.0040 | -0.30 | 0.015 | 1.14 | 9.5 | -70 |
| **3** | -0.0016 | -0.30 | 0.014 | 0.55 | 5.5 | -40 |
| **4** | -0.0030 | -0.31 | 0.027 | 1.44 | 14.0 | -90 |

**Table 9.5 Reservoir inflows ( x $10^4$ $m^3$)**

| hour | Reservoir | | | | hour | Reservoir | | | | hour | Reservoir | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | | **1** | **2** | **3** | **4** | | **1** | **2** | **3** | **4** |
| **1** | 10 | 8 | 8.1 | 2.8 | **9** | 10 | 8 | 1 | 0 | **17** | 9 | 7 | 2 | 0 |
| **2** | 9 | 8 | 8.2 | 2.4 | **10** | 11 | 9 | 1 | 0 | **18** | 8 | 6 | 2 | 0 |
| **3** | 8 | 9 | 4 | 1.6 | **11** | 12 | 9 | 1 | 0 | **19** | 7 | 7 | 1 | 0 |
| **4** | 7 | 9 | 2 | 0 | **12** | 10 | 8 | 2 | 0 | **20** | 6 | 8 | 1 | 0 |
| **5** | 6 | 8 | 3 | 0 | **13** | 11 | 8 | 4 | 0 | **21** | 7 | 9 | 2 | 0 |
| **6** | 7 | 7 | 4 | 0 | **14** | 12 | 9 | 3 | 0 | **22** | 8 | 9 | 2 | 0 |
| **7** | 8 | 6 | 3 | 0 | **15** | 11 | 9 | 3 | 0 | **23** | 9 | 8 | 1 | 0 |
| **8** | 9 | 7 | 2 | 0 | **16** | 10 | 8 | 2 | 0 | **24** | 10 | 8 | 0 | 0 |

**Table 9.6 Reservoir storage capacity limits, plant discharge limits, plant generation limits and reservoir end conditions (x $10^4$ $m^3$)**

| Plant | $V_{min}$ | $V_{max}$ | $V_{ini}$ | $V_{end}$ | $Q_{min}$ | $Q_{max}$ | $P_{h,min}$ | $P_{h,max}$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 80 | 150 | 100 | 120 | 5 | 15 | 0 | 500 |
| 2 | 60 | 120 | 80 | 70 | 6 | 15 | 0 | 500 |
| 3 | 100 | 240 | 170 | 170 | 10 | 30 | 0 | 500 |
| 4 | 70 | 160 | 120 | 140 | 13 | 25 | 0 | 500 |

## 9.6 Simulations and Results

A genetic algorithm provides a final population of solutions. The best solution, in terms of the fitness function, is usually taken as the optimal solution. The mathematical best solution might not necessarily be the best for the decision makers, who might wish to take some other factors, not implementable in the mathematical formulation, into account. The GA can act as a decision support tool by providing the analyst with a possible set of optimal solutions upon which they can base their judgement.

In the short term hydrothermal scheduling problem, the two important parameters that can be allowed to vary are the satisfaction of the final (end conditions) reservoir storage levels and the cost of thermal generation. These two objectives can be conflicting and by providing final solutions showing the best of both variables, the decision maker can be assisted in making better decisions. The optimal solution provided by the GA is the cost of thermal generation as well as the total violation of the final reservoir storage limits. A number of tests were carried out to validate the performance of the hydrothermal scheduling GA on the test network. In one set of experiments, each GA trial was allowed to run for 2000 generations, while in the other set, the GA run was terminated after 5000 generations. Each experiment was run 10 times, each starting with a different random initial population.

### 9.6.1 Results for GA runs of 2000 generations

The effect of using different parameter resolutions for different stages of the GA was investigated. Table 9.7 shows the GA performance for 2000 generations, for the single and multiple step resolution respectively. The column showing the violation of reservoir end volume indicates the sum of the violations of all the plants. An example of the individual reservoir violations is shown in table 9.8 for one of the simulations tabulated. The corresponding variations between the schedule cost and total reservoir storage (end condition) violations are shown in figure 9.9. The results show that the multiple resolution GA performs significantly better than the single resolution one, and it also takes less cpu time to find the given results.

**Table 9.7 Hydrothermal scheduling results for 2000 generations**

| Trial | Single resolution | | Multiple step resolution | |
|---|---|---|---|---|
| | Thermal cost | Total violation in **end** volume | Thermal cost | Total violation in **end** volume |
| 1 | 950,631 | 0.110 | 939,721 | 0.215 |
| 2 | 946,483 | 0.116 | 939,546 | 0.242 |
| 3 | 944,233 | 0.100 | 939,970 | 0.331 |
| 4 | 948,593 | 0.079 | 942,259 | 0.445 |
| 5 | 948,892 | 0.093 | 943,451 | 0.506 |
| 6 | 951,542 | 0.226 | 946,428 | 0.295 |
| 7 | 946,088 | 0.098 | 935,482 | 0.527 |
| 8 | 948,968 | 0.134 | 941,702 | 0.445 |
| 9 | 949,298 | 0.167 | 951,509 | 0.34 |
| 10 | 945,111 | 0.071 | 942,776 | 0.200 |
| Problem variables | Average resolution [0.1],<br>Number of bits [696],<br>Cpu time [14 min.],<br>Generations [2000] | | Average resolutions [2.7, 1.2, 0.1],<br>Number of bits [240, 336, 696],<br>Generations per stage [500, 500, 1000],<br>Cpu time [8 min.] | |

**Table 9.8 Sample reservoir final storage conditions**

| plant | Expected end volume | Final end volume | Violation of end volume |
|---|---|---|---|
| 1 | 120 | 120.0 | 0 |
| 2 | 70 | 70.0 | 0.1 |
| 3 | 170 | 170.1 | 0.1 |
| 4 | 140 | 139.9 | 0 |
| Total violation = 0.2, Scheduling cost = 939, 721 | | | |

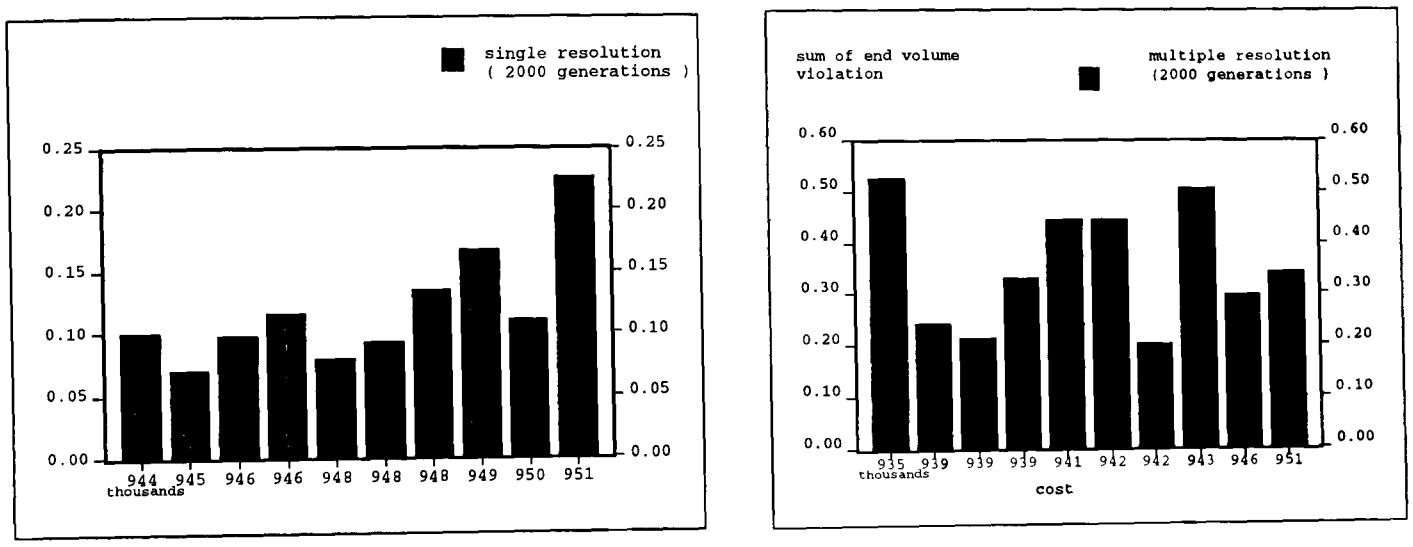



**Figure 9.9 Relationship between reservoir end volume violation and cost of thermal generation (2000 generations)**

From figure 9.9 it can be seen that the GA with multiple resolution consistently performs better than that with single resolution in terms of scheduling cost and also takes much less cpu time to obtain the improved results. It is also worth noting that the best cost solution does not necessarily result in the least violation of end volume storage requirements. The decision maker should be able to choose the best solution from those provided by the GA runs.

### 9.6.2 GA results for 5000 generations

Table 9.9 show the results of the single and 3 step parameter resolution GA for trials of 5000 generations. The corresponding variations of the scheduling cost with the total reservoir end storage violations are shown in figure 9.10. Table 9.10, derived from tables 9.7 and table 9.9 compares the performance of multiple parameter resolution GA for runs terminated after 2000 and 5000 generations. Increasing the number of generations in a run from 2000 to 5000 provides a 0.57 percent average improvement in scheduling cost and slight improvement in meeting end volume constraints. This improvement, was however accompanied by an almost 80% increase in computation time. The GA user must decide the degree of accu-

racy required of the solutions, as well as the computation time which can be tolerated, since the solution quality is usually improved by increasing the number of generations.

**Table 9.9 Hydrothermal scheduling results for 5000 generations ( Multiple parameter resolution)**

| Trial | Single resolution | | Multiple step resolution | |
|---|---|---|---|---|
| | Thermal cost | Total violation in **end** volume | Thermal cost | Total violation in **end** volume |
| 1 | 947,846 | 0.091 | 939,734 | 0.215 |
| 2 | 945,221 | 0.086 | 936,451 | 0.169 |
| 3 | 942,600 | 0.071 | 935,721 | 0.493 |
| 4 | 943,024 | 0.099 | 936,625 | 0.233 |
| 5 | 946,611 | 0.079 | 938,551 | 0.445 |
| 6 | 946,767 | 0.093 | 936,567 | 0.219 |
| 7 | 945,768 | 0.098 | 938,420 | 0.376 |
| 8 | 951,087 | 0.140 | 937,141 | 0.221 |
| 9 | 948,513 | 0.134 | 937,749 | 0.204 |
| 10 | 948,654 | 0.122 | 932,734 | 0.115 |
| Problem variables | Average resolution [0.1],<br>Number of bits [696],<br>Average cpu time [32 min.],<br>Generations [5000] | | Average resolutions [2.7, 1.2, 0.1]<br>Number of bits, [240, 336, 696]<br>Generations per stage [710, 1430, 2860],<br>Total Cpu time [20 min.] | |

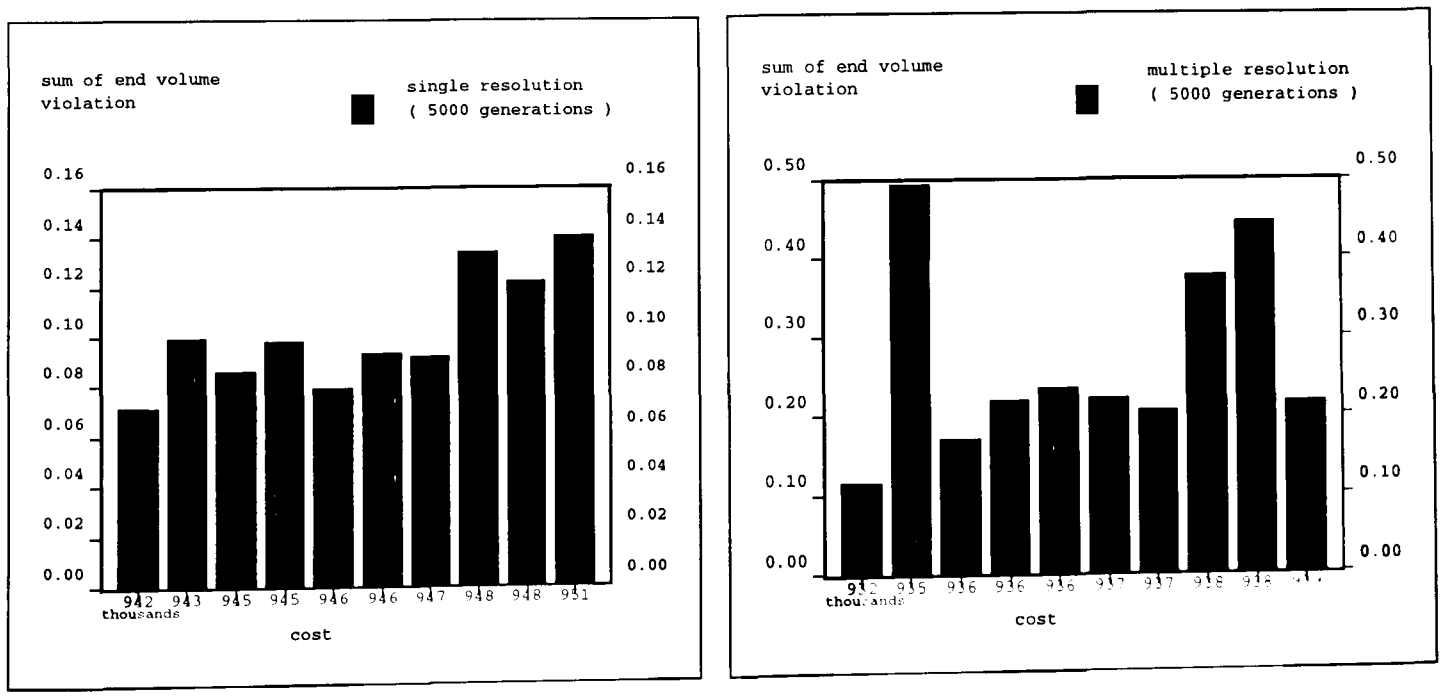



**Figure 9.10 Relationship between reservoir end volume violation and cost of thermal generation (5000 generations, multiple parameter resolution)**

**Table 9.10 Comparison of the hydrothermal scheduling GA performance for 2000 and 5000 generations in a run**

| Trial | Percentage improvement in thermal cost | Actual improvement in violation of final end storage |
|---|---|---|
| 1 | -0.001 | 0 |
| 2 | 0.33 | 0.073 |
| 3 | 0.45 | -0.162 |
| 4 | 0.60 | 0.212 |
| 5 | 0.52 | 0.061 |
| 6 | 1.05 | 0.076 |
| 7 | -0.31 | 0.151 |
| 8 | 0.49 | 0.224 |
| 9 | 1.47 | 0.136 |
| 10 | 1.08 | 0.085 |
| **Mean** | **0.57%** | **0.086** |

Figure 9.11 shows the variation of the scheduling cost with the number of generations, while Figure 9.12 shows the variation of the total *end volume* violations with the number of generations. From figure 9.11, it can be seen that the GA has nearly converged by generation 500, after which the scheduling cost changes very slowly as the number of generations in a run is increased. It is also important to observe the effects of the end volume constraints as shown in figure 9.12, otherwise the GA run might be prematurely terminated before the optimal results that also satisfy the problem constraints are obtained.

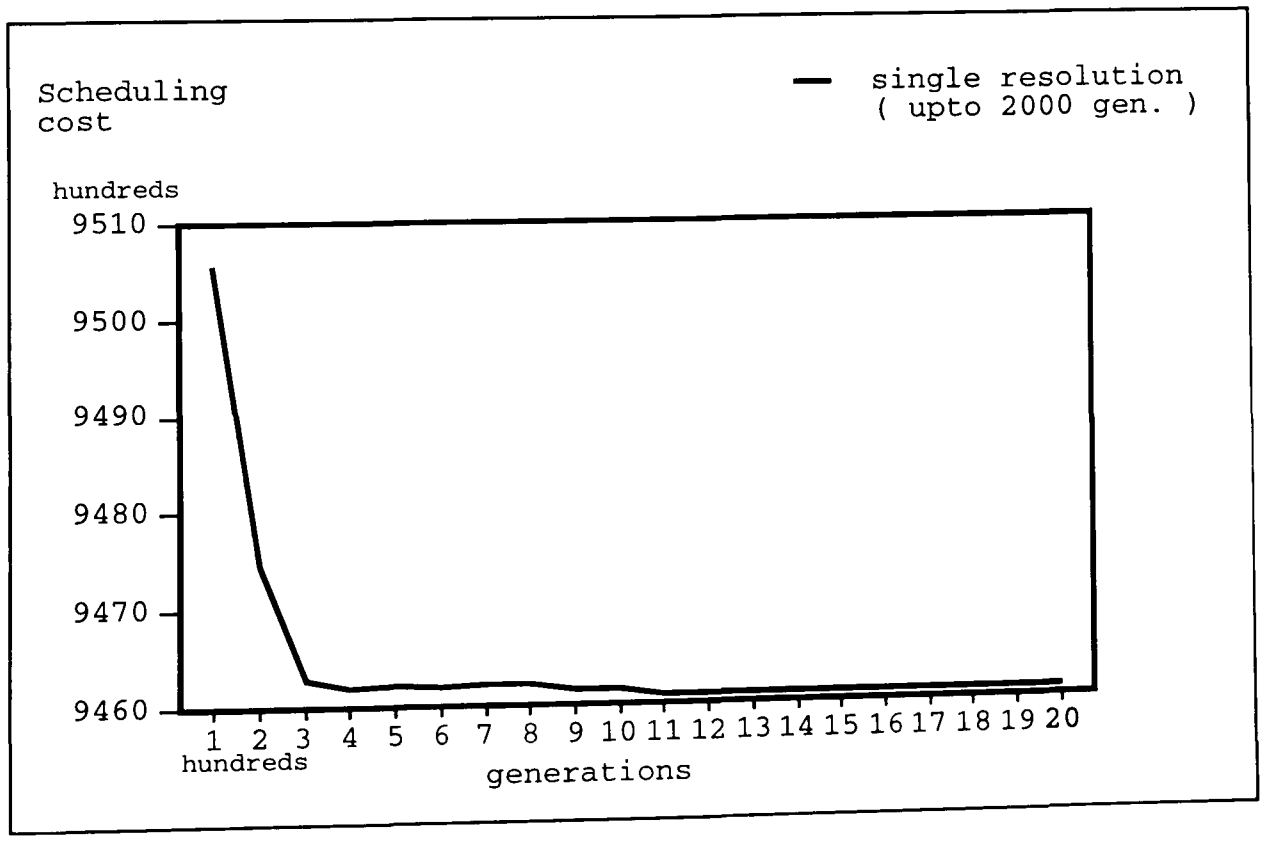



**Figure 9.11 Variation of hydrothermal scheduling cost with GA generations**

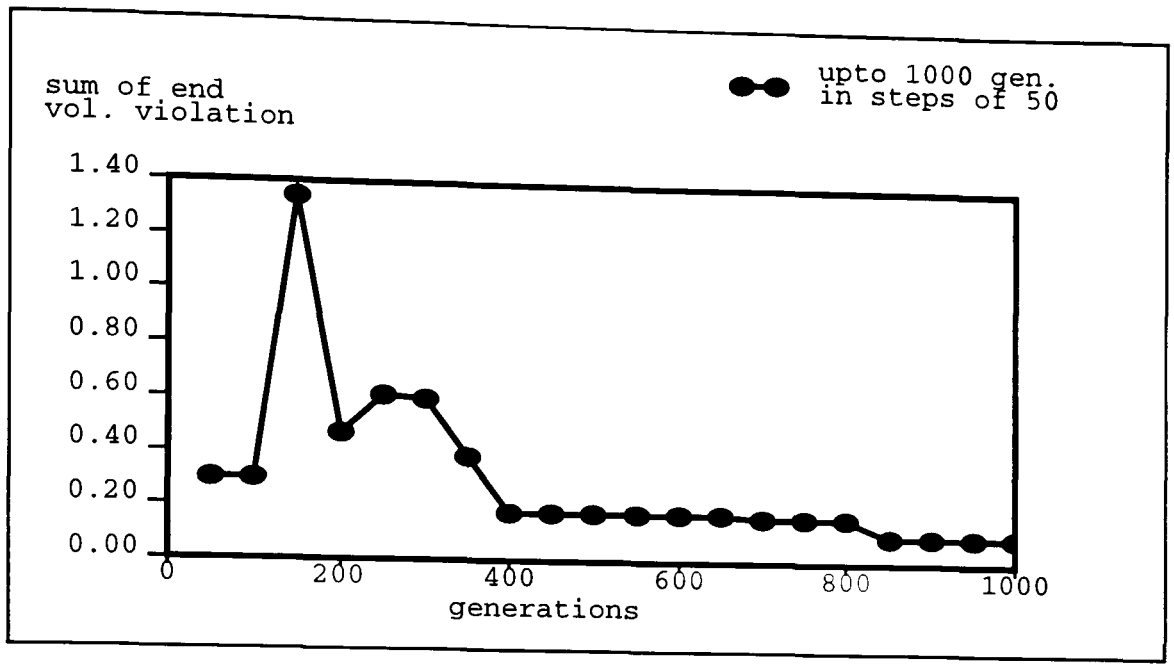



**Figure 9.12 Variation of total end volume violations with GA generations**

#### 9.6.2.1 Multiple time step, single resolution GA

Table 9.11 shows the results of a 3 step GA run with different time intervals at each stage, for a total GA run of 5000 generations, with the generations for each stage shown in the table. The corresponding variations between the scheduling cost and total reservoir end storage violations are shown in figure 9.13.

**Table 9.11 Hydrothermal scheduling results for 5000 generations (multiple time step GA)**

| Trial | Single time step | | Multiple time step | |
|---|---|---|---|---|
| | Thermal cost | Total violation in **end** volume | Thermal cost | Total violation in **end** volume |
| 1 | 947,846 | 0.091 | 945,402 | 0.100 |
| 2 | 945,221 | 0.086 | 938,577 | 0.114 |
| 3 | 942,600 | 0.071 | 939,798 | 0.104 |
| 4 | 943,024 | 0.099 | 940,269 | 0.106 |
| 5 | 946,611 | 0.079 | 939,789 | 0.110 |
| 6 | 946,767 | 0.093 | 938,370 | 0.134 |
| 7 | 945,768 | 0.098 | 941,046 | 0.114 |
| 8 | 951,087 | 0.140 | 944,006 | 0.113 |
| 9 | 948,513 | 0.134 | 945,942 | 0.119 |
| 10 | 948,654 | 0.122 | 943,734 | 0.122 |
| Average resolution [0.1], number of bits [696], Average cpu time [32 min.], Generations [5000] | | | Time steps (hours) [3, 2, 1] Number of bits, [232, 348, 696] Generations per stage [710, 1430, 2860], Cpu time [22 min.] | |

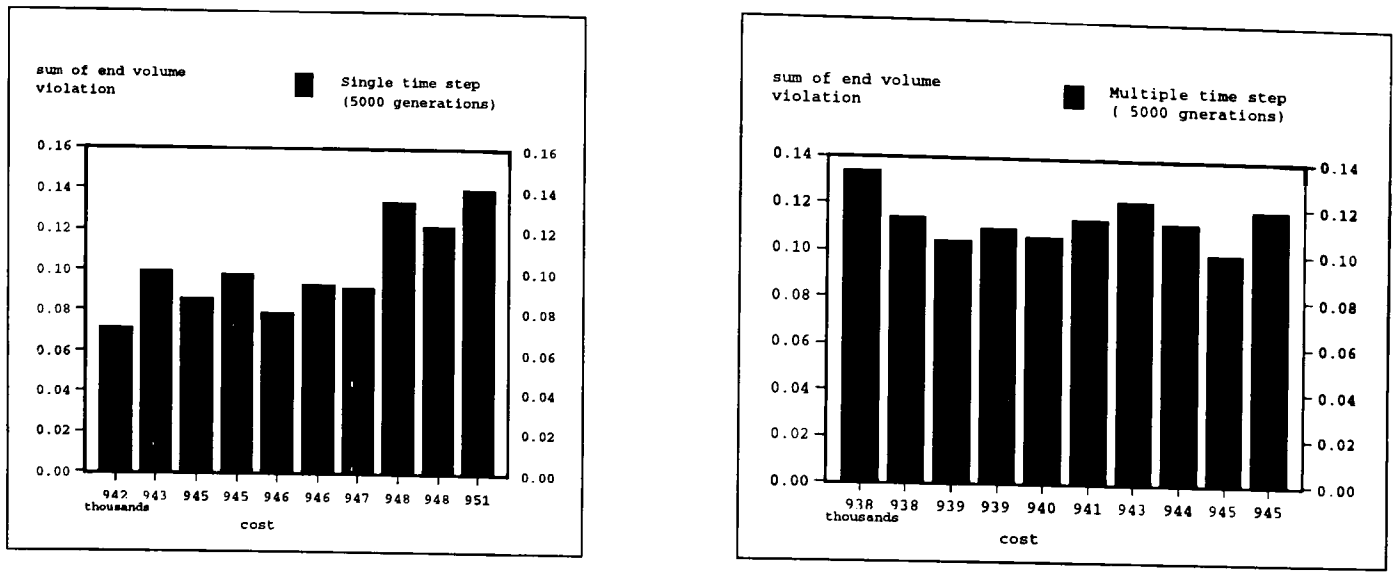



**Figure 9.13 Relationship between reservoir end volume violation and cost of thermal generation (5000 generations, multiple time step)**

### 9.6.3 Sample hydrothermal scheduling output results

Apart from the turbine discharge, which is given as the hydrothermal scheduling GA solution, it is also useful to provide as an output, quantities such as reservoir storage levels, total thermal generation and hydro unit power outputs, during each time interval. These quantities are calculated using the water discharge rates, the hourly river inflows, water transport delays and the load demand at each time interval, over the scheduling period. Sample output result of one of the GA trials showing these output quantities are given in tables 9.13 to 9.14 with the corresponding plots in figures 9.14 to 9.17.

**Table 9.12 A summary of final reservoir storage levels and cost for the sample hydrothermal scheduling GA trial**

| Reservoir | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Final storage | 119.96 | 70.03 | 170.06 | 139.96 |
| Expected final storage | 120 | 70 | 170 | 140 |
| Thermal generation cost = **936,451**, Total end volume violation = **0.169** | | | | |

**Table 9.13 Hourly plant discharge ( x $10^4$ $m^3$)**

| Hour | plant 1 | plant 2 | plant 3 | plant 4 |
|---|---|---|---|---|
| 1 | 6.259842 | 6.000000 | 28.03922 | 13.00000 |
| 2 | 9.015748 | 9.543307 | 18.07843 | 13.00000 |
| 3 | 13.50394 | 13.65354 | 23.25490 | 13.09449 |
| 4 | 6.417323 | 6.566929 | 25.84314 | 13.75591 |
| 5 | 5.000000 | 6.000000 | 14.23529 | 13.00000 |
| 6 | 7.992126 | 6.354331 | 18.00000 | 14.51181 |
| 7 | 6.417323 | 6.000000 | 27.25490 | 13.75591 |
| 8 | 10.74803 | 6.354331 | 11.56863 | 16.11811 |
| 9 | 13.50394 | 6.000000 | 27.33333 | 14.60630 |
| 10 | 10.66929 | 13.08661 | 28.27451 | 14.51181 |
| 11 | 10.66929 | 7.275590 | 24.66667 | 14.51181 |
| 12 | 5.000000 | 6.496063 | 10.15686 | 14.51181 |
| 13 | 10.66929 | 6.000000 | 26.62745 | 19.42520 |
| 14 | 7.834646 | 9.826772 | 11.88235 | 13.00000 |
| 15 | 10.66929 | 8.551181 | 10.15686 | 20.93701 |
| 16 | 7.834646 | 8.267716 | 14.00000 | 15.36220 |
| 17 | 10.74803 | 8.267716 | 12.74510 | 13.09449 |
| 18 | 5.000000 | 11.17323 | 17.76471 | 13.00000 |
| 19 | 6.417323 | 11.10236 | 11.25490 | 18.57480 |
| 20 | 5.000000 | 11.10236 | 20.90196 | 16.11811 |
| 21 | 7.834646 | 8.551181 | 10.00000 | 19.33071 |
| 22 | 6.417323 | 7.275590 | 10.00000 | 17.72441 |
| 23 | 5.000000 | 6.141732 | 10.00000 | 18.57480 |
| 24 | 6.417323 | 12.37795 | 10.00000 | 15.36220 |

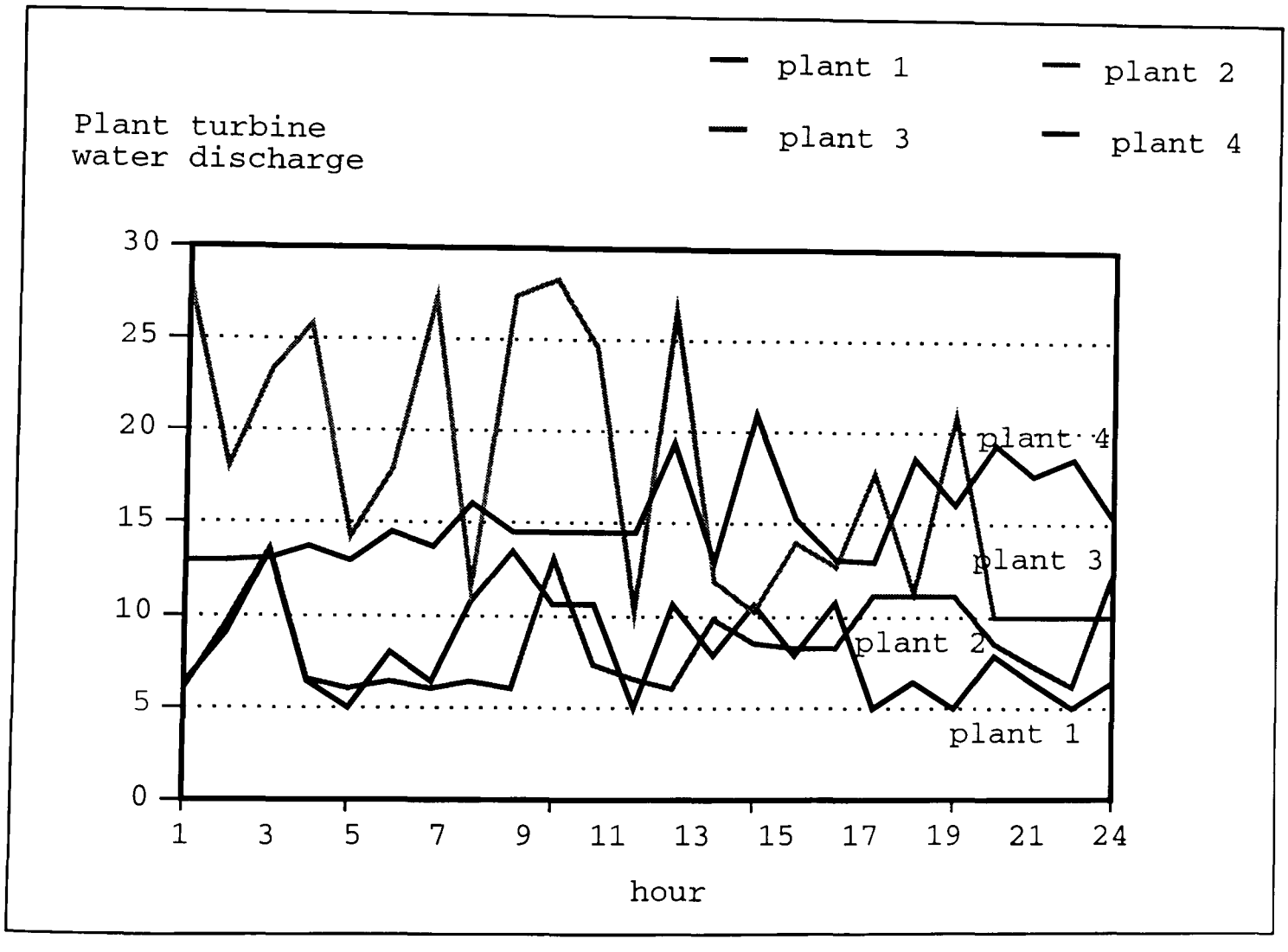



**Figure 9.14 Hourly plant discharge trajectories**

**Table 9.14 Reservoir storage levels at end of each time step ( x $10^4$ $m^3$)**

| Hour | plant 1 | plant 2 | plant 3 | plant 4 |
|---|---|---|---|---|
| 1 | 103.74 | 82.00 | 150.06 | 109.80 |
| 2 | 103.72 | 80.46 | 140.18 | 99.20 |
| 3 | 98.22 | 75.80 | 127.19 | 87.71 |
| 4 | 98.80 | 78.24 | 118.36 | 73.95 |
| 5 | 99.80 | 80.24 | 130.17 | 88.99 |
| 6 | 98.81 | 80.88 | 136.24 | 92.56 |
| 7 | 100.39 | 80.88 | 123.55 | 102.05 |
| 8 | 98.65 | 81.53 | 127.98 | 111.78 |
| 9 | 95.14 | 83.53 | 114.42 | 111.41 |
| 10 | 95.47 | 79.44 | 103.89 | 114.90 |
| 11 | 96.80 | 81.17 | 100.08 | 127.64 |
| 12 | 101.80 | 82.67 | 108.59 | 124.70 |
| 13 | 102.13 | 84.67 | 109.72 | 132.60 |
| 14 | 106.30 | 83.84 | 113.12 | 147.88 |
| 15 | 106.63 | 84.29 | 123.12 | 151.61 |
| 16 | 108.80 | 84.02 | 124.96 | 146.40 |
| 17 | 107.05 | 82.76 | 134.71 | 159.94 |
| 18 | 110.05 | 77.58 | 135.33 | 158.82 |
| 19 | 110.63 | 73.48 | 144.09 | 150.40 |
| 20 | 111.63 | 70.38 | 137.46 | 148.28 |
| 21 | 110.80 | 70.83 | 147.05 | 141.70 |
| 22 | 112.38 | 72.55 | 155.15 | 141.74 |
| 23 | 116.38 | 74.41 | 165.09 | 134.42 |
| 24 | 119.96 | 70.03 | 170.06 | 139.96 |

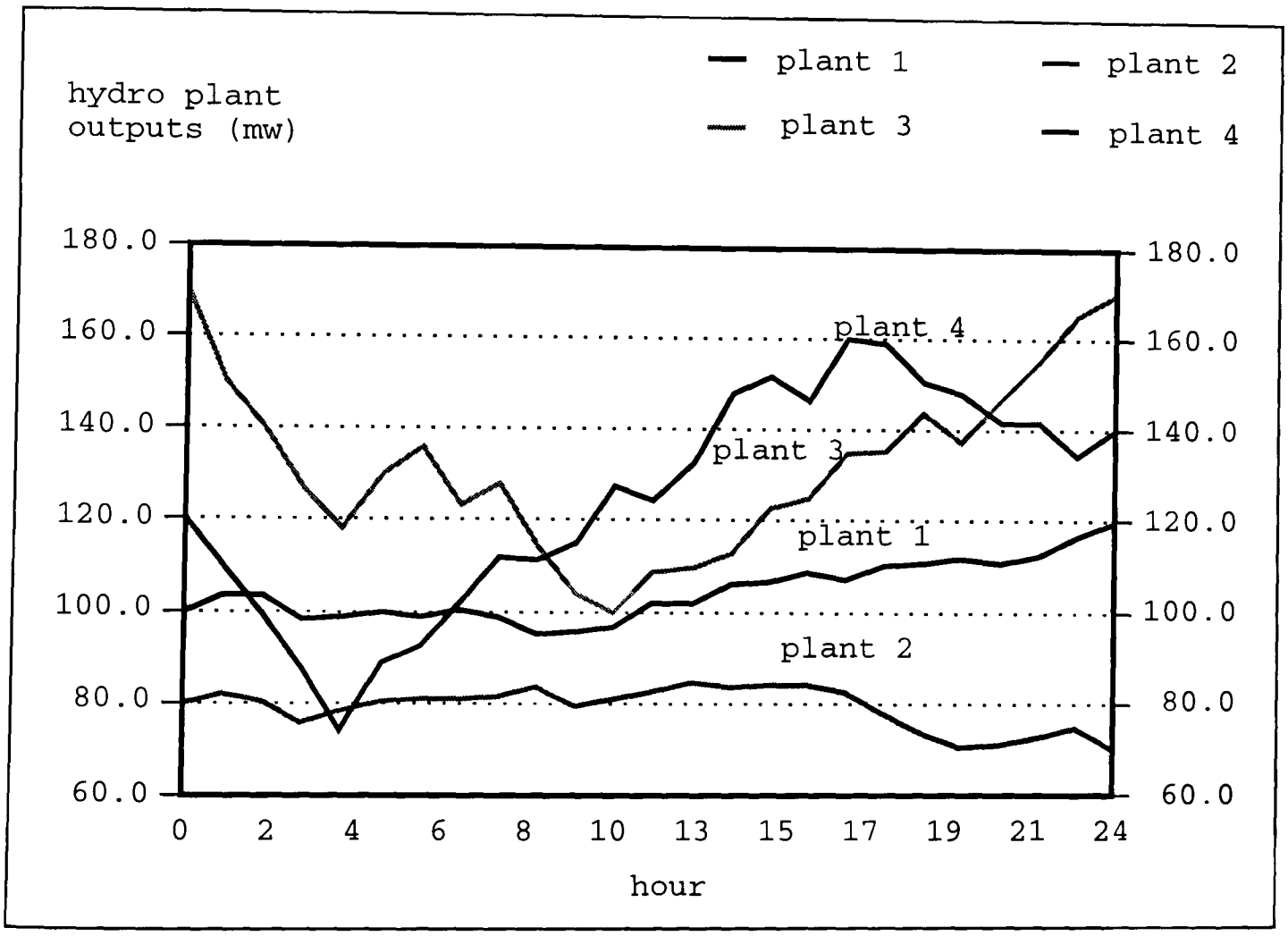



**Figure 9.15 Hourly reservoir storage levels**

**Table 9.15 Hydro plant power outputs and total thermal generation**

| Hour | hydro power generation (MW) | | | | Thermal power generation (MW) |
|---|---|---|---|---|---|
| | plant 1 | plant 2 | plant 3 | plant 4 | |
| 1 | 63 | 49 | 0 | 205 | 1051 |
| 2 | 82 | 71 | 44 | 194 | 998 |
| 3 | 97 | 84 | 14 | 181 | 982 |
| 4 | 63 | 51 | 0 | 170 | 1004 |
| 5 | 52 | 48 | 45 | 165 | 977 |
| 6 | 74 | 51 | 40 | 189 | 1053 |
| 7 | 64 | 49 | 0 | 191 | 1344 |
| 8 | 89 | 52 | 47 | 221 | 1589 |
| 9 | 95 | 50 | 0 | 215 | 1878 |
| 10 | 87 | 85 | 0 | 216 | 1930 |
| 11 | 87 | 57 | 0 | 225 | 1858 |
| 12 | 52 | 53 | 39 | 231 | 1933 |
| 13 | 89 | 51 | 0 | 268 | 1821 |
| 14 | 75 | 74 | 42 | 231 | 1775 |
| 15 | 91 | 67 | 44 | 300 | 1626 |
| 16 | 76 | 66 | 46 | 261 | 1620 |
| 17 | 91 | 65 | 48 | 244 | 1678 |
| 18 | 53 | 77 | 41 | 248 | 1717 |
| 19 | 66 | 74 | 51 | 291 | 1756 |
| 20 | 54 | 71 | 30 | 268 | 1854 |
| 21 | 76 | 58 | 50 | 286 | 1767 |
| 22 | 66 | 52 | 52 | 272 | 1676 |
| 23 | 54 | 45 | 54 | 273 | 1420 |
| 24 | 67 | 76 | 55 | 249 | 1140 |

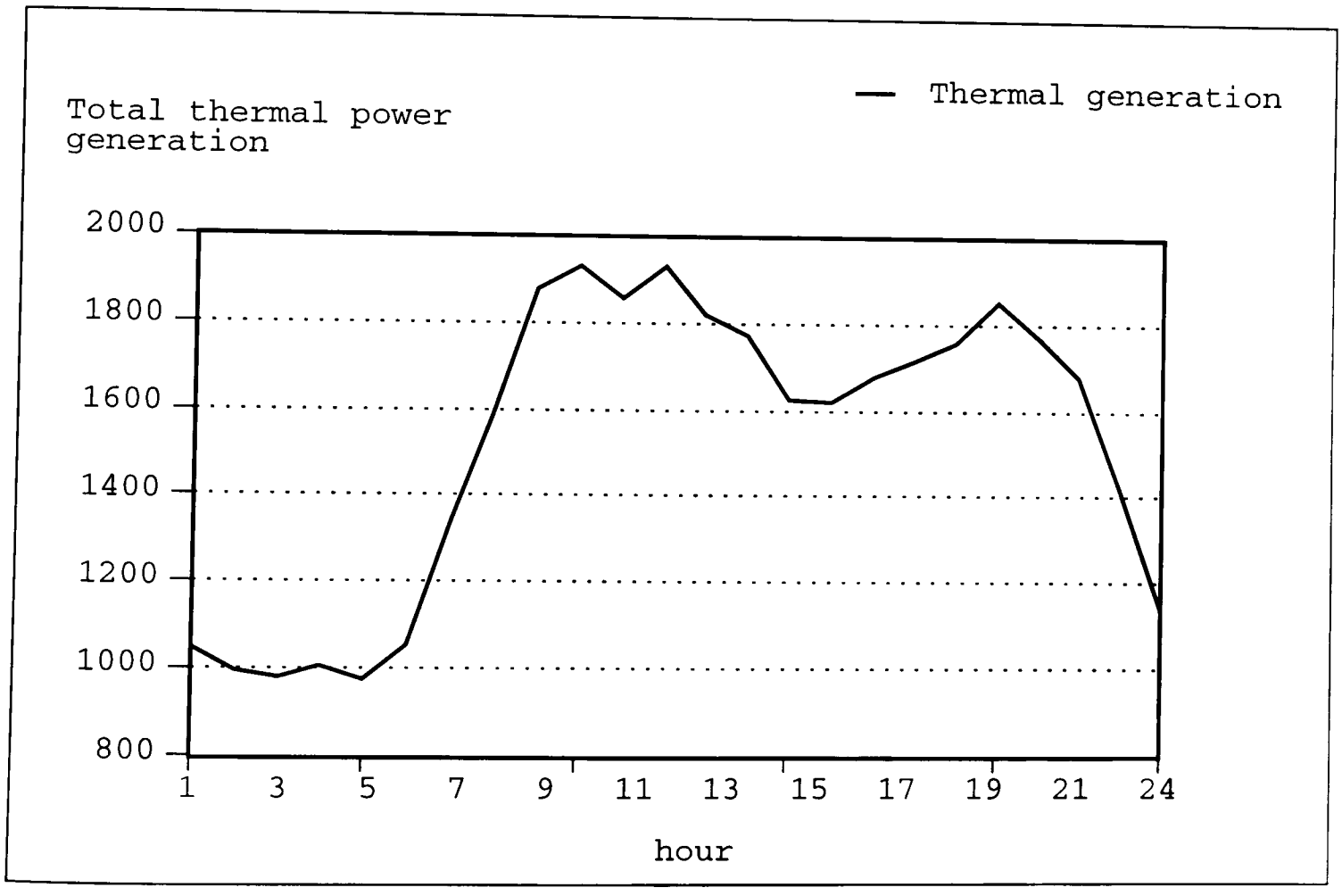



**Figure 9.16 Total hourly thermal generation**

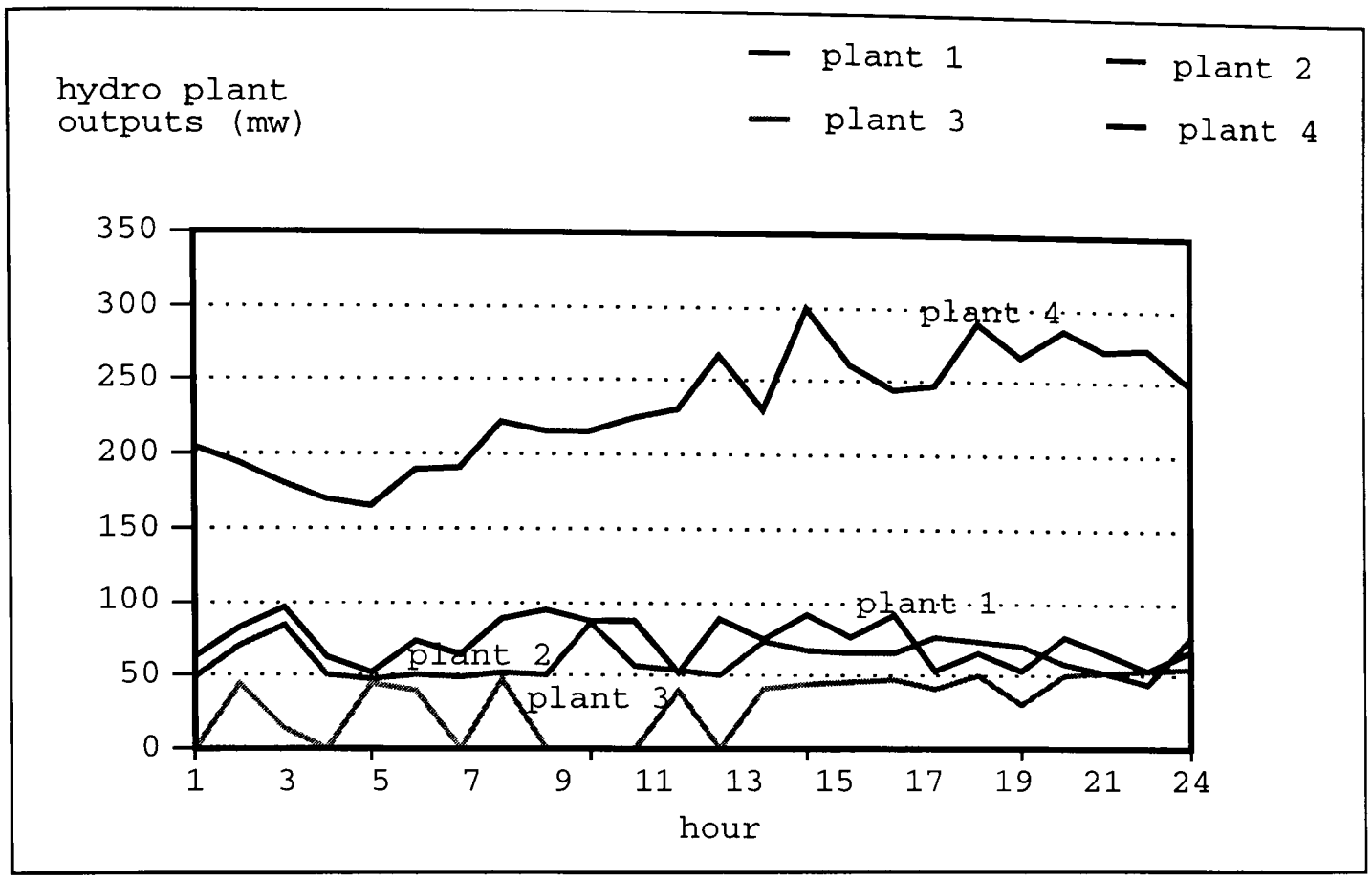



**Figure 9.17 Hourly hydro power generation**

## 9.7 Conclusions

In the hydrothermal scheduling problem, the complexity introduced by the cascade nature of the hydraulic network, the scheduling time linkage, non-linear relationships in the problem variables and the water transport delay factors has made the problem very difficult to solve using standard optimisation methods. The GA, on the other hand is able to take into account all the problem variables without making the usual simplifying assumptions, required by other techniques. This problem is *epistatic* in GA terms, since a schedule at an earlier time interval affects that at a later time, and therefore the whole scheduling period must be treated as a single solution or entity. Once the problem has been formulated in the GA framework, the only other issues to be resolved are the GA control parameters. Large scale hydrothermal scheduling problems can easily be solved by intuitive techniques such as multiple resolution in parameter variables or multiple time interval decomposition which speed up the search process.

The genetic algorithm approach provides a simple hydrothermal scheduling problem formulation and solution method and is able to take into account the variation in net head and water transport delay factors. Once good GA control parameters have been obtained, the solution to the problem under different operational scenarios can easily be obtained. The genetic algorithm method results in a simple hydrothermal scheduling problem formulation and solution method which can easily be extended to the solution of other power system optimisation tasks.

# Chapter 10. Conclusions and Future Work

## 10.1 Conclusions

The empirical success of genetic algorithms on hitherto unsolved complex problems has made them a very attractive computation technique. The GA, must however be carefully designed in order to be able to solve any given problem. This design involves the appropriate choice of the GA model, control parameters, fitness function and problem representation. The GA is best applied in an innovative way to any specific problem, by using as much of the problem knowledge as possible. The proper selection of the GA control parameters such as population size, crossover and mutation rates can result in improved solutions. Recent empirical evidence has shown that, if these variables are allowed to self adapt, as the GA run progresses, much better solutions could be obtained, thus avoiding the need to set the parameters *a priori.* At present, the GA control parameter settings are based on a mixture of experimental trials on the problem domain and theoretical insights on GA performance. It is the lack of a solid theoretical basis for a universal setting of GA control parameter settings across a wide range of problem domains that is one of the main drawbacks of the GA method. Recent research findings on the theory of GA, particularly, those on convergence analysis, have began to throw more light on the appropriate choice, mix and setting of GA control parameters.

The GA method is able to provide a number of quasi optimal solutions to a problem, either by repeated trials with different initial populations or by taking a sample of the best solutions from the final generation of the run. These alternative solutions can sometimes provide some very practical solutions that might otherwise have escaped the attention of the analyst. The inherently parallel nature of the GAs also allows their implementation on both coarse and fine grained parallel computers, resulting in significant decreases in computation time.

There have been several attempts to apply genetic algorithms to generation scheduling problems. Many of these applications work well for small problems, but do not scale well to larger problems. In this thesis, genetic algorithm implementations have been presented that are capable of solving the generation scheduling problem for small, medium and large scale power systems.

Specific conclusions that can be made on the use of the genetic algorithm for the solution of scheduling problems are presented in the following sections.

### 10.1.1 Thermal Scheduling

This work has demonstrated the feasibility of using a genetic algorithm for solving the thermal generation scheduling problem for small, medium and large size power systems. Although, the GA is generally slower than some conventional solution techniques, it provides great advantages in its modelling framework that allows the easy treatment of the majority of the problem constraints. The developed hybrid genetic algorithm thermal scheduling method allows a straightforward and simple modelling of virtually all the scheduling problem constraints. Once scheduling problems have been formulated in the GA modelling framework, the other major issues that have to be resolved include:

1. preventing premature convergence of the genetic algorithm, through an appropriate choice of the GA control parameters,
2. finding ways of improving on the computation time,
3. establishing a suitable fitness function, by the proper treatment of problem constraints.

This work has resolved the above three major questions by proposing hybrid canonical genetic algorithm and deterministic crowding genetic algorithm thermal scheduling methods that improve search efficiency through the introduction of problem specific heuristics. As stand alone systems, the GA methods were only able to solve small scale scheduling problems within a reasonable time frame. However, with the hybrid GA, it has been shown how effectively the inclusion of domain knowledge, through the simple priority list unit commitment scheme, can improve the GA convergence rates and solution quality. The importance of problem decomposition, especially for large scale scheduling problems, even when the problem is not completely decomposable, has also been demonstrated by the improvement in the GA solutions provided by the decomposed hybrid GA method. The speed limitation of the GA can be further alleviated by the use of parallel computers, through the implementation of parallel genetic algorithm models.

### 10.1.2 Hydrothermal Co-ordination

It has been shown that the genetic algorithm is capable of providing solutions to a hydrothermal co-ordination problem that incorporates most of the constraints including: multiple reservoir cascade hydro networks, variable head hydro plants, time delay between consecutive reservoirs and non-linear hydro and thermal power generation characteristics. The GA is able to take into account all the problem variables without making the usual simplifying assumptions, which are necessary for other conventional techniques. The GA model treats the whole scheduling period as a non separable objective function in order to account

for the scheduling time linkage, cascade river flow dynamics and water transport delay factors. Good solutions to the hydrothermal scheduling problems have been obtained by exploiting the scheduling problem structure resulting in the development of GA that combines adaptive GA control parameter resolution with multiple time interval decomposition, to produce a robust scheduling algorithm that not only provides high quality solutions but also speeds up the GA search process.

### 10.1.3 Economic Dispatch

Economic dispatch is a sub-problem of both the unit commitment and hydrothermal coordination problems, and assumes that the decision to commit any unit to generation has been made prior to performing economic dispatch. In this work, a genetic algorithm solution approach to the economic dispatch problem, including the constrained economic dispatch problem in which some of the units have prohibited operating regions, has been presented. The genetic algorithm method has been shown to be capable of solving the constrained economic dispatch problem for practical power systems. The proper choice of the appropriate GA model, is however important as has been demonstrated in this work. The deterministic crowding genetic algorithm has shown its superiority over the standard GA in solving this problem. When solving the economic dispatch problem, it has also been shown that the use of a parallel local hill climbing algorithm on the final genetic algorithm population can provide improved final solutions.

### 10.1.4 Genetic Algorithms

The genetic algorithm solution techniques provide an appropriate modelling framework for both the thermal and hydrothermal scheduling problems, allowing the modelling of virtually all the problem constraints. A particular strength of the GA method has been its ability to use a mixture of discrete and continuous parameters in the problem formulation, leading to an intuitive and natural representation for the problem being considered. However, it must be acknowledged, as has been shown in this work, that the GA is an optimisation technique like any other, which must satisfy the basic optimisation steps such as:

- provision of an appropriate evaluation (fitness) function,
- provision of an appropriate search step size for optimisation in a continuous parameter space or an appropriate interpretation of integer variables, for an integer programming problem,
- determination of a proper method of handling constraints in a constrained optimisation problem, as this choice is crucial to the GA performance.

It is thus not sufficient, to apply the *simple* GA to a problem and expect it to provide good solutions, without taking into account the problem structure and size.

There are several variations of genetic algorithm models, whose performance can differ very widely, and the appropriate choice of a GA model suitable for the problem at hand is also an important consideration. For example, in this work, it has been demonstrated that despite its simplicity, the deterministic crowding genetic algorithm is able to provide very good solutions to the complex problem of generator scheduling for practical power systems.

The genetic algorithm and other evolutionary computation models are useful computational tools that can help in solving some of the challenging operation and control problems facing electricity utilities today. The generation scheduling problem is one of the most challenging power system optimisation and control problems, and for the GA to successfully solve this problem attests to its ability to solve real world engineering problems. Theoretical research is continuing on the development of a more universal GA theory, which will hopefully resolve issues such as the appropriate choice of GA parameters. If this succeeds, the GA method will be much more acceptable to industry, as the design of the algorithm will no longer be an exclusive domain for the GA expert.

### 10.1.5 Power System Operation Planning Using Genetic Algorithms

Economic operation of electric power systems involves the solution of a number of different problems depending on the operation complexities of individual utilities, but the core problems remain the same. The main operation control problems to be solved include:

- Economic dispatch,
- Unit commitment,
- Hydrothermal co-ordination,
- Operation under environmental impact,
- Energy interchange (transactions).

In this thesis, a generalised genetic algorithm optimisation model allowing the determination of the most economic operation of a power system has been presented. Figure 10.1 illustrates how the genetic algorithm optimisation is used in solving the economic operation problems in a power system.

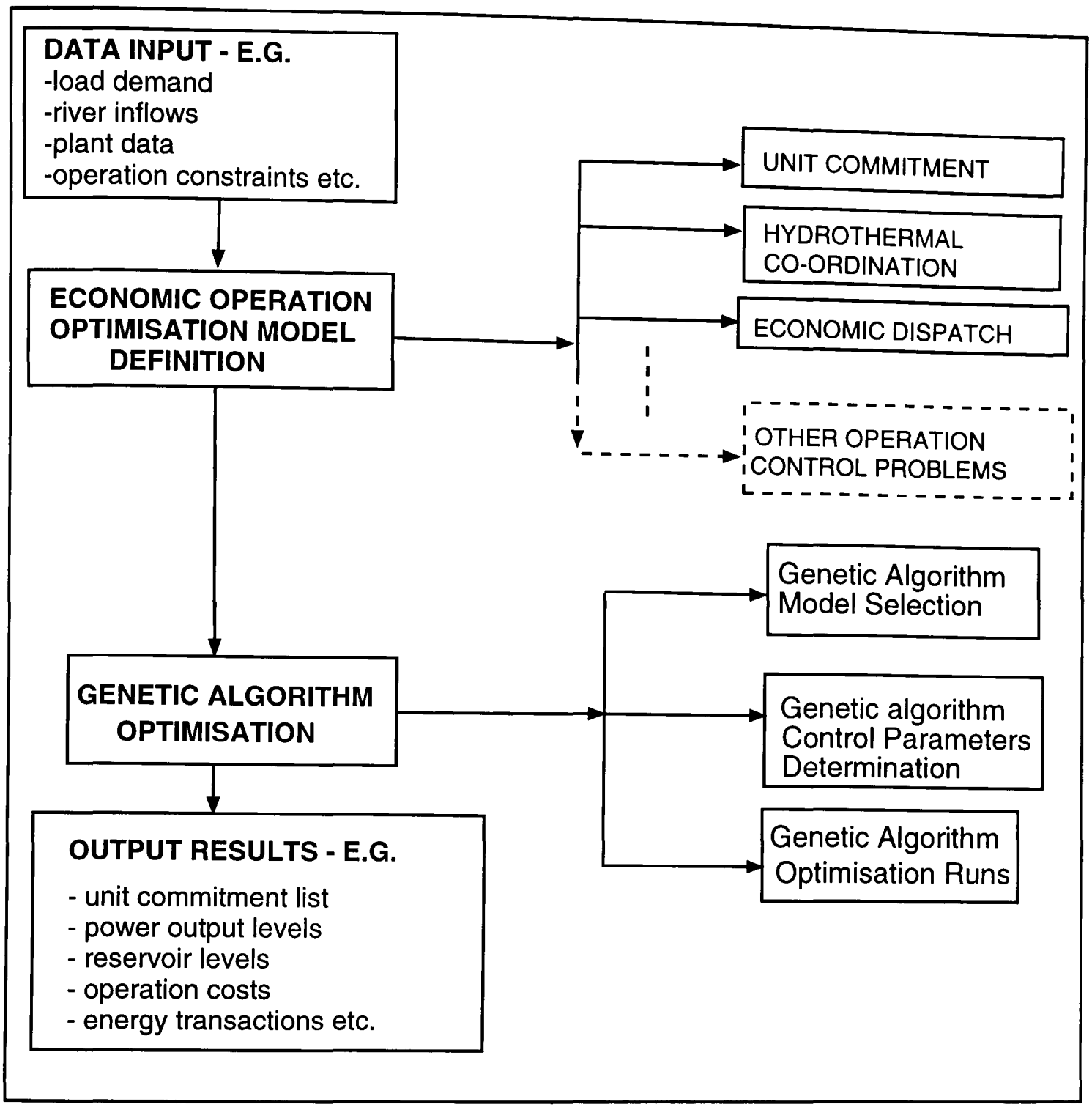



**Figure 10.1 Genetic Algorithm as an operational control optimisation tool**

The diagram illustrates the robust nature of the genetic algorithm optimisation method. The algorithm can be applied to a wide range of optimisation tasks, without changing the basic optimisation algorithm. Modelling the economic operation of power systems requires an optimisation method capable of handling discrete, discontinuous and non linear functions, as well as {0, 1} decision variables. The Genetic Algorithm method offers a global optimisation procedure which is independent of the form (convex, non convex etc.) of the solution space. Other advantages of the GA technique include:

- it is simple and hence has no major programming overheads,
- it is very robust and can be used in a number of different applications,
- it leads to an easier problem formulation and modelling.

Once a given operational control problem has been appropriately modelled, the genetic optimisation task can generally be applied to provide the desired optimisation results. The GA models described have solved the generation scheduling problems while taking into account various combinations of plant mix, problem constraints and operational scenarios. This thesis has demonstrated the ability of the genetic algorithm method in solving economic operation control problems in power systems.

## 10.2 Future Work

New possible explanations regarding the GA operation mechanism continue to emerge. The precise mathematical analysis of the complex dynamics of the GA process for practical problems is however far from being resolved. The main approach that seems to provide insight into the GA function is feedback between application results and the theoretical explanations of the results obtained. Power system optimisation is one of the large scale problems which can provide much needed feedback for further developments of the GA theory. This research has sought to answer some of the questions on the GA behaviour on real world problems, however there are a number of areas that need further work. These areas are described in the following sections.

### 10.2.1 Hybrid systems

Finding additional means of including domain knowledge with genetic algorithms to provide better solutions to the scheduling problem, where, for example, expert systems could be combined with GA methods, is an area that requires further research. Some of the possible hybrid systems are:

#### 10.2.1.1 Hybrid GA - Lagrangian Relaxation Method

For the thermal scheduling problem, an immediate practical research would involve the combination of the genetic algorithm with the Lagrangian Relaxation decomposition technique. Since the concept of a hybrid technique has been proved using the simple priority list unit commitment method, it is envisaged that better solutions can be obtained by combining the advantages of the Lagrangian Relaxation method with those of the genetic algorithm technique to produce a more powerful unit commitment algorithm. The combined GA-LR unit commitment process can be implemented in two alternative ways, as both algorithms are powerful in their own rights and are able to independently model most of the unit commitment problem constraints. In one alternative, the genetic algorithm acts as the main solution method and incorporates the lagrangian relaxation method as a part of the GA process, where the lagrangian relaxation unit commitment results provide part of the initial GA population. In the second alternative, the Lagrangian Relaxation method can either use the GA solution to estimate the initial values of the lagrange multipliers or it can use the GA to update the lagrange multipliers in the course of solution.

#### 10.2.1.2 Hybrid GA - Linear Programming Methods

Linear programming can be used to provide a coarse solution to the hydro sub system optimisation, before applying a genetic algorithm. [Hulselmann et. al.] use linear programming combined with genetic algorithms to provide optimal hydro generation scheduling for a cascaded run off river system. The merits and tradeoffs required to optimally combine these two techniques require further investigation. There are also possibilities of combining genetic algorithms with other non linear programming methods.

### 10.2.2 Detailed modelling of the scheduling problem

Further work is required in order to design GA models with the ability to handle a more detailed modelling of the generation scheduling problem that include constraints such as:

1. scheduling of multiple energy interchange contracts,
2. environmental constraints,
3. Fuel constraints.

### 10.2.3 Generalisation of the scheduling problem

In order to generalise the generation scheduling function, the following further research is necessary:

1. A re-assessment of the various generation scheduling objectives in a deregulated power supply industry, as it affects the different participants in the generation scheduling activities.
2. Extension of the GA scheduling method to cover much longer term scheduling periods, a requirement for fuel budgeting, longer term interchange contracts and maintenance planning activities.

### 10.2.4 Improvements to the GA computation method

To further improve the performance of the GA computation method, the more work is required in the following areas:

#### 10.2.4.1 General

1. Finding ways of further improving the computational efficiency of hybrid GA without compromising solution quality. This involves investigations of various methods of accelerating the algorithm convergence, as well as pre-optimisation heuristics and techniques.

2. Use of parallel processing techniques or other GA population structures, through an appropriate parallel GA implementation to speed up the scheduling GA solution process.

3. Carrying out investigations on better ways of grading the various problem constraints in the hydrothermal and thermal scheduling problem objective function formulation that applies the penalty function approach of constraint handling.

4. Investigations on the use of real number GA problem representation and other definition of problem specific crossover and mutation operators to enhance the handling of problem constraints.

5. Performing a statistical analysis using fitness landscape distance correlation analysis to determine the relationships among the major GA parameters of selection, crossover and population size, especially in cases where the problem is subject to a number of constraints, such as the hydrothermal and thermal scheduling problems. This should enable a more automated application of the GA scheduling algorithm without prior experimentation with the GA control parameter settings.

#### 10.2.4.2 Canonical (Standard) GA

1. Further research is required to establish an empirical relationship among the major GA control parameters of population size, crossover rates, mutation rates and selection mechanisms.

2. More work is required to determine the effects of selection pressure, fitness scaling mechanisms, elitism and population replacement methods on the convergence characteristics of the GA.

#### 10.2.4.3 Deterministic Crowding GA

1. Further investigations of the deterministic crowding GA methods, especially an investigation of the distance metrics used in the parent-selection replacement strategies, and other ways of improving the speed of convergence.

2. Investigating the relationships between the population size and the problem size for the deterministic crowding GA.

3. Extending the use of the deterministic crowding genetic algorithm to classification and multi objective optimisation problems in other power system operation control problems. Current problems that can benefit from the DCGA method include:

   a) classification of contingencies in power system security analysis,

   b) determination of an appropriate load model structure in ARMA load forecasting techniques,

c) the tradeoff in environmental - cost criteria in the economic dispatch problem,

d) the reactive power planning problem, such as the optimal allocation of reactive power sources in the network.

#### 10.2.4.4 Hardware GA Implementation

An interesting area of further research would be a look at the possibility of implementing the GA generation scheduling function in hardware. The technology of realising GAs in hardware is already available. [Higuchi et. al.] have developed a self adapting GA to control connections in programmable logic devices, while [Wirbel] has implemented GAs in a text compression chip. The ultimate goal of generation scheduling, is one that includes an optimal power flow sub function, which at the moment cannot be done because of software computation time limitations. Since the core GA operators of reproduction, crossover and mutation involve only random number generation, copying and partial exchanges, the GA can easily be implemented in hardware. This can be realised by the use of field programmable gate arrays that can enable easy re-programming for different GA fitness functions. Custom specific VLSI chips can also be used, although they are not as cost effective as field programmable gate arrays for low volume designs. GA hardware implementations can offer massive speed ups over the current software approaches, especially if the parallel nature of the GA is also exploited in the hardware implementation.

### 10.2.5 Implementation of GA Scheduling function in practical EMS systems

Further work is required to sort out the practical difficulties of integrating the GA scheduling functions in a practical energy management system. Daily generation scheduling in a modern EMS system involves a combination of the optimal scheduling of thermal and Hydro units and a post operation analysis. As competition becomes more prevalent in the electricity supply industry, there is a greater need for more accurate unit schedules. In the traditional scheduling program, the engine that performs the GA optimisation and production cost calculations, as well as the report generating activities are all done in languages such as FORTRAN, PASCAL or C. To speed up the whole scheduling process, and to move away from main frame computers, the auxiliary tasks, such as graphs and report generation can be done on Personal computers using tools such as spreadsheets, which are easier to use for building custom input / output displays, without complex programming. The use of click and drag tools provided by the combination of the mouse and window systems has greatly increased the versatility of the input / output system and it is important to build a good data interface between the GA optimisation function and the data storage systems. These data interfaces should be kept generic to ensure full compatibility with third party software packages. To maximise flexibility, data should stored in data bases as objects in the most basic format. Commands can be written in structured query language to retrieve the stored system and plant data which are then used in the GA scheduling process. To meet all these GA inte-

gration needs, a distributed client server architecture can be used in which PCs are used in the front end to prepare data and reports, while workstations are used as high speed servers to store data and perform the GA scheduling optimisation tasks. Further work is required to find out the optimum way of integrating the GA scheduling functions in such a multiple architecture EMS system.

### 10.2.6 Conclusion

Finally as nature continues to evolve, so will new research areas in the GA field continue to emerge, since this is a technique that emulates nature.

# Appendix A. Data for Thermal Scheduling Test Systems

## A.1 Data for 10 unit test system

**Table A.1 Total load and reserve requirements in MW**

| Time | Load | Reserve | Time | Load | Reserve |
|---|---|---|---|---|---|
| 1 | 1167 | 350 | 13 | 923 | 280 |
| 2 | 1097 | 329 | 14 | 910 | 270 |
| 3 | 1039 | 329 | 15 | 900 | 270 |
| 4 | 1028 | 300 | 16 | 876 | 260 |
| 5 | 1017 | 300 | 17 | 853 | 260 |
| 6 | 1051 | 300 | 18 | 829 | 250 |
| 7 | 1098 | 300 | 19 | 794 | 240 |
| 8 | 1051 | 300 | 20 | 782 | 240 |
| 9 | 1017 | 300 | 21 | 770 | 240 |
| 10 | 993 | 300 | 22 | 818 | 240 |
| 11 | 958 | 280 | 23 | 864 | 260 |
| 12 | 946 | 280 | 24 | 1167 | 350 |

**Table A.2 Unit characteristics and cost coefficients**

| Unit | Pmax (MW) | Pmin (MW) | MDT (hr) | MUT (hr) | IC (hr) | IP (MW) | a ($) | b ($/MW) | c ($/MW^2) | σ ($) | δ ($) | τ | SDC ($) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 60 | 15.0 | 2 | 3 | 3 | 60 | 15.000 | 1.400 | 0.0051 | 15 | 123 | 5 | 0 |
| 2 | 80 | 20.0 | 4 | 3 | 3 | 80 | 25.000 | 1.500 | 0.0040 | 15 | 123 | 5 | 0 |
| 3 | 100 | 30.0 | 4 | 4 | 4 | 100 | 40.000 | 1.350 | 0.0039 | 25 | 110 | 5 | 0 |
| 4 | 120 | 25.0 | 3 | 3 | 3 | 120 | 32.000 | 1.400 | 0.0038 | 12 | 100 | 5 | 0 |
| 5 | 150 | 50.0 | 3 | 1 | -3 | 0 | 29.000 | 1.540 | 0.0021 | 30 | 130 | 5 | 0 |
| 6 | 280 | 75.0 | 3 | 6 | -3 | 0 | 72.000 | 1.350 | 0.0026 | 30 | 146 | 6 | 0 |
| 7 | 520 | 250.0 | 4 | 10 | 10 | 520 | 105.000 | 1.395 | 0.0013 | 60 | 207 | 11 | 0 |
| 8 | 150 | 50.0 | 2 | 3 | 3 | 150 | 100.000 | 1.329 | 0.0014 | 80 | 202 | 11 | 0 |
| 9 | 320 | 120.0 | 5 | 7 | 7 | 320 | 49.000 | 1.264 | 0.0029 | 50 | 137 | 7 | 0 |
| 10 | 200 | 75.0 | 6 | 6 | 6 | 200 | 82.000 | 1.214 | 0.0015 | 70 | 157 | 9 | 0 |

where σ cold startup cost, δ hot startup cost, τ cooling time constant, SDC shutdown cost/hr, IP initial power, IC initial condition, a,b,c cost coefficients, MDT/MUT min. down/up times

## A.2 Data for 26 unit test system

### Table A.3 Load profile_1 in MW

| HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1700 | 5 | 1750 | 9 | 2540 | 13 | 2590 | 17 | 2550 | 21 | 2600 |
| 2 | 1730 | 6 | 1850 | 10 | 2600 | 14 | 2550 | 18 | 2530 | 22 | 2480 |
| 3 | 1690 | 7 | 2000 | 11 | 2670 | 15 | 2620 | 19 | 2500 | 23 | 2200 |
| 4 | 1700 | 8 | 2430 | 12 | 2590 | 16 | 2650 | 20 | 2550 | 24 | 1840 |
| Spinning reserve is set to cover the loss of the largest committed unit | | | | | | | | | | | |

### Table A.4 Load profile_2 in MW

| HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1430 | 5 | 1350 | 9 | 2300 | 13 | 2290 | 17 | 2190 | 21 | 2300 |
| 2 | 1450 | 6 | 1470 | 10 | 2380 | 14 | 2260 | 18 | 2200 | 22 | 2180 |
| 3 | 1400 | 7 | 1710 | 11 | 2290 | 15 | 2190 | 19 | 2300 | 23 | 1910 |
| 4 | 1350 | 8 | 2060 | 12 | 2370 | 16 | 2130 | 20 | 2340 | 24 | 1650 |
| Spinning reserve is set to cover the loss of the largest committed unit | | | | | | | | | | | |

### Table A.5 Unit characteristics and cost coefficients

| Unit | Pmax (MW) | Pmin (MW) | MDT (hr) | MUT (hr) | IC (hr) | IP (MW) | a ($) | b ($/MW) | c ($/MW^2) | σ ($) | δ ($) | τ | SDC ($) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.389 | 25.547 | 0.0253 | 0 | 0 | 1 | 0 |
| 2 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.411 | 25.675 | 0.0265 | 0 | 0 | 1 | 0 |
| 3 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.638 | 25.803 | 0.0280 | 0 | 0 | 1 | 0 |
| 4 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.760 | 25.932 | 0.0284 | 0 | 0 | 1 | 0 |
| 5 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.888 | 26.061 | 0.0286 | 0 | 0 | 1 | 0 |
| 6 | 20 | 4.0 | 0 | 0 | -1 | 0 | 117.755 | 37.551 | 0.0120 | 20 | 20 | 2 | 0 |
| 7 | 20 | 4.0 | 0 | 0 | -1 | 0 | 118.108 | 37.664 | 0.0126 | 20 | 20 | 2 | 0 |
| 8 | 20 | 4.0 | 0 | 0 | -1 | 0 | 118.458 | 37.777 | 0.0136 | 20 | 20 | 2 | 0 |
| 9 | 20 | 4.0 | 0 | 0 | -1 | 0 | 118.821 | 37.890 | 0.0143 | 20 | 20 | 2 | 0 |
| 10 | 76 | 15.2 | 2 | 3 | 3 | 76 | 81.136 | 13.327 | 0.0088 | 50 | 50 | 3 | 0 |
| 11 | 76 | 15.2 | 2 | 3 | 3 | 76 | 81.298 | 13.354 | 0.0089 | 50 | 50 | 3 | 0 |
| 12 | 76 | 15.2 | 2 | 3 | 3 | 76 | 81.464 | 13.380 | 0.0091 | 50 | 50 | 3 | 0 |
| 13 | 76 | 15.2 | 2 | 3 | 3 | 76 | 81.626 | 13.407 | 0.0093 | 50 | 50 | 3 | 0 |
| 14 | 100 | 25.0 | 2 | 4 | -3 | 0 | 217.895 | 18.000 | 0.0062 | 70 | 70 | 4 | 0 |
| 15 | 100 | 25.0 | 2 | 4 | -3 | 0 | 218.335 | 18.100 | 0.0061 | 70 | 70 | 4 | 0 |
| 16 | 100 | 25.0 | 2 | 4 | -3 | 0 | 218.775 | 18.200 | 0.0060 | 70 | 70 | 4 | 0 |
| 17 | 155 | 54.3 | 3 | 5 | 5 | 155 | 142.735 | 10.694 | 0.0046 | 150 | 150 | 6 | 0 |
| 18 | 155 | 54.3 | 3 | 5 | 5 | 155 | 143.029 | 10.715 | 0.0047 | 150 | 150 | 6 | 0 |
| 19 | 155 | 54.3 | 3 | 5 | 5 | 155 | 143.318 | 10.737 | 0.0048 | 150 | 150 | 6 | 0 |
| 20 | 155 | 54.3 | 3 | 5 | 5 | 155 | 143.597 | 10.758 | 0.0049 | 150 | 150 | 6 | 0 |
| 21 | 197 | 68.9 | 4 | 5 | -4 | 0 | 259.131 | 23.000 | 0.0026 | 200 | 200 | 8 | 0 |
| 22 | 197 | 68.9 | 4 | 5 | -4 | 0 | 259.649 | 23.100 | 0.0026 | 200 | 200 | 8 | 0 |
| 23 | 197 | 68.9 | 4 | 5 | -4 | 0 | 260.176 | 23.200 | 0.0026 | 200 | 200 | 8 | 0 |
| 24 | 350 | 140.0 | 5 | 8 | 10 | 350 | 177.057 | 10.862 | 0.0015 | 300 | 200 | 8 | 0 |
| 25 | 400 | 100.0 | 5 | 8 | 10 | 400 | 310.002 | 7.492 | 0.0019 | 500 | 500 | 10 | 0 |
| 26 | 400 | 100.0 | 5 | 8 | 10 | 400 | 311.910 | 7.503 | 0.0019 | 500 | 500 | 10 | 0 |

## A.3 Data for 100 unit test system

### Table A.6 Unit characteristics and cost coefficients

| Unit | Pmax (MW) | Pmin (MW) | MDT (hr) | MUT (hr) | IC (hr) | IP (MW) | a ($) | b ($/MW) | c ($/MW^2) | σ ($) | δ ($) | τ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.000 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 2 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.000 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 3 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.000 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 4 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.000 | 16.500 | 0.0021 | 1120 | 560 | 4 |
| 5 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.000 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 6 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.000 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 7 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.000 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 8 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.000 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 9 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.000 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 10 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.000 | 27.790 | 0.0017 | 60 | 30 | 0 |
| 11 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.100 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 12 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.100 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 13 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.100 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 14 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.100 | 16.500 | 0.0021 | 1120 | 560 | 4 |
| 15 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.100 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 16 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.100 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 17 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.100 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 18 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.100 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 19 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.100 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 20 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.100 | 27.790 | 0.0017 | 60 | 30 | 0 |
| 21 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.200 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 22 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.200 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 23 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.200 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 24 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.200 | 16.500 | 0.0021 | 1120 | 560 | 4 |
| 25 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.200 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 26 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.200 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 27 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.200 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 28 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.200 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 29 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.200 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 30 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.200 | 27.790 | 0.0017 | 60 | 30 | 0 |
| 31 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.300 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 32 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.300 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 33 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.300 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 34 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.300 | 16.500 | 0.0021 | 1120 | 560 | 4 |
| 35 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.300 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 36 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.300 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 37 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.300 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 38 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.300 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 39 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.300 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 40 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.300 | 27.790 | 0.0017 | 60 | 30 | 0 |
| 41 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.400 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 42 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.400 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 43 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.400 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 44 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.400 | 16.500 | 0.0021 | 1120 | 560 | 4 |
| 45 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.400 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 46 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.400 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 47 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.400 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 48 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.400 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 49 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.400 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 50 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.400 | 27.790 | 0.0017 | 60 | 30 | 0 |
| 51 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.500 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 52 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.500 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 53 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.500 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 54 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.500 | 16.500 | 0.0021 | 1120 | 560 | 4 |

## Table A.7 Unit characteristics and cost coefficients ( continued from table A.6)

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 55 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.500 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 56 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.500 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 57 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.500 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 58 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.500 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 59 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.500 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 60 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.500 | 27.790 | 0.0017 | 60 | 30 | 0 |
| 61 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.600 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 62 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.600 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 63 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.600 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 64 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.600 | 16.500 | 0.0021 | 1120 | 560 | 4 |
| 65 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.600 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 66 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.600 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 67 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.600 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 68 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.600 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 69 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.600 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 70 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.600 | 27.790 | 0.0017 | 60 | 30 | 0 |
| 71 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.800 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 72 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.800 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 73 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.800 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 74 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.800 | 16.500 | 0.0021 | 1120 | 560 | 4 |
| 75 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.800 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 76 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.800 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 77 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.800 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 78 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.800 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 79 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.800 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 80 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.800 | 27.790 | 0.0017 | 60 | 30 | 0 |
| 81 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.900 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 82 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.900 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 83 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.900 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 84 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.900 | 16.500 | 0.0021 | 1120 | 560 | 4 |
| 85 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.900 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 86 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.900 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 87 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.900 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 88 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.900 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 89 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.900 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 90 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.900 | 27.790 | 0.0017 | 60 | 30 | 0 |
| 91 | 455 | 150.0 | 8 | 8 | 8 | 455 | 1000.990 | 16.190 | 0.0005 | 9000 | 4500 | 5 |
| 92 | 455 | 150.0 | 8 | 8 | 8 | 455 | 970.990 | 17.260 | 0.0003 | 10000 | 5000 | 5 |
| 93 | 130 | 20.0 | 5 | 5 | -5 | 130 | 700.990 | 16.600 | 0.0020 | 1100 | 550 | 4 |
| 94 | 130 | 20.0 | 5 | 5 | -5 | 130 | 680.990 | 16.500 | 0.0021 | 1120 | 560 | 4 |
| 95 | 162 | 25.0 | 6 | 6 | -6 | 162 | 450.990 | 19.700 | 0.0040 | 1800 | 900 | 4 |
| 96 | 80 | 20.0 | 3 | 3 | -3 | 80 | 370.990 | 22.260 | 0.0071 | 340 | 170 | 2 |
| 97 | 85 | 25.0 | 3 | 3 | -3 | 85 | 480.990 | 27.740 | 0.0008 | 520 | 260 | 2 |
| 98 | 55 | 10.0 | 1 | 1 | -1 | 55 | 660.990 | 25.920 | 0.0041 | 60 | 30 | 0 |
| 99 | 55 | 10.0 | 1 | 1 | -1 | 55 | 665.990 | 27.270 | 0.0022 | 60 | 30 | 0 |
| 100 | 55 | 10.0 | 1 | 1 | -1 | 55 | 670.990 | 27.790 | 0.0017 | 60 | 30 | 0 |

## Table A.8 Load demand in MW

| HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7000 | 5 | 10000 | 9 | 13000 | 13 | 14000 | 17 | 10000 | 21 | 13000 |
| 2 | 7500 | 6 | 11000 | 10 | 14000 | 14 | 13000 | 18 | 11000 | 22 | 11000 |
| 3 | 8500 | 7 | 11500 | 11 | 14500 | 15 | 12000 | 19 | 12000 | 23 | 9000 |
| 4 | 9500 | 8 | 12000 | 12 | 15000 | 16 | 10500 | 20 | 14000 | 24 | 8000 |
| Spinning reserve is set at 10% of the load demand | | | | | | | | | | | |

## A.4 Data for 110 unit test system

### Table A.9 Unit characteristics and cost coefficients

| Unit | Pmax (MW) | Pmin (MW) | MDT (hr) | MUT (hr) | IC (hr) | IP (MW) | a ($) | b ($/MW) | c ($/MW^2) | σ ($) | δ ($) | τ | SDC ($) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.389 | 25.547 | 0.0253 | 0 | 0 | 1 | 0 |
| 2 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.411 | 25.675 | 0.0265 | 0 | 0 | 1 | 0 |
| 3 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.638 | 25.803 | 0.0280 | 0 | 0 | 1 | 0 |
| 4 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.760 | 25.932 | 0.0284 | 0 | 0 | 1 | 0 |
| 5 | 12 | 2.4 | 0 | 0 | -1 | 0 | 24.888 | 26.061 | 0.0286 | 0 | 0 | 1 | 0 |
| 6 | 20 | 4.0 | 0 | 0 | -1 | 0 | 117.755 | 37.551 | 0.0120 | 20 | 20 | 2 | 0 |
| 7 | 20 | 4.0 | 0 | 0 | -1 | 0 | 118.108 | 37.664 | 0.0126 | 20 | 20 | 2 | 0 |
| 8 | 20 | 4.0 | 0 | 0 | -1 | 0 | 118.458 | 37.777 | 0.0136 | 20 | 20 | 2 | 0 |
| 9 | 20 | 4.0 | 0 | 0 | -1 | 0 | 118.821 | 37.890 | 0.0143 | 20 | 20 | 2 | 0 |
| 10 | 76 | 15.2 | 2 | 3 | 3 | 76 | 81.136 | 13.327 | 0.0088 | 50 | 50 | 3 | 0 |
| 11 | 76 | 15.2 | 2 | 3 | 3 | 76 | 81.298 | 13.354 | 0.0089 | 50 | 50 | 3 | 0 |
| 12 | 76 | 15.2 | 2 | 3 | 3 | 76 | 81.464 | 13.380 | 0.0091 | 50 | 50 | 3 | 0 |
| 13 | 76 | 15.2 | 2 | 3 | 3 | 76 | 81.626 | 13.407 | 0.0093 | 50 | 50 | 3 | 0 |
| 14 | 100 | 25.0 | 2 | 4 | -3 | 0 | 217.895 | 18.000 | 0.0062 | 70 | 70 | 4 | 0 |
| 15 | 100 | 25.0 | 2 | 4 | -3 | 0 | 218.335 | 18.100 | 0.0061 | 70 | 70 | 4 | 0 |
| 16 | 100 | 25.0 | 2 | 4 | -3 | 0 | 218.775 | 18.200 | 0.0060 | 70 | 70 | 4 | 0 |
| 17 | 155 | 54.3 | 3 | 5 | 5 | 155 | 142.735 | 10.694 | 0.0046 | 150 | 150 | 6 | 0 |
| 18 | 155 | 54.3 | 3 | 5 | 5 | 155 | 143.029 | 10.715 | 0.0047 | 150 | 150 | 6 | 0 |
| 19 | 155 | 54.3 | 3 | 5 | 5 | 155 | 143.318 | 10.737 | 0.0048 | 150 | 150 | 6 | 0 |
| 20 | 155 | 54.3 | 3 | 5 | 5 | 155 | 143.597 | 10.758 | 0.0049 | 150 | 150 | 6 | 0 |
| 21 | 197 | 68.9 | 4 | 5 | -4 | 0 | 259.131 | 23.000 | 0.0026 | 200 | 200 | 8 | 0 |
| 22 | 197 | 68.9 | 4 | 5 | -4 | 0 | 259.649 | 23.100 | 0.0026 | 200 | 200 | 8 | 0 |
| 23 | 197 | 68.9 | 4 | 5 | -4 | 0 | 260.176 | 23.200 | 0.0026 | 200 | 200 | 8 | 0 |
| 24 | 350 | 140.0 | 5 | 8 | 10 | 350 | 177.057 | 10.862 | 0.0015 | 300 | 200 | 8 | 0 |
| 25 | 400 | 100.0 | 5 | 8 | 10 | 400 | 310.002 | 7.492 | 0.0019 | 500 | 500 | 10 | 0 |
| 26 | 400 | 100.0 | 5 | 8 | 10 | 400 | 311.910 | 7.503 | 0.0019 | 500 | 500 | 10 | 0 |
| 27 | 500 | 140.0 | 5 | 8 | 5 | 500 | 210.000 | 12.000 | 0.0014 | 500 | 800 | 4 | 0 |
| 28 | 500 | 140.0 | 7 | 8 | -2 | 0 | 180.000 | 12.100 | 0.0013 | 250 | 800 | 4 | 0 |
| 29 | 200 | 50.0 | 4 | 4 | 1 | 200 | 240.000 | 12.200 | 0.0026 | 40 | 300 | 3 | 0 |
| 30 | 100 | 25.0 | 3 | 2 | -2 | 0 | 220.000 | 12.500 | 0.0039 | 10 | 60 | 2 | 0 |
| 31 | 50 | 10.0 | 2 | 1 | -3 | 0 | 60.000 | 23.000 | 0.0051 | 25 | 10 | 1 | 0 |
| 32 | 20 | 5.0 | 1 | 1 | -2 | 0 | 50.000 | 13.500 | 0.0050 | 10 | 15 | 1 | 0 |
| 33 | 80 | 20.0 | 2 | 3 | -4 | 0 | 200.000 | 13.200 | 0.0078 | 40 | 30 | 2 | 0 |
| 34 | 250 | 75.0 | 4 | 4 | -1 | 0 | 140.000 | 12.400 | 0.0012 | 50 | 20 | 3 | 0 |
| 35 | 360 | 110.0 | 4 | 5 | -2 | 0 | 120.000 | 10.300 | 0.0038 | 200 | 35 | 4 | 0 |
| 36 | 400 | 130.0 | 8 | 8 | 3 | 400 | 90.000 | 9.900 | 0.0043 | 400 | 30 | 5 | 0 |
| 37 | 40 | 10.0 | 1 | 1 | -1 | 0 | 80.000 | 13.400 | 0.0011 | 10 | 20 | 1 | 0 |
| 38 | 70 | 20.0 | 1 | 1 | -2 | 0 | 70.000 | 13.300 | 0.0023 | 50 | 300 | 1 | 0 |
| 39 | 100 | 25.0 | 2 | 2 | -1 | 0 | 115.000 | 12.900 | 0.0034 | 10 | 150 | 2 | 0 |
| 40 | 120 | 20.0 | 2 | 4 | -3 | 0 | 150.000 | 12.800 | 0.0067 | 15 | 120 | 3 | 0 |
| 41 | 180 | 40.0 | 3 | 4 | -5 | 0 | 40.000 | 12.700 | 0.0056 | 50 | 80 | 3 | 0 |
| 42 | 220 | 50.0 | 4 | 5 | -1 | 0 | 300.000 | 12.600 | 0.0023 | 150 | 50 | 3 | 0 |
| 43 | 440 | 120.0 | 8 | 7 | 2 | 440 | 250.000 | 7.400 | 0.0012 | 450 | 30 | 4 | 0 |
| 44 | 560 | 160.0 | 8 | 8 | -6 | 0 | 100.000 | 6.600 | 0.0045 | 300 | 45 | 5 | 0 |
| 45 | 660 | 150.0 | 9 | 9 | 4 | 660 | 160.000 | 6.500 | 0.0022 | 400 | 50 | 6 | 0 |
| 46 | 700 | 200.0 | 12 | 12 | 4 | 700 | 130.000 | 6.200 | 0.0067 | 650 | 70 | 8 | 0 |
| 47 | 32 | 5.4 | 0 | 0 | -1 | 0 | 34.389 | 26.547 | 0.0353 | 0 | 0 | 1 | 0 |
| 48 | 32 | 5.4 | 0 | 0 | -1 | 0 | 34.411 | 26.675 | 0.0365 | 0 | 0 | 1 | 0 |
| 49 | 52 | 8.4 | 1 | 1 | -1 | 0 | 34.638 | 26.803 | 0.0380 | 0 | 0 | 1 | 0 |
| 50 | 52 | 8.4 | 1 | 1 | -1 | 0 | 34.761 | 26.932 | 0.0384 | 0 | 0 | 1 | 0 |
| 51 | 52 | 8.4 | 1 | 1 | -1 | 0 | 34.888 | 27.061 | 0.0386 | 0 | 0 | 1 | 0 |
| 52 | 60 | 12.0 | 1 | 2 | -1 | 0 | 127.755 | 38.551 | 0.0320 | 30 | 30 | 2 | 0 |
| 53 | 60 | 12.0 | 1 | 2 | -1 | 0 | 128.108 | 38.664 | 0.0326 | 30 | 30 | 2 | 0 |
| 54 | 60 | 12.0 | 1 | 2 | -1 | 0 | 128.458 | 38.777 | 0.0236 | 30 | 30 | 2 | 0 |
| 55 | 60 | 12.0 | 1 | 2 | -1 | 0 | 128.821 | 38.890 | 0.0243 | 30 | 30 | 2 | 0 |

## Table A.10 Unit characteristics and cost coefficients (continued from table A.9)

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 56 | 96 | 25.2 | 2 | 3 | 3 | 96 | 82.136 | 14.327 | 0.0098 | 60 | 60 | 3 | 0 |
| 57 | 96 | 25.2 | 2 | 3 | 3 | 96 | 82.298 | 14.354 | 0.0099 | 60 | 60 | 3 | 0 |
| 58 | 100 | 35.0 | 3 | 3 | 3 | 100 | 82.464 | 14.380 | 0.0092 | 60 | 60 | 3 | 0 |
| 59 | 100 | 35.0 | 3 | 3 | 3 | 100 | 82.626 | 14.407 | 0.0094 | 60 | 60 | 3 | 0 |
| 60 | 120 | 45.0 | 3 | 4 | -3 | 0 | 218.895 | 19.000 | 0.0072 | 80 | 80 | 4 | 0 |
| 61 | 120 | 45.0 | 3 | 4 | -3 | 0 | 219.335 | 19.100 | 0.0071 | 80 | 80 | 4 | 0 |
| 62 | 120 | 45.0 | 3 | 4 | -3 | 0 | 219.775 | 19.200 | 0.0070 | 80 | 80 | 4 | 0 |
| 63 | 185 | 54.3 | 4 | 5 | 5 | 185 | 143.735 | 11.694 | 0.0066 | 160 | 160 | 6 | 0 |
| 64 | 185 | 54.3 | 4 | 5 | 5 | 185 | 144.029 | 11.715 | 0.0057 | 160 | 160 | 6 | 0 |
| 65 | 185 | 54.3 | 4 | 5 | 5 | 185 | 144.318 | 11.737 | 0.0058 | 160 | 160 | 6 | 0 |
| 66 | 185 | 54.3 | 4 | 5 | 5 | 185 | 144.597 | 11.758 | 0.0059 | 160 | 160 | 6 | 0 |
| 67 | 197 | 70.0 | 4 | 5 | -4 | 0 | 269.131 | 24.000 | 0.0036 | 210 | 210 | 8 | 0 |
| 68 | 197 | 70.0 | 4 | 5 | -4 | 0 | 269.649 | 24.100 | 0.0036 | 210 | 210 | 8 | 0 |
| 69 | 197 | 70.0 | 4 | 5 | -4 | 0 | 270.176 | 24.200 | 0.0036 | 210 | 210 | 8 | 0 |
| 70 | 360 | 150.0 | 5 | 8 | 10 | 360 | 187.057 | 11.862 | 0.0025 | 310 | 310 | 8 | 0 |
| 71 | 400 | 160.0 | 6 | 8 | 9 | 400 | 320.002 | 8.492 | 0.0029 | 510 | 510 | 10 | 0 |
| 72 | 400 | 160.0 | 6 | 8 | 9 | 400 | 321.910 | 8.503 | 0.0030 | 510 | 510 | 10 | 0 |
| 73 | 300 | 60.0 | 4 | 4 | -1 | 0 | 52.136 | 13.327 | 0.0054 | 40 | 60 | 3 | 0 |
| 74 | 250 | 50.0 | 3 | 3 | -1 | 0 | 42.298 | 12.354 | 0.0055 | 65 | 70 | 2 | 0 |
| 75 | 90 | 30.0 | 2 | 2 | -1 | 0 | 32.464 | 11.380 | 0.0099 | 60 | 90 | 2 | 0 |
| 76 | 50 | 12.0 | 1 | 1 | -1 | 0 | 23.626 | 9.407 | 0.0031 | 68 | 30 | 2 | 0 |
| 77 | 450 | 160.0 | 5 | 6 | 5 | 450 | 220.000 | 14.000 | 0.0024 | 600 | 900 | 4 | 0 |
| 78 | 600 | 150.0 | 7 | 8 | -2 | 0 | 190.000 | 13.100 | 0.0023 | 350 | 900 | 4 | 0 |
| 79 | 200 | 50.0 | 4 | 4 | 1 | 200 | 250.000 | 13.200 | 0.0036 | 50 | 400 | 3 | 0 |
| 80 | 120 | 20.0 | 3 | 3 | -2 | 0 | 230.000 | 13.500 | 0.0049 | 20 | 70 | 2 | 0 |
| 81 | 55 | 10.0 | 1 | 1 | -3 | 0 | 70.000 | 24.000 | 0.0061 | 35 | 20 | 1 | 0 |
| 82 | 40 | 12.0 | 1 | 1 | -2 | 0 | 60.000 | 14.500 | 0.0070 | 40 | 25 | 1 | 0 |
| 83 | 80 | 20.0 | 2 | 2 | -4 | 0 | 210.000 | 14.200 | 0.0088 | 50 | 40 | 2 | 0 |
| 84 | 200 | 50.0 | 4 | 4 | 1 | 200 | 150.000 | 13.400 | 0.0022 | 60 | 30 | 3 | 0 |
| 85 | 325 | 80.0 | 4 | 4 | -2 | 0 | 130.000 | 11.300 | 0.0048 | 300 | 45 | 4 | 0 |
| 86 | 440 | 120.0 | 5 | 6 | 3 | 440 | 80.000 | 8.900 | 0.0053 | 500 | 40 | 5 | 0 |
| 87 | 35 | 10.0 | 0 | 0 | -1 | 0 | 90.000 | 14.400 | 0.0021 | 20 | 30 | 1 | 0 |
| 88 | 55 | 20.0 | 1 | 1 | -2 | 0 | 80.000 | 14.300 | 0.0033 | 60 | 400 | 1 | 0 |
| 89 | 100 | 20.0 | 3 | 2 | -1 | 0 | 125.000 | 13.900 | 0.0034 | 20 | 160 | 2 | 0 |
| 90 | 220 | 40.0 | 2 | 3 | -3 | 0 | 160.000 | 13.800 | 0.0037 | 25 | 130 | 3 | 0 |
| 91 | 140 | 30.0 | 3 | 3 | -4 | 0 | 50.000 | 13.700 | 0.0066 | 60 | 90 | 3 | 0 |
| 92 | 100 | 40.0 | 3 | 2 | -1 | 0 | 400.000 | 13.600 | 0.0043 | 160 | 40 | 3 | 0 |
| 93 | 440 | 100.0 | 6 | 6 | 2 | 440 | 260.000 | 8.400 | 0.0022 | 460 | 40 | 4 | 0 |
| 94 | 500 | 100.0 | 8 | 8 | -6 | 0 | 110.000 | 7.600 | 0.0055 | 310 | 55 | 5 | 0 |
| 95 | 600 | 100.0 | 9 | 8 | 4 | 600 | 170.000 | 7.500 | 0.0032 | 410 | 60 | 6 | 0 |
| 96 | 700 | 200.0 | 12 | 12 | 4 | 700 | 140.000 | 7.200 | 0.0077 | 660 | 80 | 8 | 0 |
| 97 | 15 | 3.6 | 0 | 0 | -1 | 0 | 26.389 | 26.547 | 0.0353 | 0 | 0 | 1 | 0 |
| 98 | 15 | 3.6 | 0 | 0 | -1 | 0 | 25.411 | 26.675 | 0.0365 | 0 | 0 | 1 | 0 |
| 99 | 22 | 4.4 | 0 | 0 | -1 | 0 | 25.638 | 26.803 | 0.0380 | 0 | 0 | 1 | 0 |
| 100 | 22 | 4.4 | 0 | 0 | -1 | 0 | 25.760 | 26.932 | 0.0384 | 0 | 0 | 1 | 0 |
| 101 | 60 | 10.0 | 1 | 3 | -1 | 0 | 65.000 | 15.300 | 0.0210 | 20 | 85 | 5 | 15 |
| 102 | 80 | 10.0 | 1 | 3 | -1 | 0 | 82.000 | 16.000 | 0.0230 | 20 | 101 | 5 | 25 |
| 103 | 100 | 20.0 | 2 | 4 | 1 | 100 | 86.000 | 20.200 | 0.0240 | 22 | 114 | 5 | 40 |
| 104 | 120 | 20.0 | 2 | 4 | 5 | 120 | 84.000 | 20.200 | 0.0350 | 10 | 94 | 5 | 32 |
| 105 | 150 | 40.0 | 3 | 5 | -7 | 0 | 75.000 | 25.600 | 0.0340 | 18 | 113 | 5 | 29 |
| 106 | 280 | 40.0 | 2 | 5 | 3 | 280 | 56.000 | 30.500 | 0.0370 | 27 | 176 | 6 | 42 |
| 107 | 520 | 50.0 | 7 | 7 | -5 | 0 | 67.000 | 32.500 | 0.0390 | 34 | 267 | 11 | 75 |
| 108 | 150 | 30.0 | 2 | 4 | 3 | 150 | 68.000 | 26.000 | 0.0350 | 45 | 282 | 11 | 49 |
| 109 | 320 | 40.0 | 5 | 5 | -6 | 0 | 69.000 | 25.800 | 0.0280 | 38 | 187 | 7 | 70 |
| 110 | 200 | 20.0 | 5 | 5 | -3 | 0 | 72.000 | 27.000 | 0.0260 | 26 | 227 | 9 | 62 |

## Table A.11 Load demand in MW

| HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD | HR | LOAD |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 11600 | 5 | 10500 | 9 | 13500 | 13 | 13200 | 17 | 14000 | 21 | 16500 |
| 2 | 10900 | 6 | 11200 | 10 | 14500 | 14 | 13000 | 18 | 14700 | 22 | 15000 |
| 3 | 9500 | 7 | 12500 | 11 | 14600 | 15 | 14500 | 19 | 15600 | 23 | 14300 |
| 4 | 9300 | 8 | 12900 | 12 | 14000 | 16 | 14600 | 20 | 16200 | 24 | 13500 |
| Spinning reserve is set to cover the loss of the largest committed unit | | | | | | | | | | | |

# Appendix B. Personal Communications

## Personal Communication from A. Bakirtzis

Date: Sat, 16 Sep 95 20:16:09 +0300
From: bakiana@eng.auth.gr (Anastasios Bakirtzis)
Message-Id: <9509161716.AA02625@vergina.eng.auth.gr>
Organisation: School of Engineering Aristotle University of Thessaloniki
Thessaloniki 54006 Macedonia Greece
To: Shadrack.Orero@brunel.ac.uk
Subject: **Correction to GA-UC 110unit results**
Status: RO

Dear Mr. Orero,
Thank you for the unit commitment results of your 110 unit test system. Unfortunately due to a silly mistake I made in modeling your version of the exponentialy time dependent start up cost, which I discovered yesterday, the results I e-mailed you are not correct. So, in a separate e-mail I send you the results after the correction of the error.
Comparing the operating cost of both the 100 unit and the 110 unit systems it seems that your GA implementation works much better than ours since it gives about 1% lower cost in both cases. You don't need to bother checking our dispatch routine. I checked your results with our dispatch routine and I got almost the same operating cost you got. If you have any publications on your GA solution to UC I would be greatful if you could send me a copy of your work. Congratulations on your nice work.

Yours sincerely,
Tasos Bakirtzis

## Personal Comminucation from B.F. Wollenberg

To: Shadrack.Orero@brunel.ac.uk
From: "Bruce F. Wollenberg" <wollenbe@ee.umn.edu>
Subject: Re: **LANGRANGIAN RELAXATION FOR UC AND NEW BOOK.**
Date: Mon, 4 Mar 1996 07:06:05

Dear Shadrack:

Indeed, the LR method is critically dependent on the lambda update procedures and the initial lambda values. However, usual practice is to use a prioirty list scheme or a dynamic programming scheme to get an initial solution, and perhaps to make the final adjustments at the end.

The prioirty list schemes willnot always come up with the best schedule because of the lack of a complete set of unit states, however as soon as one uses a complete set of unit states you are faced with an overwhelming number of states and cannot solve the problem - LR overcomes this nicely. In my opinion, LR is the state of the art and is advancing to where it includes all kinds of constraints like pollution, unit ramping, and even transmission security constraints.

Bruce Wollenberg

---

Bruce F. Wollenberg
Electrical Engineering Dept.
University of Minnesota Phone (612) 626-7192
200 Union Street SE Fax: (612) 625-4583
Minneapolis MN 55455 email wollenbe@ee.umn.edu

---

# Chapter 11. Bibliography and References


[1] Amado, S.M., Ribeiro, C.C., "Short-term generation scheduling of hydraulic multi-reservoir, multi-area interconnected systems.", *IEEE Trans. PWRS*, Vol. 2, No. 3, Aug. 1987, pp. 758-763.

[2] Annakkage, U.D., Numnonda, T., Pahalawaththa, N.C. "Unit commitment by parallel simulated annealing.", *Proc. IEE*, Pt. C, Vol. 142, No. 6, Nov. 1995, pp. 595-600.

[3] Antonisse, J., "A new interpretation of schema notion that overturns the binary encoding constraint.", *In D. Schaffer, (Ed.) Proc. of Third Int. Conf. on Genetic Algorithms,* 1989, pp. 86-91.

[4] Aoki, K., Satoh, T., Itoh, M., "Unit commitment in a large-scale power system including fuel constrained thermal and pumped-storage hydro.", *IEEE Trans. PWRS*, Vol. 1, No. 4, Nov. 1986, pp. 1077-1084

[5] Arvanitidis, N.V., Rosing, J., "Composite representation of multi-reservoir hydrothermal power system.", *IEEE Trans. PAS*, Vol. 89, No. 2, Feb. 1970, pp. 319-326.

[6] Ayoub, A. K., Patton, A.D., "Optimal thermal generating unit commitment.", *IEEE Trans. PAS*, Vol. 78, Jan. 1960, pp. 1272-1284.

[7] Back, T., "Order statistics for convergence velocity analysis in simplified evolutionary algorithms.", *In Whitley, D., Vose, M., (Eds.), Proc. of Foundations of Genetic Algorithm Workshop III,* 1994, Cambridge, MIT press.

[8] Back, T., "Optimal mutation rates in genetic search.", *In S. Forrest (Ed.), Proc. of the Fifth Int. Conf. on Genetic Algorithms, San Mateo, CA: Morgan Kauffman*, 1993, pp. 2-8.

[9] Back, T., "The interaction of mutation rate, selection and self adaptation within a genetic algorithm.", *In R. Manner, Manderick, B. (Eds.), Parallel Problem Solving in Nature II*, 1992, pp. 85-94, Elsevier: Amsterdam.

[10] Back, T., Schwefel, H.-P., "An overview of evolutionary algorithms for parameter optimisation", *Jnl. of Evolutionary Computation,* Vol. 1, No. 1, 1993, pp. 1-24.

[11] Back, T., Hoffmeister, F., "Extended selection mechanisms in genetic algorithms.", *In R.K. Belew, L.B. Booker (Eds.), Proc. of the fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 92-99.

[12] Back, T., Hoffmeister, F., Schwefel, H.-P., "A survey of evolution strategies." *In R.K. Belew, L.B. Booker (Eds.), Proc. of the Fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 2-9.

[13] Bainbridge, E.S., McNamee, J.M., Robinson, D.J., Nevison, R.D., "Hydrothermal dispatch with pumped storage.", *IEEE Trans. PAS*, Vol. 85, No. 5, 1966, pp. 472-485.

[14] Baker, J.E. "Reducing bias and inefficiency in the selection algorithm.". *In J.J. Grefenstette(Ed.). Proc. of the Second Int. Conf. on Genetic Algorithms*, 1987, pp. 14-21.

[15] Bakirtzis, A., Petridis, V., Kazarlis, S., "A genetic algorithm solution to the economic dispatch problem.", *Proc. IEE Pt. C*, Vol. 141, No .4, 1994, pp. 377-382.

[16] Baldwin, C.J., Dale, K.M., Dittrich, R.F., "A study of economic shut down of generating units in daily dispatch.", *IEEE Trans. PAS*, Vol. 90, Jan. 1971, pp. 1752-1756.

[17] Bard, J.F., "Short term scheduling of thermal-electric generators using Lagrangian relaxation.", *Operations Research*, Vol. 36, No. 5, Sept. / Oct. 1988, pp. 756-766.

[18] Beasley, D., Bull, D.R., Martin, R.R., "A sequential niche technique for multimodal function optimisation.", *Jnl. of Evolutionary Computation*, Vol. 1, No. 2, 1993, pp. 101-125.

[19] Belding, T., "The distributed genetic algorithm revisited.", *In D. Eshelman, (Ed.) , Proc. of the Sixth Int. Conf. on Genetic Algorithms*, San Mateo : Morgan Kaufmann, 1995.

[20] Belew, R.K., "When both the individuals and populations search: Adding simple learning to the genetic algorithm.", *In J. D. Schaffer (Ed.), Proc. of the Third Int. Conf. on Genetic Algorithms*, 1989, pp. 34-41.

[21] Bellman, R.E., Drefus, S.E., "Applied Dynamic Programming.", *Princeton University press*, NJ, 1962.

[22] Bertsekas, D.P., "Constrained Optimisation and Lagrange Multipliers.", *New York: Academic Press*, 1982.

[23] Bertsekas, D.P., Lauer, G.S., Sander, Posbergh, T.A., "Optimal short term scheduling of large scale power systems.", *IEEE Trans. AC,* Vol. 28, No. 1, Jan. 1983, pp. 1-11.

[24] Bissonette, V., Lafond, L., Cote, G.," A hydrothermal scheduling model for the hydro-Quebec production system.", *IEEE Trans. on PWRS*, Vol. 1, No. 2, 1986, pp. 204-210.

[25] Bonaert, A.P., El-Abiad, A.H., Koivo, A.J., "Effects of hydro dynamics on optimum scheduling of thermal hydro power systems.", *IEEE Trans. PAS*, Vol. 91, 1972, pp. 1412-1419.

[26] Boone G., Chiang, H., "Optimal capacitor placement in distribution systems by genetic algorithm.", *Int. Jnl. of Electrical Power and Energy Systems*, Vol. 15, No. 3, 1993, pp. 155-162.

[27] Box, G.E.P., "Evolutionary operation: A method for increasing industrial productivity.", *Jnl. of the Royal Statistical Society*, 6 (2), 1957, pp. 81-101.

[28] Brannud, H., Bubenko, J.A., Sjelvgren, D., "Optimal short term operation planning of a large hydrothermal power system based on a non linear network flow concept.", *IEEE Trans. PWRS*, Vol. 1, No. 4, Nov. 1986, pp. 75-82.

[29] Brannud, H., Sjelvgren, D., Bubenko, J.A., "Short term generation scheduling with security constraints.", *IEEE Trans. PWRS*, Vol. 3, No. 1, Feb.. 1988, pp. 310-316.

[30] Bremerman, H.J., "Numerical optimisation procedures derived from biological evolution processes.", *In H.L. Oestreicher, D.R. Moore (Eds.), Cybernetic Problems in Bionics* 1968, pp. 597-615. New York: Gordon and Breach.

[31] Bridges, C.L., Goldberg, D.E., "An analysis of reproduction and crossover in a binary-coded genetic algorithm." *In J. J Grefenstette (Ed.), Proc. of the Second Int. Conf. on Genetic Algorithms*, 1987, pp. 9-13.

[32] Bubenko, J.A., Waern, B.M., "Short range hydro optimisation by the pontraygin maximum principle.", *Proc. Fourth PSCC*, Grenoble, France, 1972, Paper 2.1/13.

[33] Calderon, L.R., Galiana, F.D., "Continuous solution simulation in the short term hydrothermal co-ordination problem.", *IEEE Trans. PWRS*, Vol. 2, No. 3, Aug. 1987, pp. 737-743.

[34] Caneiro, A.A.F, Soares, S., Bond, P.S., "A large scale application of an optimal deterministic hydrothermal scheduling algorithm.", *IEEE Trans. on PWRS*, Vol. 5 No. 1, 1990, pp. 204-211.

[35] Cavalho, M.F., Soares, S., "An efficient hydro-thermal scheduling algorithm.", *IEEE Trans. PWRS,* Vol. 2, No. 3, Aug. 1987, pp. 537-542.

[36] Chandler, W.G., Dandeno, P.L., Glimn, A.F., Kirchmayer, L.K., "Short range economic operation of a combined thermal and hydroelectric power system.", *AIEE Trans. PAS*, Vol. 72, Part III, 1953, pp. 1057-1065.

[37] Chang., S., Chen, C., Fong, I., Luh, P.B., "Hydroelectric generation scheduling with an effective differential dynamic programming.", *IEEE Trans. PWRS*, Vol. 5, No. 3, 1990, pp. 737-743.

[38] Chen, C.L., Wang, S.C., "Branch and bound scheduling for thermal units.", *IEEE Trans. on Energy Conversion*, Vol. 8, No. 2, June 1993, pp. 184-189.

[39] Chen, P.-H, , Chang, H.-C., "Large Scale economic dispatch by genetic algorithm.", *IEEE Trans. PWRS*, Vol. 10, No. 4, Nov. 1995, pp. 1919-1925.

[40] Christensen, G.S., Soliman, S.A., "Optimal Long Term Operation of Electric Power Systems.", *Plenum Press, New York,* 1988.

[41] Christensen, G.S., Soliman, S.A., "On the application of functional analysis to the optimisation of the production of hydroelectric power", *IEEE Trans. PWRS*, Vol. 2, No. 4, Nov. 1987, pp. 841-847.

[42] Cohen, A.I., "Modelling ramp limitations in unit commitment.", *Proc. 12th PSCC*, 1987, pp. 1107-1114.

[43] Cohen, A.I., Sherkat, V.R., "Optimisation-Based Methods for Operation Scheduling.", *Proc. IEEE* Vol. 75, Dec. 1987, pp. 1574-1591.

[44] Cohen, A.I., Yoshimura, M., "A branch and bound algorithm for unit commitment.", *IEEE Trans. PAS*, Vol. 102, No. 2, Feb. 1983, pp. 444-451.

[45] Cohen, A.I., Wan, S.H., "A method for solving fuel constrained unit commitment problem.", *IEEE Trans. PWRS*, Vol. 2, No. 3, Aug. 1987, pp. 608-614.

[46] Cohoon, J.P., Martin, W.N., Richards, D.S., "A multi-population genetic algorithm for solving the K-partition problem on hyper-cubes." *In R. K. Belew, L. B. Booker (Eds.), Proceedings of the Fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 244-248.

[47] Dahlin, E.B., Shen, D.W.C., "Application of dynamic programming to optimisation of hydroelectric - steam power system operation.", *Proc. IEE*, Vol. 112, No. 12, Dec. 1967, pp. 2255-2260.

[48] Dahlin, E.D., Shen, D.W.C., "Optimal solution to the hydro steam dispatch problem for certain practical systems.", *IEEE Trans. PAS*, Vol. 85, No. 5, 1966 437-458.

[49] Dantzig, G.B., "Linear Programming and Extensions.", *Princeton University Press*, 1963.

[50] Dasgupta, D. and McGregor, D.R., "Thermal unit commitment using genetic algorithms.", *Proc. IEE Pt. C*, Vol. 141, No. 5, Sept. 1994, pp. 459-465.

[51] Davis, L., (Ed.), "A Handbook of Genetic Algorithms,", *New York: Van Nostrand Reinhold*, 1991.

[52] Davis, L., "Adapting operator probabilities in genetic algorithms.", *In J. D. Schaffer (Ed.), Proc. of the Third Int. Conf. on Genetic Algorithms*, 1989, pp. 61-69.

[53] Davis, L., (Ed.), "Genetic Algorithms and Simulated Annealing.", *New York: Van Nostrand Reinhold,* 1987.

[54] Davis, T.E., Principe, J.C., "A Markov chain framework for the simple genetic algorithm.", *Jnl. of Evolutionary Computation,* Vol. 1, No. 3, 1993, pp. 269-288.

[55] Dawkins, R., "The selfish gene.", *Oxford University Press*, 1989.

[56] De Jong, K.A., "Are genetic algorithms function optimisers?.", *In R. Manner, Manderick, B. (Eds.), Parallel Problem Solving in Nature II*, Elsevier: Amsterdam, 1992, pp. 3-13.

[57] De Jong, K.A., "Genetic algorithms: A 10 year perspective.", *In J.J. Grefenstette (Ed.), Proceedings of First Int. Conf. on Genetic Algorithms and Their applications*, 1985, pp. 169-177.

[58] De Jong, K.A., "Adaptive system design: A genetic approach.", *IEEE Trans. on Systems, Man, and Cybernetics, SMC-109*, 1980, pp. 566-574.

[59] De Jong, K.A., "An analysis of the behaviour of a class of genetic adaptive systems.", *Doctoral Dissertation, University of Michigan, Ann Arbor,* Dissertation Abstracts International, 1975.

[60] De Jong, K.A., Spears, W.M., Gordon, D.F., "Using markov chains to analyse genetic algorithms for function optimisation, (GAFOS).", *In Whitley, D., Vose, M., (Eds.), Proc. of Foundations of Genetic Algorithm Workshop III,* 1994, Cambridge: MIT Press.

[61] De Jong, K.A., Spears W.M., "On the state of evolutionary computation.", *In S. Forrest (Ed.), Proc. of the Fifth Int. Conf. on Genetic Algorithms*, San Mateo, CA: Morgan Kauffman, 1993, pp. 618-626.

[62] De Jong, K.A., Spears W.M., "Formal analysis of the role of multi-point crossover in genetic algorithms.", *Annals of Mathematics and Artificial Intelligence*, Vol. 5, No. 1, 1992, pp. 1-26.

[63] De Jong , K.A., Spears, W.M., "An analysis of the interacting roles of population size and crossover in genetic algorithms.", *In H.-P. Schwefel, R. Manner (Eds.), Parallel Problem Solving from Nature*, 1991, pp. 38-47.

[64] De Jong, K.A., Spears, W.M., "Using genetic algorithms to solve NP-complete problems.", *In J.D. Schaffer (Ed.), Proceedings of the Third Int. Conf. on Genetic Algorithms*, 1989, pp. 124-132.

[65] Deb, K., Horn, J., Goldberg, D.E., "Multimodal deceptive functions.", *Complex Systems*, 7 (2), 1993, pp. 131-153.

[66] Deb, K., Goldberg, D.E., "An investigation of niche and species formation in genetic function optimisation.", *In J. D. Schaffer (Ed.), Proc. of the Third Int. Conf. on Genetic Algorithms, 1989*, pp. 42-50.

[67] Dillon, T.S., Edwin, K.W., Kochs, H.-D., Taud, R.J., "Integer programming approach to the problem of optimal unit commitment with probabilistic reserve determination.", *IEEE Trans. PAS*, Vol. 97, No. 6, 1978, pp. 2154-2166.

[68] Ding, H., El-Keib, A.A., Smith, R.E., "Optimal clustering of power system networks using genetic algorithms.", *Electrical Power Systems Research*, 30 (3), 1994, pp. 209-214.

[69] Dommel, H.W., Tinney, W.F. "Optimal power flow solutions.", *IEEE Trans. PAS*, Vol. 87, No. 10, 1968, pp. 1866-1874.

[70] Drake, J.H., Kirchmayer, L.K., Mayall, R.B., Wood, H., "Optimum operation of a hydrothermal power system.", *AIEE Trans. PAS*, Vol. 82, Part III, 1962, pp. 242-250.

[71] Duran, R.A., Puech, C., Diaz, J., "Optimal operation of multi reservoir systems using an aggression-decomposition approach.", *IEEE Trans. PAS*, Vol. 104, No. 8, Aug. 1985, pp. 2086-2092.

[72] El-Hawary, M.E., Christensen, G.S., "Optimal Economic Operation of Electric Power Systems.", *Academic Press, New York*, 1979.

[73] EPRI TR-103697 Technical report, "Optimisation of the unit commitment problem by a coupled gradient network and by a genetic algorithm.", *EPRI*, April 1994.

[74] Eshelman, L.J., Schaffer, J.D., "Real coded genetic algorithms and interval schemata.", *In D. Whitley, (Ed.), Proc. Foundations of Genetic Algorithm Workshop II*, San Mateo, CA: Morgan Kauffman, 1992, pp. 187-202.

[75] Eshelman, L.J., Schaffer, J.D., "Preventing premature convergence in genetic algorithms by preventing incest.", *In R. K. Belew, L. B. Booker (Eds.), Proceedings of the Fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 115-122.

[76] Eshelman, L.J., Caruana, R.A., Schaffer, J.D. "Biases in the crossover landscape.", *In J.D. Schaffer (Ed.), Proc. of the Third Int. Conf. on Genetic Algorithms*, 1989, pp. 10-19.

[77] Fan, J.Y., McDonald, J.D., "A practical approach to real time economic dispatch considering unit's prohibited operating zones.", *IEEE Trans. PWRS*, Vol. 9, No. 4, 1994, pp. 1737-1743.

[78] Fiacco, A.V., McCormick, G.P., "Non Linear Programming: Sequential Unconstrained Minimisation Technique.", *John Wiley and Sons*, 1966.

[79] Finch, J.W., Besmi, M.R., "Genetic algorithms applied to a power system stabiliser.", *Proc. of First IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems : Innovations and applications*, U.K., IEE conference publication No. 414, pp. 100-105.

[80] Fischer, M.L., "Lagrangian relaxation method for solving integer programming problems.", *Management Science*, Vol. 27, 1981, pp. 1-17.

[81] Fogarty, T.C., "Varying the probability of mutation in the genetic algorithm.", *In J. D. Schaffer (Ed.), Proceedings of the Third Int. Conf. on Genetic Algorithms*, 1989, pp. 104-109.

[82] Fogel, L.J., Owens, A.J., Walsh, M.J., "Artificial Intelligence Through Simulated Evolution.", *John Wiley*, 1966.

[83] Fogel, D.B., "Asymptotic convergence properties of genetic algorithms and evolutionary programming.", *Cybernetics and Systems*, 25 (3), 1994, pp. 389-407.

[84] Fonseca, C.M., Fleming, P.J., "Genetic algorithms for multiobjective function optimisation: Formulation, discussion and generalisation.", *In S. Forrest (Ed.), Proceedings of the Fifth Int. Conf. on Genetic Algorithms*, San Mateo, CA: Morgan Kauffman. 1993, pp. 416-423.

[85] Forrest, S., Mitchel, M., "Relative building block fitnes and the building block hypothesis.", *In D. Whitley, (Ed.), Proc. Foundations of Genetic Algorithm Workshop II*, San Mateo, CA: Morgan Kauffman, 1992, pp. 109-126.

[86] Forrest, S., Mitchel, M., "What makes a problem hard for genetic algorithms? Some anomalous results and their explanations.", *Machine Learning* 13, 1992, pp. 285-319.

[87] Garver, L.L., "Power generation scheduling by integer programming - development of a theory.", *IEEE Trans. PAS*, 82, Feb. 1963, pp. 730-735.

[88] Geoffrion, A.M., "Lagrangian relaxation method for integer programming", *Mathematical Programming Study*, Vol. 2, 1974, pp. 82-114.

[89] Glover, F., "Tabu search - part I.", *ORSA Jnl. of Computing*, 1 (3), 1989, pp. 190-206.

[90] Glover, F., "Tabu search - part II.", *ORSA Jnl. of Computing*, 2 (1), 1990, pp. 4-32.

[91] Goldberg, D.E., "A Wright brothers theory for genetic-algorithm flight.", *Jnl. of Inst. of Systems, Control and Information Engineers*, 37, (8), 1993, pp. 450-458.

[92] Goldberg, D.E., "Real coded genetic algorithms, virtual alphabets and blocking.", *Complex Systems*, 5, 1991, pp. 139-167.

[93] Goldberg, D.E., "A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing.", *Complex Systems*, 44, 1990, pp. 445-460.

[94] Goldberg, D.E., "Genetic Algorithms in Search, Optimisation, and Machine Learning.", *Reading, MA: Addison-Wesley*, 1989.

[95] Goldberg, D.E., "Sizing populations for serial and parallel genetic algorithms.", *In J.D. Schaffer (Ed.), Proc. of the Third Int. Conf. on Genetic Algorithms*, 1989, pp. 70-79.

[96] Goldberg, D.E., Deb, K., "A comparative analysis of selection schemes used in genetic algorithms.", *In G. J. E. Rawlins (Ed.), Foundations of Genetic Algorithms*, 1991, pp. 69-93.

[97] Goldberg, D.E., Deb, K., Clark, J.M., "Genetic algorithms, noise and the sizing of populations.", *Complex Systems*, 6, 1992, pp. 333-362.

[98] Goldberg, D.E., Deb, K., Horn, J., "Massive multimodality, deception and genetic algorithms.", *In R. Manner, Manderick, B. (Eds.), Parallel Problem Solving in Nature II*, 1992, pp. 37-46, Elsevier: Amsterdam.

[99] Goldberg, D.E., Lingle, R., "Alleles, loci and the travelling salesman problem.", *In J.J. Grefenstette (Ed.), Proc. of the First Int. Conf. on Genetic Algorithms*, 1985.

[100] Goldberg, D.E., Korb, B., Deb, K., "Messy genetic algorithms: Motivation, analysis, and first results.", *Complex Systems*, 35, 1989, pp. 493-530.

[101] Goldberg, D.E., Milman, K., Tidd, C., "Genetic algorithms: A bibliography.", *IlliGAL report 92008, University of Illinois at Urbana-Champaign*, 1992, available via ftp, **GAL4.GE.UIUC.EDU /pub/papers/IlliGALs/92008.ps.Z.**

[102] Goldberg, D.E., Richardson, J. "Genetic algorithms with sharing for multimodal function optimisation.", *In J.J. Grefenstette (Ed.), Proc. of the Second Int. Conf. on Genetic Algorithms*, 1987, pp. 41-49.

[103] Goldberg, D.E., Rudnick, M., "Genetic algorithms and the variance of fitness.", *Complex Systems*, 53, 1991, pp. 265-278

[104] Goldberg, D.E., and Segrest, P., "Finite markov chain analysis of genetic algorithms.", *In J.J. Grefenstette (Ed.) Proc. of the Second Int. Conf. on Genetic Algorithms*, 1987, pp. 1-8.

[105] Gordon, S., Whitley, D., "Serial and parallel genetic algorithms as function optimisers.", *In Forrest, S., (Ed.), Proceedings of the Fifth Int. Conf. on Genetic Algorithms*, San Mateo, CA: Morgan Kauffman, 1993, pp. 177-183.

[106] Gorges-Schleuter, M., "Explicit parallelism of genetic algorithms having population structures.", *In H.-P. Schwefel and R. Manner, (Eds.) Parallel Problem Solving in Nature*, 1991, pp. 150-159.

[107] Gorges-Schleuter, M., "APARAGOS: An asynchronous parallel genetic algorithm.", *In J. D. Schaffer (Ed.), Proc. of the Third Int. Conf. on Genetic Algorithms*, 1989, pp. 422-427.

[108] Grefenstette, J.J., "Virtual genetic algorithms: First results.", *Technical Report, U.S.A. Navy Centre for applied Research in Artificial Intelligence*, Feb. 1995.

[109] Grefenstette, J.J., "Predictive models of fitnes distributions of genetic operators.", *In Whitley, D., Vose, M., (Eds.), Proc. of Foundations of Genetic Algorithm Workshop III*, 1994, Cambridge, MIT press.

[110] Grefenstette, J.J., "Strategy acquisition with genetic algorithms.", *In L. Davis (Ed.), Handbook of Genetic Algorithms*, 1991, pp. 186-201.

[111] Grefenstette, J.J., "Conditions for implicit parallelism.", *In G. J. E. Rawlins (Ed.), Foundations of Genetic Algorithms*, 1991, pp. 252-264.

[112] Grefenstette, J.J., "Lamarckian learning in multi-agent environments.", *In R. K. Belew, L. B. Booker (Eds.), Proc. of the Fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 303-310.

[113] Grefenstette, J.J., "Incorporating problem specific knowledge into genetic algorithms.", *In L. Davis (Ed.), Genetic Algorithms and Simulated Annealing*, 1987 pp. 420-460.

[114] Grefenstette, J.J., "Optimisation of control parameters for genetic algorithms.", *IEEE Trans. on Systems, Man, and Cybernetics, SMC-161*, 1986, pp. 122-128.

[115] Grefenstette, J.J., Baker, J.E., "How genetic algorithms work: A critical look at implicit parallelism.", *In J. D. Schaffer (Ed.), Proc. of the Third Int. Conf. on Genetic Algorithms*, 1989, pp. 20-27.

[116] Habibollahzadeh, H., Brannud, H., Bubenko, J.A., "Optimal short term planning of hydro-thermal power system.", *8th PSCC, Helsinki*, Aug. 1984, pp. 322-336.

[117] Habibollahzadeh, H., Bubenko, J.A., "Application of decomposition techniques to short term operation planning of hydro-thermal power system", *IEEE Trans. PWRS*, Vol. 1, No. 1, Feb. 1986, pp. 41-47.

[118] Habibollahzadeh, H., Frances, D., Sui, U., "A new generation scheduling program at Ontario", *IEEE Trans. PWRS*, Vol. 5, No. 1, Feb. 1990, pp. 65-73.

[119] Hajmihail, A.T., "Genetic algorithm based design and optimisation of statistical quality control methods.", *Clinical chemistry* 39 9, 1993, pp. 1972-1978.

[120] Hano, I., Tamura, Y., Narita, S., "An application of maximum principle to the most economical operation of power systems.", *IEEE Trans. PAS*, Vol. 85, No. 5, May 1966, pp. 486-494.

[121] Hanqing, Q.Y., Richards, G.G., "Optimum distribution system harmonic filter design using a genetic algorithm.", *Electrical Power Systems Research*, 30 (3), 1994, pp. 263-268.

[122] Happ, H.H., Johnson, R.C., Wright, W.J., "Large scale hydrothermal unit commitment method and results.", *IEEE Trans. PAS*, Vol. 90, 1971, pp. 1373-1383.

[123] Happ, H.H., "Optimal power dispatch: A comprehensive survey.", *IEEE Trans. PAS*, Vol. 96, 1977, pp. 841-854.

[124] Hesser, J., Manner, R., "Towards an optimal mutation probability for genetic algorithms.", *In H.-P. Schwefel, R. Manner (Eds.) , Parallel Problem Solving from Nature*, 1991, pp. 23-32.

[125] Higuchi, T., "Darwin on a chip.", *The Economist,* 326 (7798):85, February 1993.

[126] Himmelblau, D.M., "Applied Non Linear Programming.", *McGraw Hill, New York*, 1972.

[127] Hobbs, W.J., Hermon, G., Warner, S., Sheble, G.B., "An enhanced dynamic programming approach for unit commitment.", *IEEE Trans. PWRS*, Vol. 3, No. 1, Jan. 1988, pp. 1201-1205.

[128] Hoffmeister, F., Back, T., "Genetic algorithms and evolution strategies: Similarities and differences.", *In H.-P. Schwefel, R. Manner (Eds.), Parallel Problem Solving from Natur*e, 1991, pp. 445-469.

[129] Holland, J.H., "Genetic algorithms.", *Scientific American,* 267 1, 1992, pp. 44-50.

[130] Holland, J.H., "Adaptation in Natural and Artificial Systems.", *Ann Arbor: University of Michigan Press*, 1975.

[131] Hooke, R., Jeeves, T.A., "Direct search solution of numerical and statistical problems.", *J. Assoc. Comp. Mach.* , 8, 1961, pp. 212-229.

[132] Hopefield, J, Tank, D., "Neural computation of decisions in optimisation problems.", *Biological Cybernetics,* Vol. 52, 1985, pp. 141-152.

[133] Horn, J., "Finite markov chain analysis of a genetic algorithm with niching.", *In Forrest, S., (Ed.), Proceedings of the Fifth Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kauffman, 1993, pp. 110-117.

[134] Horn, J., Goldberg, D.E., Deb, K., "Implicit niching in learning classifier systems: natures way." , *Jnl. of Evolutionary Computation,* Vol. 2, No. 1, 1994, pp. 37-66.

[135] Hulselmann, M., Muller, H., Wertzlinger, H., "Economic short term scheduling for a run of river plant chain by combined linear programming and genetic optimisation.",

*Proc. of Fourth IEE Int. Conf. on Power System Control and Management (PSCM96), U.K.*, IEE conference publication No. 421, 1996, pp. 42-46.

[136] Iba, K., "Reactive power optimisation by genetic algorithm.", *IEEE Trans. PWRS*, Vol. 9, No. 2, May 1994, pp. 685-692.

[137] IEEE Committee Working Group., "Description and bibliography of major economy-security functions.", *IEEE Trans. PAS*, Vol. 100, No. 1, Jan. 1981, pp. 211-235.

[138] Irving, M.R., Sterling, M.J.H., "Optimum network tearing using simulated annealing.", *Proc. IEE Pt. C*, Vol. 137, No. 1, 1990, pp. 69-72.

[139] Irving, M.R., Sterling, M.J.H., "Economic dispatch of active power by quadratic programming using a sparse linear complementary algorithm.", *Int. Jnl. of Electrical Power and Energy Systems,* Vol. 7, No. 1, Jan. 1985, pp. 1-6.

[140] Irving, M.R., Sterling, M.J.H., "Economic dispatch of active power with constraint relaxation.", *Proc. IEE, Pt. C*, Vol. 130, No. 4, July 1983, pp. 172-177

[141] Janikow, C. Z., Michalewicz, Z., "An experimental comparison of binary and floating point representations in genetic algorithms.", *In R. K. Belew, L. B. Booker (Eds.), Proc. of the Fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 31-36.

[142] Jarmo, T.A., "An indexed bibliography of genetic algorithms in power engineering.", *Art of CAD, Ltd., Vasa, Finland*, 1996, available via ftp, **ftp.uwasa.fi /cs/report94-1/gaPOWERbib.ps.Z.**

[143] Jog, P., Suh, J. Y., Van Gucht, D., "The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the travelling salesman problem.", *In J. D. Schaffer (Ed.), Proc. of the Third Int. Conf. on Genetic Algorithms*, 1989, pp. 110-115.

[144] Karzalis, S.A., Barkitzis, A.G., Petridis, "A genetic algorithm solution of the unit commitment problem.", *IEEE Trans. PWRS, Vol. 11, No. 1*, Feb. 1996, pp. 83-90.

[145] Khodaverdia, E., Bramellar, A., Dunnet, R.M., "Semi rigorous thermal unit commitment for large scale electrical power systems.", *Proc. IEE Pt. C*, Vol. 133, No. 4, May 1986 pp. 157-164.

[146] Kido, T., Kitano, H., Nakanishi, M., "A hybrid search for genetic algorithms: Combining Genetic algorithms, Tabu search and Simulated annealing.", *In Forrest, S., (Ed.), Proceedings of the Fifth Int. Conf. on Genetic Algorithms*, San Mateo, CA: Morgan Kauffman, 1993, pp. 614-619.

[147] Kirchmayer, L.K., "Economic Operation of Power Systems.", *Wiley, New York,* 1958.

[148] Kirkpatrick, S., Gelatt, C.D., Vechi, M.P., "Optimisation by simulated annealing.", *Science*, 220, 1983, pp. 671-680.

[149] Koza, J.R., "Genetic Programming: On the Programming of Computers by Means of Natural Selection.", *Cambridge: MIT,* 1992.

[150] Kuester, J.L., Mize, J.H., "Optimisation Techniques with FORTRAN.", *McGraw-Hill* 1973.

[151] Kuloor, S., Hope, G.S., Malik, O.P., "Environmentally constrained unit commitment.", *Proc. IEE., Pt. C*, Vol. 139, 1992, pp. 122-128.

[152] Langdon, W.B., "Scheduling planned maintenance of the National Grid.", *Evolutionary Computing, AISB Workshop*, Lecture Notes in Computer Science 865, Springer Verlag, 1994.

[153] Lansberry, J.E., Wozniack, P.E., Goldberg, D.E., "Optimal hydro generator governor tuning with a genetic algorithm.", *IEEE Trans. Energy Conversion*, Vol. 7, No. 4, Dec. 1992, pp. 623-630.

[154] Lauer, G.S., Bertsekas, D.P., Sander, Posbergh, T.A., "Solution of large scale optimal unit commitment problems.", *IEEE Trans. PAS*, Vol. 101, No. 1, Jan. 1982, pp. 79-86.

[155] Lee, B.Y., Park, Y.M., Lee, K.Y., "Optimal generation planning for a thermal system with pumped storage based on analytical production costing model.", *IEEE Trans. PWRS*, Vol. 2, No. 2, May 1987, pp. 486-493.

[156] Lee, K.Y., Bai, X., Park, Y.M., "Optimisation method for reactive power planning by using a modified simple genetic algorithm.", *IEEE Trans. PWRS*, Vol. 10, No. 4, Nov. 1995, pp. 1843-1849.

[157] Lee, F.N., "The application of commitment utilisation factor (CUF) to thermal unit commitment.", *IEEE Trans. PWRS*, Vol. 6, No. 2, May 1991, pp. 691-698.

[158] Lee, F.N., "A fuel constrained unit commitment method.", *IEEE Trans. PWRS*, Vol. 4, No. 3, Aug. 1989, pp. 1208-1218.

[159] Lee, F.N., Breiphol, A.M., "Reserve constrained economic dispatch with prohibited operating zones.", *IEEE Trans. PWRS*, Vol. 8, No. 1, 1993, pp. 246-253.

[160] Lee, F.N., Huang, J., Adapa, R., "Multi-area unit commitment via sequential method and a d.c. power flow network model.", *IEEE Trans. PWRS*, Vol. 9, No. 1, Feb. 1994, pp. 279-287.

[161] Levitin, G., Mazal-Tov, S., Elmakis, D., "Optimal sectionalizer allocation in electrical distribution systems by genetic algorithm.", *Electric Power Systems Research*, 31 (2), 1994, pp. 97-102.

[162] Levitin, G., Mazal-Tov, S., Elmakis, D., "A genetic algorithm for open loop distribution system design.", *Electric Power Systems Research*, 32 (2), 1995, pp. 81-87.

[163] Li, S., Shahidehpour, S.H., "Promoting the application of expert systems in short term unit commitment.", *IEEE Trans. PWRS*, Feb. 1993, pp. 287-292.

[164] Li, F., Morgan, R., Song, Y.H., "Handling constrained power dispatch with genetic algorithms.", *Proc. of First IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems : Innovations and applications, 1995, U.K.*, IEE conference publication no 414, pp. 181-187.

[165] Liang, Z.X., Glover, J.D., "A zoom feature for dynamic programming solution to the economic dispatch problem including transmission losses.", *IEEE Trans. PWRS*, Vol. 7, No. 2, 1992, pp. 544-550.

[166] Liang, R.H., Y.Y. Hsu., "Scheduling of hydroelectric generation units using artificial neural networks.", *Proc. IEE, Pt. C*, Vol. 141, No. 5, Sep. 1994, pp. 452-458.

[167] Liang, R.H., Y.Y. Hsu., "Fuzzy linear programming: An application to hydroelectric generation scheduling.", *Proc. IEE, Pt. C*, Vol. 141, No. 6, Nov. 1994, pp. 568-574.

[168] Lidgate, D., Amir, B.H., Wirasinha, R., "An algorithm for the optimal operation of hydro and hydrothermal generation systems.", *Proc. of IEE Int. Conf. On Power System Monitoring and Control*, 1986, pp. 321-326.

[169] Lidgate, D., Khalid, B.M.N., "Unit commitment in a thermal generation system with multiple pumped storage power stations.", *Int. Jnl. of Electrical Power and Energy Systems*, Vol. 6, No. 2, April. 1984, pp. 101-111.

[170] Liu, Z., Villaseca, F.E., Renovich, F., "Neural networks for generation scheduling in power systems.", *Proc. of Int. Joint Conf. on Neural Networks*, Baltimore, MD, 1992, pp. 233-242.

[171] Lowery, P.G., "Generating unit commitment by dynamic programming.", *IEEE Trans. PAS*, Vol. 85, No. 5, May 1966, pp. 422-426.

[172] Luo, G.X., Habibollahzadeh, H, Semylen, A., "Short-term hydrothermal scheduling, detailed model and solutions", *IEEE Trans. PWRS*, Vol. 1, No. 4, Oct. 1989, pp. 1452-1462.

[173] Lyra, C., Tavares, H., Soares, S., "Modelling and optimisation of hydrothermal scheduling", *IEEE Trans. PAS*, Vol. 103, No. 8, Aug. 1984, pp. 2126-2133.

[174] Ma, X., El-Keib, A.A., Smith, R.E., Ma, H., "A genetic algorithm based approach to thermal unit commitment of electric power systems.", *Electric Power Systems Research*, 34, 1995, pp. 29-36.

[175] Mahfoud, S., "Niching Methods for Genetic Algorithms", *Doctoral Dissertation, University of Illinois at Urbana Champaign"*, Dissertation Abstracts International, 1995

[176] Mahfoud, S., "Simple analytical models of genetic algorithms for multimodal function optimisation.", *In S. Forrest (Ed.), Proc. of the Fifth Int. Conf. on Genetic Algorithms*, San Mateo, CA: Morgan Kauffman, 1993, pp. 643.

[177] Mahfoud, S., "Finite markov chain models of an alternative selection strategy for the genetic algorithm.", *Complex Systems,* 7 (2), 1993, pp. 155-170.

[178] Mahfoud, S., "Crowding and preselection revisited.", *In R. Manner and B. Manderick (Eds.), Parallel problem solving from nature 2,* Amsterdam: Elsevier, 1992, pp. 27-36.

[179] Mahfoud, S.W., Goldberg, D.E., "A genetic algorithm for parallel simulated annealing.", *In R. Manner and B. Manderick (Eds.), Parallel problem solving from nature 2,* Amsterdam : Elsevier, 1992, pp. 301-310.

[180] Maifeld, T., Sheble, G. B., "Short term load forecasting by a neural network and a refined genetic algorithm.", *Electric Power Systems Research*, 31 (3), 1994, pp. 147-152.

[181] Merlin, A., Sandrin, P., "A new method of for unit commitment at Electricite de France.", *IEEE Trans. PAS*, Vol. 102, 1983, pp. 1218-1225.

[182] Michalewicz, Z., Janikow, C. Z., "Handling constraints in genetic algorithms.", *In R. K. Belew, L. B. Booker (Eds.), Proc. of the Fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 151-157.

[183] Miranda, V., Ranito, J.V., Proenca, L.M., "Genetic algorithms in optimal multi-stage distribution network planning.", *IEEE Trans. PWRS*, Vol. 9, No. 4, 1994, pp. 1927-1933.

[184] Misra, N., Baghzouz, Y., "Implementation of unit commitment problem on super computers.", *IEEE Trans. PWRS*, Vol. 9, No. 1, 1994, pp. 305-309.

[185] Muckstadt, J.A., Koenig, S.A., "An application of Lagrangian relaxation to scheduling in power generation systems.", *Operations Research*, Vol. 25, No. 3, May/June 1977, pp. 387-403.

[186] Muckstadt, J.A., Wilson, R.C., "An application of mixed integer programming duality to scheduling thermal generating systems.", *IEEE Trans. PAS*, Vol. 87, Dec. 1968, pp. 1968-1977.

[187] Muhlenbein, H., "Parallel genetic algorithms, population genetics, and combinatorial optimisation.", *In J.D. Schaffer, (Ed.) Proc. of third Int. Conf. on Genetic Algorithms*, 1989 , pp. 416-421.

[188] Muhlenbein, H., Schomisch, M., Born, J., "Parallel genetic algorithm as a function optimiser.", *In R.K. Belew, and L.B. Booker, (Eds.) Proc. of fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 271-278.

[189] Muller, H., "Power flow optimisation in electric networks by evolutionary strategic search.", *Proc. of Eighth PSCC*, pp. 401-408.

[190] Muller, H., Petritsch, G., "Genetic programming and simulated annealing for optimisation of unit commitment.", *Proc. of Eleventh PSCC*, pp. 1097-1102.

[191] Musoke, H.S., Biswas, S.K., Ahmed E., Cliff P., Kazibwe W., "Simultaneous solution of unit commitment and dispatch problems using artificial neural networks.", *Int. Jnl. of Electrical Power and Energy Systems,* Vol. 15, No. 3, 1993 pp. 193-199.

[192] Nandwa, J., Bijwe, P.R., Kothari, D.P., "An application of progressive optimality algorithm to optimal hydrothermal scheduling considering deterministic and stochastic data.", *Int. Jnl. of Electrical Power and Energy Systems*, Vol. 8, No. 1, April 1986, pp. 61-64.

[193] Nara, K., Satoh, T., Kitagawa, M., "Distribution systems loss minimum reconfiguration using genetic algorithms.", *IEEE Trans. PWRS*, Vol. 7, No. 3, Aug. 1992, pp. 1044-1057.

[194] Nelder, J.A., Mead, R., "A simplex method for function minimisation.", *Computer Jnl.* 7, 1964, pp. 308-313.

[195] Nicholson, Sterling, M.J.H., "Optimum dispatch of active and reactive generation by quadratic programming.", *IEEE Trans. PAS*, Vol. 92 , 1973, pp. 644-654.

[196] Nix, A.E., Vose, M.D., "Modelling a genetic algorithm with markov chains.", *Annals of Mathematics and Artificial Intelligence,* Vol. 5, No. 1, 1992, pp. 79-88.

[197] Oei, C.K., Goldberg, D.E. and Chang, S.J., "Tournament selection, niching and the preservation of diversity.", *In D. Whitley, (Ed.), Proc. Foundations of Genetic Algorithm Workshop II*, San Mateo, CA: Morgan Kauffman, 1992.

[198] Orero, S.O., and Irving, M.R., "Economic dispatch of generators with prohibited operating zones : A genetic algorithm approach", *To appear in IEE Proc. Part C, Gen., Tran. and Dist.*

[199] Orero, S.O., and Irving, M.R., "A genetic algorithm for network partitioning in power system state estimation.", *To appear in IEE Int. Conf. on Control, Exeter, U.K.*, 1996.

[200] Orero, S.O., and Irving, M.R., "A genetic algorithm for generator scheduling in power systems", *Int. Jnl. of Electric Power and Energy Systems,* Vol. 18, No. 1, 1996, pp. 19-26.

[201] Orero, S.O., Irving, M.R., "Scheduling of generators with a hybrid genetic algorithm.", *Proc. of First IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and applications, U.K.*, IEE conference publication No. 414, 1995, pp. 201-206.

[202] Orero, S.O., "Short term load forecasting with special application to the Kenya power system.", *Master of Science Thesis*, University of Nairobi, Kenya, 1988.

[203] Ouyang, Z., Shahidepour, S.M., "An intelligent dynamic programming for unit commitment application.", *IEEE Trans. PWRS,* Vol. 6, No. 3, Aug. 1991, pp. 1204-1209.

[204] Ouyang, Z., Shahidepour, S.M., "A hybrid artificial neural network-dynamic programming approach to unit commitment.", *IEEE Trans. PWRS*, Vol. 7, No. 1, Feb. 1992, pp. 236-242.

[205] Pang, C.K., Chen, H.C., "Optimal short-term thermal unit commitment.", *IEEE Trans. PAS*, Vol. 95, No. 4, July/ Aug. 1976, pp. 1336-1346.

[206] Pang, C.K., Sheble, G.B, Albuyeh, F., "Evaluation of dynamic programming based methods and multiple area representation for thermal unit commitments.", *IEEE Trans. PAS*, Vol. 100, No. 3, March 1981, pp. 1212-1218.

[207] Papageorgiou, M., "Optimal multi reservoir network control by the discrete maximum principle.", *Water Resources Research*, 21 (2), 1985, pp. 1824-1830.

[208] Pereira, M.V.F., Pinto, L.M.V.G., "application of decomposition techniques to the mid and short term scheduling of hydrothermal systems.", *IEEE Trans. PAS*, Vol. 102, No. 11, Nov. 1983, pp. 3611-3618.

[209] Pereira, M.V.F., Pinto, L.M.V.G. "A decomposition approach to the economic dispatch of the hydrothermal systems.", *IEEE Trans. PAS*, Vol. 101, No. 10, 1982 3851-3860.

[210] Piekutowski, M.R., Litwinowicz, T., Frowd, R.J., " Optimal short term scheduling for a large scale cascaded hydro system.", *Power Industry Computer Applications Conference,* Phoenix, AZ, 1993, pp. 292-298.

[211] Powell, M.J.D., "An efficient method for finding the minimum of a function of several variables without calculating derivatives.", *Computer Jnl.* 7, 1964, pp. 155-164.

[212] Radcliffe, N.J., "The algebra of genetic algorithms.", *Annals of Mathematics Artificial Intelligence*, 10 (4), 1994, pp. 339-384.

[213] Radcliffe, N.J., "The Random respectful recombination.", *Proc. Foundations of Genetic Algorithm Workshop II, In D. Whitley, (Ed.)*, San Mateo, CA: Morgan Kauffman, 1992.

[214] Radcliffe, N.J., "Non linear genetic representations.", *In R. Manner, Manderick, B. (Eds.), Parallel Problem Solving in Nature II*, 1992, pp. 259-268, Elsevier: Amsterdam.

[215] Radcliffe, N.J., "Equivalence class analysis of genetic algorithms.", *Complex Systems*, 5 (2), 1991, pp. 183-206.

[216] Radcliffe, N.J., "Formal analysis and random respectful recombination.", *In R.K. Belew, and L.B. Booker, (Eds.) Proc. of fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 222-229.

[217] Reeves, C.R., "Using genetic algorithms with small populations.". *In Forrest, S., (Ed.), Proceedings of the Fifth Int. Conf. on Genetic Algorithms*, San Mateo, CA: Morgan Kauffman, 1993.

[218] Richards, G.G., Yang, M., "Distribution system harmonic worst case design using a genetic algorithm.", *IEEE Trans. Power Delivery*, Vol. 8, No. 3, July 1993. pp. 1484-1489.

[219] Richardson, J.T, Palmer, M.R., Liepins, G.E., Hilliard, M., "Some guidelines for genetic algorithms with penalty functions.", *In J.D. Schaffer, (Ed.), Proc. of Third Int. Conf. on Genetic Algorithms*, 1989, pp. 191-197.

[220] Robertson, G., "Parallel implementation of genetic algorithms on a classifier system.", *In J.J. Grefenstette (Ed.), Proc. of the Second Int. Conf. on Genetic Algorithms*, 1987.

[221] Ruzic, S., Rajakovic, N., "A new approach for solving extended unit commitment.", *IEEE Trans. PWRS*, Vol. 6, No. 1, Feb. 1991, pp. 269-275.

[222] Saha, T.N., Khapade, S.A., "An application of a direct method for the optimal scheduling of hydrothermal power systems.", *IEEE Trans. on PAS*, Vol. 97, No. 3, 1978 977-985.

[223] Sasaki, H., Watanabe, M., Kubokawa, J., Yorino, N., Yokoyama, R., "A solution method of unit commitment by artificial neural networks.", *IEEE Trans. PWRS*, Vol. 7, 1992, pp. 974-981.

[224] Schaffer, J.D., "Multiple objective optimisation with vector evaluated genetic algorithms.", *In J.J. Grefenstette (Ed.), Proc. of the First Int. Conf. on Genetic Algorithms*, 1985, pp. 93-100.

[225] Schaffer, J.D., Caruana, R.A., Eshelman, L.J. Das, R., "A study of control parameters affecting online performance of genetic algorithms for function optimisation.", *In J.D. Schaffer, (Ed.) Proc. of third Int. Conf. on Genetic Algorithms*, 1989, pp. 51-60.

[226] Muhlenbein, H., Schlierkamp-Voosen, D.,"Predictive models for the breeder genetic algorithm.", *Jnl. of Evolutionary Computation*, Vol. 1, No. 1, 1993, pp. 25-49.

[227] Schwefel, H.-P., "Numerical Optimisation of Computer Models.", *Chichester, Wiley*, 1981.

[228] Shaw, J.J., Bertsekas, D.P., "Optimal scheduling of large hydrothermal power systems.", *IEEE Trans. PAS*, Vol. 104, No. 2, Feb. 1985, pp. 286-294.

[229] Sheble, G.B., "Solution of the unit commitment problem by the method of unit periods.", *IEEE Trans. PWRS*, Vol. 5, No. 1, Feb. 1990, pp. 257-260.

[230] Sheble, G.B., Brittig, K., "Refined genetic algorithm - Economic dispatch example.", *IEEE Trans. PWRS*, Vol. 19, No. 1, Feb. 1994, pp. 128-135.

[231] Sheble G., Fahd, G.N., "Unit commitment literature synopsis.", *IEEE Trans. PWRS*, Vol. 10, No. 1, Feb. 1995, pp. 117-123.

[232] Sheble G., Maifeld T., "Unit commitment by genetic algorithm and expert system.", *Electrical Power System Research* 30 (2) 1994, pp. 115-121.

[233] Shen, C.M., Laughton, M.A., "Determination of optimum power system operating conditions under constraints.", *Proc. IEE*, Vol. 116 (2), 1969, pp. 225-239.

[234] Sherkat, V.R., Campo, R., Moslehi, K., "Stochastic long-term hydrothermal optimisation for a multi reservoir system.", *IEEE Trans. PAS*, Vol. 104, No. 8, Aug. 1985, pp. 2040-2050.

[235] Shoults, R.R., Chang, S.K., Helmick, S., Grady, M.M., "A practical approach to unit commitment, economic dispatch and savings allocation for multiple area pool operation with import/export constraints.", *IEEE Trans. PAS*, Vol. 99, No. 2, Aug. 1980, pp. 625-633.

[236] Siedlecki, W., Sklanky, J., "Constrained genetic optimisation via dynamic reward balancing and its use in pattern recognition.", *In J.D. Schaffer, (Ed.), Proc. of Third Int. Conf. on Genetic Algorithms*, 1989, pp. 141-150.

[237] Sjelvegren, D., Anderson, S., Dillon, T.S., "Optimal operations planning in a large hydro thermal power system.", *IEEE Trans. PAS*, Vol. 102, 1983 3644-3651.

[238] Smith, A., Tate, D., "Genetic optimisation using a penalty function.", *In Forrest, S., (Ed.), Proceedings of the Fifth Int. Conf. on Genetic Algorithms*, San Mateo, CA: Morgan Kauffman, 1993, pp. 499-505.

[239] Snyder, W.L., Powell, H.D., Rayburn, J.C., "Dynamic programming approach to unit commitment.", *IEEE Trans. PAS*, Vol. 2, No. 2, May 1987, pp. 339-350.

[240] Soares, S., Ohishi, T., "Hydro dominated short-term hydrothermal scheduling via a hybrid simulation-optimisation approach: a case study.", *Proc. IEE Pt. C*, Vol. 142, No. 6, Nov. 1995, pp. 569-575.

[241] Soares, S., Lyra, C., Tavayes, W.O., "Optimal generation scheduling of hydrothermal power system.", *IEEE Trans. PAS*, Vol. 99, No. 3, 1980 1106-1114.

[242] Sokkapa, B.G., "Optimum scheduling of hydrothermal systems - A generalised approach.", *IEEE Trans. PAS*, Vol. 82, 1963, pp. 97-104.

[243] Soliman, S.A., Christensen, G.S., "Application of functional analysis to optimisation of variable head multi reservoir power system for long term regulation.", *Water Resources Research,* 22 (6), 1986, pp. 852-858.

[244] Spears, W.M., "Crossover or mutation?.", *In D., Whitley, (Ed.), Proc. Foundations of Genetic Algorithm Workshop II,* San Mateo, CA: Morgan Kauffman, 1992, pp. 221-236.

[245] Spears, W.M., De Jong, K.A., "An analysis of multi-point crossover.", *In G.J.E., Rawlins (Ed.), Foundations of Genetic Algorithms*, 1991, pp. 301-315.

[246] Spears, W.M., De Jong, K.A., "On the virtues of parameterised uniform crossover.", *Proc. Fourth Int. Conf. on genetic algorithms, (Eds.) R. Belew and L. Booker*, San Mateo, CA: Morgan Kauffman, 1991, pp. 230-236

[247] Spears, W.M., De Jong, K.A., Back, T., "An overview of evolutionary computation.", *In P. Bradzil (Ed.), Proc. of European Conf. on Machine Learning*, Lecture notes in AI 667: Berlin: Springer Verlag, 1993, pp. 442-459.

[248] Spiessens, P., Manderick, B., "A massively parallel genetic algorithm: Implementation and first analysis.", *In R.K. Belew, and L.B. Booker, (Eds.) Proc. of fourth Int. Conf. on Genetic Algorithms*, 1991, pp. 279-287.

[249] Starkweather, T, Whitley, D., Mathias, K., "Optimisation using a distributed genetic algorithm.", *In H.-P. Schwefel, R. Manner (Eds.), Parallel Problem Solving form nature*, 1991, pp. 176-185.

[250] Sterling, M.J.H., "Power System Control", *IEE Peter Peregrinus, Ltd. London*. 1978.

[251] Sterling, M.J.H, Irving, M.R., "Optimisation methods for economic dispatch in electric power systems.", *Trans. Institute of Measurements and Control*, Vol. 6, No. 5, Oct. 1984, pp. 247-252.

[252] Su, C.C., Hsu, Y.Y., "Fuzzy dynamic programming: An application to unit commitment.", *IEEE Trans. PWRS*, Vol. 6, No. 3, Aug. 1991, pp. 1231-1237.

[253] Suh, J., Van Gutch, D., "Incorporating heuristic information into genetic search.", *In J.J. Grefenstette (Ed.), Proc. of the Second Int. Conf. on Genetic Algorithms*, 1987, pp. 100-107.

[254] Sundharajavan, S., Pahwa, A., "Optimal selection of capacitors for radial distribution systems using a genetic algorithm.", *IEEE Trans. PWRS*, Vol. 9, No. 3, 1994, pp. 1499-1507.

[255] Suzuki, J., "A markov chain analysis of a genetic algorithm.", *In S. Forrest (Ed.), Proc. of the Fifth Int. Conf. on Genetic Algorithms,* San Mateo, CA: Morgan Kauffman, 1993, pp. 146-153.

[256] Syswerda, G., "Uniform crossover in genetic algorithms.", *J.D. Schaffer, (Ed.) Proc. of third Int. Conf. on Genetic Algorithms*, 1989, pp. 2-9.

[257] Syswerda, G., "Schedule optimisation using genetic algorithms.", *In L. Davis (Ed.), Handbook of Genetic Algorithms*, 1991, pp. 332-349.

[258] Syswerda, G., "A study of reproduction in generational and steady state genetic algorithms.", *In G.J.E., Rawlins (Ed.), Foundations of Genetic Algorithms*, 1991, pp. 94-101.

[259] Tanese, R., "Parallel genetic algorithms for a hyper cube.", *In J.J. Grefenstette (Ed.), Proc. of the Second Int. Conf. on Genetic Algorithms*, 1987, pp. 177-183.

[260] Tanese, R., "Distributed genetic algorithms.", *In D. Schaffer, (Ed.) Proc. of third Int. Conf. on Genetic Algorithms*, 1989, pp. 434-440.

[261] Taylor, A.J.E., Irving, M.R., Sterling, M.J.H., "Power system partitioning using genetic optimisation.", *Universities Power Engineering Conference,* U.K., *1990*, pp. 607-610.

[262] Tong, S.K., Shahidehpour, S.M., "Combination of Lagrangian relaxation and linear programming approaches for fuel-constrained unit commitment problems.", *Proc. IEE, Pt. C*, Vol. 136, No. 3, May 1989, pp. 162-174.

[263] Tong, S.K., Shahidehpour, S.M., "Hydrothermal unit commitment with probabilistic constraints.", *IEEE Trans. PWRS*, Vol. 5, No. 1, Feb. 1990, pp. 276-282.

[264] Tong, S.K., Shahidehpour, S.M., Ouyang, Z. "A heuristic short term unit commitment.", *IEEE Trans. PWRS*, Vol. 6, No. 3, April, 1991, pp. 1210-1216.

[265] Turgeon, A., "Optimal scheduling of thermal generating units.", *IEEE Trans. AC*, Vol. 23, No. 6, Dec. 1978, pp. 1000-1005.

[266] Turgeon, A., "Optimal short-term hydro-scheduling from the principle of progressive optimality.", *Water Resources Research*, Vol. 17, No. 3, June 1981, pp. 481-486.

[267] Turgeon, A., "Optimal operation of multi reservoir power systems with stochastic inflows." *Water Resources Research*, Vol. 16, No. 2, April 1980, pp. 275-283.

[268] Vaan Laarhoven, P.J.M., Aarts, E.H.L., "Simulated Annealing: Theory and Applications.", *D. Reidel,* 1989.

[269] Van Den Bosch, P.P.J., Honderd, G., "A solution of the unit commitment problem via decomposition and dynamic programming.", *IEEE Trans. PAS*, Vol. 104, No. 7, July 1985, pp. 1684-1690.

[270] Van Meeteren, H.P., "Scheduling of generation and allocation of fuel using dynamic and linear programming.", *IEEE Trans. PAS*, Vol. 103, No. 7, July 1984, pp. 1562-1568.

[271] Virmani, S., Anderian, E.C., Imhof, K., Mukherjee, S., "Implementation of a Lagrangian relaxation based unit commitment.", *IEEE Trans. PWRS*, Vol. 4, No. 4, Oct. 1989, pp. 1373-1380.

[272] Vose, M.D. "A closer look at mutation in genetic algorithms.", *Annals of Mathematics and Artificial Intelligence*, 10 (4), 1994, pp. 423-434.

[273] Vose, M.D., "Modelling simple genetic algorithms.", *Foundations of Genetic Algorithm Foundations of Genetic Algorithm Workshop workshop, In D., Whitley, (Ed.),* San Mateo, CA: Morgan Kauffman, 1992, pp. 63-74.

[274] Vose, M.D., "Generalising the notion of schema in genetic algorithms.", *Annals of Mathematics and Artificial Intelligence,* 50 (3), 1991, pp. 385-396.

[275] Juliany, J., Vose, M.D., "The genetic algorithm fractal.", *Jnl. of Evolutionary Computation,* Vol. .2, No. 2, 1994, pp. 165-180.

[276] Vose, M.D., Liepins, G.E., "Punctuated equilibria in genetic search.", *Complex Systems*, 5 (1), 1991, 31-34.

[277] Wakamori, F., Masui, S., Morita, K., "Layered network model approach to optimal daily hydro scheduling.", *IEEE Trans. PAS*, Vol. 101, No. 9, Sep. 1982, pp. 3310-3314.

[278] Walters, D.C., Sheble, G. B., "A genetic algorithm solution to the economic dispatch with valve point loading.", *IEEE Trans. PWRS*, Vol. 8, no 3, 1993, 1325-1332.

[279] Wang, C., Shahidehpour, S.H., "Effects of ramp_rate limits on unit commitment and economic dispatch.", *IEEE Trans. PWRS*, Vol. 8, No. 3, Aug. 1993, pp. 1341-1350.

[280] Wang, C., Shahidehpour, S.M., "Power generation scheduling for multi-area hydrothermal power systems with tie-line constraints, cascaded reservoirs and uncertain data.", *IEEE Trans. PWRS*, Vol. 8, No. 3, 1993, pp. 1333-1340.

[281] Wen, F., Han, Z., "Fault section estimation in power systems by a genetic algorithm.", *Electric Power Systems Research*, 34, 1995, pp. 165-172.

[282] Whitley, D., "A genetic algorithm tutorial.", *Statistics and Computing, U.K.*, 4 (2), 1994, pp. 65-85.

[283] Whitley, D., "An executable model of a genetic algorithm.", *In D. Whitley, (Ed.), Foundations of Genetic Algorithm Workshop,* San Mateo, CA: Morgan Kauffman, 1992, pp. 45-62.

[284] Whitley, D., "The Genitor algorithm and selective pressure: Why ranked based allocation of reproductive trials is best.", *In D. Schaffer, (Ed.) Proc. of third Int. Conf. on Genetic Algorithms*, 1989, pp. 116-123.

[285] Wirbel, L., "Compression chip is first to use genetic algorithms.", *Electronic Engineering Times,* December 1992, pp. 17.

[286] Wong, K.P., Wong, Y.N., "Short term hydrothermal scheduling. Part I: Simulated annealing approach. ", *Proc. IEE., Pt. C*, Vol. 141, No. 5, 1994, pp. 497-501.

[287] Wong, K.P., Wong, Y.N., "Short term hydrothermal scheduling. Part I: Parallel simulated annealing approach. ", *Proc. IEE., Pt. C*, Vol. 141, No. 5, 1994, pp. 502-507.

[288] Wong, K.P., Cheung, H.N., "Thermal generator scheduling algorithm using hybrid genetic / simulated annealing approach.", *Proc. IEE, Pt. C*, Vol. 142, No. 4, July 1995, pp. 372-380.

[289] Wong, K.P., Wong, Y.W., "Genetic and genetic / simulated annealing approaches to economic dispatch.", *Proc. IEE, Pt. C*, Vol. 141, No. 5, Sep. 1994, pp. 507-513.

[290] Wood, A.J., Wollenberg, B.F., "Power Operation, Generation and Control.", *John Wiley and Sons*, 1996.

[291] Wright, A., "Genetic algorithms for real parameter optimisation.", *In G.J.E., Rawlins (Ed.), Foundations of Genetic Algorithms*, 1991, pp. 205-220.

[292] Wu, Q.H., Ma, J.T., "Power system optimal reactive power dispatch using evolutionary programming.", *IEEE Trans. PWRS*, Vol. 10, No. 3, Aug. 1995, pp. 1243-1249.

[293] Xia, Q., Xiang, N., Wang, S., Zhang, B., Huang, M., "Optimal daily scheduling of cascaded plants using a new algorithm of non-linear minimum cost network flow concept.", *IEEE Trans. PWRS, Vol. 3, No. 3*, 1988, pp. 929-935.

[294] Xiaofang, Q., Palmieri, F., "Theoretical analysis of evolutionary algorithms with infinite population size in continuous space, part I: Basic properties.", *IEEE Trans. on Neural Networks*, Special edition on evolutionary computation, 5 (1), Jan. 1994, pp. 102-119.

[295] Xiaofang, Q., Palmieri, F., "Theoretical analysis of evolutionary algorithms with infinite population size in continuous space, part II: Analysis of the diversification role of crossover.", *IEEE Trans. on Neural Networks*, Special edition on evolutionary computation, 5 (1), Jan. 1994, pp. 120-129.

[296] Yang, H.T., Huang, C.M., Huang, C.L., "Identification of ARMAX model for short term load forecasting: an evolutionary approach.", *IEEE Trans. PWRS, Vol. 11, No. 1*, Feb. 1996, pp. 403-408.

[297] Yang, H.T., Yang, P.C., , Huang, C.L., "Evolutionary programming based economic dispatch for units with non smooth fuel cost functions.", *IEEE Trans. PWRS, Vol. 11, No. 1*, Feb. 1996, pp. 112-118.

[298] Yang, J.S., Chen, N.M., "Short term hydrothermal co-ordination using multi-pass dynamic programming", *IEEE Trans. PWRS, Vol. 4, No. 3*, Aug. 1989, pp. 1050-1056.

[299] Yeh, E.C., Venkata, S.S., Sumic, Z., "Improved distribution system planning using computational evolution.", *Proc. of IEEE PICA Conf.*, 1995, pp. 530-536.

[300] Yin, X., Germay, N., "Investigations on solving load flow problem by genetic algorithms.", *Electrical Power Systems Research*, Vol. 22 (3), 1991, pp. 151-163.

[301] Zhuang, F., Galiana F.D., "Towards a more vigorous and practical unit commitment by Lagrangian relaxation.", *IEEE Trans. PWRS*, Vol. 3, May 1988 pp. 763-770.

[302] Zhuang, F., Galiana, F., "Unit commitment by simulated annealing.", *IEEE Trans. PWRS*, Vol. 5, No. 1, Feb. 1990 pp. 311-317.