CRANFIELD UNIVERSITY

John M. Oliver

Multi-Objective Optimisation Methods Applied to Complex Engineering Systems

SCHOOL OF ENGINEERING

PhD

Academic Year: 2013 - 2014

Supervisor: Professor A. Mark Savill

Co-supervisor: Dr Timoleon Kipouros

September 2014

CRANFIELD UNIVERSITY

SCHOOL OF ENGINEERING

PhD

Academic Year: 2013 - 2014

John M. Oliver

Multi-Objective Optimisation Methods Applied to Complex Engineering Systems

Supervisor: Professor A. Mark Savill

Co-supervisor: Dr Timoleon Kipouros

September 2014

This thesis is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Declaration of Authorship

I, John M. Oliver, declare that this thesis titled, 'Multi-Objective Optimisation Methods Applied to Complex Engineering Systems' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____________________

Date: ____________________

"Nothing at all takes place in the universe in which some rule of maximum or minimum does not appear."

Leonhard Euler

"It always takes longer than you expect, even when you take into account Hofstadter's Law."

Douglas Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid*, 1979

"What limit can be put to this power, acting during long ages and rigidly scrutinising the whole constitution, structure, and habits of each creature, favouring the good and rejecting the bad? I can see no limit to this power, in slowly and beautifully adapting each form to the most complex relations of life."

Charles Darwin, *On the Origin of Species by Means of Natural Selection*, 1859

Abstract

This research proposes, implements and analyses a novel framework for multi-objective optimisation through evolutionary computing aimed at, but not restricted to, real-world problems in the engineering design domain.

Evolutionary algorithms have been used to tackle a variety of non-linear multi-objective optimisation problems successfully, but their success is governed by key parameters which have been shown to be sensitive to the nature of the particular problem, incorporating concerns such as the number of objectives and variables, and the size and topology of the search space, making it hard to determine the best settings in advance. This work describes a real-encoded multi-objective optimising evolutionary algorithm framework, incorporating a genetic algorithm, that uses self-adaptive mutation and crossover in an attempt to avoid such problems, and which has been benchmarked against both standard optimisation test problems in the literature and a real-world airfoil optimisation case.

For this last case, the minimisation of drag and maximisation of lift coefficients of a well documented standard airfoil, the framework is integrated with a free-form deformation tool to manage the changes to the section geometry, and XFoil, a tool which evaluates the airfoil in terms of its aerodynamic efficiency. The performance of the framework on this problem is compared with those of two other heuristic MOO algorithms known to perform well, the Multi-Objective Tabu Search (MOTS) and NSGA-II, showing that this framework achieves better or at least no worse convergence.

The framework of this research is then considered as a candidate for smart (electricity) grid optimisation. Power networks can be improved in both technical and economical terms by the inclusion of distributed generation which may include renewable energy sources. The essential problem in national power networks is that of power flow and in particular, optimal power flow calculations of alternating (or possibly, direct) current. The aims of this work are to propose and investigate a method to assist in the determination of the composition of optimal or high-performing power networks in terms of the type, number and location of the distributed generators, and to analyse the multi-dimensional results of the evolutionary computation component in order to reveal relationships between the network design vector elements and to identify possible further methods of improving models in future work. The results indicate that the method used is a feasible one for the achievement of these goals, and also for determining optimal flow capacities of transmission lines connecting the bus bars in the network.

Keywords

Evolutionary, Algorithm, Self-Adaptive, Framework, Electrical Power, Plexos, Power Flow, Network, Grid, MOOEA, Multi-Objective, Optimization, MOO, MOOP, Airfoil

Acknowledgements

This research was carried out at the School of Engineering of Cranfield University and was enabled and supported by funding from the Engineering and Physical Sciences Research Council (EPSRC), for which I am glad to have this opportunity to express my gratitude.

I acknowledge gratefully and give my wholehearted thanks to:

My PhD supervisor, Professor Mark Savill, for the opportunity to undertake this work, his support, and his invaluable guidance.

My co-supervisor, Dr Timoleon Kipouros, for his insights, suggestions and liaising with colleagues in other institutions.

My fellow PhD students past and present with whom I was lucky to share an office, for discussions, their support and friendship, and other colleagues in Professor Savill's group, for the same reasons.

My wife, for support, love and tea.

Contents

List of Figures

List of Tables

Glossary

ACO	Ant Colony Optimisation.
C_D	Coefficient of Drag.
CAD	Computer-aided design.
CAE	Computer-aided Engineering.
CFD	Computational Fluid Dynamics.
C_L	Coefficient of Lift.
C_P	Coefficient of Pressure.
DE	Differential Evolution.
DG	Distributed Generation or Generators; also describes the variables (gene alleles) that hold the number of units of generators of each type.
DoE	Design of Experiments.
EA	Evolutionary Algorithm.
EC	Evolutionary Computing.
EP	Evolutionary Programming.
ES	Evolutionary Strategies.
FE	Function Evaluation.
FEM	Finite Element Methods.
GA	Genetic Algorithm.
GUI	Graphical User Interface.
ISO	Independent System Operator.
JRE	Java run-time environment.
JVM	Java Virtual Machine.

LC	Line Capacities - The Maximum Flow ratings, in MW, of transmission lines; also describes the variables (gene alleles) that hold the values.
LMP	Locational Marginal Pricing.
MaOOP	Many objective optimisation.
MOO	Multi-Objective Optimisation.
MOOEA	Multi-Objective Optimising Evolutionary Algorithm.
MOGA	Multi-Objective Genetic Algorithm.
MOOP	Multi-Objective Optimisation Problem.
MOTS	Multi-Objective Tabu Search.
NFL	No Free Lunch - theorem.
OCGT	Open-Cycle Gas Turbine.
OF	Objective Function.
OPF	Optimal Power Flow.
PFapprox	Approximate Pareto front - not the globally optimal front.
PFopt	Pareto-optimal front - the globally optimal front.
PSC	Pseudo-code.
PSO	Particle Swarm Optimisation.
PT	Polynomial Time.
PV	Photovoltaic, of solar power generation.
RDO	Robust Design Optimisation.
SA	Simulated Annealing.
SAMPSC	Self-Adaptive Multi-point Swap Crossover.
SAUBC	Self-Adaptive Uniform Blend Crossover.
SBX	Simulated Binary Crossover.
Plugin	A software component that adds a specific feature to an existing software application.
UCP	Unit Commitment Problem.

Physical Constants

Speed of light	c	$=$	$2.997\ 924\ 58 \times 10^{8}$ m$^{-\mathrm{S}}$ (exact)
Elementary positive charge	e	$=$	$1.602176565(35) \times 10^{-19}$ C (approx.)

Symbols

P	power	MW (Js^{-1} x 10^6)
v	voltage	kV (V x 10^3)
I	current	kA (A x 10^3)

Dedicated to my parents

to my mother, for the sacrifices she made for me

to my father, for the love of books

Chapter 1

Introduction

This chapter is intended to address the following:

- To describe the structure of the thesis.
- To introduce the research of the thesis.
- To provide the motivation for the research.
- To highlight the novelties of the work.

1.1 Thesis organisation

The thesis is organised as set out below:

Chapter 1, here, sets out to describe the organisation of the thesis and to briefly introduce the work.

Chapter 2 provides a literature review of the subject areas encompassed in the research, considering the background and recent literature and identifying areas for the new research presented herein to be worthwhile and commensurate with the thesis objectives.

Chapter 3 sets out the value and description of the framework and multi-objective optimising evolutionary algorithm (MOOEA) produced in the course of this research, and reports on the benchmarks used and their outcomes.

Chapter 4 describes the optimisation of an airfoil using the framework and MOOEA of this work and an analysis and exploration of the results obtained.

Chapter 5 describes the optimisation of a model electrical power grid using the framework and MOOEA of this work, and analyses the results obtained.

Chapter 6 considers the work as a whole, describing its limitations, offering conclusions, and the possibilities of further work.

1.2 Background and motivation

Engineering design is a discipline that, through its applicability to so many areas (Rao, 1996) of our societies that are built upon technology, has a major impact upon contemporary living.

The increasing complexity of engineering designs, especially power and propulsion systems, require the use of advanced computer modelling techniques to represent such systems in an accessible manner, and the deployment of multi-dimensional analysis techniques to visualise their complex interdependencies in the most easily understandable manner.

Bloebaum and McGowan (2010) describe complex engineering systems as those systems in which the "... tightly coupled interacting phenomena yield a collective behavior [sic] that cannot be derived by the simple summation of the behavior [sic] of the parts", where such systems tend to exhibit the following properties:

- Have very many parameters defining the design
- Have a challenging design space - constraints, discontinuities, non-linear
- Have emergent behaviour - that behaviour which is not designed in

- Challenge *a priori* preferences
- Require trade-off compromises between goals of the system

The availability of relatively low cost computing hardware and software means that there is a greater possibility of improving engineering designs through the exploitation of optimisation in wider subject domains than in previous decades (Forrester *et al.*, 2008). The targeting of multi-objective optimisation techniques at industrial engineering (Zalzala and Fleming, 1997) has been shown to be tractable and successful and they have emerged, together with the techniques mentioned above, as key tools for establishing the appropriate degrees of complexity with which to replace previous simpler descriptions and solutions of designs and models. This is particularly true with regards to establishing design trade-offs, performing complex risk analysis, and ensuring the requisite variety to avoid and manage emerging tipping points.

Evolutionary computing (EC) techniques have been used to tackle a variety of non-linear (Nicolis, 1995) multi-objective optimisation problems successfully (Haupt and Haupt, 2004, p. 174), and have been specifically applied to real-world complex engineering system optimisation problems (Gen and Cheng, 2000), but their success is governed by key parameters which have been shown to be sensitive to the nature of the particular problem, incorporating concerns such as the number of objectives and variables, and the size and topology of the search space, making it hard to determine the best settings in advance. The works of this thesis attempts to address this issue.

Improving electrical power grids, as an engineering design problem and as a matter of concern in many technological societies due to the relative fragility fo such networks, (Amin, 2003) taken together with the diminution of fossil fuel supplies and the need to cut pollution emissions, particularly of greenhouse gases, has become a matter of high research interest (Rylatt *et al.*, 2013). This work seeks to add an optimisation based-approach as a facilitator of design in the inception of (smart) power grids.

1.3 Thesis aims and objectives

This work therefore set out to create an evolutionary algorithm framework that is able to work on real-world engineering design problems, having the above characteristics, with the aim of being able to self-adapt in order to at least partially obviate the problem of determining the best parameter settings in advance. While work has been done in this area, commencing with Evolutionary Strategies (ES) by Schwefel and Genetic Algorithms (GA), as described by Bäck (1992), this work creates both a novel framework advantageous to real-world situations, and an evolutionary algorithm (EA) with novelty in its self-adaptability, and certain other aspects, as detailed in Chapter 3. The EA is a multi-objective optimising EA (MOOEA) that is not limited by its construction in being able to work on any number of objective function dimensions, while the framework incorporates the MOOEA and provides means of applying it to new optimisation problems in a manner that enables the framework to be independent of the particular problem, as well as providing means of overriding certain default behaviours. However, although not architecturally limited in the number of objective functions (OFs) it can handle, in practice it has been found, as set out further in Chapter 2, that for a general EA, the more OFs are present, especially above three, the more likely the algorithm will find it hard to converge to an optimal set of solutions.

The framework of this research is then considered as a candidate for (smart) electrical power grid optimisation. Power networks can be improved in both technical and economical terms by the inclusion of distributed generation which may include renewable energy sources. This work therefore sets out to propose and investigate a method to assist in the determination of the composition of optimal or high-performing power networks in terms of the type, number and location of the distributed generators, and to analyse the multi-dimensional results of the evolutionary computation component in order to reveal relationships between the network design vector elements and to identify possible further methods of improving models in future work.

In order to achieve the aims set out above, the evolutionary algorithm framework produced is benchmarked with standard tests from the literature (Zitzler *et al.*, 2000), to establish both correct functionality and its ability to converge with acceptable or better performance. A complex real-world engineering problem is then to act as a further more exacting test of its capabilities, for which an airfoil optimisation case is used. The case chosen had been used in other studies of a similar nature, for which results of other algorithms were available to also act as benchmarks.

The airfoil optimisation case concerned the minimisation of drag and maximisation of lift coefficients of a well documented standard airfoil. The framework is integrated with a free-form deformation tool to manage the changes to the section geometry, and XFoil, a tool which evaluates the airfoil in terms of its aerodynamic efficiency. The performance of the framework/EA on this problem is compared with those of two other heuristic MOO algorithms known to perform well (Kipouros *et al.*, 2012), the Multi-Objective Tabu Search (MOTS) and NSGA-II.

1.4 Publications

The following peer-reviewed papers were produced and published or accepted for publication in the course of this research:

1. *A Self-adaptive Genetic Algorithm Applied to Multi-Objective Optimization of an Airfoil*, (Oliver *et al.*, 2013), presented by the author at the Evolve 2013 [1] conference in Leiden, NL, published in the book of the proceedings.

2. *An Evolutionary Computing-based Approach to Electrical Power Network Configuration*, (Oliver *et al.*, June 2015), presented by the author at an ECCS'13 [2] conference satellite workshop, in Barcelona; published in the journal *Emergence: Complexity and Organization.*

[1]EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation

[2]European Conference on Complex Systems : Integrated Utility Services IUS'13 workshop

3. *Electrical Power Grid Network Optimisation by Evolutionary Computing*, (Oliver *et al.*, 2014), presented on the author's behalf at the ICCS'14 [3] conference in Cairns, AU, published in a special issue of the journal *Procedia Computer Science*.

4. *Multi-Objective Optimization by Self-Adaptive Evolutionary Algorithm*, (Oliver *et al.*, expected 2016), an invited chapter in the book: *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation 5, Series: Studies in Computational Intelligence*, publication expected 2016.

1.5 Software Produced

There were two main software items produced in the course of this PhD, these being:

1. The self-adaptive multi-objective optimising evolutionary algorithm framework (named Ganesh, for brevity).

2. The set of plugins[4] comprising the various optimisation problem definitions used in this work, which are available to be dynamically loaded and executed by Ganesh.

A number of utilities were also produced, to facilitate data manipulation and plotting, and these are detailed in Appendix B.

All these codes were produced with the intention that they be open source and are made available with an open-source licence, currently at: `http://www.logiprime.com/ganesh/`.

[3]The International Conference on Computational Science

[4]Although correctly spelled 'plug-in', it is commonly referred to as 'plugin' in technical contexts (see Glossary).

Chapter 2

Heuristic multi-objective optimisation algorithms

2.1 Introduction

This chapter provides a literature review of the general subject areas germane to this research, namely optimisation, heuristics, evolutionary computing, and related topics. In doing so, the context of the motivation for the aims of thesis are also made apparent.

Specific optimisation subject domains, namely airfoils and electrical power, are dealt with in those specific chapters.

2.2 Features of real-world optimisation

Optimisation is applicable to all areas of engineering which is inherently an area of real-world interaction, consequently engineers and researchers in all areas need to be aware of the possibilities in both theory and practice of optimisation (Rao, 1996). The engineering discipline sectors utilising optimisation include: structural, thermal systems, chemical and metallurgical, electronics and electrical, mechanical, Aerodynamics, Combustion, propulsion, control, power and general engineering design.

Rao (1996) notes that one way of classifying optimisation problems is by the nature of the decision vector elements, where *static* problems are those dependent upon some function of the decision vector, whereas *dynamic* problems are those in which each element of the design vector is a function of one or more other non-decision problem parameters. He gives the following characteristics in his assessment of real-world engineering design problems:

- The design is in some way amenable to optimisation.
- At least one of the objective functions of the optimisation is non-linear.
- There are constraints that need to be adhered to.
- At least some components of the design vector require real numbers.
- The objective function outputs real numbers.

Gen and Cheng (2000) add that real-world complex engineering system problems tend to be optimisation problems with complex constraints, and that such problems are often multi-dimensional in both decision space and objective space.

Engineering may include unusual real-life matters that can be hard to model such as health and safety factors, aesthetics and other concerns, or may be computationally complex and could be classed as NP-complete, meaning they could be both in NP and NP-hard problem classes (Michalewicz *et al.*, 1999).

Computational complexity (Garey and Johnson, 1979) is concerned with the inherent difficulty of problems, giving P as those decision problems[1] that can be solved in polynomial time (PT) deterministically (by a deterministic Turing machine), which implies their time complexity function is given by $O(p(n))$ where p is some polynomial function and n denotes the size of the input. Problems not in

[1]decision problems have solutions: *true* or *false*

P are only solvable in exponential time, given by $O(c^n)$ (where c is some constant), which means their solution time increases exponentially as n increases. P type problems may be solvable through deep insight, while those not in P require some kind of exhaustive search approach. Problems not in P are considered intractable for other than small n, with some exceptions; exceptions arise where the worst-case does not occur generally, such as for the Simplex linear programming algorithm. The class of NP problems is then defined as those decision problems that can be solved by PT nondeterministic algorithms[2], which implies that their solution when = true is verifiable in deterministic PT. NP-hard problems are those 'at least as hard as the hardest problems in NP', implying that they might not be solvable in PT. Figure 2.1 illustrates these set relationships, with the split between left and right being the unsolved question '$P \neq NP$?' which essentially means whether every problem whose solution can be verified in PT can be solved in PT. If $P \neq NP$, then there are NP problems (such as the NP-complete ones) that might not be solvable in PT, but whose solution can be verified in PT.

A decision problem can be derived from an optimisation problem (Garey and Johnson, 1979), whether it is maximising or minimising, by re-framing the problem as a bounded one and asking whether a solution exists within that bound B, for example in the Travelling Salesman problem, is there a tour having length $\leq B$, or in a maximisation, is there a solution with utility $\geq B$. If the objective function can be evaluated 'quickly' (in PT at the most), then the decision problem is no harder than the original problem and can be possibly solved by a PT nondeterministic algorithm.

Thus real-world engineering problems often deal with NP-complete problems, and require algorithms with characteristics that give them a chance in tackling them pragmatically.

[2]nondeterministic algorithms have two parts: a stochastic solver and a deterministic verifier

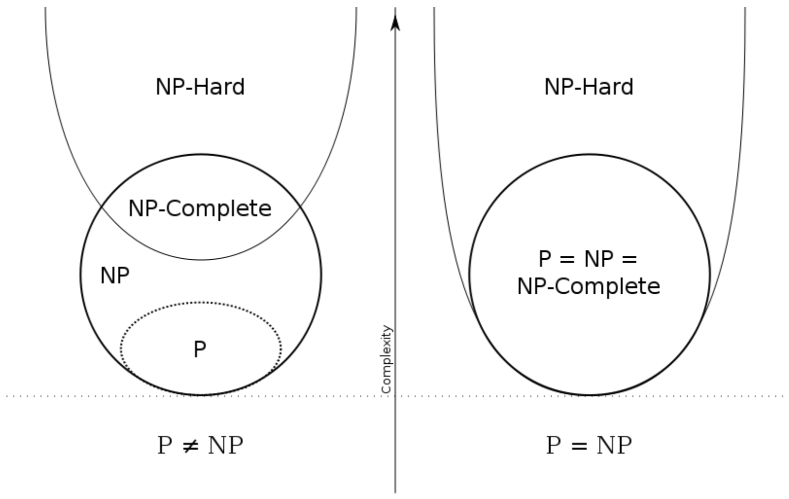


Figure 2.1: Euler diagram for P, NP, NP-complete, and NP-hard set of problems, where left side shows $P \neq NP$ and right side $P = NP$.

2.3 Optimisation

2.3.1 Optimisation overview

Stated informally, optimisation is the process by which the minimum or maximum of a proposed function is found, and also the conditions, which means the values of the decision variables that the function takes as inputs, under which the result is obtained (Rao, 1996). The result of the function, which is termed the *objective function* (OF), thus represents the best outcome. When the optimisation seeks a minimum, the OF is often termed the *cost* function as one generally seeks to lower costs. Conversely, when seeking a maximum value, the OF is often termed the *utility* or *fitness* function, since these are naturally seen as 'good' and hence desirable.

However, there are essential limitations imposed upon candidate solutions that may otherwise be considered optimal; a solution must be both *feasible* and *legal* (Gen and Cheng, 2000). A *legal* solution is one that can successfully be translated from the internal representation of the optimiser into the problem domain, whereas a *feasible* one is legal and resides in the feasible solution region of the domain, as in Figure 2.2. For example, a solution could represent a theoretically possible wing and hence be *legal*, but not generate sufficient lift for it to be *feasible* in the real world.

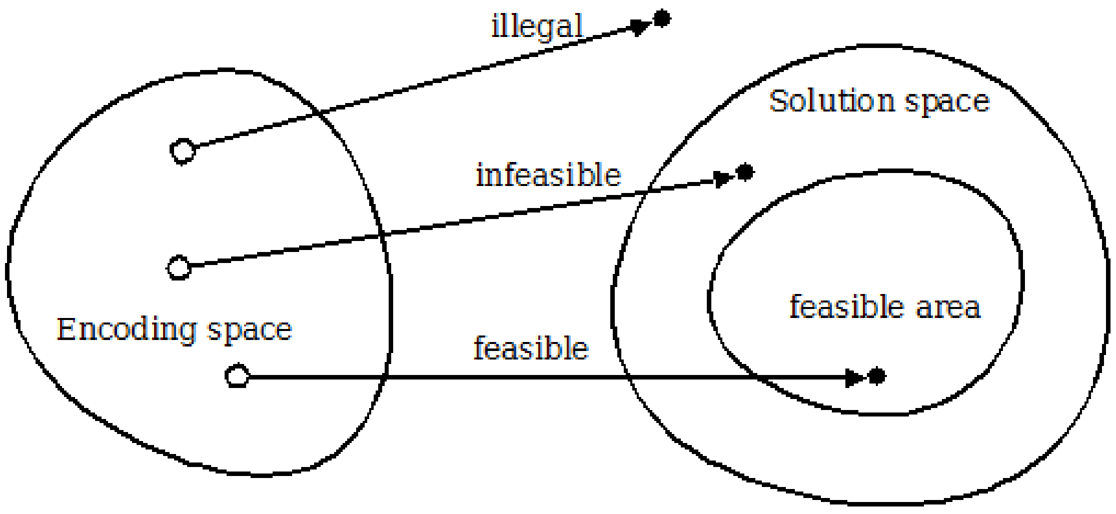


FIGURE 2.2: Coding & solution spaces showing illegal and infeasible regions.

Without loss of generality, optimisation can be taken to be a minimisation, since by the principle of duality (Rao, 1996), a minimum can be transformed into a maximum, and vice-versa, by multiplying the result by -1, as the maximum of a function is equal to the minimum of the negative of that function. Henceforth unless specifically declared otherwise, minimisation will be assumed, as it indeed is in the internal mechanisms of the produced MOOEA (see Chapter 3).

A formal statement of optimisation can therefore be made as follows (Bäck *et al.*, 1997):

$$\begin{aligned} &\vec{x} \in M \\ &f : M \to \mathbb{R} \\ &f(\vec{x}) \to \min. \end{aligned} \tag{2.1}$$

in which $\vec{x}$ is a vector of M parameters for the system, such that the objective function $f(\vec{x})$ is minimised. To find the global optimum (2.1) it is necessary to:

$$\begin{aligned} &\text{find the vector } \vec{x}^* \\ &\text{such that } \forall \vec{x} \in M : f(\vec{x}) \geq \vec{x}^* = f^*. \end{aligned} \tag{2.2}$$

Optimisation problems occur in two broad classes: *continuous* and *discrete*, in which the former encompasses problems where M is infinite and $M = \mathbb{R}^n$ or where M is defined by equations and inequalities, and where the latter is defined by M being in some way finite or in which points are isolated from each other, examples of which are combinatorial and integer problems (Bertsekas, 1999).

Non-linear problems are of the continuous class, and can be thought of in two different ways, from a mathematics (a) or physics (b) viewpoint: (a) equations in which the unknowns are variables of a polynomial of degree two or more, or in the argument of a function which is not a polynomial of degree one; (b) The superposition principle does not hold, so have output not directly proportional to the input (Nicolis, 1995).

Having defined the process of global optimisation, in which the best solution from the entire search space is sought, it is then necessary to note that there may be regions within the solution space that have small values locally, the *local* minima, that may deceive an optimiser into taking one of those as the global minimum, erroneously. Figure 2.3 illustrates.

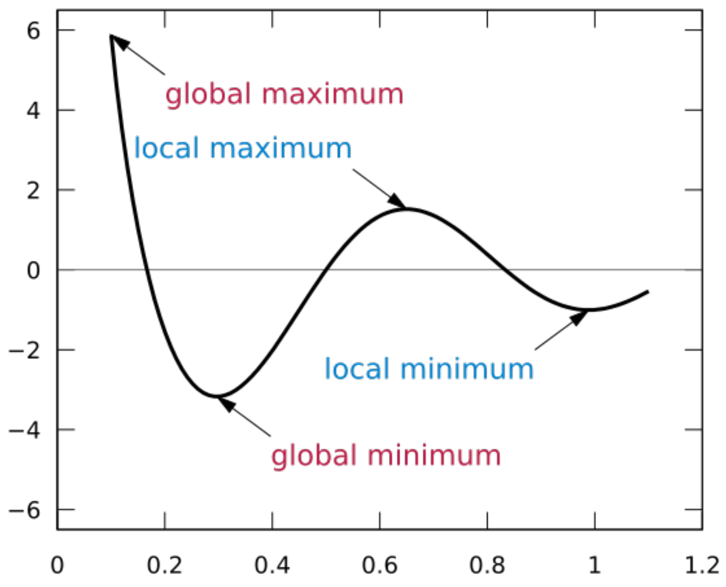


FIGURE 2.3: Global and local minima and maxima

There is also the possibility that the objective function has two or more locally optimal solutions, in which case it is known as multimodal (Deb *et al.*, 1993). An extreme example is given in Figure 2.4 which shows a surface plot from the R language of Rastrigin's function in its 2D form (Mühlenbein *et al.*, 1991), being a highly multimodal function having many local minima and maxima, and the global minimum (equation 2.3). Figure 2.5 shows a contour plot of the same function. Multimodal optimisation seeks to obtain a set of good solutions, comprising the best of the local optima along with, ideally, the global optimum if it can be found. When using methods which do not guarantee to find the global optimum, then it is more important to have a set to choose from since the global optimum may not be present and there may be different advantages and disadvantages between one local optimum and another.

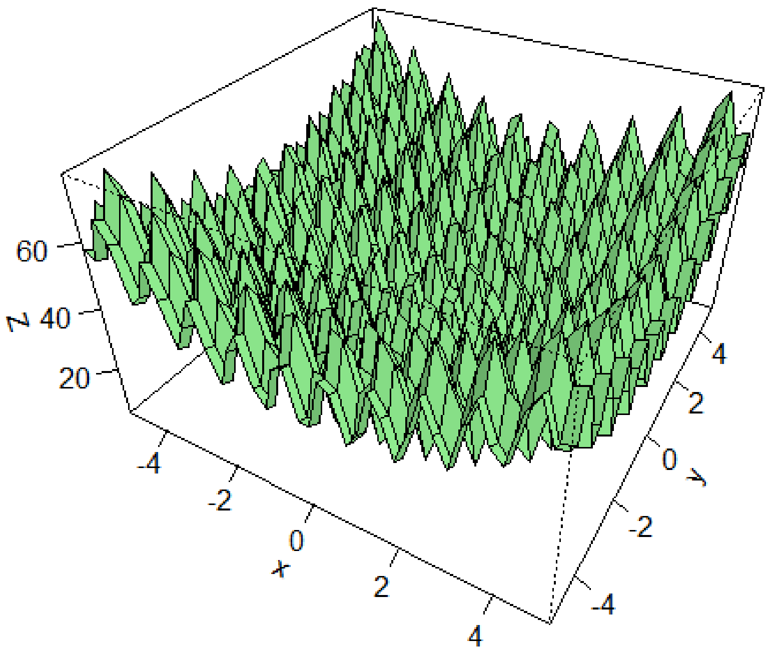


FIGURE 2.4: Rastrigin's 2D function as a surface plot, showing its many local minima and maxima, where the global minimum is at (0,0) and equal to 0.

Rastrigin's function in general form of n dimensions

$$f(x) = An + \sum_{i=1}^{n}[x_i^2 - Acos(2\pi x_i)] \tag{2.3}$$

$$\text{where } A = 10 \text{ and } x_i \in (-5.12, 5.12)$$

$$f(x) = 0 \text{ at } x = 0$$

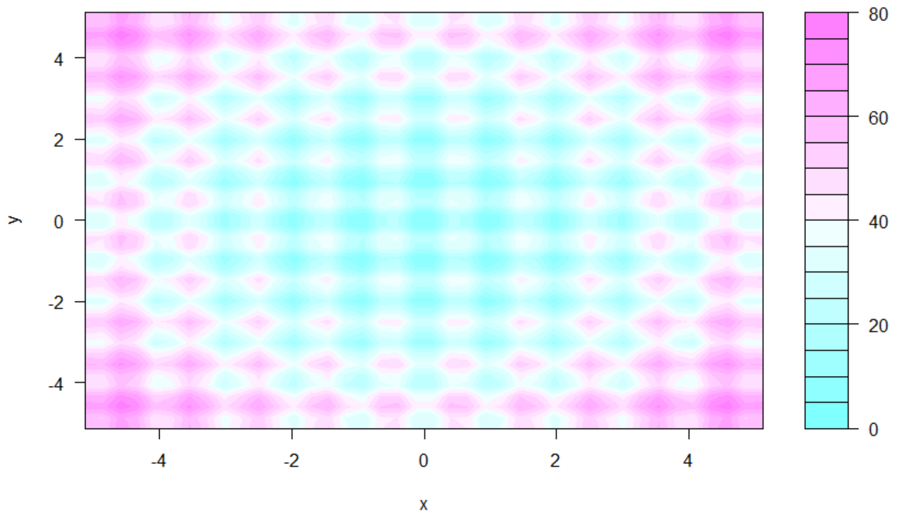


FIGURE 2.5: Rastrigin's 2D function as a contour plot, showing the many minima and maxima as countour lines with colour indicating the amplitude.

In some optimisation problems the optimiser has no (or little) in-built 'knowledge' about the subject domain of the problem upon which it works, thus the optimiser acts as a *black-box* process (Schaefer and Nolle, 2006), transforming the input into an output (Figure 2.6). On these types of investigations, evolutionary computing is of particular relevance, and this is dealt with below in its own section.

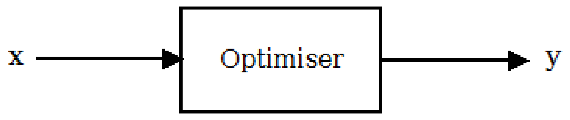


FIGURE 2.6: A general *black-box* optimiser that finds the appropriate settings of x to yield the best value for y, in which x and y can take any form.

The "No free lunch" (NFL) investigation by Wolpert and Macready (1997) addresses the issue of choosing the best algorithm for a class of problems or even a particular problem, and whether such a choice is *a priori* possible. Their investigation shows that not only is there no best algorithm, but that any algorithm that performs especially well on a particular problem class, will perform correspondingly worse than another on all other problem classes. This is true even for purely random search strategies. In particular, it is shown that if some knowledge of the problem can be incorporated, then performance tends to be better, although such knowledge is not always available. Uncertainty, in the form of lack of exact knowledge, about the precise objective function, f, can be expressed as a probability distribution, $P(f)$, and the performance of the optimiser then depends on how well its strategies are aligned with $P(f)$ (Wolpert, 2012).

For example, in the electrical grid optimisation of Chapter 5, three of the objective functions are effectively 'black-box', as they themselves depend on iterative procedures inside Plexos (commercial software that performs power calculations), about which the MOOEA has no information. The MOOEA can therefore not be stated to be *a priori* the best algorithm for this work; however it is not readily apparent that any other algorithm could substantiate that claim either. The future work described in Chapter 6, section 6.3.3 suggests a hybrid that could treat the GA as the originator of training examples for supervised learning by a genetic program (GP), with the goal of making the GA run in shorter overall times by using fast OFs produced by the GP.

Similarly, the airfoil optimisation of Chapter 4 is also effectively black-box, as it is the XFoil codes which perform the airfoil assessment for the MOOEA.

2.3.2 Multi-objective optimisation

So far this work has been concerned with the optimisation of a single quality criterion, a given problem's objective function (OF), whereas in Chapter 1 it was noted that many real-world complex engineering systems tend to have more than one OF to be concerned about. This section therefore addresses the area of multi-criterion optimisation, which is more usually termed *multi-objective* optimisation (MOO), contemporarily.

Fonseca and Fleming (1998) made the point that optimisation constraints can also be taken as hard objectives, needing to be satisfied prior to the optimisation of the further, soft, objectives, and that on the other hand, problems having many objectives have in the past been transformed into single objective ones, with hard constraints, in order to solve them. Their proposed framework enabled constraints and OFs, both treated as functions, to be manipulated together. The effect of constraints can be visualised as in Figure 2.7.

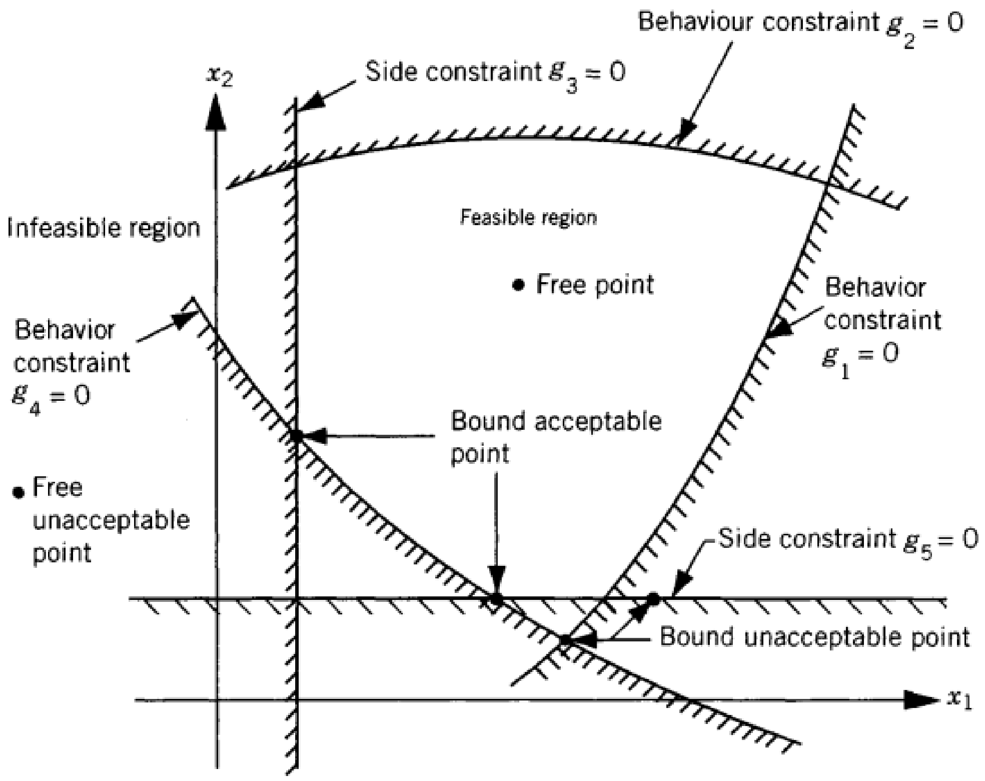

Behaviour constraint $g_2 = 0$
x_2
Side constraint $g_3 = 0$
Infeasible region
Feasible region
• Free point
Behavior constraint $g_4 = 0$
Behavior constraint $g_1 = 0$
Bound acceptable point
• Free unacceptable point
Side constraint $g_5 = 0$
Bound unacceptable point
x_1

FIGURE 2.7: Visualising the effect of constraints in a 2D objective space, $\mathcal{Z}$ (Rao, 1996).

However, the essence of MOO, is that there are two or more *conflicting* objective functions that need to be simultaneously optimised, such that each is satisfied. Systems having many OFs that are not competing, are not effectively MOO. An optimisation with OFs that are not in competition with each other, even where this has not been recognised, will naturally give rise to one solution to the problem, as is the case in single objective optimisation.

In problems that *do* have conflicting objectives, it naturally arises that there are a *set* of optimal solutions rather than just one, because no one solution can be better than all of the others with respect to all OFs, since to improve one OF necessarily degrades the other OFs (Deb, 1999). The global optimal set (in decision space) is known as the Pareto-optimal set, but other solution sets which approximate the global one, may be found and would be termed the *local* Pareto set or front. In objective space, the optimal set is known as the *non-dominated* set, since each solution cannot be said to be dominated (be more optimal) by any other. Hence MOO requires trade-offs to be made, by some higher-level decision maker (whether human or otherwise), in choosing a compromise solution to be the answer to the problem, as Figure 2.8 illustrates. Bearing in mind the *Principle of Duality* introduced at the start of 2.3, a two-dimensional (2D) MOO problem (MOOP), can be optimised in a number of ways, as Figure 2.9 illustrates, in which the direction of the convergence of the solutions are depicted depending upon the type of optimisation being performed.

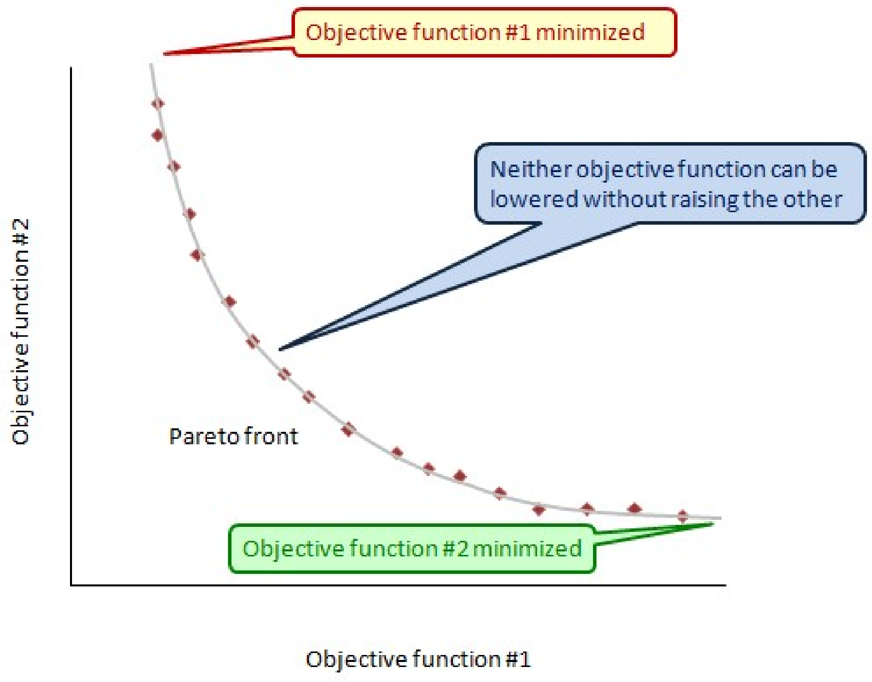


FIGURE 2.8: The Pareto-optimal front of a bi-objective minimising optimisation problem.

A corollary is that, unlike single objective optimisation, MOO gives rise to a new multi-dimensional space called the *objective space*, $\mathcal{Z}$, in which the values of the OFs exist. There is then, a mapping between a given solution in *decision space* and its corresponding point in the *objective space*, noting that their vectors are of different dimensions, the former being of n decision variables, and the latter of M objective functions.

Moreover, Coello *et al.* (2006) note, referencing Bäck (1995), that for a general MOOP (and many particular ones), searching for the global optimum is an NP-complete problem (Garey and Johnson, 1979) for any system that is of more than minimal complexity, due to the exponential increase in the size of the search space,

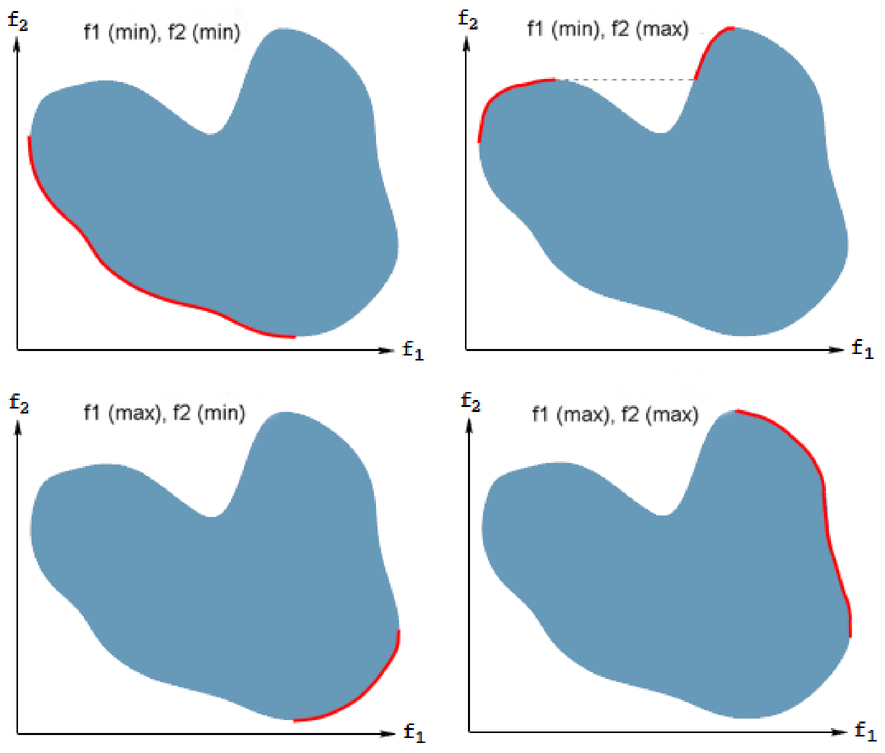


FIGURE 2.9: Bi-objective optimisation Pareto front quadrants for OFs f_1 & f_2, depending on the optimisaton of each OF being maximisation or minimisation (Deb, 2013).

which depends upon the cardinality of both the decision vector and the permitted range of its components.

To formally define Multi-objective optimisation, the following can be stated, without loss of generality (Fonseca and Fleming, 1998):

Minimise simultaneously n components $f_j, j = 1, \cdots, n$ of a function f of a general decision variable x in a universe $\mathcal{U}$, where:

$$f(x) = (f_1(x), \cdots, f_n(x)) \tag{2.4}$$

The *Pareto dominance* relation can then be defined as follows (assuming minimisation as above): A given vector $u = (u_1, \cdots, u_n)$ is said to dominate another vector $v = (v_1, \cdots, v_n)$ if and only if u is at least partly less than $v(u_p < v)$; stated formally thus:

$$\forall i \in \{1, \cdots, n\}, \quad u_i \leq v_i \wedge \exists i \in \{1, \cdots, n\} : u_i < v_i. \tag{2.5}$$

Pareto optimality is then defined as: A solution $x_u \in \mathcal{U}$ is said to be Pareto-optimal if and only if there is no $x_v \in \mathcal{U}$ for which $v = f(x_v) = (v_1, \cdots, v_n)$ *dominates* $u = f(x_u) = (u_1, \cdots, u_n)$.

In practice, optimisation problems may be subject to restrictions on the values that one or more of their decision variables may take, or on the values held to be useful in the problem solution. Such constraints can usually take the form expressed as a function inequality: $f(x) \leq c$, or $f(x) < c$, where c is a constant value and f is real-value function of x.

An inequality constraint function $g(x)$ can manage a ≤ 0 constraint (or vice-versa) by multiplying both sides of the inequality by -1 or swapping one side of the inequality for the other (Deb, 2012). For example, $g(x) \leq y$ becomes $-g(x) \geq -y$, or $y \geq g(x)$.

Equality constraints are harder to deal with, should be avoided if possible, and are a subject in their own right, especially in robust design optimisation (RDO), in which uncertainties are modelled to minimise their effects. Rangavajhala *et al.* (2007) examine approaches in RDO for equality constraint handling and provide a strategy, but here it is noted that if possible, relax an equality $f(x) = c$ by replacing it with a combination of inequalities for special cases.

A constrained multi-objective optimisation problem, having functions $(f_1, \cdots, f_k)$, can thus be expressed without loss of generality as:

$$(f_{k+1}(x), \cdots, f_n(x)) \leq (g_{k+1}, \cdots, g_n)$$

...in which x is a generic decision variable, in the universe $\mathcal{U}$, and is subject to a positive number of constraints $n - k$ applied component-wise (Fonseca and Fleming, 1998). It is not necessarily true that a solution exists in $\mathcal{U}$ which satisfies such a constrained problem.

Zitzler *et al.* (2003) extended the definition of dominance by recognising that in practice, it is often very difficult to obtain the true global Pareto-optimal set, whereas there may be one or more *local* Pareto sets, which they term approximation sets, that may suffice to provide a choice of good enough solutions. The existence of two or more approximation sets motivates the definition for further dominance relations, as follows:

Approximation Set - Let $A \subseteq \mathcal{Z}$ be a set of objective vectors for which A is an approximation set if any element of A does not dominate or is not equal to any other vector in A, and for which the set of approximation sets is given by Ω. This enables all dominated solutions to be simply disregarded.

There then may exist the following dominance relations (Tables 2.1 and 2.2), considering objective vectors (solutions) z^1 and z^2, and approximation sets A & B:

Table 2.1: Extended dominance relations to include approximation sets, from the perspective of objective vectors

Relation	Symbol	Description
strictly dominates	$z^1 \succ\succ z^2$	z^1 is better than z^2 in all objectives
dominates	$z^1 \succ z^2$	z^1 is not worse than z^2 in all objectives and better in at least one objective
weakly dominates	$z^1 \succeq z^2$	z^1 is not worse than z^2 in all objectives
incomparable	$z^1 \parallel z^2$	neither z^1 weakly dominates z^2 nor vice-versa

Table 2.2: Extended dominance relations to include approximation sets, from the perspective of approximation sets

Relation	Symbol	Description
strictly dominates	$A \succ\succ B$	every $z^2 \in B$ is strictly dominated by at least one $z^1 \in A$
dominates	$A \succ B$	every $z^2 \in B$ is dominated by at least one $z^1 \in A$
better	$A \rhd B$	every $z^2 \in B$ is weakly dominated by at least one $z^1 \in A$ and $A \neq B$
weakly dominates	$A \succeq B$	every $z^2 \in B$ is weakly dominated by at least one $z^1 \in A$
incomparable	$A \parallel B$	neither A weakly dominates B nor vice-versa

The Pareto dominance relation, defined in equation 2.5, can then be used to compare solutions within a population and rank them by how dominant and non-dominated they are with respect to the rest of the population (Fonseca and Fleming, 1993). Figure 2.10 shows an example ranking in which 'level 1' is the best performing set.

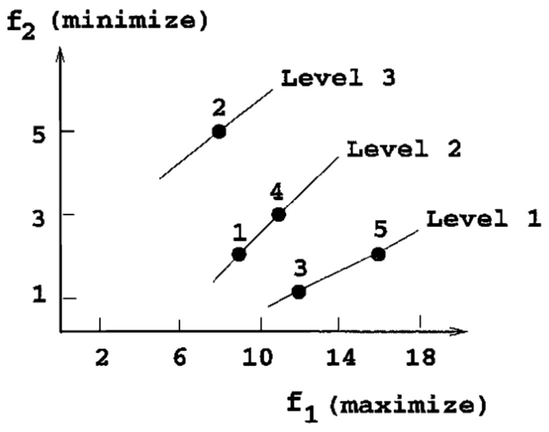


FIGURE 2.10: Pareto ranking example showing successive ranks

2.4 Performance of optimisation algorithms

The requirement to define what is meant by performance and to understand how algorithms perform arises from the desire to design algorithms and to gain insight into how well they work in practice, which in turn means how to measure their performance. Obtaining metrics for performance enables the comparison of algorithms, provides input into the design process, which may also include producing estimates of time and computational complexity and to assist in deciding how the algorithm should terminate.

Fundamentally, performance comprises both the quality of solutions produced and the effort (CPU time) or time (elapsed) required to elucidate them (Fonseca and Fleming, 1996). Effort is naturally a reflection of the *time complexity* of the algorithm running on the problem itself, usually expressed as 'Big O' notation, in which the time needed to run the algorithm is a function of the size of its input, where O(n) indicates a linear relationship (Garey and Johnson, 1979). Other considerations include computational complexity (a measure of the theoretical difficulty of the problem), and resource usage such as memory, and possibly disk space.

The quality of the solutions achieved essentially means their objective function values in the final generation or iteration. If the global optima are known *a priori* then the quality is relative to their closeness to them.

A necessary step in defining performance is to draw a distinction between the utility or cost function, more generally termed the objective function (OF), and the concept of fitness of solution. The optimisation problem is defined by the specification of the OF, whereas fitness is a measure of the desirability of a candidate solution at a given point in the execution of the algorithm. While the value of one may be equal to the value of the other in some single objective algorithms, the distinction is certainly necessary when considering algorithms in which the assessment of a given solution is dependent upon a vector of objective values (Fonseca and Fleming, 1997), and in MOOPs this is the case by definition.

For MOOPs then, 'performance' encompasses a number of considerations, foremost of which are those related directly to the Pareto optimal (PFopt) front which is the set of solutions which are globally optimal. Thus convergence to the PFopt front, both in terms of being able to achieve it and the rate of progress towards it, are basic concerns. The other primary concern is the diversity and spread of solutions along the PFopt front, or along the locally optimal approximation sets prior to convergence, since it is desirable, in the absence of knowledge about the possible solution set (that a domain expert might have), to obtain results from across the width of the (hyperplane of the) front (Fleming and Purshouse, 2001). The subsidiary concern of maintaining a degree of lateral diversity of solutions across objective space, in which solutions are perpendicular to the PFopt front, can be seen as a putative mechanism of performance improvement rather than as a direct measure of performance itself. Both cases of diversity are important in the performance of a GA as potentially good solutions should not be lost, where possible (Laumanns *et al.*, 2001). Diversity of solutions helps prevent premature convergence in the earlier stages of the process, and ameliorates the tendency to search in non productive directions later.

It is the case that the PFopt front may not be known, thus whether maximal convergence has been achieved is not always knowable, in which case performance in this regard can only be measured from an alternative known reference, such as an existing approximation front or a pre-defined set, typically of low quality points in objective space.

There are a variety of quantitative metrics for determining the spread or density of solutions in objective space, which are required both for a quality measure of the algorithm final solution set (the wider and more even the spread, the better) but also to help the algorithm while running to assign a fitness to a solution, since it is usually the case that an algorithm wants to remove one or more solutions from a densely clustered location and allow more solutions at sparsely populated locations.

A density/crowding/sparsity metric may require finding the *distance* between two or more given solutions in objective space and there are a number of ways such a thing can be calculated, for example the distances of Manhattan, Euclidean, Chebyshev, Minkowski, among others. Normalisation of the objective function values may be necessary where OFs are scaled differently, in order that the distances computed for one dimension do not swamp those of another. These values may then be used to define a region in which other solutions are to be counted, or for example by summing for a given solution the distances of all other solutions to it (Deb *et al.*, 2000). Diversity of solutions, however, comes at the price of increased computational expense (Deb *et al.*, 2003). The fitness of a solution can then be derived as a function of both its quality and its proximity to other solutions, either through a sharing scheme (Fonseca and Fleming, 1995a) in which fitness is degraded as a function of higher population density, or through the secondary application of density as a criterion when ranking would otherwise be the same.

Measures of convergence are calculated for the approximation front(s), in which diversity of solutions may be taken into account. Such metrics include:

Hypervolume (Zitzler and Thiele, 1998) was originally described as "the size of the space covered", is a measure of how much space (area or hypervolume, depending on the dimensionality of the objective space), is dominated by an approximation front, thus it is not only an indicator of convergence but also of breadth of front.

(Unary) ε-indicator (Zitzler *et al.*, 2003) is a measure of the minimum distance of translation needed to move every solution in the discovered front, so that the front weakly dominates the most converged front found, thus is an intuitive measure of Pareto dominance.

***R* indicators** Hansen and Jaszkiewicz (1998) are three different but related metrics that provide an assessment of the difference between approximation fronts, but do not work in all cases. R1 estimates the number of times the solutions of one front are better than the other. R2 is a graded estimate of the superiority of one front over the other, and R3 indicates the proportion by which one front is better than the other. They require a set of utility functions together with assumed probabilities of their occurrence and a numerical integration technique to solve the integral. Deb and Jain (2002) suggest a weighted set of Tchebycheff metrics for the utility functions.

It is important to remember that each convergence metric is in some way deficient by itself in that they may not always be applicable, depending on the distribution of points in the approximation or reference fronts being compared, or may not provide *all* the information required. Where fronts are comparable, different metrics may indicate opposite conclusions, again depending on the nature of the fronts, for example a wide spread of solutions related to a front which is only partially better converged. Thus it is advisable to use more than one metric in such comparisons, with both qualitative (better) but also quantitative (better by how much) metrics being available (Zitzler *et al.*, 2003). Zitzler *et al.* (2002)

showed that at least M metrics are needed to compare two or more fronts of an M objective problem.

2.5 Heuristics and meta-heuristics

2.5.1 Heuristics

It has been noted above that MOOPs are often hard to solve as they tend not to be amenable to analytical methods due to their usual non-linearity and multi-dimensionality (both in decision and objective space), and also often or usually, having very large search spaces from which solutions must be picked. The sizes of the search spaces of such problems tends to preclude the use of exhaustive and complete searching, which would take too long to be practical, or even possible (since they may in theory take many years).

Moreover, multimodality tends to occur as soon as the number of OFs exceed one, for non-trivial competing objectives, even where each OF by itself is of single modality and is a convex function (Fonseca and Fleming, 1995b).

The inherent characteristics of these problems leads to the adoption of other methods which are determinedly less than complete explorers of the problem space, with the corollary that solutions found may not always be the best, but either the method returns the best solution often enough, or approximate solutions are deemed “good enough”.

Heuristics then, are “criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal.”, (Pearl, 1984). Heuristic methods require knowledge about the problem domain, and may be of a stochastic nature, but tend to have simple, incomplete or unreliable information about the exact problem. There is an essential dichotomy at the heart of these methods: to be simple, yet effective. A common phrase used to describe them is “rule of thumb”, which in turn may imply some sort of summation or scoring of elements or steps in the problem. The so-called *Greedy* heuristic (Atallah and Blanton, 2009) is a good example of such a method: it adopts the general strategy of adopting the locally optimal choice at each decision point with the hope that this will eventually find the global optimum, while accepting that this will not always be the case; yet it needs specific knowledge about the problem in order to be used. The book of Polya (2004) is itself a heuristic guide to solving problems generally, even though aimed specifically at mathematical problems.

The term *meta-heuristic* was introduced by Glover (1986) in his discussion of the Tabu search algorithm, which uses a heuristic, as above, but upon which is imposed a further strategy - that of penalizing moves that take a path already taken, and an organised memory mechanism. Various metaheuristic implementations have found use in the real-world, tackling necessary tasks such as job scheduling and vehicle or network routing (Sörensen and Glover, 2013), in which any solution that works is "good enough" even if such may not be the global optimum.

It is frequently the case that in pursuing difficult problems, especially real-world engineering ones, that the Pareto-optimal set, PFopt, is neither known, nor indeed knowable analytically (Veldhuizen and Lamont, 2000), therefore until proven otherwise, it is usually necessary to assume that a given set of solutions obtained in a front, is an approximation to the global: PFapprox.

2.5.2 Meta-heuristics

The term metaheuristic is often found in the literature and as it is by nature inherently general, the definitions given for it tend to vary, nevertheless the following set of common properties are fundamental in describing it (Blum and Roli, 2003) :

- Have strategies that in some way influence the direction of search.

- Have a goal of exploring the search space efficiently to find (near) optimal solutions.
- Use a variety of techniques from local searches to complex learning strategies.
- Are often, or usually, non-deterministic, and approximate.
- May possess mechanisms to avoid becoming trapped in local minima.
- Are often described at a level of abstraction from a specific problem.
- Are not problem-specific.
- May be hybridised with some domain-specific knowledge as heuristics that are controlled by a higher level strategy.
- May use organised memory to leverage search experience to determine the direction of the search.

A metaheuristic thus is both a framework and a set of concepts, strategies and mechanisms that are consistent, for the design of heuristic algorithms. Sörensen and Glover (2013) refine the definition *metaheuristic* to be "a high-level problem-independent algorithmic *framework* that provides a set of guidelines or strategies to develop heuristic optimization algorithms". They also note that problem-specific implementations of heuristic algorithms within a metaheuristic framework, are also metaheuristics.

Figure 2.11 depicts the broad categories of metaheuristcs that have been inspired by nature, with the common thread that living organisms in some way necessitate optimisation in the struggle to survive long enough to reproduce, except for the simulated annealing (SA) algorithm (Kirkpatrick *et al.*, 1983), which uses the metaphor of the annealing process of metallurgy, as a system having multiple potential energy states therefore being amenable to statistical mechanics (analysis of aggregate properties of atoms in solids or liquids).

Particle Swarm optimisation (PSO) introduced by Kennedy and Eberhart (1995) is based on velocities of moving particles in 2, 3 or n-dimensional (hyper-)spaces, and acceleration towards solutions nearer to the global optima, where behaviour was inspired by flocking and foraging of birds and fish, but also the desire to model human social behaviour. However, there are similarities with evolutionary algorithms (EA), in that: it is stochastic; there is an operator (agent velocity adjustment) similar in effect to that of crossover in genetic algorithms (GA); the concept of *fitness* of a solution is used; and the group 'knowledge' of the best position found so far (*gbest*), being analogous to elitism. Each particle, termed *agent*, stores the coordinates of the best solution location it has visited (*pbest*) and tracks other agents in its vicinity, while maintaining a minimum separation, that it is informed are the current local optima (*lbest*). At each step, the agent evaluates its fitness and is accelerated towards *pbest* and *lbest* positions, with weighted random adjustments to its acceleration vector. A significant difference to GAs is that a PSO agent only has access to *lbest* in other agents, whereas GAs swap chromosomes with many individuals. Another significant difference is that a PSO can be considered to be a "directed mutation" method, in that individuals may be modified, related to their fitness, but no re-sampling occurs, unlike in evolutionary methods. Re-sampling is the creation of new solutions from existing ones; in canonical PSOs the population is fixed.

Ant colony optimization (ACO) (Dorigo *et al.*, 1996), (Dorigo and Blum, 2005) and its derivatives, share some common themes with PSO of animal behaviour and foraging, in which colonies of the same species attempt to maximise their net energy gain (food consumption versus the energy expended to acquire it) over time. In ACO, the agents are metaphorical ants that move stochastically over a landscape, communicating with each other by assigning values to the paths they have taken that are relative to their fitness (the metaphor has it that the fitness is indirectly proportional to the distance to the food and the path value is a pheromone trail, for example when used for the Travelling Salesman problem (TSP), shorter paths are travelled by more ants in a given time, thus acquire higher values). The path values are adjusted by the number of agents travelling them, and by a deterministic degradation function (a metaphorical pheromone evaporation) that helps to prevent premature convergence, thus the path value provides a bias guiding the agents in their decision to take it.

Various comparisons have been performed to investigate the relative performance of these approaches, but as specific algorithm details vary (for example, whether a GA is real or binary encoded, if elitism is used, and so on), and as the NFL theorem necessitates that a given algorithm will always be out-performed by another on some problems, it cannot be said of any specific metaheuristic algorithm that it is "better" than the others, generally.

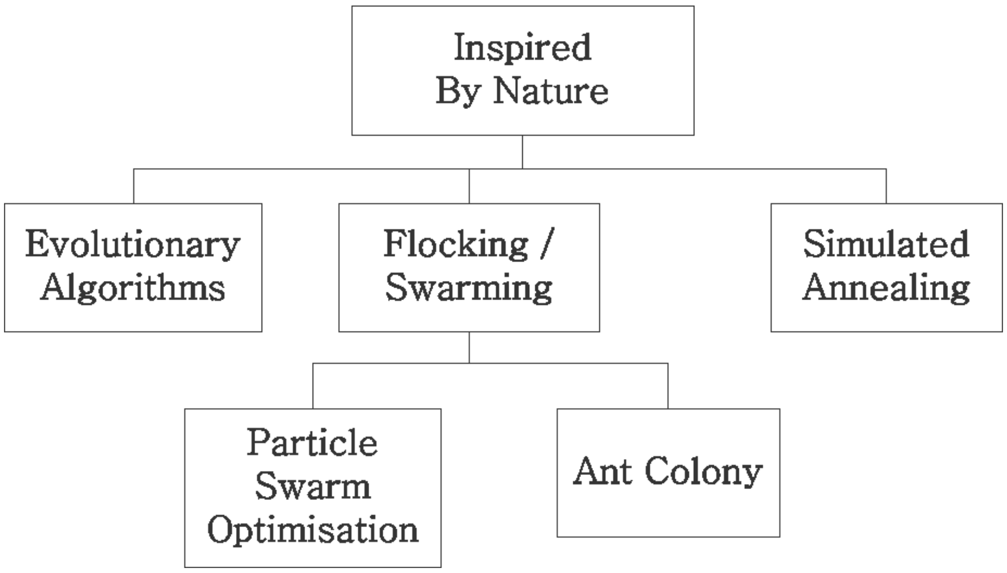


FIGURE 2.11: Nature-inspired metaheuristics.

However, so many derivative metaheuristics have arisen that are inspired by nature, from EAs through various insect and bacterial foraging, to flocking or swarming, shark hunting strategies using Lévy flights (Chechkin *et al.*, 2008) and even music-based metaphors, that it becomes a matter of intrigue to wonder how much such strategies really differ fundamentally, given that they are all attempting essentially the same thing: to both search widely in a very large space and yet to home in on regions containing good solutions in the hope of finding better ones; also known as the diversify/intensify or explore/exploit dichotomy. For example, Bäck (1995, p. 257) declares that "Evolutionary Algorithms are representatives of the mathematical concept of a Markov process (respectively chain, in discrete spaces)". Although diversify and explore both have the meaning of searching more widely in the search space, while intensify and exploit have the meaning of focusing in on promising regions, the pair diversify/intensify now are largely used in the context of some deliberate strategy, often involving specific memory caches or divisions, whereas explore/exploit tend to be more associated with stochastic undirected approaches (Blum and Roli, 2003).

Along with Sörensen (2013), the conclusion could be reached that it would be better to research existing approaches to refine them or define them analytically, in the sense of analysing how they work, rather than introduce new derivative methods that are not really novel in their fundamentals unless they can provably be shown to contribute to the field.

Therefore, the following section considers the Evolutionary Algorithm class of metaheuristics that has shown itself to be particularly useful in tackling MOOPs, and the Genetic Algorithm in particular, as a refinement was identified that was novel and potentially of use for MOOPs in general and complex engineering systems in particular.

2.6 Evolutionary algorithms

The "classic" optimisation approaches such as those derived from Simplex or gradient-based methods, are not inherently adaptable to MOOPs as they were designed with single objectives in mind, which means a MOOP needs to be transformed in some way into a single OF problem if such methods are to be used, and this is always a compromise, as it biases the result to one objective or another. This transformation also means that each run produces but one solution, when we know that a MOOP by definition (see above) gives rise to many candidate solutions. It is not only the "classic" optimisation methods that struggle with MOOPs; metaheuristics such as simulated annealing, which were also conceived for single objective problems, are subject to the same issues too.

The Evolutionary computing (EC) field historically had problems with recognising names and terminology from one sub-discipline to another, but now it is generally recognised that EC is synonymous with EA and the categories in the given figure are now generally accepted. EA is a class of metaheuristic that encompasses a number of different type of algorithms that have been inspired, at different times and in various ways, by evolutionary biology, in particular neo-Darwinian natural selection (Figure 2.12).

The defining characteristics of algorithms in the EA field is that they:

- are composed of a population of potential candidate solutions, which in the canonical case is created stochastically initially.
- have some way of selecting good solutions from the population.
- have some way of using the decision vectors of selected solutions to create new solutions.
- may provide some way to change the decision vectors of existing solutions.
- through some or all of the above, the solutions are able to improve over time without external influence.

Essentially then, the solutions are evaluated and compared; good ones are allowed to persist and the less good are ignored or discarded, and new ones are created from the existing ones, iteratively. In this way, solutions evolve over time to become better, in terms of their OFs. The various algorithm types (see Figure 2.12) have different strategies to accomplish these broad and general goals.

Because EAs contain a population of solutions, rather than just one, they are inherently more suitable to MOOPs which must also, by definition, provide multiple solutions, as this enables EAs to provide many solutions per run. Indeed, one EA run can theoretically find all the possible Pareto optima, whereas non EA approaches would be able to find one only per run and have to be run n times to find n further optima (Coello, 2006).

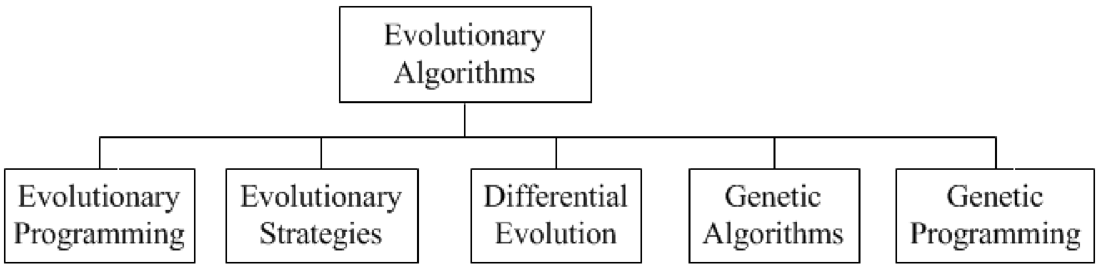


FIGURE 2.12: The main sub-classes of evolutionary algorithms

EAs are able to manage discontinuous, non-differentiable, multi-modal, noisy, and otherwise unconventional response surfaces Schwefel (1997), and because of this robustness are able to cope with a broad field of applications. Fonseca and Fleming (1995b) also note that EAs are inherently able to be parallelised due to being population-based. EAs according to Bäck *et al.* (1997) are "especially well suited for solving difficult optimization problems", and all these attributes of EAs provide good reasons for using them to tackle real world-engineering problems.

Of course, EAs have their limitations and when considering MOOPs, those EAs using a Pareto-dominance strategy can struggle on problems that have four or more objectives to be solved simultaneously (Ishibuchi *et al.*, 2008). The performance of convergence can deteriorate as a large proportion of the solutions in the population become non-dominated, which has the effect of decreasing the pressure of selection upon them. This phenomenon occurs since each new objective adds a degree of freedom in which the solution could excel, and the way dominance relations are defined means an otherwise poor solution can be non-dominated if it is 'good' in just one objective. In these hyper-dimensional objective spaces, exponentially more solutions are needed to approximate the Pareto-optimal front.

However, despite these challenges, EAs remain among the best approaches to tackle such problems, and there are strategies that can be used to at least ameliorate such problems in some circumstances as described further in 2.6.1.

Most implementations of EAs descend from the closely related approaches which were proposed independently in different places although at similar times, these being genetic algorithms (GA), evolutionary programming (EP), and evolution strategies (ES), as depicted in Figure 2.12. Genetic programming (Koza, 1998) is fundamentally different to the other approaches, in that its purpose is to produce a set of optimal computer programmes, rather than a set of optimal decision vectors as the others do.

ES was introduced by Rechenberg and refined by Schwefel (Rechenberg, 1973, and Schwefel, 1981, cited in Bäck, 1992), and EP was first posited by Fogel (Fogel, Owens, and Walsh, 1966, cited in Bäck and Schwefel, 1993). The three main classes, ES, EP and GA, are quite similar from a broad perspective (Fogel, 1994), since each consists of a population of candidate solutions that may be changed stochastically, and from which some are selected based on some quality related to 'fitness' while others may be in some way discarded. Originally GAs were binary-encoded while ES and EP were based on real values, and GAs had both mixing (crossover) and mutation operators, while ES and EP had just mutation, but over time they have become functionally more similar as ESs adopted crossover and GAs adopted real-encoding.

Beyer and Schwefel (2002) describe ESs as steady-state algorithms in which the population is constant with members being replaced by better offpsring as they occur, depending on the exact scheme used, giving the following ES types:

$(1+1)$: has just two solutions at each iteration: an old parent and a new child. Mutation occurs to all parent decision variables with random change and if the offpsring is better than the parent it is kept, and if not, discarded.

$(\mu+1)$: 2 parents produce 1 offpsring which replaces one of the parents if the latter is worse, or is discarded otherwise.

$(\mu+\lambda)$: in which not only one offspring is created at a time or in a generation, but $\lambda \geq 1$ descendants, and, to keep the population size constant, the λ worst out of all $\mu+\lambda$ individuals are discarded.

(μ, λ): in which the selection takes place among the λ offspring only, whereas their parents are forgotten no matter how good or bad their tness was compared to that of the new generation. This scheme must ensure $\lambda > \mu$.

Schwefel (1997) found that GAs were the most popular type of EA among EA researchers, at that time, and that the *Building Block* (BB) concept, descended from Holland's (Holland, 1992) *Schema* explanation of GA performance although he also noted that the BB theory had never been proved, and this remains the case. Some research has shown the Schema theory to not hold true for all cases, in particular Mühlenbein (1991) showed, using a deceptive function[3], that "The estimate of the fitness of a schema is equal to the exact fitness in very simple applications only. Therefore interpretations using the exact fitness cannot be applied in connection with the schema theorem. But if the estimated fitness is used in the interpretation, then the schema theorem is almost a tautology, only describing proportional selection"; a comparison is also made with (Grefenstette and Baker, 1989) whose criticism is said to be of a similar vein.

The GA is a very flexible kind of EA, because its ability to be targeted at a particular problem, NFL theory notwithstanding, is primarily dependent upon the data structure (and associated functions) of its problem representation, or *chromosome* as the metaphor has it. Thus GAs can be used on discrete and continuous problems and objective spaces of all sorts of topologies, for example convex or concave function spaces. ESs however, are more naturally associated with parameter optimisation, due to their being continuous in nature, while EPs flexiblity lies between the ES and the GA (Bäck and Schwefel, 1993).

Moreover, Deb and Agrawal (1999) found that using GAs with tournament selection and both recombination and mutation operators, with an adequate but not excessive population size, were able to not only tackle difficult optimisation problems, but were also not necessarily overly computationally expensive, in contrast to popular belief.

[3]A deceptive function has a low-order schema fitness average that favours a particular local optimum, but has a global optimum located at that optimum's complement (Goldberg *et al.*, 1989).

The GA is thus used as the template for this work, as it has been shown to be effective on real world problems in a variety of domains, in science and engineering (Gen and Cheng, 2000), (Fleming and Purshouse, 2002), (Deb, 2012), and the GA of this work adopts a variety of *chromosomes* to enable it to be of general applicability, and the mechanism of doing so is another aspect of its novelty.

Genetic Algorithms (GA) are a class of evolutionary algorithm (EA) that first were used in incipient form by Rosenberg (1967) and subsequently systematised by Holland (1992) for the binary form, then expanded upon by Schaffer (1984) who introduced the first MOGA (called VEGA), and then Goldberg (1989). They are stochastic and characterized by populations of potential solutions that converge towards local or global optima through evolution by algorithmic selection (Jones *et al.*, 2002) as inspired by neo-Darwinian (Coello, 2006) evolutionary processes. An initial population of random solutions is created and through the evaluation of their fitnesses for selection for reproduction, and by the introduction of variation through mutation and recombination (crossover), the solutions are able to evolve towards the optima.

The biological metaphor is used heavily in usual discussion of GA mechanisms and structures, in which a given candidate solution to the optimisation problem is termed an *individual* or *organism*. The decision vector of the solution is termed the *chromosome* which in turn is composed of a set of *genes*, each of which is a variable of the appropriate data type (which may be primitive or a class[4]) and whose value is an *allele*. The result of the evaluation of the OFs, which depends on the chromosome, is then the *fitness*. The chromosomes may then be subject to manipulation by certain functions, termed *operators*, the most common of which are *crossover* and *mutation*. Crossover, also known as *recombination* in the biological domain, occurs during the creation of new solutions, in which two existing solutions (*parents*) are combined to create one or more new ones (*children*) by mixing the chromosomes in certain defined but stochastic ways. This is metaphorical sexual reproduction. Mutation is the process of altering one or more genes in one solution's chromosome. A variety of other similar operators exist which in some specific way also alter the chromosome(s) or gene(s) in order to explore the search space, and whose quality depends upon the specific details of the problem concerned. The operators are given a certain probability of being invoked, as part of the GA run parameters.

Fonseca and Fleming (1993) implemented a rank-based fitness scheme, inspired by Goldberg, in their MOGA that also enabled an external decision maker to choose solutions; further research over more recent years by Fonseca and Fleming (1998), Fleming *et al.* (2005), Deb (2001), Zitzler *et al.* (2003), and others, has improved their ability in handling multi-objective optimization problems.

GA performance on a given problem has been shown, since De Jong (De Jong, 1975), to be extremely sensitive to the settings of its parameters, these being the probabilities of mutation and crossover occurring, the population size and the number of players in a tournament selection (when this selection method is used). Moreover, for certain real/continuous encoded GAs, it is necessary to consider operators' polynomial distribution indices (Deb and Agrawal, 1995).

2.6.1 Algorithm adjuncts and concerns

Evolutionary algorithms, such as GAs, can be improved upon by a variety of further tactics (not discussed as yet) which have the purpose to in some way explore the search space more effectively or exploit good solutions in order to find better ones or other solutions in regions not populated as was discussed previously under diversity in performance.

Selection is the term used to describe which solutions are chosen to either derive new ones from ('breed') or which ones persist into the next generation, depending on the life-cycle scheme used. Goldberg (1989) described the Roulette Wheel approach in which the probability of a solution being selected (to breed with another) is proportional to its fitness, under stochastic choice. This was found to lead to genetic drift due to the bias of the procedure. The Tournament variation

[4]A *class* being a complex data type, in the sense of the object-oriented analysis and design methodology of software development (Rumbaugh *et al.*, 1991).

(Goldberg and Deb, 1991) chooses two (or more) solutions at random, compares them (for fitness[5]) and chooses the most fit for breeding. This has been shown to be less biased and good for parallel implementation. Gen and Cheng (2000) provide references to other possible methods of selection: the μ & λ methods are deterministic ones that select the best of (parents *and* offspring); *truncation* and *block* are deterministic also, ranking in order of fitness and choosing the best, while the steady state and generational selections take the similar approach, choosing a subset of the population to replace, where the subset might be all, or replacing the worst n with n offspring.

Elitism, first proposed by De Jong (1975), acts as adjunct to selection, in that it archives the best solutions (or some subset of them) and allows (some of) them to be re-inserted into the population for comparison or breeding. This prevents these best solutions from being lost through normal action of genetic operators or random loss due to non-selection.

A variety of archiving schemes are used in different algorithms, having the common characteristic of maintaining some subset of best solutions, and differing in how solutions enter and exit the archive; exit might be by time (or generation number) expiry, or replacement by better solutions; entry might be governed by quality criteria or by archive size. For example, Knowles and Corne (2000) in their PAES algorithm, use a size-limited archive to which entry is gained by an offspring when it is better than its parent or when it is better than a solution in the archive in which case it replaces the one it is better than.

An alternative, one might say opposite, approach to elitism and archiving is that of *restart* (Fukunaga, 1998) in which one run is halted and instead a new one begun with a (probably) different starting population. In this case it is the multiplicity of runs which leads to the best solutions found, as more initial conditions are experienced and it is known that with stochastic algorithms, some starting populations lead to better results than others. The 'new' runs can of course be carried out in the same run, by discarding the final population (having

[5]which might be by rank then distance, for example

written it out to storage or archived it internally) and generating another starting population.

The time complexity of an optimisation problem can be exacerbated by objective functions which themselves are computationally expensive, as is the case, for example, in computational fluid dynamics (CFD), in which a (very) large number of equations need to be solved numerically per *function evaluation*. Under such circumstances, an objective function might take many minutes, or even hours or days to calculate. Population based algorithms such as some Evolutionary ones suffer from this due to the large numbers of function evaluations (population size $\times$ number of generations) needing to be performed.

Surrogate models (Forrester *et al.*, 2008) are one way to circumvent this practical problem. These models, which might also be known as metamodels, once created act as alternative objective function sources, but being much cheaper computationally in use, perhaps orders of magnitude cheaper, through curve-fitting or response surface modelling depending upon the number of decision variables. As well as being less time complex, surrogate models also need a useful degree of accuracy.

The models are created as data in a bottom-up manner, through sampling of well considered data points (through a design of experiments) using the expensive computational route. The model is assessed at various quality points (such as noise, constraints, robustness), with data being replaced or additionally created as necessary for the validity of the final model. It is assumed that the (engineering) function behaves in a smooth continuous way, such that non-sampled data points can be calculated with requisite accuracy.

An alternative to a surrogate model is to use the Kriging method which acts as the surrogate, as created by Krige, and as discussed by Forrester *et al.* (2008). In this approach, it is assumed that the observed responses (the objective functions) derive from a stochastic process even though they may arise from deterministic codes. A set of random vectors is generated from the sample data and a set of correlation values is created between them using the Kriging basis function, from which a correlation matrix is created for all the sample and observed response data, along with a covariance matrix. The covariances then govern a Gaussian process modelling interpolated values for data points not in the sampled set. This method has been shown to be the best (linear) unbiased predictor of interpolated values.

Evolutionary algorithms have been applied successfully to a variety of multi-objective optimisation problems to various extents, where *multi* had tended to be between 2 objective functions (OFs) and up to around 4. Problems having more than this number of OFs are now commonly referred to as *many* to highlight this boundary, above which success has been seen to diminish either in ability to converge or in providing diversity of solutions, or with requisite longer run times (Khare *et al.*, 2003).

A study by Hughes (2005) found that the hypervolume quality metric indicated that the Pareto ranking method tends to inhibit algorithm performance compared to those which use other than Pareto ranking methods. That study also found that while it is better to generate an entire Pareto set in one run where it is possible, splitting a many objective optimisation problem (MaOOP), especially for the higher dimensions, into a collection of single objective problems, works better (for convergence) than their Pareto ranking algorithm equivalent MOOPs, such as NSGA-II.

Knowles and Corne (2007) found that for MaOOPs with ten or more OFs, a purely random search could show better convergence than an MOEA. This seems to be largely because with more OFs, the more likely it is that a solution is non-dominated in at least one dimension thus reducing the Pareto ranking hierarchy. The study found that the mode rank shifted away from 1, tending to lead to middle ranks given a higher fitness than warranted, and selection pressure being reduced. They also found that MOEAs could be efficient for MaOOPs with 10 or more OFs, if part of the Pareto front could be discarded/ignored, through preference direction. They noted that investigation of degeneracy might be fruitful, in which several distinct decision vector points map to the same objective vector, where less uniform degeneracy has a greater effect on possible out performance of random search.

With the above in mind, approaches based on the supplanting of Pareto dominance with alternative techniques have been variously proposed, as outlined by Purshouse and Fleming (2007) in which they also use standard ranking and diversity techniques to illustrate the concepts of dominance resistance and active diversity promotion. The work showed that techniques for promoting diversity, as well as being very influential on outcome, could be in some cases deleterious. As noted above, in higher dimensions it is more likely that solutions become non-dominated, leading to selection based solely upon the secondary diversity criterion, and they noted that even in an enumeration of search space, the proportion of non-dominated soltions could be very high, and that standard operators struggle to find newly dominant solutions. As Knowles and Corne (2007) found that preference direction could be way to enable MOEAs to work in higher dimensions, they highlight the potential merit of *preferability* operators and suggest investigation of non-Pareto dominance comparators such as using perfomance indicators e.g. hypervolume, as criteria.

The Ishibuchi *et al.* (2008) review gives an alternative dominance ranking that eliminates more solutions, thus increasing selection pressure, and a ranking scheme which takes account of the number of objectives by which one a solution may be better than another. Decreasing the dimensionality of objective space by bundling up objectives together may be possible, where a translation between domains is feasible.

Saxena and Deb (2007) showed that MOEAs could work well in higher dimensions using an approach based on principal components analysis (PCA) with non-linear dimensionality reduction. This has been shown to be effective for problems with up to 50 objectives. They suggest that their parameter 'k' for their "MVU-PCA-NSGA-II" proposal, should take the value 'k' $= \lceil \sqrt{M} \rceil$ not only for the problems they tests, but for use in general (where M is the number of objectives.)

2.6.2 Self-adaptation

Real-encoded GAs can be thought of as being similar to ESs, depending on the details of the operators, where ESs are able to adapt their control parameter (or strategy parameter as it was called). The GA described in this work adopts this extra capability to adapt the control parameters but in a self-adaptive manner.

The mechanism used by ESs originally was not self-adaptive as they used a meta-mutation rate to control the rate of change of the mutation rate control itself. The term *self-adaptive* used here is meant in the sense of that coined by Eiben *et al.* (2006b), to indicate control parameters of the GA that are encoded in the chromosome along with the problem definition parameters applying to the objective functions (the *main* parameters), and that these control parameters are subject to change along with the *main* parameters due to mutation and crossover. This is different from a purely *adaptive* control parameter strategy as in that case the change is instigated algorithmically by some feedback at the higher level of the GA rather than the lower level of each chromosome/solution in the population. The *deterministic* approach is rule-based and is not considered adaptive.

Bäck *et al.* (2000) used an initial random mutation rate in which the mutation rate of each solution was initialised to a random number in the range (0.001, 0.25), however they suspected that this randomness slowed down convergence to some extent. Problems can be highly sensitive to the mutation rate, so starting off from ones which are 'far' from the ideal would indeed likely inhibit progress.

Eiben *et al.* (2006a) showed how population size and tournament selection size can be made to be self-adaptive, although in the former case to the detriment of performance of the optimization. Nonetheless, the latter case was shown to improve performance, and the method by which a parameter whose context is the population can be set through the aggregation of its representation at the individuals within the population, can be extended to other parameters having the same high-level context. However, the above work only uses mutation to affect each self-adapting parameter gene, rather than including the parameter genes in the crossover of the chromosome as a whole, and the model used is a steady-state GA (SSGA) with relatively low replacement strategy rather than a generational one (GGA). This work uses a fixed tournament size in order to restrict all the self-adaptation occurring to the level of the individual, rather than by aggregation, since this is the focus of the work.

Smith and Fogarty (1996) Discuss their steady-state GA having adaptive mutation, in which mutation first occurs to the gene encoding the mutation rate and then the new mutation rate is applied to the *main* genome. This is similar to the (real-encoded) steady-state ESs described by Beyer and Schwefel (2002).

Zhang and Sanderson (2008), (Zhang and Sanderson, 2007) describe differential evolution (DE) algorithms that use self-adaptation, including their multi-objective (MO) JADE2 and JADE algorithms, that generate new values for mutation factors and crossover probabilities based on probability distributions governed by self-adapting *means*. DEs (Storn and Price, 1997) are similar to GAs but new solutions are produced by adding the weighted difference of two population vectors to a third, to create a new donor vector which is recombined (crossover) with a target (parent) vector to produce the trial (child) vector. Differences between GAs and DEs, both algorithmic and from a performance perspective, are discussed in Tusar and Filipic (2007). In a DE scheme, the mutation factor is a weight rather than a probability as in a GA, and notably crossover acts on whole parameters (the genes in a GA) rather than parts of parameters (Holland's *schemas*).

Sareni *et al.* (2004) describe self-adaptation in a multi-objective genetic algorithm (MOGA) in which there is a self-adaptive choice between three different crossover operators for crossover, and in which mutation is self-adapted by the standard deviation of the amount of perturbation applied to a gene. Both of these mechanisms are different to the ones employed by the MOOEA in this work.

Tan *et al.* (2006) expounded their binary MOGA in which their mutation operator (adoptive mutation or AMO) modifies the mutation rate deterministically as a function of time. Their AMO is a modified bit flip operator, and acts upon a binary-encoded chromosome with each decision variable (DV) having the same probability of undergoing change. This probability is set so that on average 1 DV is changed per chromosome, and the mutation rate is the probability of a bit of the DV changing if selected for change. The mutation rate is set by a deterministic function proportionally to the generation number, such that it tends to start off relatively high and diminishes over time. The mutation rate is thus a global one applying to all chromosomes equally. Their scheme has the property that the start rate is much higher than normal (0.8) and that it is switched to a lower one (0.05 or less) suddenly rather than having a smooth decrease over time. Also of interest is their 'enhanced exploration strategy' (EES), which is essentially a deterministically adaptive diversity promoter that acts by creating a sub-population whose purpose is to explore lesser populated areas. The size of the sub-population is determined at each generation deterministically and is a function of the rate of progress towards convergence, where progress is measured by the ratio (epr) of new non-dominated solutions in $g + 1$ to the total number of non-dominated solutions in g. The number of solutions in the exploring sub-population is set higher when epr is low and conversely lower when epr is higher. Their results showed both improved performance. The EES approach is probably applicable to real-encoded algorithms too.

Tan *et al.* (2009) discuss a binary MOGA in which they define an 'adaptive variation operator' having two elements: (1) a deterministic control to modulate the degree of exploration or exploitation; (2) an adaptive control that coordinates crossover and mutation operators. Each gene (DV) of each chromosome has its own control parameter for the rates of crossover and mutation, which are deterministically assigned by (1) with the intention that they change linearly over time (starting higher and decreasing). They have a scheme to modify bits relative to their position in the gene, which is specific to the binary chromosome; that bits towards the high end (most signigicant bit (MSB)) change more at the beginning and less at the end due to the size of the effect they have, and other bits change proportionally to their position. Their adaptive control (2) is predicated on the idea that mutation and crossover should not be independent operations as mutation can destroy valuable schema swapped in crossover (remembering that it is a binary chromosome), and that the performance of crossover, and mutation, varies throughout the life-cycle. They suppose in general that dissimilar chromosome structures are often more dominant in the initial stages while similar ones dominate in the later stages as fitness selection or drift direct. Thus the control considers the chromosome content and only allows mutation to occur when crossover occurs and only for regions in each chromosome which are similar. Their test results showed these approaches would work well.

It seems reasonable to suppose that having two control parameters per gene is effectively extending the search space by a possibly signigicant factor, leading to the adaptive behaviour needing more time in which to improve itself. It is arguable whether there is a theoretical equivalent scheme for control (2) in a real-encoded chromosome.

Ho *et al.* (1999) used a binary GA for single objective optimisation in which sub-population groups adapted their mutation or crossover rates based on feedback from average fitness increase, while Li *et al.* (2004) investigated diversity-guided mutation and deterministically adaptive mutation and crossover rates in a binary single-objective GA. These works all found their implementations of the various adaptive methods provided advantages on mathematically based benchmark problems.

2.7 Optimisation frameworks

2.7.1 Synopsis

Optimisation frameworks are pre-existing software codes that provide a way to more or less easily add optimisation problems into them. Their purpose is commonly to provide an optimisation algorithm (or a choice of them) with which to run newly constructed problems. These have been produced in a variety of languages to run on a variety of platforms. Their existence enables researches to use MOEAs, whose primary purpose is not the investigation of MOEAs necessarily.

There are many EA resources available in some shape or form, but not all take the framework approach. There are a growing number of frameworks supporting solely genetic programming, but these are considered out of scope here.

The work of this thesis provides such a framework and is described in Chapter 3 and in particular in section 3.2.4.

2.7.2 Non-commercial frameworks

Within this section are those frameworks which are available on a non-commercial basis.

MOEA Framework (Hadka, 2016) is a Java-based comprehensive library supplying a variety of EA algorithms and also particle swarm and genetic programming options. It can interact with jMetal libraries and also PISA, and provides problems also. Although it has a quick start guide, the documentation is not free which although reasonable given what is on offer, is nevertheless a hurdle, especially in making an assessment of how easy it is to use in practice.

jMetal (Durillo and Nebro, 2015) is a framework implemented in Java which is focused on running MOOPs and it supplies a good number of EA algorithms for both MOO and single objective problems, and it also supplies some problems too. It has the particular ability to run parallel algorithms. It provides the ability to mix real and binary encodings, and provides quality indicators. Not as comprehensive as the MOEA Framework but still offering much.

Evolving Objects (Evolving Objects Team - various, 2015) is described as an Evolutionary Computation Framework, and is implemented with C++. It is said to run on Linux and Windows and provides a comprehensive set of facilities for someone who knows C++. In addition to EAs it also provides other algorithm options such as particle swarm, ESs, a variety of operators, parallelisation tools and a Mersenne random number generator for better randomisation performance. It seems to be more of a library than a framework and seems more targeted at Linux.

EvA2 (Zell *et al.*, 2016) is a framework in Java also offering EA algorithms along with others such as particle swarm, DE, ES, and also classical ones such as simulated annealing (SA) and others. It offers the ability to run algorithms distributed over network nodes and is GUI rather than command line based.

Opt4J (Lukasiewycz *et al.*, 2016) is a seemingly simpler framework in Java also offering EA algorithms along with particle swarm, DE, and simulated annealing, along with some optimisation problems, and is GUI based.

2.7.3 Commercial frameworks

Within this section are those frameworks which are available on a commercial only basis.

It is common, indeed even usual, that commercial software is not *open source* in the sense that its source code is not available for inspection, with a prominent exception being in the case of some security products. The Free Software Foundation, Inc (2016) extends the definition of *open source* to "software that gives you the user the freedom to share, study and modify it" and calls this *free* in the sense of freedom rather than cost (although there is a debate about the definitions and benefits of both *open* and *free*).

The existence of free (cost), free (freedom) and open source software to science and engineering academics, is a great benefit not only because issues of licenses, terms and conditions of use, financing and availability are obviated, but also because the correctness of the code itself, in terms of the science or engineering theory being addressed, can be inspected (or even changed), and where the code may be considered to be part of the experimental design. The lack of a commercial goal also decreases or obviates conflicts of interest which otherwise can inhibit collaboration (Jacob, 2016). Some commercial software may made available (binary executables and much more rarely, source code) to academics, but licence terms and conditions can prove limiting.

However, while it is generally preferable to have open source/free software for research purposes, specifically required functionality may not necessarily be available, in which case commercial codes may be the only recourse. It may also be the case that commercial software is simply better, in terms of quality or more extensive functionality, through being well-funded, hence having many more person-hours of development, or providing more or higher quality documentation. Many of these products provide GUI front-ends or GUI data visualisations, and this is the case for the products given below.

MATLAB (The MathWorks, Inc., 2016a) is an environment and programming language which supports operations on vectors and matrices, enabling set operations to be performed by single language instructions which in other languages such as C++, Fortran or Java, would require loops to be coded for. As an additional product requiring purchase, the Optimization Toolbox™ (The MathWorks, Inc., 2016b) is available, which provides solvers for linear programming, mixed-integer linear programming, quadratic programming, nonlinear optimization, and nonlinear least squares. These solvers can find optimal solutions to continuous and discrete problems and perform trade-off analyses. This product does not provide a framework as such, but the codes can be made to be called from other software, though it does have a GUI.

The modeFrontier product (ESTECO SpA, 2016) provides both an optimisation framework and a workflow pipeline for multi-disciplinary design optimisation, enabling computer aided engineering (CAE), computational fluid dynamics (CFD), or finite element methods (FEM), to be brought together along with post-processing, visualisation and statistical analysis. The integration of specific examples of the aforementioned tool types are given, but it is not readily apparent that other types of software codes can be integrated. The product also has a response surface method functionality, enabling meta-models to be produced for faster online optimisations. There seems to be a comprehensive set of single and multi objective optimisation algorithms available, from various heuristic global searches (GA, ES, PSO, SA), through deterministic (Simplex, game theory) and gradient-based searches.

Optimus (Noesis Solutions, 2016) is a process integration and design optimisation platform for design space exploration, engineering optimisation and robustness and reliability. It has seemingly similar functionality to modeFrontier, in that it provides workflow, a variety of specific external product integration possibilities, and single and multi objective optimisation algorithms (differential evolution, simulated annealing, particle swarm optimization and covariance matrix adaptation evolution strategy, and a small number of others it says are evolutionary in nature but for which the description is vague). It also provides an interactive optimisation capability, enabling designers to direct progress through preference choices.

OptiY (OptiY GmbH, 2016) is a multi-disciplinary analysis and optimisation product with a focus on computer-aided design (CAD) and CAE, robust design, reliability analysis, and optimisation, along with workflow. Integration of external products is available, although since the interface architecture is dependent on Microsoft's (MS) ".Net" architecture, these are limited to MS Windows™ operating systems. A variety of design of experiments (DoE) and post-processing analyses are also available. The optimisation possibilities include single and multi objective codes Hooke-Jeeves local search, grid-Search, adaptive response surface, ES, GA, Pareto strength evolutionary algorithm and surrogate modelling.

The optiSLang product (Dynardo GmbH, 2016) is robust design optimisation software focused around CAE, with multi-disciplinary workflow and interfaces to a variety of other products. It provides the following multi-objective optimisation algorithms: gradient methods, genetic algorithms, evolutionary strategies, and Adaptive Response Surface Methods. It has the facility of Python and C++ interfaces allowing further integration to other products having compatibility with those alternatives, though there is no immediately available information about binary compatibility.

Kimeme (Cyber Dyne s.r.l, 2016) is a platform for multi-objective and multi-disciplinary optimisation, providing process integration, workflows, experimental design, and optimisation heuristics (which they say are NSGA-III, MOjDE, BOBYQA and BFGS, and unspecified others). It is not readily apparent what external interfaces it provides.

Chapter 3

The self-adaptive MOOEA

3.1 Introduction

As part of the work of this thesis, a meta-heuristic framework and multi-objective optimising evolutionary algorithm were designed and implemented in the Java language (Oracle, 2014), which for ease of reference are collectively named Ganesh[1]. The framework provides an environment in which the MOOEA runs, and a novel mechanism for incorporating the problems upon which the algorithm operates, as well as mechanisms for interacting with the external environment in which the framework runs. Sections 3.2.3 and 3.6 provide further details about aspects of the latter.

Apart from any specific novelty Ganesh offers, the development of the codes was of benefit to the author as it enabled full understanding and control of the codes and in itself provided insight into the issues of MOOEAs.

The MOOEA part of Ganesh is enabled to be independent of the optimisation problems it runs, by having the problems defined as separately compiled plugins which are given as parameters to the MOOEA at run time, to dynamically load them and execute them in the appropriate way.

[1]The name Ganesh comes from borrowing some letters from 'GA with non-domination ranking and elitism', roughly.

The MOOEA part of Ganesh is a genetic algorithm (GA) employing elitism and Pareto non-domination ranking, originally posited by Goldberg (1989), first implemented by Fonseca and Fleming (1993), and subsequently developed by Deb (2001). This GA adopts the general principle but implements it in a simplified manner which is nonetheless of similar performance in terms of computational complexity.

In writing about the MOOEA part of Ganesh, the biological metaphor is used extensively, as explained in Section 2.6. Here, a distinction is drawn between a solution's decision vector, which following the biological metaphor are termed the *main* genes, and the control parameters governing the self-adaptation, which are termed the *control* genes. In the self-adaptation scheme described further below, both *main* and *control* genes are encoded within the chromosome, wherein the *control* genes are themselves also subject to both mutation and crossover, along with the *main* genes.

The GA uses a novel crossover mechanism in order to recombine the mutation and crossover rates, as well as each of their perturbation control parameters, and unlike other GAs, provides a novel tunable control for the number of duplicate chromosomes in each generation. Benefits and novelties of the framework and algorithm are set out in section 3.2.4.

Through the mechanism of providing a choice of chromosome types, Ganesh caters for a variety of possible data types for MOOP's decision vectors. The chromosome consists of the set of genes, each of which stores the value of one component of the vector, and which are usually of the same type.

Ganesh provides the choice of using real-encoded chromosomes, in addition to that of the binary type in which each gene is either a 1 or 0, and furthermore provides a mixed type (for Reals and Integers) which has the novelty of allowing discrete value sets, in which a gene can take one of a set of defined values, whose definition is given as part of the optimisation problem definition.

Because of the availability of these different chromosome types, Ganesh is able to be used on a wide variety of both discrete and continuous multi-objective optimisation problems, where the problem is encoded by an appropriate scheme.

3.2 Ganesh: Framework and algorithm

3.2.1 The GA Algorithm

This section provides an overview of the algorithm and framework capabilities, which are expanded upon where necessary further down in section 3.2.4.

The GA developed in this work was inspired by the NSGA-II algorithm (Deb *et al.*, 2000) and its predecessors including Fonseca and Fleming's (1993) MOGA and that of Goldberg (1989), with some modifications, to: initialisation of population and solution, the non-domination sorting method, the construction of the new generation, the addition of repairable Hard Constraints, the adoption of a plug-in architecture, and of course the self-adaptive aspect, as well as some practical considerations given below.

Soft constraints are implemented, following Fonseca and Fleming (1998), as secondary considerations of Pareto dominance, requiring the constraint definition to return an increasingly negative number indicating the increasing degree of violation, and where 0 (or a positive number) indicates no violation. Any solution violating a constraint is dominated by any that does not, and the degree of violation is used to establish dominance between the constraint breakers.

Internally the GA is constructed to minimize, requiring objective functions that maximize to return a negative number, by the principle of duality (Deb, 2001).

A tournament selection method of degree two is used, polynomial mutation (Deb and Goyal, 1996) is used along with a simulated binary crossover (SBX) (Deb and Agrawal, 1995) for real parameters, and the crossover strategy used in the problem discussed here is uniform for real-encoded crossover, although a multi-locus gene-swap crossover is available also. Self-adaptive crossover requires further consideration and is discussed more fully, below.

This GA provides the novel ability to choose the cardinality of duplicate solutions in each generation, meaning that 0, 1 or many duplicates may be kept, with the default being many, where a duplicate is defined as all corresponding genes (by their location) in both chromosomes having the same alleles (values).

Hard constraints may be either of pre- or post-evaluation types, where pre-evaluation hard constraints are allowed to be repairable, whereas post-evaluation hard constraints must cause discarding of failing solutions. Repair is effected by changing the parameters (the *genes*) until the solution is once again within the constraint. Since repair occurs before evaluation of the solution (the determining of the objective function values), repaired solutions are available in the current generation as normal. Of course hard constraints which rely upon the values of objective functions as part of their violation detection, may only be of the post-evaluation type. Soft constraints may be of either type, with the same proviso, but do not provide a discard option.

Initialisation at both population level and solution level have defaults which are able to be over-ridden by the problem definition, enabling pre-defined data to be included, and alternative distribution functions to be used.

The plug-in architecture of the algorithm enables the optimization problem to be specified separately as a new code module, thus each new optimization problem adds code, rather than requiring changes to the existing code base, and enables the optimization problem to be effectively a parameter of the algorithm.

Figure 3.1 shows a high level view of the algorithm as a flow-chart.

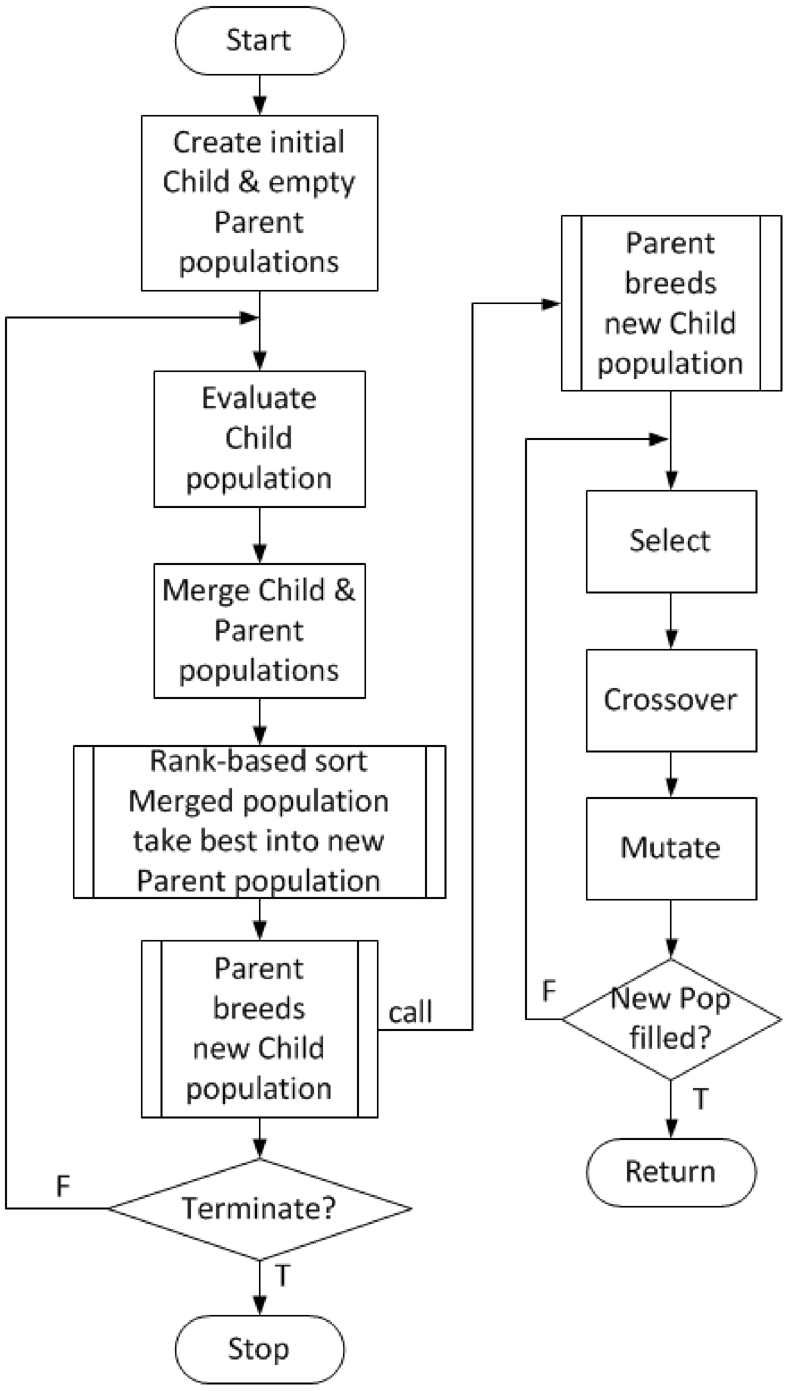

Start
Create initial Child & empty Parent populations
Evaluate Child population
Merge Child & Parent populations
Rank-based sort Merged population take best into new Parent population
Parent breeds new Child population
call
Terminate?
F
T
Stop
Parent breeds new Child population
Select
Crossover
Mutate
New Pop filled?
F
T
Return

FIGURE 3.1: This is a high level view of the Ganesh algorithm expressed as a flow-chart, in which the production of the next generation population is shown as a sub-process for convenience.

3.2.1.1 Simplified non-dominated sorting

The non-domination sorting is amended from NSGA-II to ensure that each solution is compared with every other one once in a simple and efficient manner which is entirely for-loop based, with the number of comparisons being of the same order as that of the *continuously updated* method (Deb, 2001), and the method of updating dominated-by count and dominated-solutions lists are modified accordingly.

In order to establish the order of non-dominated rankings, as shown in the 'Rank-based sort Merged population' box of the flow chart in Figure 3.1, it is necessary to compare the corresponding objective function values of solutions in the population, in order to determine which solutions dominate which others, using the definition of *dominates* given in Table 2.1 ($z^1 \succ z^2$). For objective functions that are computationally expensive to evaluate, the dominance ranking can be time consuming and therefore needs to be efficient.

The non-domination sorting used here differs from Deb's (2001, pp. 36-38) improved *continuously updated* method, by using a nested *for-loop* mechanism as follows, which takes less code and is therefore correspondingly faster and simpler, while of the same order of computational complexity.

A population P of N solutions such as the set $P = \{A, B, C, D, E, F, G\}$ can be compared thus, where *compare* means perform the $z^1 \succ z^2$ *dominance* test (of Table 2.1):

A : B C D E F G : compare A to all the others
B : C D E F G : compare B to all the others, except A
C : D E F G : compare C to all the others, except A, B ...
and so on ...
...then for a population p, where m is the number of objective functions for the given problem , the maximum number of comparisons c to compare every solution

in the population with every other is:

$$c = m \cdot \big(\,(N-1) + (N-2) + (N-3) + \cdots + (N-(N-1)\,)\big)$$
$$\text{and where } n = N-1, \text{ then } c = \frac{m \cdot n \cdot (n+1)}{2}$$

thus the number of comparisons to be made is of the order $O(mN^2)$, and the actual number of comparisons made will on average be the same as Deb's *continuously updated* method.

For the population P having N solutions in which p & q are individual solutions within P numbered from 0 to N-1, the c comparisons above is achieved for the population using loops of the form:

```
for  p in 0 to N-1
    for  q in (p + 1) to N-1
        compare q to p
```

or more formally:

$$\forall p \in \{0, \cdots, n-1\}, \forall q \in \{p+1, \cdots, n-1\} : (q \succ p) \rightarrow q$$

The pseudo-code (PSC) 3.1 below, which relates to the box in the flow-chart of Figure 3.1 entitled *Rank-based sort*, gives more detail of the above and the description of the following two paragraphs. The functions referenced in that PSC are defined in the PSCs that follow it, and note that text preceded by the ▷ symbol are comments.

In order to incorporate the above, this method differs further from Deb's, primarily concerning how each solution builds up its list of the solutions it dominates, and the count of solutions it is dominated by, and these fall simply into place with an object-oriented design, in which a solution *class* encapsulates its own data.

The new generation is produced by pruning one solution at a time from the merged parent and child populations and recalculating the distance/crowding metric each time, giving a more accurate estimate of the best solution to remove with respect to the crowding (and non-dominated ranking) metric (Kukkonen and Deb, 2006). Other algorithmic approaches to the production of the new generation can be used, in a similar way to how alternative chromosome and population initialisers may be specified (Section 3.2.4.10).

Pseudo-code 3.1: Ganesh: Fast Non-Dominated Sort

```
Data: Population: P merge of parent & child Populations
Data: Solution: p,q
Population Function FastNDS( mergedPopulation P )
    N ← popSize
    for i in 0 to N-1 do
        p ← P(i)
        for j in (i+1) to N-1 do
            q ← P(j)
            p.dominanceCompare(q)
            if $p \succ q$ then
                q.dominatedByCount : increment
                p.dominatesList.add(q)
            else if $q \succ p$ then
                p.dominatedByCount : increment
                q.dominatesList.add(p)
        if p.dominatedByCount = 0 then
            Rank1-List.add(p)
    return BuildNewPop(Rank1-List,popSize)
```

Pseudo-code 3.2: Ganesh: dominanceCompare

```
Data: Solution: this
Data: int: result
int Function dominanceCompare( Solution other )
    ▷ Return Domination order, where 1: this Solution dominates,
      -1: other Solution dominates, 0: both non-dominated
    ▷ n.b. assumes MINIMISE
    if $thisSumViolations < 0$ and $othersumViolations < 0$ then
        if $thisSumViolations = othersumViolations$ then
            result = 0
        else if $thisSumViolations < othersumViolations$ then
            result = -1
        else
            result = 1
    else
        if $thisSumViolations < 0$ and $othersumViolations = 0$ then
            result = -1
        else if $thisSumViolations = 0$ and $othersumViolations < 0$ then
            result = 1
        else
            for each OF do
                if $thisVal < otherVal$ then
                    thisDoms = true
                if $thisVal > otherVal$ then
                    otherDoms = true
            if thisDoms and not otherDoms then
                result = 1
            else if not thisDoms and otherDoms then
                result = -1
            else
                result = 0
    return result
```

Pseudo-code 3.3: Ganesh: Build New Population by rank

```
Data: LinkedList: newPopList, ThisFront, NextFront
Data: Solution: p, q
Data: int Rank = 1
Population Function BuildNewPop( Rank1-List, popSize )
    ThisFront ← Rank1-List
    while (ThisFront not empty) and (newPopList.size < popSize) do
        NextFront = new LinkedList
        while ThisFront not empty do
            p ← ThisFront.nextSolution
            p.rank = Rank
            while p.dominatesList not empty do
                q ← p.dominatesList.nextSolution ▷ next solution dominated
                 by p
                q.dominatedByCount : decrement
                if q.dominatedByCount = 0 then
                    NextFront.add(q)                          ▷ for next Rank
        increment Rank
        if newPopList.size < PopSize then
            if numberDuplicates < Integer.MAX_VALUE then
                pruneDuplicates(ThisFront);
            ofCount ← number of OFs
            ▷ Array of LinkedList of Solutions, saving Solutions in
              sorted order, for each OF, for this Front.
            OFOrgsSorted = new List[ofCount]
            setCrowdingMetric( ThisFront, newPopList.size(), popSize, ofCount,
             OFOrgsSorted )
            newPopList.add( ThisFront )
        ThisFront = NextFront
    ▷ Create the new empty parent population and add to it most fit
      solutions first
    Create new Population: NewPop
    sort(newPopList) ▷ in descending order of solution rank & distance
     (using RankDistComp)
    for each Solution in sorted newPopList : i do
        if new population size < required size then
            newPop.add(newPopList.get(i))
    return newPop
```

Pseudo-code 3.4: Ganesh: Set Crowding Metric

Data: List of Solutions: ThisFront
Data: Solution: thisSoln,prevSoln,nextSoln

```
Void Function setCrowdingMetric( ThisFront, popListSize(), popSize, ofCount,
 OFOrgsSorted )
    ▷ Calculate a crowding/distance metric - (a cuboid of the
      surrounding two neighbours in each OF dimension).
    for each Solution in ThisFront do
        Solution.metric ← 0
    for each OF: i do
        sort(ThisFront)              ▷ into OF order (min,max), for this OF
        OFOrgsSorted[i] ← ThisFront                ▷ Save the sorted order
        ▷ Set boundary points' metrics to Infinity so they are always
          included
        ThisFront.Solution.metric[0] ← MAXDIST
        ThisFront.Solution.metric[N-1] ← MAXDIST
        ofMin = ThisFront.SolutionOFvalue(i)
        ofMax = ThisFront.SolutionOFvalue(i)
        for each Solution (except 1st and last) in ThisFront : j do
            thisSoln ← ThisFront[j]
            prevSoln ← ThisFront[j-1]
            nextSoln ← ThisFront[j+1]
            dist ← thisSoln.metric
            dist = dist + (nextSoln.OFValue(i) - prevSoln.OFValue(i) ) / (ofMax
             - ofMin)
            thisSoln.metric ← dist
```

Pseudo-code 3.5: Ganesh: RankDistComp function, to sort by

```
Data: int result
Data: Solution: s1, s2
int Function RankDistComp(Solution s1, Solution s2)
    ▷ Compare solutions by their OFs and crowding metric
    ▷ Returns a negative integer, zero, or a positive integer as the
      first argument is less than, equal to, or greater than the
      second.  n.b.  assumes Minimising
    result ← 0
    if s1.rank < s2.rank then
        result = +1;
    else if s1.rank > s2.rank then
        result ← -1
    else if s1.rank = s2.rank then
        if s1.dist < s2.dist then
            result ← -1;
        else if s1.dist > s2.dist then
            result ← +1;
        else
            result ← 0;
    return result
```

3.2.2 Self-adaptation

Unlike Bäck *et al.* (2000) (who used an initial random mutation rate), Ganesh allows mutation and crossover rates to be specified for the initial population, or to be set to random values in a uniform distribution, or to default to certain values. The default mutation rate of each solution would be set to 1/n where n is the number of variables of the objective functions (OFs), and the default crossover rate would be 0.6, both as probability of occurrence. This GA also allows alternative initializers to be written and specified per problem, allowing for different probability distributions, such as the uniform or Gaussian, however this work uses the uniform distribution. Initial random values stand a good chance of being poor choices, however, and are not used in this work.

Similarly to Bäck (1992) and Smith and Fogarty (1996) (a steady-state GA), mutation first occurs to the gene encoding the mutation rate and then the new mutation rate is applied to the *main* genome, but unlike the previous studies, this is based on a generational GA, that is one in which the entire population is in theory able to be replaced by fitter solutions, and for which the variables, and operator parameters, are encoded as real numbers in the genes. In this work, extra self-adaptation is added in the form of recombination (crossover), and the rate is encoded in the chromosome.

The GA control parameters undergoing self-adaptation are the mutation probability pM (per gene) and the crossover probability pC (per chromosome), and also the associated polynomial distribution indices, (Deb and Goyal, 1996) & (Deb and Agrawal, 1995), for each, ηM and ηC respectively, which are all real values. Each solution has a chromosome encoding its objective function parameters and its control parameters. Mutation occurs to all of the parameters including the control ones and their indices, but mutation occurs first to the control ones at the current rate of mutation, and then the *main* ones using the newly mutated values.

A novel operator is defined here, being a self-adaptive uniform blend crossover (SAUBC) of real-encoded chromosomes. The uniform crossover specifies that each gene has a 50% chance of crossing over if the chromosome is to undergo crossover at all, and the probability of chromosome crossover occurring is given by pC. When a real gene is selected for recombination, it is blended with its corresponding allele at the same location of the other parent chromosome, using the SBX operator (Deb and Agrawal, 1995). The uniform crossover was chosen in preference to an n-locus crossover, since it has been shown (De Jong and Spears, 1992) to maintain performance independently of the distribution of allele values, and because its disruptive effect is outweighed by its 'productivity', which means its ability to increase diversity in the population.

However, since crossover occurs between chromosomes but each chromosome has its own pC, the pC to be used is chosen stochastically at 50% probability from either of the parent chromosomes selected for breeding, and the ηC is taken from the same chromosome. The ηC value is then used in the crossover of the respective controls from each chromosome (pC, ηC, pM and ηM) and the *main* chromosome genes, with the new control values being written to the recombined chromosomes, as shown in Figure 3.2. This crossover operator strategy can also be applied with the real gene components of the mixed chromosome that Ganesh also provides.

The MOOEA of this work then, differs from the works mentioned above since it uses novel self-adaptation mechanisms in a real-encoded multi-objective generational GA in which rate and degree parameters are held within each solution (chromosome). The MOOEA of this work uses its own mutation and crossover operators to self-adapt its control parameters in the same way that they cause self-adaptation in the decision vector (the *main* genes).

The following provides more information about the design aspects of the self-adaption used by Ganesh. *Chromosome* is an abstract class from which all other Chromosome types are derived, with the concrete classes[2] *BinaryChromosome*, *RealChromosome* and *MixedChromosome* at the next level down in the class hierarchy, and the self-adaptive variants are derived from those in turn. Thus self-adaptation is an option, rather than an enforced behaviour, as the problem class designer can choose from any of the concrete chromosome classes to use for the

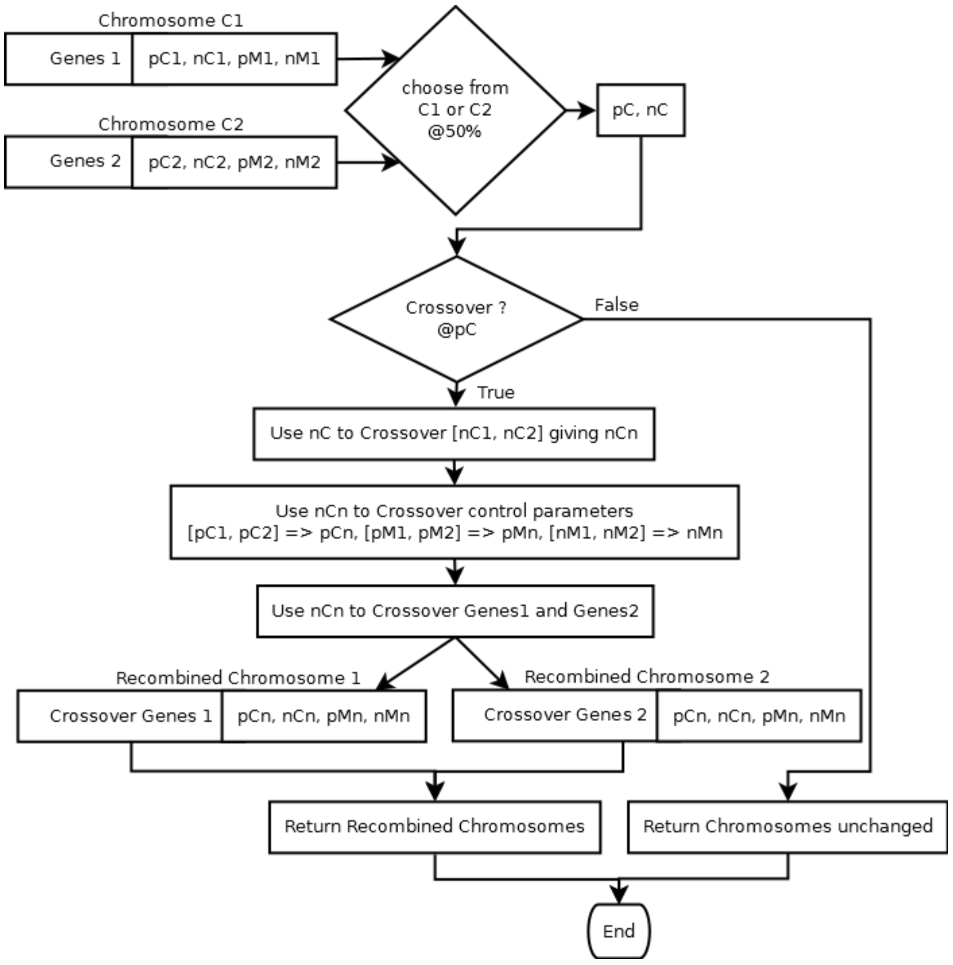

Chromosome C1
Genes 1
pC1, nC1, pM1, nM1
Chromosome C2
Genes 2
pC2, nC2, pM2, nM2
choose from C1 or C2 @50%
pC, nC
Crossover ? @pC
False
True
Use nC to Crossover [nC1, nC2] giving nCn
Use nCn to Crossover control parameters [pC1, pC2] => pCn, [pM1, pM2] => pMn, [nM1, nM2] => nMn
Use nCn to Crossover Genes1 and Genes2
Recombined Chromosome 1
Crossover Genes 1
pCn, nCn, pMn, nMn
Recombined Chromosome 2
Crossover Genes 2
pCn, nCn, pMn, nMn
Return Recombined Chromosomes
Return Chromosomes unchanged
End

FIGURE 3.2: Flow-chart explaining the novel self-adaptive uniform blend crossover operator (SAUBC).

decision vector. The existing self-adaptive classes use one variable for each control aspect, thus there is one crossover rate and one mutation per chromosome and the self-adaptation is therefore similar to the *isotropic* type of Evolution Strategy (Deb, 2001), except that here there is no rule to determine how these control rates change: they are purely driven by stochastic processes undergoing artificial (in contrast to natural) selection, whereas the ES has a logarithmic update rule. The perturbation factors, ηM and ηC, which are the mutation and crossover polynomial distribution indices respectively, of real number *genes*, are also assigned one of each per chromosome, and can be thought of as further tuning controls.

[2]A concrete class can have objects created from it, unlike abstract classes which cannot.

3.2.3 Framework and architecture

This section describes how a given optimisation problem is defined within the Ganesh framework and its architecture, how the novelty of the architecture allows the Ganesh framework to be independent of a specific optimisation problem that it would be used to execute, and how optimisation problems are able to be tailored with the various overrideable[3] options that Ganesh facilitates and provides. Ganesh here identifies both the framework and the optimisation algorithm which it runs.

A little terminology from the object-oriented methods of analysis and software design (Rumbaugh *et al.*, 1991) is used below, albeit kept to the minimum, and is briefly explained here: A *class* is a description of some concept from the subject domain, is typically a noun, and consists of both data and the subroutines that work with the data, and is in effect an abstract data type. An *object* is a particular identified occurrence of a class. The subroutines are termed *methods* and represent behaviours from the subject domain and are typically verbs. In Java, the methods are termed *functions*, although they differ from mathematical functions in that they can have side-effects. For example, Ganesh has a *Chromosome* class which represents the decision vector of a solution (using the biological analogy) and has methods *crossover* and *mutate* which change the Chromosome in certain ways, as explained previously. A given candidate solution in the population of solutions is represented by an *object* of type *Chromosome*.

Here, the class used to code the optimisation problem under consideration, is termed the *problem class* and it is this, along with any other helper classes specific to this problem, that constitute the plugin, which is dynamically loaded by Ganesh at run-time to execute this optimisation.

Figure 3.3 is a Unified Modeling [*sic*] Language (UML) (The Object Management Group, 2014) package diagram, which shows that plugins are dependent upon the Ganesh package but not vice-versa. This indicates that some classes

[3]Overrideable in object-oriented analysis and design means using a locally (in a derived class) defined artefact such as a method, that takes precedence over its parent class definition.

used in Ganesh are available to be used by plugins, and indeed it is necessary to do so, and Figure 3.4 provides more detail. Ganesh provides a point at which to call the objective functions (OF) defined in the plugin, and provides the necessary classes, as an API, for the problem to define the OFs in the plugin, along with other necessary components of the problem definition.

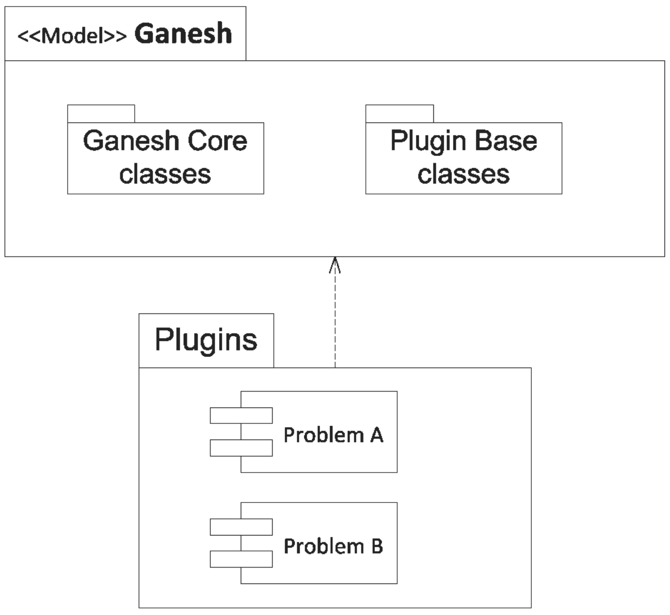


FIGURE 3.3: UML showing the dependency of Plugins upon Ganesh.

Figure 3.4 is a UML diagram that shows the classes involved in defining and calling an optimisation problem that has been encoded in a plug-in. Every optimisation problem must be defined as a derived class (sub-class) of Experiment, which is in turn derived from Plugin and which is defined in the Ganesh package, and it is this that enables optimisation problems to be compiled separately and independently yet to be dynamically loaded and run by Ganesh. It should be borne in mind that it is not the intention to specify every class within Ganesh, whether by UML or other means, but to merely highlight the most salient ones in order to explain key points, where necessary.

The plugin, in defining an optimisation problem, must as a minimum define the data type of the decision vector (such as a vector of, say, 12 floating point (*Real*) parameters), by choosing a *Chromosome* of the desired type and a list of the desired number of *Variables*, and also define at least one objective function. A multi-objective problem is created by the user defining many OFs, each of which is accomplished by creating a new nested class, derived from the *ObjFunc* class, within the problem class. Ganesh provides a default initialiser for the population and a default initialiser for each *Chromosome* type, thus the problem is fully defined. The problem class may then be compiled into an independent library file, named GA-Plugins.*jar*, which may contain as many different optimisation problems as desired.

In this way, the optimisation problem is created completely separately and without needing to change any of the source code of Ganesh. In order to execute a particular optimisation problem, Ganesh is simply called from the command line with the name of the problem class as a parameter. The independent file containing the plugin may contain as many different optimisation problems as is desired, enabling easy testing or switching between problems.

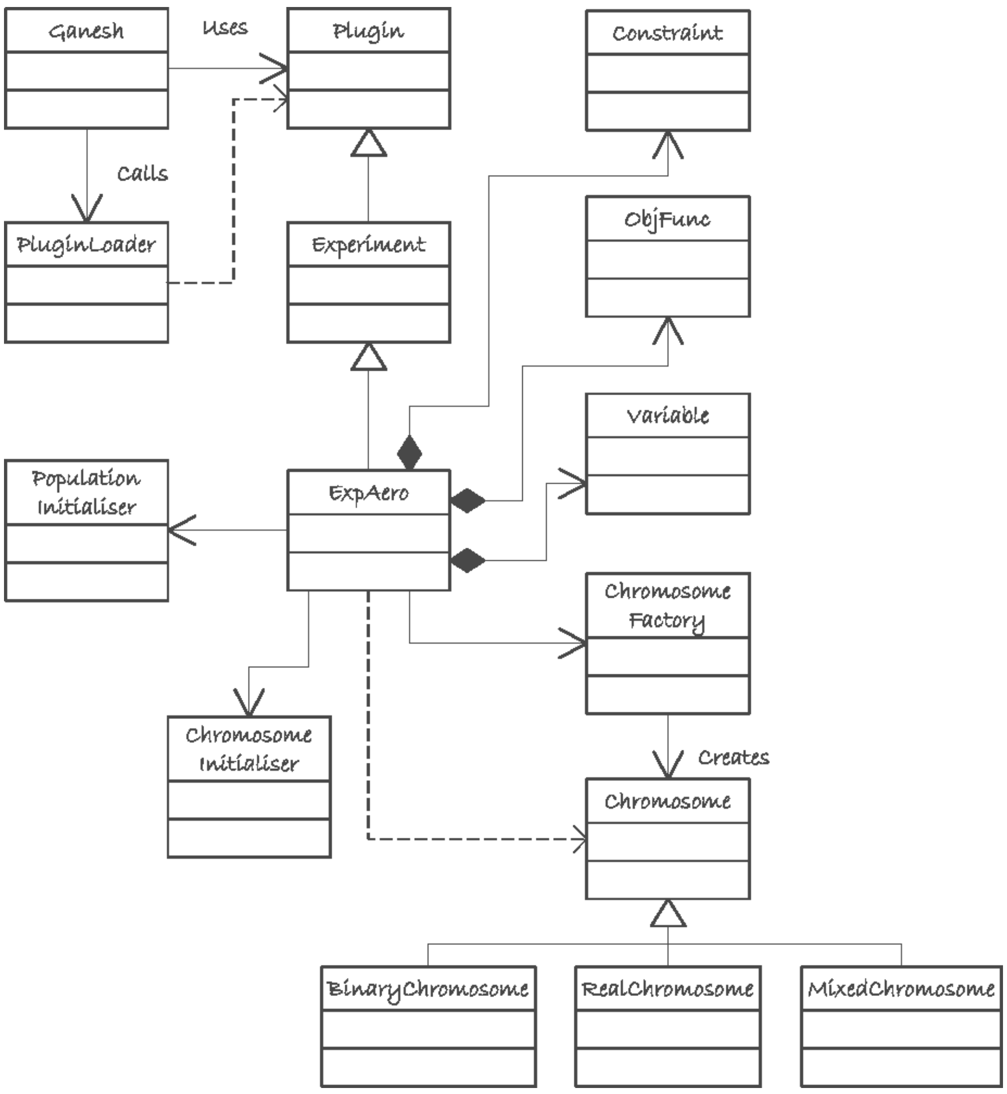


FIGURE 3.4: A UML class diagram showing how plugins are integrated with Ganesh. The ExpAero class is an example of any optimisation problem to be used, and implements an airfoil optimisation. All problem classes must be derived from the Experiment abstract class, and are dependent upon the other associated classes shown, which are available in the Ganesh library (.jar).

3.2.4 Algorithm characteristics, benefits and novelty

In this section, further details of the framework and algorithm are given, the benefits thereof and the novelty of the work. It is intended that Ganesh be easy to understand and use with practical benefits.

While Ganesh may not offer many of the facilities of other frameworks given in the literature review, it does provide the following very specific benefits which all or most of the others do not do in the same way. Its relative simplicity can be a benefit in getting started.

3.2.4.1 Synopsis

Here the benefits are listed briefly and where appropriate further details are given in subsequent sections:

- Self-adaptivity - see 3.2.4.2
- Crossover mechanisms - see 3.2.4.3
- Chromosome types - see 3.2.4.4
- Plug-in experiment code - see 3.2.4.5
- Using external software as (supplier of) objective functions - see 3.2.4.6
- Callable from external software - see 3.2.4.7
- Duplicate solutions control - see 3.2.4.8
- Constraints - see 3.2.4.9
- Chromosome initialisers - see 3.2.4.10
- Population initialisers - see 3.2.4.11
- Operator configuration - see 3.2.4.12
- Problem-specific parameters - see 3.2.4.13
- Resume from previous run - see 3.2.4.14
- Command line run-time parameters - see 3.2.4.15
- User manual - Ganesh also comes with a user manual, in addition to the automatically produced Java class documentation.

3.2.4.2 Self-adaptivity

Detail about the novel self-adaptive features of Ganesh has been given in the preceding section 3.2.2. The benefit of this is to improve performance in some cases, as seen in the results of sections 3.5 and 4.5.2.

3.2.4.3 Crossover mechanisms

The GA implements novel crossover mechanisms for self-adaptivity which recombines the control parameters for mutation and crossover rates, as well as each of their perturbation control parameters in addition to the data (*main*) parameters. The benefits of these are associated with the benefits of self-adaptivity which is shown to improve performance in some cases, as seen in the results of sections 3.5 and 4.5.2.
More detail on this has been given in section 3.2.2.

3.2.4.4 Chromosome types

A novel mixed chromosome is provided that it is able not only to have genes of different data types, but also of discrete value sets in which a gene can take one of a set of predefined values, whose definition is given as part of the optimisation problem definition. Other types available are real-encoded chromosomes, binary, and integer (as the whole or part of the mixed type). Real genes are of Java's *Double* type, which is a 64-bit floating type (IEEE 754) (Goldberg, 1991). The integer gene can be of 8, 16, 32 or 64 bits.
The benefit of this is that the optimisation can be easily and simply defined for problems in which a sub-set of the decision vector must be limited to certain values, including *real* values.

3.2.4.5 Plug-in experiment code

The architecture of Ganesh enables the complete separation of independent problem definition code from the Ganesh framework and algorithm code, and detail of this is given in section 3.2.3. Similarly, different optimisation algorithms can be specified as plugins too, e.g. Ganesh or Moorand as seen in section 3.4.
This provides the following benefits:

- Minimises the knowledge of Ganesh needed to define new problems
- Enables independent version control of Ganesh and of optimisation problem code, giving clarity and traceability for repeatability of experiments.
- The optimisation problem is specified at run-time as a parameter to Ganesh, enabling several problems or different configurations of the same problem to be worked on without needing to modify Ganesh.
- The optimisation problem code can be distributed to other researchers independently of Ganesh
- The optimisation problem definition is clearer by its separation from Ganesh.

3.2.4.6 Using external software as (supplier of) objective functions

The plugins for the optimisation problems defined in both Chapters 4 and 5, use software external (FFD/Xfoil and Plexos, respectively) to Ganesh to act as surrogates for objective functions. These software are executed as independent sub-processes of Ganesh, using Ganesh's multi-threading capability to manage the effectively parallel I/O which would otherwise cause deadlock.
The benefit of this is the ability to have performed these optimisation problems at all, and the ability to use other external software if needed.

3.2.4.7 Callable from external software

Ganesh is also directly callable from other Java codes and codes in other languages which can interface to Java, which alongside the ability to define the optimisation problem at run-time, enables Ganesh to be incorporated into other workflows. An example of this benefit has been given in section 3.2.4.13 since both capabilities are involved.

3.2.4.8 Duplicate solutions control

This GA provides the novel ability to choose the cardinality of duplicate solutions in each generation, meaning that 0, 1 or many duplicates may be kept, with the default being many. Zero duplicates means one solution having no duplicates, and so on, where a duplicate is defined as all corresponding genes (by their location) in both chromosomes having the same alleles (values). The ability to control the existence of duplicates is achieved here through the use of a linked hash map data structure in which the map of <*key*, *value*>, has the *key* being the chromosome and the *value* being both a count of and list of individuals having the same genes, up to the permitted number of duplicates. The problem of duplicates may arise as convergence approaches due to elitism, even in problems having only real-encoded decision variables.

The benefit of this is to ameliorate premature convergence since each surviving duplicate candidate solution is effectively preventing a hypothertical better solution from being present.

3.2.4.9 Constraints

Constraints, either *soft* or *hard* can be added to the problem by creating sub-classes of the appropriate type, such as *HardConstraint*. A *soft* constraint is one that can be relaxed, by allowing it to be exceeded but acquiring a penalty associated with it proportional to the excess, which is reflected in the fitness of the solution when solutions are ranked. A *hard* constraint is one which must be adhered to, so solutions which break it must either be repaired or removed from the population.

Another novelty of this work is that constraints can be designated as either pre-evaluation or post-evaluation (of the objective functions), enabling Ganesh to consider just the decision vector (pre-evaluation) or the objective function value too (or instead) (post-evaluation). This designation is given to enforce proper consideration of hard constraints, since those tested pre-evaluation that fail, allow the solution to be repaired, by changing the decision vector until the constraint is no longer broken, or removed from the population. The way this is performed is problem-specific and is defined by the problem class, in the way the designer wishes it to be. Hard constraints tested post-evaluation that fail cannot be repaired and therefore can only be removed from the population.

An example of repair of a solution's design vector that fails a constraint check, is given at the end of section 5.4.

The above are achieved by methods of the Constraint family of classes (pre & post evaluation types, and soft and hard) defined within the Ganesh package. The problem class implements its constraint classes as derivations of the Ganesh ones and overrides the methods that it needs to, which is the *checkViolation* (mandatory) and *repair* and *remove* (optional depending on the factors mentioned).

The benefits of this approach is that the problem definition has a clear interface to either the hard or soft constraint and makes it obvious which is in place. Early repair of a decision vector helps improve performance since the evaluation of an unfeasible solution does not take place, and similarly for the case of a rejected solution.

3.2.4.10 Chromosome Initialisers

Each chromosome type has its own default initialiser, that defines how its genes are assigned appropriate values when a solution is created, and the default is that each gene is randomly given a value within its permitted range, assuming a uniform distribution. The *Range* is a class that the problem class designer uses to specify the range of each gene (each of which may have different ranges). However, the user may also use their own initialising function by writing one in the problem class which overrides the default. The method to achieve this is provided as part of the *Experiment* class from which all plugins must be derived. The optimisation of Chapter 5 uses this ability to create its own chromosome initialiser as it has two independent sets of genes that need to be initialised in different ways, and in one set of genes it uses a Gaussian rather than uniform distribution of random numbers.

The benefit of this ability is the flexibility to choose how an initial population can be constructed, and an appropriately distributed initial population can improve performance.

3.2.4.11 Population Initialisers

The Experiment class comes with a default population initialiser that creates the specified number of solutions in the population, using the chromosome initialiser that is active, as detailed in section 3.2.4.10. The problem class may override the default population initialiser by creating its own one. The optimisation of Chapter 5 uses this ability to ensure that there is at least one solution in the initial population that has the datum design.

The benefit of this is as stated in 3.2.4.10, and also that specific decision vectors can be constructed, such as a datum design, to enable optimisation to take place from a known starting point.

3.2.4.12 Operator Configuration

Each chromosome type available in Ganesh provides its own default crossover and mutation operator. For the binary chromosome there is a simple one-locus crossover, for the real chromosome there is a uniform simulated binary (SBX) and for the mixed there is a uniform with SBX for the real genes and gene swap with the other gene types. There is also a multi-locus gene swap crossover operator available for the real chromosome, in which the number of loci is configurable up to the maximum length of the chromosome. More crossover operators are easily added by simply creating a new class for them and using the general-purpose *cross-Genes* method, which works for 1 or many loci, with the appropriate parameters. Mutation operators are similarly dealt with.

When a crossover or mutation operator changes a given gene, it is possible that the resulting allele value will not be within the accepted range (which is defined by the problem class), and when this occurs, the operator will automatically repair the gene by assigning the nearest range boundary value.

As in the case of the initialisers, the problem class may then choose whether or not to override the default operators with an existing one of the correct type for the chosen chromosome.

The benefit here is one of flexibility and extendability.

3.2.4.13 Problem-specific parameters

Ganesh takes a number of parameters at run-time to control its behaviour (see Appendix B.1.1), but these are all concerned with the general operation of the algorithm and apply to any problem that Ganesh might run.

Often, optimisation problems will have behaviours or configuration issues that the user might wish to parameterise, rather than hard-coding them as part of the problem class, so the framework makes it possible for the problem class to define its own initialisation using an optional separate parameter file. The parameter file to be used is able to be specified as a run-time parameter along with the problem plugin and the other standard run-time parameters. An associated utility provides a simple way of reading the parameters from the file, in such a way that certain basic tests of the expected data against the actual data are carried out (data type and number of parameters), and errors arising are trapped and reported automatically, thus simplifying the specification of the problem-specific parameter-reading code. The parameter reader also accommodates nesting of repeated simple parameters, for additional flexibility.

This is achieved by the *Experiment* class which provides a default empty *initialise* method which plugins may optionally override if required, to read and initialise themselves with values from the parameter file.

The benefits of this ability can be seen in the following examples:
This ability was used to concurrently run the 20 samples per range of the Airfoil optimisation, to gather statistics for the analysis of its performance, as detailed in Chapter 4. Without this ability, concurrent running would not be possible, due to the interface imposed on Ganesh by the FFD/XFoil codes, which would increase greatly the actual elapsed time needed to perform these runs, serially.

The optimisations of the power network as described in chapter 5, use this ability to specify both configuration issues, such as locations of the Plexos model and H2 database, and to specify limits to be used in the constraints, such as the total number of DG units permitted. This enables changes to the run-time configuration of the problem and environment to be made without impacting the code under configuration management, that defines the problem, thus simplifying version control by obviating the need to define many versions of the problem that differ only in minor details.

Ganesh is already being used by a number of other research group students and collaborators (T.Kipouros - private communication). Its ability to be called from other software, and its nested repeating parameter facility, are enabling stochastic variation of the length of their optimisation problem's decision vector, under repeated invocations.

3.2.4.14 Resume from previous run

Real-world optimisation problems such as those presented in Chapters 4 & 5, tend to have objective functions that are computationally expensive to evaluate and which are therefore time-consuming. In these cases, an optimisation problem can take days, weeks or even months to run, which increases the chance that they may be interrupted by some unexpected external event such as a power cut, risking the loss of data, progress made and much time.

Ganesh obviates these problems by providing the ability to resume from the point a previous run was unexpectedly (or even, expectedly) terminated. This facility also provides a simple method of reusing a population of solutions used elsewhere (where the population size could also be just one), for example a starting population of another run used to start a new run too, for comparison. This is possible in two ways that differ by whether the default text log or optional binary log was used in the other run.

The text log by default contains, in text form, the decision vector, objective function values and where applicable, the self-adaptive control parameters, for each solution in the population, for each entire generation produced and evaluated (with respect to the objective function values and non-dominated sorting). Since only evaluated generations are held in the text log, a generation currently under evaluation is lost if interruption occurs. Nevertheless, it may be sufficiently acceptable to be able to resume from the last complete generation. The text log is cumulative and contains the entire history of the run to date, and to completion if it runs fully.

Alternatively, the user may specify, using appropriate parameters (Appendix B.1.1), to use a binary log which captures the exact state of the generation currently under evaluation (only), so that should interruption occur, only the data of the objective function of the solution currently under evaluation is lost, thus the file very likely contains partially or not evaluated solutions as well as evaluated ones.

Ganesh enables either the text log or the binary log or both to be used, moreover it enables Ganesh to retain the last generation only, as a binary log. The latter option enables Ganesh to execute optimisation problems which have objective functions that are quick to evaluate but require vary large numbers of generations to be run, which may otherwise create undesirably large text log files.

The resume is performed by running Ganesh again, specifying which logs to obtain details from, so that the first generation created in the new output log is the old one read, and new generations are created based on those solutions, as normal. Thus having resuming from a text log, when considering the data in the new output, the first generation (numbered from 0) should be ignored as it is a repetition of the old last one. Similarly, when resuming from a binary log, the new generation 0 should also be ignored as some of its solutions will have missing objective function values.

3.2.4.15 Command line run-time parameters

The program implementing the framework and algorithm takes many run-time parameters from the command line, which specify the problem to be run (from the plugins) and how the run should be performed, taking such things as population size, number of generations and so on as given in the appendices. Run-time parameters given on the command line take precedence of those coded (if indeed they are), giving flexibility and control. Being not a GUI means it can be run concurrently from script files, e.g. when undertaking statistical sample trials. As mentioned in 3.2.4.7, it is also callable from other software directly, since the interface allows the same run-time parameters to be passed.

3.2.4.16 Conclusion

The benefit to the author of this dissertation of producing the framework of Ganesh, rather than using another existing application or framework, should not be underestimated. Being the producer/owner had the following impacts:

- Intimate knowledge of the whole body of code, enabling flexibility of work.

- Freedom to choose how and with what to perform configuration management (version control).
- Flexibility of design, since it is a characteristic of research that one does not necessarily know in which precise direction one is heading.
- Minimising concerns of licence terms and conditions of other potential codes.
- Ability to change the architecture as required. In the case of the power network optimisation (in chapter 5), I could share code between a stand-alone application and Ganesh. This enabled the deconstruction of the Plexos internal data model, in turn enabling the ability to share line transmission characteristics and feedback line capacities into Plexos at run-time via XML, since the relevant XML nodes of Plexos could be stored in memory. This was a reasonably complex undertaking, since the Plexos XML was essentially an alternative representation of an in-memory SQL-type database, and Plexos was not designed to have feedback of transmission line data.

3.3 Benchmark test problems and results

This section sets out some standard benchmarks from the literature (Deb *et al.*, 2000). These test problems were primarily used to test that Ganesh was functioning correctly, but also to show that Ganesh was performing well. The airfoil optimisation problem defined in Chapter 4, is the focus of performance testing since it is a real world problem with complex objective function definitions and evaluation times of lengthy duration.

3.3.1 Problem definitions

The problems are defined in the following table (Table 3.1), where n is the number of variables used by the objective functions (OFs) to be optimised. All the OFs are to be minimised, and the GA is to be run with the same parameter settings as in the literature, to act as a benchmark, and are as follows: 25,000 function evaluations, which means for a population size of 100 the number of generations is to be 250.

Crossover probability = 0.9, Mutation probability = $1/n$ or $1/l$ (where n is as given in the table for real-encoded chromosomes, and l is the number of bits in the binary chromosome). The real-encoded problems have parameters $\eta C = 20$ and $\eta M = 20$, which are the polynomial probability distribution indices for the real crossover and mutation operators, respectively. Binary-encoded versions of the problem use 30-bit variables.

Table 3.1: Benchmark test optimisation problem definitions, showing the number of variables n and the range of values the variables can take.

Problem	n	Variable Ranges	Objective Functions
SCH	1	$[-10^3, 10^3]$	$f_1(x) = x^2$
			$f_2(x) = (x-2)^2$

FON	3	$[-4,4]$	$f_1(x) = 1 - exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2)$
			$f_2(x) = 1 - exp(-\sum_{i=1}^{3})(x_i + \frac{1}{\sqrt{3}})^2)$
POL	2	$[-\pi,\pi]$	$f_1(x) = [1 + (A_1 - B_1)^2 + (A_2 - B_2)^2]$
			$f_2(x) = [(x_1+3)^2 + (x_2+1)^2]$
			$A_1 = 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2$
			$A_2 = 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2$
			$B_1 = 0.5\sin x_1 - 2\cos x_1 + \sin x_2 - 1.5\cos x_2$
			$B_2 = 1.5\sin x_1 - \cos x_1 + 2\sin x_2 - 0.5\cos x_2$
KUR	3	$[-5,5]$	$f_1(x) = \sum_{i=1}^{n-1}\Big(-10 \cdot exp\big(-0.2\sqrt{(x_i^2 + x_{i+1}^2)}\big)\Big)$
			$f_2(x) = \sum_{i=1}^{n}\big(\lvert x_i\rvert^{0.8} + 5\sin x_i^3\big)$
ZDT1	30	$[0,1]$	$f_1(x) = x_1$
			$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}}]$
			$g(x) = 1 + \frac{9 \cdot (\sum_{i=2}^{n} x_i)}{(n-1)}$
ZDT2	30	$[0,1]$	$f_1(x) = x_1$
			$f_2(x) = g(x)[1 - (\frac{x_1}{g(x)})^2]$
			$g(x) = 1 + \frac{9 \cdot (\sum_{i=2}^{n} x_i)}{(n-1)}$
ZDT3	30	$[0,1]$	$f_1(x) = x_1$
			$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)}\sin(10\pi x_1)]$
			$g(x) = 1 + \frac{9 \cdot (\sum_{i=2}^{n} x_i)}{(n-1)}$
ZDT4	10	$x_1 \in [0,1]$	$f_1(x) = x_1$
		$x_i \in [-5,5]$	$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)}\sin(10\pi x_1)]$
		$i = 2, \cdots, n$	$g(x) = 1 + \frac{9 \cdot (\sum_{i=2}^{n} x_i)}{(n-1)}$
ZDT5	11	$x_i \in [0,1]$	$f_1(x) = 1 + u(x_1)$

			$f_2(x) = g(x)\left[1 - \sqrt{\frac{x_1}{g(x)}}\right]$
			$g(x) = \sum_{i=2}^{11} v(u(x_i))$
			$v(u(x_i)) = \begin{cases} 2 + u(x_i), & if u(x_i) < 5; \\ 1, & if u(x_i) = 5. \end{cases}$
			Where this is a binary only problem, $u(x_i)$ is the number of 1s in the x_i bit vector and x_1 has 30 bits
ZDT6	10	$[0, 1]$	$f_1(x) = 1 - exp(-4x_1)\sin^6(6\pi x_1)$
			$f_2(x) = g(x)\left[1 - \left(\frac{f_1(x)}{g(x)}\right)^2\right]$
			$g(x) = 1 + 9\left[\frac{\left(\sum_{i=2}^{n} x_i\right)}{(n-1)}\right]^{0.25}$
CONSTR	10	$x_1 \in [0.1, 1.0]$	$f_1(x) = x_1$
		$x_2 \in [0, 5]$	$f_2(x) = \frac{(1 + x_2)}{x_1}$
			with the following constraints:
			$g_1(x) = x_2 + 9x_1 \geq 6$
			$g_2(x) = -x_2 + 9x_1 \geq 1$

3.3.2 Benchmark test results

This section provides the results of the test defined in the previous section (Section 3.3.1), to show how the MOOEA, Ganesh, performs on the set of standard optimization problems from the literature, and in particular those that the NSGA-II algorithm (Deb *et al.*, 2000) was originally tested with. These act as a benchmark to show that this GA implementation is performing correctly, in the functional sense, and well, in the sense of performance, which means convergence.

The following figures, Figure 3.5 to Figure 3.16, show graphs of the solutions found in the last generation, the 250th, of a sample run of each problem, and all solutions are non-dominated. Unless otherwise stated in a figure caption, the results are for a real-encoded version of the problem using the *prune-and-recalculate* method for the distance metric and hence ranking and selection. The results are depicted as two-dimensional scatter plots in which each dot represents at least one non-dominated solution, and in which objective function $f_1(x)$ (OF1) is shown on the x-axis and $f_2(x)$ (OF2) on the y-axis, both of which are being minimised.

The test results obtained here were compared (reading values from graphs) with the results in the literature to establish that the GA is performing correctly and performs at least as well as the convergence obtained by the benchmark, although of course, the results will not be exactly the same as the benchmark since the optimisation process here is a stochastic one (as is the benchmark process).

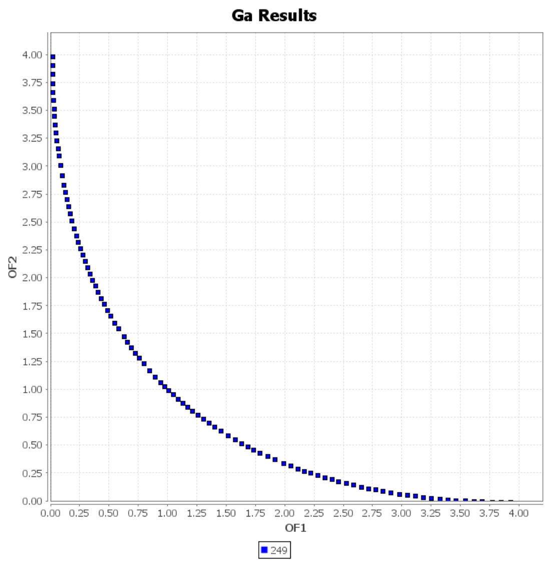


FIGURE 3.5: Benchmark test SCH, Schaffer's F1 function. Binary encoded.

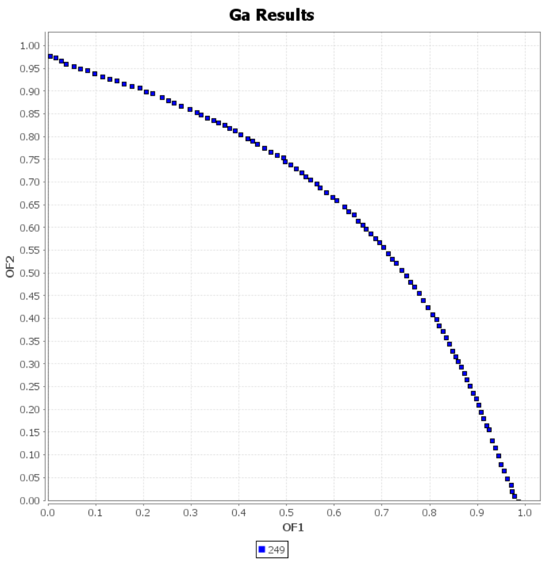


FIGURE 3.6: Benchmark test FON, Fonseca & Fleming's function.

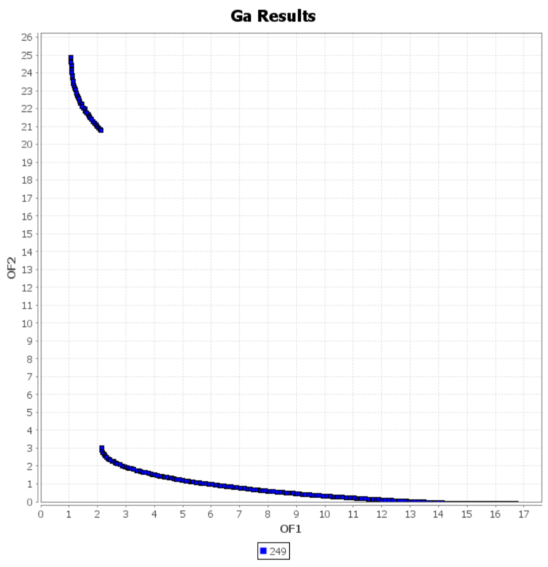


FIGURE 3.7: The benchmark test POL, Poloni's discontinuous function.

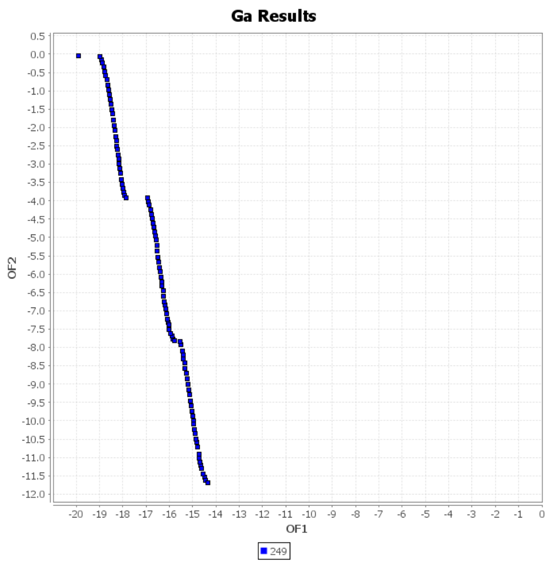


FIGURE 3.8: The benchmark test KUR, being Kursawe's discontinuous function.

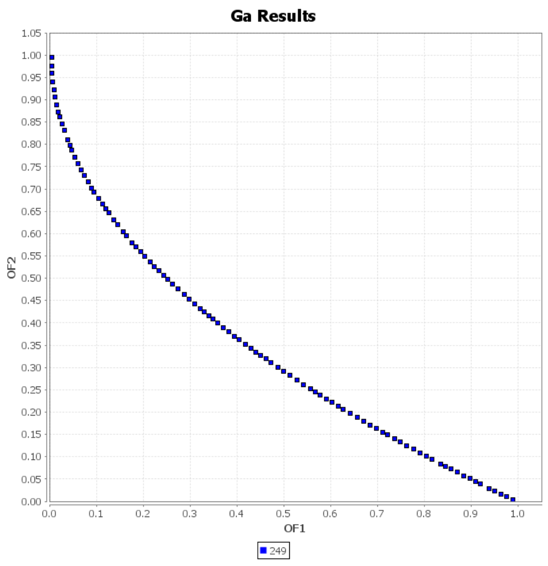


FIGURE 3.9: The benchmark test ZDT1.

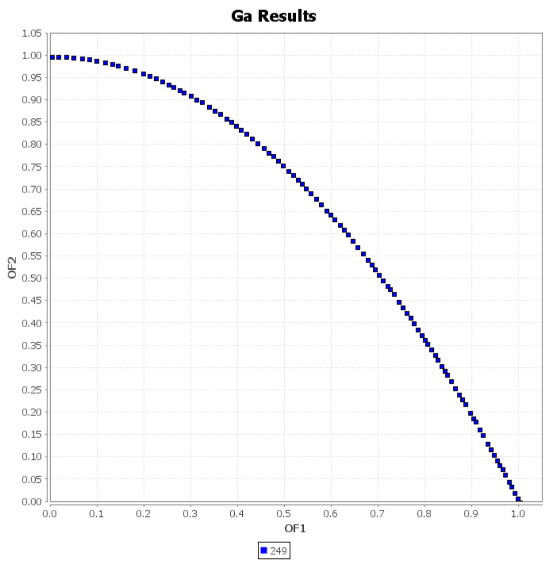


FIGURE 3.10: The benchmark test ZDT2.

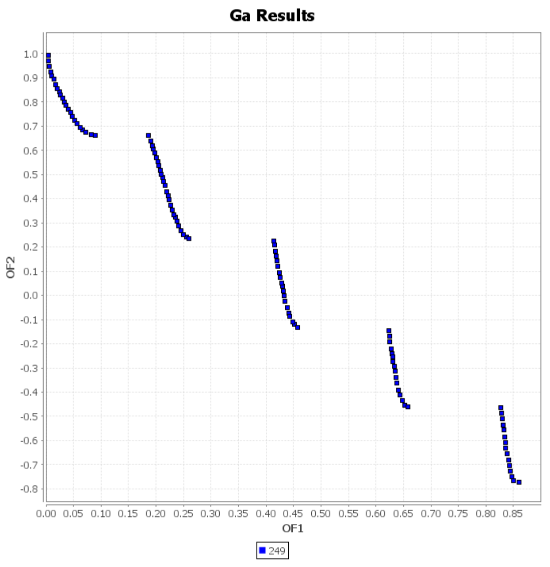


FIGURE 3.11: The benchmark test ZDT3 discontinuous.

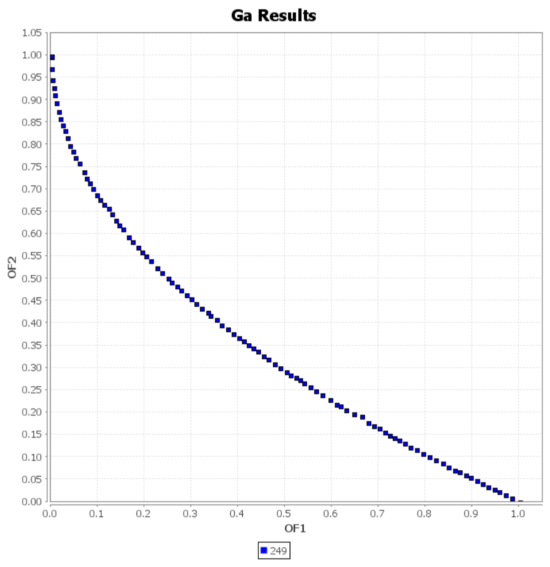


FIGURE 3.12: The benchmark test ZDT4.

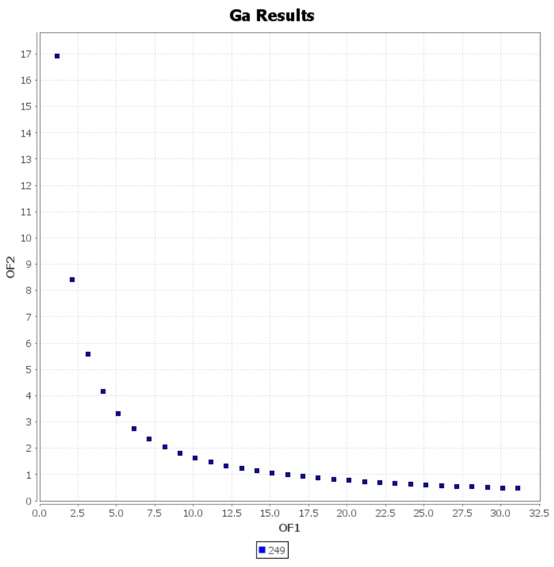


FIGURE 3.13: The benchmark test ZDT5, binary-encoded.

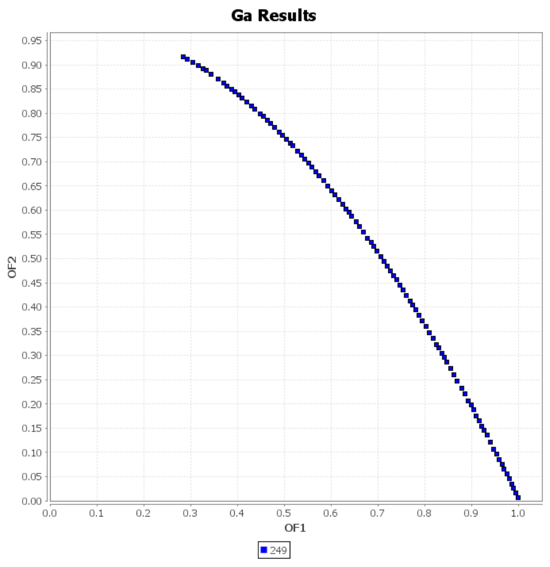


FIGURE 3.14: The benchmark test ZDT6.

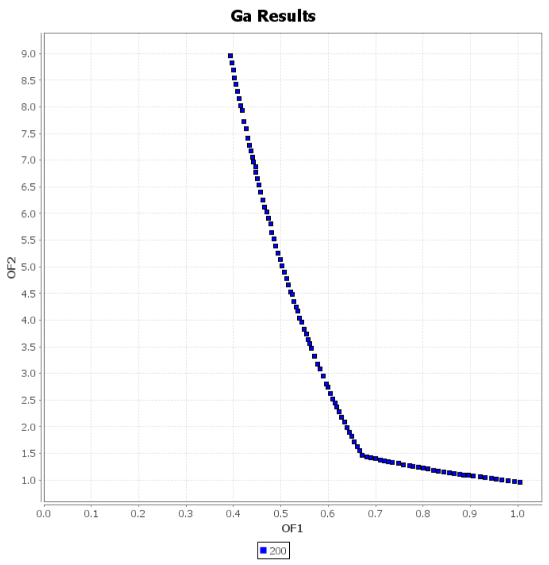


FIGURE 3.15: The benchmark test CONSTR, Deb's problem having two constraints.

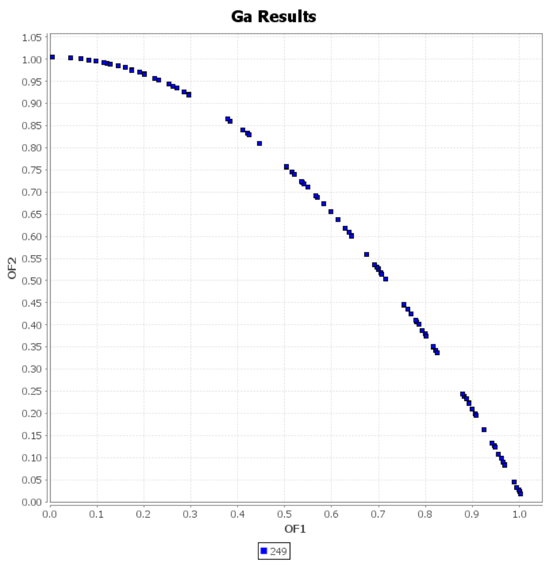


FIGURE 3.16: The benchmark test ZDT2, binary-encoded; when compared to the real-encoded result of Figure 3.10, it is noticeable that this front is much more sparse.

3.4 Comparison with random search

This section adds further test problems to the previous benchmark set, in order to act as a comparison between the performance of Ganesh and a random search algorithm, on problems which have some of the characteristics of the optimal power flow (OPF) problems of Chapter 5 as introduced briefly in section 5.6.

Due to the non-availability of a Plexos version licence (the software used in Chapter 5 in conjunction with Ganesh, for OPF), it was not possible to run direct comparisons on those problems.

Firstly, a random search algorithm (MooRand) was designed and implemented to fit into the Ganesh architecture. This was so that the differences between Ganesh and the new algorithm would be as little as possible with the exception of the exact part of the algorithm under test. Elitism and non-dominated sorting are thus carried out in the same way for both by the same code base, and the problem definition is likewise also shared. In this way, it enables and facilitates the attribution of any differences found in the results, to the specific algorithmic behaviour under test.

Each problem under test is run 20 times by each algorithm (Ganesh and MooRand) and their results compared. Performance measurement is carried out in the same way as in Chapter 4, using PISA with the *hypervolume* indicator (Zitzler and Thiele, 1999) and the (unary) ε-indicator (Zitzler *et al.*, 2003), as further described in sections 4.5.1 and 4.5.2.1. The Mann-Whitney two-tailed test (Hollander *et al.*, 2014) is used to compare the result sets.

The test problems defined here adopt all of the aspects that they can while being relevant, of the optimisation problem definition in Chapter 5 for Plexos, specifically in section 5.4, namely the size (72) and data type (integer) of the design vector, and the number of *Real* objective functions (4). Similarly, the test problems were carried out with the same run parameters: the population size (30), the number of generations (67), and the mutation (0.01389) and crossover (0.9) rates (used for Ganesh).

The significance level for the tests is taken as $\alpha = 0.05$ and the test is a non-parametric one, so compares ranks rather than raw data, and the test is for mean ranks, where $H0$ is that any difference seen is within normal bounds for randomly fluctuating values and that therefore the sample groups can be considered to be from the same population. This means that neither algorithm can be thought of as 'better' than the other. Conversely, if $H0$ is rejected, this means that, in that specific context, the relevant algorithm can be thought of as providing better performance than the other.

3.4.1 Random search algorithm

This section describes the random search algorithm used for these optimisation tests, which is designated 'MooRand' for ease of reference.

This algorithm uses pseudo-random numbers generated from a uniform distribution. Let randomF(n,p) mean generally to find a random floating point number in the interval [n,p], and let $minFloat$ be the minimum possible difference between one floating point number and another for the machine running the algorithm. Let *geneLow* & *geneHigh* mean the lowest and highest values that the gene may takes, respectively.

The expression randomF(n + $minfloat$, p) thus finds a random number in the interval (n,p], and randomF(n, p - $minfloat$) acts similarly for the interval [n,p).

Let randomI(0, g) mean find a random integer i in the interval [0, g] where g is the number of components of a solution's decision vector (genes) in the solution. The probability pG of change of a given gene of the solution thus becomes $pG = i/g$.

The random search algorithm is defined by pseudo-code 3.6. A *population* is a vector of given size (*popSize*), of solutions. The function *Evaluate* means determine the fitness functions and carry out constraint checks. *Merge* means combine two populations into one. *Non-Dominated sorting* means use the solutions' fitnesses to rank them in successive fronts. *Extract* means find the top *popSize* solutions from the given population and assign them to the new one. All these are performed in the same way as for Ganesh.

Pseudo-code 3.6: MooRand: Random search algorithm

```
begin
    Initialise Parent population to set of random solutions
    output Parent population

    while stopping criteria not met do

        Child population ← deep copy of Parent population

        for each Solution in Child population do

            pG ← randomI(0, g) / g

            for each Gene in Solution do

                if randomF(0,1) ≤ pG then

                    if randomF(0,1) ≤ 0.5 then
                        Gene ← randomF(current value + minFloat, geneHigh)
                    else
                        Gene ← randomF(geneLow, current value - minFloat)

        Evaluate (Child population)

        Mixed population ← Merge(Parent, Child)

        Non-Dominated Sorting( Mixed population)

        Parent population ← Extract ( popSize, Mixed population)

        output Parent population
```

Thus for each solution, between 0 and all of its genes may change, and they may change by increasing or decreasing from their current value to the appropriate end of their permitted range, where the current value is not included in the interval.

3.4.2 Further optimisation test problems

The DTLZn problems were defined by Deb *et al.* (2001) to be scalable test problems. Here in DTLZ1 to DTLZ9, they are scaled to a decision vector of size 72 with 4 objective functions, as described in the introduction section, 3.4.

In general, M is the number of objective functions, the first $(M-1)$ decision variables of the decision vector $\boldsymbol{x}$, $(x_1, x_2, \cdots, x_{M-1})$ of size n, govern convergence. The decision variables $(x_M, x_{M+1}, \cdots, x_n)$ govern the diversity of the solutions and their location. The total number of decision vector variables $n = M+k-1$, where k scales the difficulty of the problem. Here, $n = 72 = 4+69-1$.

$$f_i(\mathbf{x}) \text{ takes variables in } [x_1 \cdots x_{M-1}],$$
$$g(\mathbf{x}_M) \text{ takes } |\mathbf{x}_M| = k \text{ variables in } [x_M \cdots x_n], \text{ as } k = n+1-M.$$

3.4.2.1 DTLZ1

Here, the Pareto-optimal front is linear and corresponds to $\mathbf{x}_M = 0$, and the objective function values lie on the linear hyper-plane: $\sum_{m=1}^{M} f_m = 0.5$ which is difficult to converge to. The search space contains $(11^k - 1)$ local Pareto-optimal fronts, each of which can attract an MOEA. Where M > 3, all Pareto-optimal solutions on a 3D plot involving f_M and any two other objectives, will lie on or below the hyper-plane as given above. The problem is formulated as a minimisation, thus (Equations 3.1 & 3.2):

$$\left.\begin{aligned}
&\text{Min } f_1(\mathbf{x}) = \tfrac{1}{2}x_1x_2\cdots x_{M-1}(1+g(\mathbf{x}_M)),\\
&\text{Min } f_2(\mathbf{x}) = \tfrac{1}{2}x_1x_2\cdots(1-x_{M-1})(1+g(\mathbf{x}_M)),\\
&\vdots\\
&\text{Min } f_{M-1}(\mathbf{x}) = \tfrac{1}{2}x_1(1-x_2)(1+g(\mathbf{x}_M)),\\
&\text{Min } f_M(\mathbf{x}) = \tfrac{1}{2}(1-x_1)(1+g(\mathbf{x}_M)),\\
&\qquad 0 \le x_i \le 1, \quad \text{for } i = 1,2,\ldots,n.
\end{aligned}\right\} \tag{3.1}$$

where $g(\mathbf{x}_M)$ takes $|\mathbf{x_M}| = k$ variables (and any function having $g \geq 0$), and is defined thus:

$$g(\mathbf{x}_M) = 100 \left[|\mathbf{x_M}| + \sum_{x_i \in \mathbf{x_M}} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)) \right] \tag{3.2}$$

3.4.2.2 DTLZ2

DTLZ2 is defined to have multiple global optima over a non-convex Pareto front, having a design vector with components of two types, where one type governs convergence, and the other governs distribution (of solutions). These act in the objective space. This problem has a well-defined Pareto front thus is a good choice for a test problem comparing optimisation algorithms.

The generalised problem is formulated as a minimisation, thus (Equation 3.3):

$$\left.\begin{array}{l}
\text{Min } f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \cdots \cos(x_{M-2}\pi/2) \cos(x_{M-1}\pi/2), \\
\text{Min } f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \cdots \cos(x_{M-2}\pi/2) \sin(x_{M-1}\pi/2), \\
\text{Min } f_3(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \cdots \sin(x_{M-2}\pi/2), \\
\vdots \\
\text{Min } f_{M-1}(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \sin(x_2\pi/2), \\
\text{Min } f_M(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \sin(x_1\pi/2), \\
\qquad 0 \leq x_i \leq 1, \quad \text{for } i = 1, 2, \ldots, n, \\
\text{where } g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2.
\end{array}\right\} \tag{3.3}$$

3.4.2.3 DTLZ3

This problem is defined by using the equations for the objective functions of DTLZ2 (equation 3.3) but replacing $g(\mathbf{x}_M)$ with that defined for DTLZ1 (equation 3.2).

Now there are $(3^k - 1)$ local Pareto-optimal fronts which are parallel to the global Pareto-optimal front (at $g^* = 0$) corresponding to $\mathbf{x}_M = (0.5, \cdots, 0.5)^T$. The next nearest locally optimal front being at $g^* = 1$.

3.4.2.4 DTLZ4

In this problem, the equations for the objective functions of DTLZ2 (equation 3.3) are used but with an alternative mapping for the objective function subset of variables as given in equation 3.4, which raises them to the power of α, which is used here as $\alpha = 100$. This use of α causes a denser group of solutions near the $f_M - f_1$ plane, hence a varying density of solutions throughout.

$$\left.\begin{aligned}
&\text{Min } f_1(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2)\cdots\cos(x_{M-2}^\alpha \pi/2)\cos(x_{M-1}^\alpha \pi/2),\\
&\text{Min } f_2(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2)\cdots\cos(x_{M-2}^\alpha \pi/2)\sin(x_{M-1}^\alpha \pi/2),\\
&\text{Min } f_3(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2)\cdots\sin(x_{M-2}^\alpha \pi/2),\\
&\vdots\\
&\text{Min } f_{M-1}(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1^\alpha \pi/2)\sin(x_2^\alpha \pi/2),\\
&\text{Min } f_M(\mathbf{x}) = (1+g(\mathbf{x}_M))\sin(x_1^\alpha \pi/2),\\
&\qquad 0 \le x_i \le 1, \quad \text{for } i = 1, 2, \ldots, n,\\
&\text{where } g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2.
\end{aligned}\right\} \tag{3.4}$$

3.4.2.5 DTLZ5

In this problem, the equations for the objective functions of DTLZ2 (equation 3.3) are used again, but with an alternative mapping for the objective function subset of variables as given in equation 3.5. This amended problem tests the ability to converge to a curve, and it can be noted that the performance could be visualised by a graph plot of f_M with any other of the objective functions.

$$\left.\begin{aligned}
&\text{Min } f_1(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\cos(\theta_{M-2}\pi/2)\cos(\theta_{M-1}\pi/2),\\
&\text{Min } f_2(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\cos(\theta_{M-2}\pi/2)\sin(\theta_{M-1}\pi/2),\\
&\text{Min } f_3(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\sin(\theta_{M-2}\pi/2),\\
&\vdots\\
&\text{Min } f_{M-1}(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\sin(\theta_2\pi/2),\\
&\text{Min } f_M(\mathbf{x}) = (1+g(\mathbf{x}_M))\sin(\theta_1\pi/2),\\
&\text{where} \quad g(\mathbf{x}_M) = \sum_{x_i\in\mathbf{x}_M}(x_i-0.5)^2,\\
&\qquad\qquad \theta_i = \frac{\pi}{4(1+g(r))}(1+2g(r)x_i),\ \text{for } i=2,3,\cdots,(M-1),\\
&\qquad\qquad 0\le x_i\le 1, \quad \text{for } i=1,2,\ldots,n.
\end{aligned}\right\} \tag{3.5}$$

3.4.2.6 DTLZ6

This test problem is DTLZ5 above, made harder by modifying function $g(\mathbf{x}_M)$ as in equation 3.6.

$$g(\mathbf{x}_M) = \sum_{x_i\in\mathbf{x}_M} x_i^{0.1} \tag{3.6}$$

3.4.2.7 DTLZ7

This problem has a disconnected set of size 2^{M-1} of Pareto-optimal regions, thus testing whether the algorithm can maintain solutions at or around each disparate Pareto-optimal front (which correspond to $\mathbf{x}_M = 0$). Equation 3.7 specifies the problem.

$$\left.\begin{array}{lr}
\text{Min} & f_1(\mathbf{x_1}) = x_1, \\
\text{Min} & f_2(\mathbf{x_2}) = x_2, \\
\vdots & \\
\text{Min} & f_{M-1}(\mathbf{x_{M-1}}) = x_{M-1}, \\
\text{Min} & f_M(\mathbf{x}) = (1+g(\mathbf{x}_M))h(f_1, f_2, \cdots, f_{M-1}, g), \\
\text{Where} & g(\mathbf{x}_M) = 1 + \frac{9}{|\mathbf{x}_M|}\sum_{x_i \in \mathbf{x}_M} x_i, \\
 & h(f_1, f_2, \cdots, f_{M-1}, g) = M - \sum_{i=1}^{M-1}\left[\frac{f_i}{1+g}(1+\sin(3\pi f_i))\right], \\
 & 0 \le x_i \le 1, \quad \text{for } i = 1, 2, \ldots, n.
\end{array}\right\} \quad (3.7)$$

3.4.2.8 DTLZ8

This problem constrains solutions to a (hyper-)surface in which the Pareto-optimal front is a combination of a hyper-plane and a straight line, where the line is the intersection of the first $M-1$ constraints with $f_1 = f_2 = f_{M-1}$ and g_M is the constraint representing the hyper-plane. Equation 3.8 specifies the problem.

$$\left.\begin{array}{lll}
\text{Min} & f_j(\mathbf{x}) = \frac{1}{\lfloor \frac{n}{M} \rfloor} \sum_{i=\lfloor (j-1)\frac{n}{M} \rfloor}^{\lfloor j\frac{n}{M} \rfloor} x_i, & \text{for } j = 1, 2, \cdots, M, \\
\text{Subject to} & g_j(\mathbf{x}) = f_M(\mathbf{x}) + 4f_j(\mathbf{x}) - 1 \ge 0, & \text{for } j = 1, 2, \cdots, (M-1), \\
 & g_M(\mathbf{x}) = 2f_M(\mathbf{x}) + \min_{\substack{i,j=1 \\ i\mathbf{not}=j}}^{M-1} [f_i(\mathbf{x}) + f_j(\mathbf{x})] - 1 \ge 0, & \\
 & 0 \le x_i \le 1, \quad \text{for } i = 1, 2, \ldots, n. &
\end{array}\right\} \quad (3.8)$$

3.4.2.9 DTLZ9

Similarly to DTLZ8, this problem also employs a constraint surface, with a Pareto-optimal front that is a curve with $f_1 = f_2 = \cdots = f_{M-1}$ similar to DTLZ5, where solutions are less dense towards the Pareto-optimal region. The Pareto-optimal curve lies occurs at the intersection of all $(M-1)$ constraints. Equation 3.9 specifies the problem.

$$\left.\begin{array}{lll} \text{Min} & f_j(\mathbf{x}) = \displaystyle\sum_{i=\lfloor (j-1)\frac{n}{M}\rfloor}^{\lfloor j\frac{n}{M}\rfloor} x_i^{0.1}, & \text{for } j = 1,2,\cdots,M, \\ \text{Subject to} & g_j(\mathbf{x}) = f_M^2(\mathbf{x}) + 4_j^2(\mathbf{x}) - 1 \geq 0, & \text{for } j = 1,2,\cdots,(M-1), \\ & 0 \leq x_i \leq 1, \quad \text{for } i = 1,2,\ldots,n. & \end{array}\right\} \quad (3.9)$$

3.4.2.10 MOKP 0/5

This is a multi-objective knapsack problem, derived from the MOKP 0/1 problem defined by Zitzler and Thiele (1999), and tailored to have the same dimensions as the Plexos problem as described in the introduction section, 3.4.

The original MOKP 0/1 used a binary string decision vector whose variables took values in the interval [0,1], whereas in this problem, the vector is composed of integer variables (of 8 bits each) which take values in the interval [0,5].

It is required to pack each of n knapsacks with items giving a certain profit and having a particular weight, and where each knapsack has a weight capacity which is not able to be exceeded (Equation 3.10), thus a hard constraint. Each knapsack is thus an objective function having its own constraint, and n is set to 4. There are $m = 72$ items available, whose profit (p) and weight (w) values are set independently and randomly in the interval [10,100], per knapsack, and where the capacity of each knapsack (of $i = (1...n)$) is set to half the total available weight, $c_i = 0.5 \cdot (\sum_{j=1}^{m} w_{i,j})$. The objective (Equation 3.11) is to maximise the profit of

m items of each of n knapsack, where

$$\begin{aligned} P_{i,j} &= \text{profit of item } j \text{ in knapsack } i \\ W_{i,j} &= \text{weight of item } j \text{ in knapsack } i \\ c_i &= \text{capacity of knapsack } i \end{aligned}$$

find a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_m) \in \{0, 5\}^m$, such that

$$\forall i \in \{1, 2, \ldots, n\} : \sum_{j=1}^{m} w_{i,j} \cdot x_j \leq c_i \tag{3.10}$$

and for which $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ is maximum, where

$$f_i(\boldsymbol{x}) = \sum_{j=1}^{m} P_{i,j} \cdot x_j \tag{3.11}$$

and where $0 < x_j \leq 5$ *iff* item j is selected.

As in the original MOKP 0/1, a greedy repair method is used since many possible codings can lead to infeasible solutions. The repair removes items iteratively until all constraints are met, removing the items in order of their maximum profit/weight ratio, the lowest value being removed first. This tends to fill the sacks as much as possible within the constraint thus maximising the profit. The maximum profit/weigh ratio q_j is given as follows:

$$q_j = \max_{i=1}^{n} \left\{ \frac{P_{i,j}}{W_{i,j}} \right\}$$

3.4.3 Results

The results below were obtained by the Mann-Whitney statistical test (discussed in section 3.4) comparing the performance of Ganesh and MooRand as measured by the epsilon and hypervolume indicators, and are given along with the means and standard deviations of the indicators. As specified in the introduction section, 3.4, 20 runs of each algorithm were used for each problem.

3.4.3.1 DTLZ1

Here, Table 3.2 gives the Mann-Whitney test results, and Table 3.3 gives the means and standard deviations of the indicators. It can be seen from Table 3.2 that Ganesh performed the better.

Table 3.2: DTLZ1. Mann-Whitney Test p-values for $\alpha = 0.05$, for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M).

Ind	M > G	G > M	Best
eps	0.998775651	0.00122434869	Ganesh
hyp	0.999999881	1.19216374e-007	Ganesh

Table 3.3: Means of ε- and hypervolume indicators for DTLZ1 test, accompanying Table 3.2.

DTLZ1 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.163808	0.023371	0.040257	0.010879
MooRand	0.186602	0.014125	0.064708	0.006220

3.4.3.2 DTLZ2

Here, Table 3.4 gives the Mann-Whitney test results, and Table 3.5 gives the means and standard deviations of the indicators. It can be seen from Table 3.4 that Ganesh performed the better.

Table 3.4: DTLZ2. Mann-Whitney Test p-values for $\alpha = 0.05$, for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M).

Ind	M > G	G > M	Best
eps	0.999999968	3.15092411e-008	Ganesh
hyp	0.999999968	3.15092411e-008	Ganesh

Table 3.5: Means of ε- and hypervolume indicators for DTLZ2 test, accompanying Table 3.4.

DTLZ2 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.146438	0.040633	0.019289	0.009570
MooRand	0.399339	0.018624	0.302794	0.026990

3.4.3.3 DTLZ3

Here, Table 3.6 gives the Mann-Whitney test results, and Table 3.7 gives the means and standard deviations of the indicators. It can be seen from Table 3.6 that Ganesh performed the better.

Table 3.6: DTLZ3. Mann-Whitney Test p-values for $\alpha = 0.05$, for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M).

Ind	M > G	G > M	Best
eps	0.999999789	2.1143139e-007	Ganesh
hyp	0.999999923	7.6997691e-008	Ganesh

Table 3.7: Means of ε- and hypervolume indicators for DTLZ3 test, accompanying Table 3.6.

DTLZ3 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.219473	0.032111	0.111651	0.035446
MooRand	0.285546	0.017656	0.236112	0.033971

3.4.3.4 DTLZ4

Here, Table 3.8 gives the Mann-Whitney test results, and Table 3.9 gives the means and standard deviations of the indicators. It can be seen from Table 3.8 that Ganesh performed the better.

TABLE 3.8: DTLZ4. Mann-Whitney Test p-values for $\alpha = 0.05$, for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M).

Ind	M > G	G > M	Best
eps	0.999999968	3.15092411e-008	Ganesh
hyp	0.999994811	5.1885088e-006	Ganesh

TABLE 3.9: Means of ε- and hypervolume indicators for DTLZ4 test, accompanying Table 3.8.

DTLZ4 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.174196	0.027986	0.040812	0.061204
MooRand	0.424469	0.032960	0.190972	0.020375

3.4.3.5 DTLZ5

Here, Table 3.10 gives the Mann-Whitney test results, and Table 3.11 gives the means and standard deviations of the indicators. It can be seen from Table 3.10 that Ganesh performed the better.

TABLE 3.10: DTLZ5. Mann-Whitney Test p-values for $\alpha = 0.05$, for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M).

Ind	M > G	G > M	Best
eps	0.999999968	3.15092411e-008	Ganesh
hyp	0.999999968	3.15092411e-008	Ganesh

TABLE 3.11: Means of ε- and hypervolume indicators for DTLZ5 test, accompanying Table 3.10.

DTLZ5 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.182641	0.029279	0.069079	0.018115
MooRand	0.338402	0.013972	0.248108	0.020745

3.4.3.6 DTLZ6

Here, Table 3.12 gives the Mann-Whitney test results, and Table 3.13 gives the means and standard deviations of the indicators. It can be seen from Table 3.12 that MooRand performed the better.

TABLE 3.12: DTLZ6. Mann-Whitney Test p-values for $\alpha = 0.05$, for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M).

Ind	M > G	G > M	Best
eps	0.00133859648	0.998661404	MooRand
hyp	3.15092411e-008	0.999999968	MooRand

TABLE 3.13: Means of ε- and hypervolume indicators for DTLZ6 test, accompanying Table 3.12.

DTLZ6 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.251443	0.021198	0.280420	0.020621
MooRand	0.231840	0.012692	0.212012	0.018785

3.4.3.7 DTLZ7

Here, Table 3.14 gives the Mann-Whitney test results, and Table 3.15 gives the means and standard deviations of the indicators. It can be seen from Table 3.14 that Ganesh performed the better.

TABLE 3.14: DTLZ7. Mann-Whitney Test p-values for $\alpha = 0.05$, for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M).

Ind	M > G	G > M	Best
eps	0.999999943	5.73277303e-008	Ganesh
hyp	0.999999968	3.15092411e-008	Ganesh

Table 3.15: Means of ε- and hypervolume indicators for DTLZ7 test, accompanying Table 3.14.

DTLZ7 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.165262	0.045021	0.117447	0.048186
MooRand	0.362949	0.036670	0.471258	0.053817

3.4.3.8 DTLZ8

Here, Table 3.16 gives the Mann-Whitney test results, and Table 3.17 gives the means and standard deviations of the indicators. It can be seen from Table 3.16 that Ganesh performed the better.

Table 3.16: DTLZ8. Mann-Whitney Test p-values for $\alpha = 0.05$, for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M).

Ind	M > G	G > M	Best
eps	0.999993349	6.65139211e-006	Ganesh
hyp	0.993083251	0.00691674946	Ganesh

Table 3.17: Means of ε- and hypervolume indicators for DTLZ8 test, accompanying Table 3.16.

DTLZ8 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.210905	0.043863	0.332330	0.076100
MooRand	0.285523	0.042366	0.392731	0.060583

3.4.3.9 DTLZ9

Here, Table 3.18 gives the Mann-Whitney test results, and Table 3.19 gives the means and standard deviations of the indicators. It can be seen from Table 3.18 that Ganesh performed the better.

TABLE 3.18: DTLZ9. Mann-Whitney Test p-values for $\alpha = 0.05$, for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M).

Ind	M > G	G > M	Best
eps	0.994911528	0.00508847225	Ganesh
hyp	0.996584872	0.00341512793	Ganesh

TABLE 3.19: Means of ε- and hypervolume indicators for DTLZ9 test, accompanying Table 3.18.

DTLZ9 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.474524	0.095193	0.620900	0.170833
MooRand	0.525815	0.031291	0.703415	0.015519

3.4.3.10 MOKP 0/5

Here, Table 3.20 gives the results obtained by the Mann-Whitney statistical test comparing the performance of Ganesh and MooRand as measured by the epsilon and hypervolume indicators, and Table 3.21 gives the means and standard deviations of the indicators. As specified in the introduction section, 3.4, 20 runs of each algorithm were used, and it can be seen from Table 3.20 that Ganesh performed the better.

TABLE 3.20: MOKP 0/5. Mann-Whitney test results for 2 independent sample groups of 20 runs, for Ganesh (G) and MooRand (M), showing p-values for $\alpha = 0.05$.

Ind	M > G	G > M	Best
eps	0.991967993	0.00803200657	Ganesh
hyp	0.994911528	0.00508847225	Ganesh

TABLE 3.21: Means and standard deviations of ε- and hypervolume indicators for MOKP 0/5 test, accompanying Table 3.20.

MOKP 0/5 statistics				
	Epsilon		Hypervolume	
Algorithm	Mean	SD	Mean	SD
Ganesh	0.396714	0.091842	0.745523	0.197681
MooRand	0.471243	0.095706	0.879651	0.164171

3.4.4 Summary

These tests are intended to show that Ganesh can perform better than random for problems having some of the same characteristics, as the distributed generation optimisations of Chapter 5. The specific similarities are problems having the same dimensions in their decision vectors and number of objective functions.

Since it was not possible to run the MooRand algorithm with Plexos itself, as a Plexos licence was not available, these test problems were chosen instead as they leant themselves to being scaled.

It is not proposed that the problems themselves are similar to those of Chapter 5, rather they have been chosen to provide a spread of problem types in order to show that at least in some cases, Ganesh can perform better than random for problems of these dimensions, particularly considering the number of elements in the decision vector compared with the size of the population of candidate solutions used.

Of the 10 optimisation problems used here, in 9 of them Ganesh was seen to achieve better performance, by both the hypervolume and εpsilon performance indicators, than MooRand, despite the relatively small population size compared with the number of components in the decision vector. Only for the DTLZ6 test problem was MooRand seen to be better. It should be noted that Ganesh could be tuned to that problem better, by adjusting its rates of mutation or crossover, probably by making it more random, though this is not the point of the tests so it has not been tried.

Ten problems, 20 runs per problem per algorithm, with a population size of 30 and 67 generations, has given 402,000 function evaluations per algorithm.

3.5 Experiments in self-adaptation

In this section, the self-adaptation used by Ganesh is compared with Ganesh in non-self-adaptive mode through the simple expedient of choosing the non-self-adaptive Real *Chromosome* to run the latter.

The airfoil optimisation problem of Chapter 4 is used as the basis of comparison and it is run here similarly to how it is descrubed in that chapter, namely that each mode (self-adaptive and non-self-adaptive) has 20 runs of each range (0.3, 0.6, 0.8, 1.0), thus giving a total of 80 runs per mode, or 160 runs overall. The population of solutions is set at 120 and the number of generations is set to 50, thus giving 6,000 function evaluations (FEs) per run, which gives 480,000 FEs per mode and 960,000 FEs overall. The crossover rate and mutation rate are set to 0.9 and 0.5 respectively, and the number of duplicate solutions allowed per generation is 0.

The results obtained from the runs described above are summarised in the following tables, 3.22, 3.23 & 3.24, in which the table 3.22 gives the results of the Mann-Whitney two-tailed hypothesis tests, and the other tables give the means and standard deviations of the respective performance indicators for the given range of the stated mode. The null hypothesis is that the performance indicators show no statistical difference other than might be obtained by chance alone, thus neither mode of operation can be said to be better than the other. A significance level of $\alpha = 0.05$ is used with which to compare the p-values obtained.

The Mann-Whitney test results are marked as $H0$ when the test indicates that the null hypothesis should not be rejected. If the test shows that the null hypothesis could be rejected, the result is marked as A or N if the self-adaptive or non-self-adaptive mode respectively is indicated as better for the given performance indicator.

TABLE 3.22: Mann-Whitney test results for 2 independent sample groups of 20 runs, for self-adaptive mode (A) and non-self-adaptive mode (N), showing p-values for $\alpha = 0.05$.

Range	Ind	N > A	A > N	Best
0.3	eps	0.997119	0.002881	A
0.3	hyp	0.998547	0.001453	A
0.6	eps	0.585725	0.414274	$H0$
0.6	hyp	0.950660	0.049339	A
0.8	eps	0.995320	0.004680	A
0.8	hyp	0.941794	0.058206	$H0$
1.0	eps	0.158369	0.841630	$H0$
1.0	hyp	0.133636	0.866364	$H0$

TABLE 3.23: Means and standard deviations of ε- and hypervolume indicators of Ganesh in self-adaptive mode, accompanying Table 3.22.

Self-adaptive mode				
	Epsilon		Hypervolume	
Range	Mean	SD	Mean	SD
0.3	0.077564	0.052772	0.045849	0.022552
0.6	0.111192	0.046011	0.101889	0.031940
0.8	0.095531	0.024549	0.115987	0.048946
1.0	0.134706	0.045658	0.164167	0.067991

TABLE 3.24: Means and standard deviations of ε- and hypervolume indicators of Ganesh in non-self-adaptive mode, accompanying Table 3.22.

Non-self-adaptive mode				
	Epsilon		Hypervolume	
Range	Mean	SD	Mean	SD
0.3	0.128665	0.058027	0.064399	0.021486
0.6	0.108141	0.034664	0.118921	0.028030
0.8	0.124487	0.033144	0.149136	0.054489
1.0	0.120090	0.033025	0.139473	0.049614

3.5.1 Summary

Table 3.22 shows that the Mann-Whitney two-tailed tests give results allowing the null hypothesis ($H0$) to be rejected for at least one of the ε- and hypervolume indicators, for 3 out of the 4 ranges under test. Where the null hypothesis is rejected, the self-adaptive mode (A) of Ganesh is always seen to perform better.

Range ± 0.3 gives the self-adaptive mode (A) best for both indicators, range ± 0.6 has A for the hypervolume indicator, and range ± 0.8 has A for the ε-indicator.

It therefore seems reasonable to conclude that there is a real advantage in the self-adaptive functionality used by Ganesh.

3.6 Methods and materials

As described in the thesis introduction, having performed a literature review (which was also refined at various points through further investigation) the Ganesh framework and algorithm were analysed, designed and produced and then benchmarked as above. Ganesh was then applied to a test case real-world problem, the airfoil optimisation, and then used on the electrical power optimisation problem, having built confidence.

The Agile Software Development methodology (Martin, 2002) was used to analyse, design and develop the software developed in this work. This places emphasis on producing simple and easily testable codes that can be enhanced and extended through incremental improvements, in which units testing not only provides confidence but also drives the development.

The framework, algorithm, and the plugins encoding the optimisation problems were developed in Java version *Java Platform, Standard Edition 7 (Java SE 7)* (Oracle, 2014). The software from which Ganesh is built is dependent upon features which were new to an older version, *Java 2 Platform, Standard Edition 5.0 (J2SE 5.0)*, in particular Enumerations and Generics, therefore Ganesh will not run on versions of Java older than that. Some additional programs were also developed in Java to collect and manage data produced by Ganesh, and to prepare the data for processing by further external tools, such as PISA (Bleuler *et al.*, 2003).

Software *Patterns* (Gamma *et al.*, 1995) and specifically patterns for Java (Grand, 1998) & (Grand, 1999) were used both directly and as inspiration during the design process, and UML (Martin, 2003) was used both as a documentation method and as a design tool. Software design pattens are recognised general solutions arising experientially, that have applicability to many specific design problems and can be thought of as design re-use. Since they are generalised, their use can lead to higher levels of abstraction than may be necessary, leading to increased complexity, thus judgement is needed in deciding their applicability.

The software sources, and the source of the thesis (which was created using LaTeX 2_ε (LaTeX project team, 2014)) were kept under configuration management using Mercurial (Mercurial community, 2014), a scalable distributed source control management (SCM) tool.

The R language (R Development Core Team, 2014) was used to post-process some result data and to produce graph plots.

GNUPlot (Geeknet inc., 2014) was used for some other graph plots.

The above software, tools and Ganesh all run on both Microsoft Windows™, Unix and Linux platforms. This proved to be crucial for the success of this work, since the Airfoil optimisation needed to be carried out on a Linux platform, while the power optimisation using Plexos needed to be carried out on a Windows™ platform.

Chapter 4

A real-world airfoil application test case

4.1 Introduction to airfoil optimization

This chapter considers the application of the MOOEA of this work, Ganesh, to a real-world engineering problem, that of airfoil design by multi-objective optimisation. This work acts as both a test-case and as a comparison with a similar study. An unexpected outcome of this work was the identification of some improvements that could be made to a tool commonly used for airfoil design and which had been used in the study with which this was compared.

The approach is to use a standard airfoil having well understood characteristics, to iteratively modify its geometry and assess the new geometry in terms of aerodynamic performance, and then allow the set of solutions to evolve over time. Ganesh creates an initial population of random design vectors, each of which undergoes the above process, and Ganesh uses the aerodynamic performance criteria to perform its optimisation in the usual manner, thus evolving a set of more highly performing airfoils over time.

The optimisation performed by Ganesh is then compared with results obtained by two other high performing multi-objective optimising algorithms, NSGA-II (Deb *et al.*, 2000) and Multi-Objective Tabu Search (MOTS)(Jaeggi *et al.*, 2008).

The standard airfoil used was the NACA 0012 section (Abott and von Doenhoff, 1959), Fig. 4.1, which is a standard symmetric airfoil having a 12% thickness to chord length ratio, defined originally by the U.S. National Advisory Committee for Aeronautics, which is now part of the National Aeronautics and Space Administration (NASA). This airfoil is used in a wide variety of aircraft (Lednicer, 2010), therefore there is a well established body of knowledge about it that is practical as well as theoretical.

4.2 Airfoil geometry

Airfoil shape modification is carried out by free-form deformation (FFD) (Sederberg and Parry, 1986) using an implementation used by Kipouros *et al.* (2012). FFD is a mathematical technique for deforming 'solid geometric models' in a free-form manner, based upon a chosen set of control points from which the deformation is governed, thus performing a mapping $\mathcal{R}^2 \rightarrow \mathcal{R}^2$. Here, the airfoil has four control points of a surrounding hull, each of which has a vertical and horizontal displacement component, giving rise to eight parameters (being four pairs) as illustrated in Fig. 4.2, in which a positive value gives a right or up shift and a negative one a left or down shift, for the relevant axis.

The displacement of one or more control points thus produces a new geometry via the mapping, that is translated into a vector of Cartesian coordinates that in turn can be used by Drela's XFoil tool (Drela, 1989).

The modified airfoil geometry is evaluated by XFoil for aerodynamic efficiency, calculating coefficients of lift C_L and drag C_D which are then used in Ganesh as objective function results. Figure 4.3 illustrates the flow between the component codes.

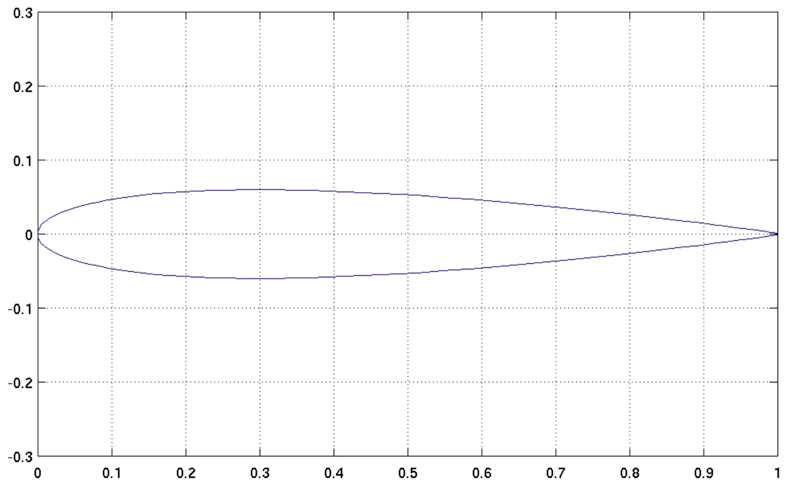


FIGURE 4.1: The NACA 0012 Airfoil, Selig (2014).

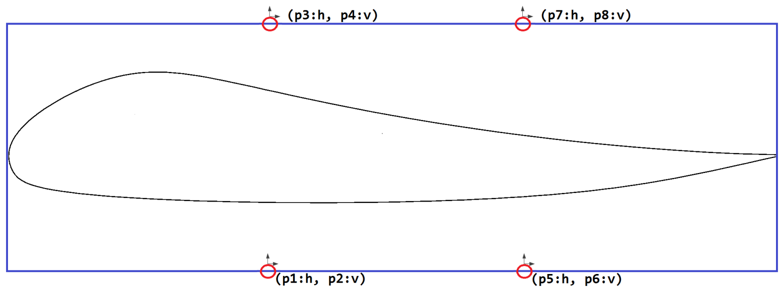


FIGURE 4.2: An airfoil is a cross-section of a wing, hence inherently two dimensional, and is shown here enclosed in its free-form deformation hull with the four control points, each of which has a horizontal and vertical displacement component, giving eight deformation parameters that define its shape altering.

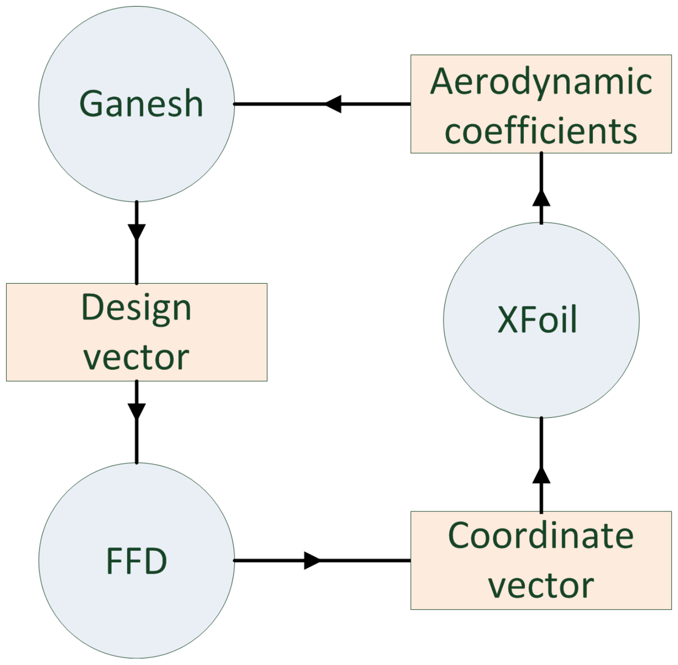


FIGURE 4.3: Schematic diagram showing the interaction of Ganesh, FFD and XFoil.

4.3 Modifying XFoil

Previous work with Ganesh and XFoil, (Oliver *et al.*, 2013) following on from Kipouros *et al.* (2012), had located extreme minima that are not feasible airfoil shapes, as under certain conditions XFoil would not converge and would not feed-back the convergence failure to Ganesh. Under these circumstances, some solutions would be unfeasible but have high fitness rankings in Ganesh due to erroneously assigned high performing lift or drag coefficients, hence would still be selected for by Ganesh, ensuring they remained in final results. Limiting boundaries had been applied to both lift and drag coefficients in an attempt to minimise this problem by keeping solutions within a region more likely to be feasible, and this had improved the performance yet not eliminated the problem entirely. The post-evaluation constraint checking that Ganesh provides enables these limiting boundaries of the OF values to be treated as soft-constraints, thus solutions breaking the limits are penalised by the degree of infraction. The constraints were as given in the Equations 4.1 to 4.3.

$$C_L \geq -3 \tag{4.1}$$

$$C_L \leq 0 \tag{4.2}$$

$$C_D \leq 2 \tag{4.3}$$

However, even with these limiting boundaries on $\mathrm{C_L}$ and $\mathrm{C_D}$, results obtained for larger deformations (range ±1.0, as defined at the end of section 4.4), included solutions that had extreme values in both objectives and which represented design vectors that were unfeasible, therefore the constraints were redefined to the following, to further exclude the obviously unfeasible regions:

$$C_L \geq -2 \tag{4.4}$$

$$C_L \leq 0 \tag{4.5}$$

$$C_D \leq 1.5 \tag{4.6}$$

In order to preclude the necessity of removing unfeasible designs from future result sets, I further enhanced the XFoil software by ensuring that all error codes indicating convergence problems, were correctly trapped and passed up, enabling the proper setting of extremely poor values for the lift and drag coefficients to be returned to Ganesh. Ganesh can then assign extremely low fitness values for each objective function of any candidate solution for which convergence in XFoil is a problem, and thus is able to eliminate these unfeasible designs through its normal fitness ranking process. The challenges found using XFoil in batch mode originally were due to the modifications made in converting it to work in batch mode from interactive mode, and were not due to the original author (Drela, 1989), nor this author.

4.4 Defining the optimisation

Having modified XFoil, the work here reiterates that of Oliver *et al.* (2013) without the problems of unfeasible results, thus enabling proper statistical analysis, and so the original results are not detailed here as they are effectively superseded by the better ones obtained with the new XFoil codes.

The airfoil is subject to two hard model constraints, these being enforced internally within XFoil, and are the thickness of the airfoil section at (a) 25% and (b) 50% along the airfoil chord, which ensure there is a minimum volume in which to place strengthening spars towards the leading and trailing edges, thus discovered optimised designs should in theory be feasible and practicable. See Figures 4.4, 4.5 and 4.6.

The optimisation definition ensures that each candidate design has the same angle of attack (α) of $\alpha = 15°$, as shown in figure 4.6, so that the objective function results are comparing equivalent measurements, by choosing FFD parameters

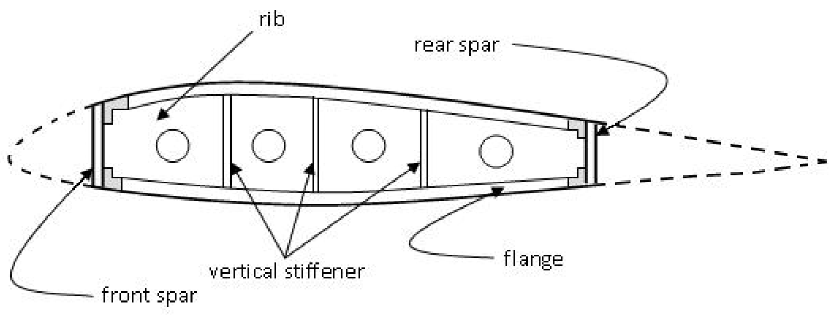


FIGURE 4.4: An airfoil showing strengthening spars and vertical stiffeners.

FIGURE 4.5: A photograph showing a cross-section of an aircraft wing.

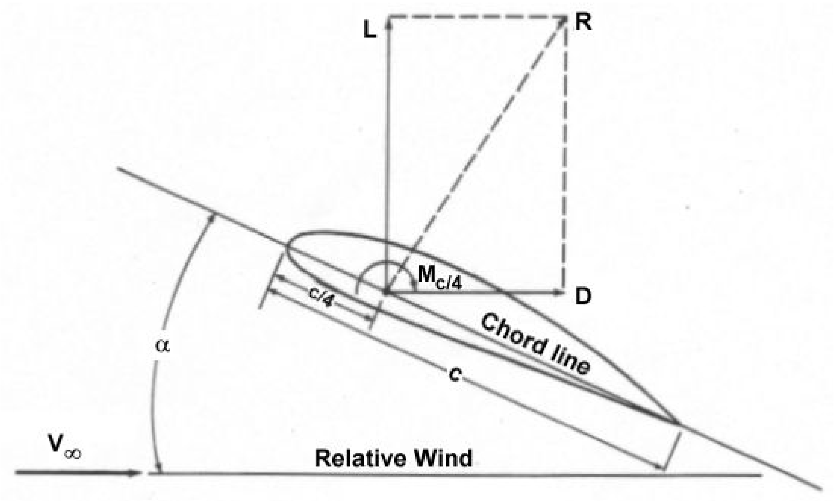


FIGURE 4.6: Diagram showing α the Angle of Attack, lift, drag and resultant vectors, and the airfoil chord. Aerospaceweb (2012)

which do not alter the position of the leading and trailing edges of the airfoil or the chord.

The FFD's eight design parameters are encoded in the GA as real numbers in the genes of each solution's chromosome, and the FFD code modifies the airfoil relative to a given datum design vector defining the geometry, based on the parameters from the GA design vector. FFD expresses the modified geometry as sets of x-y coordinates in a form that XFoil can receive, XFoil then calculates the coefficients of moment, drag (C_D), and lift (C_L) of the modified geometry and returns the latter two results, C_D and C_L, to the GA. Since the goal of this work is to optimize the airfoil with respect to drag and lift as a bi-objective problem, the coefficient of moment (C_M) is not used at this time.

The Equations 4.7 & 4.8 define the optimisations to be performed, being maximization of the lift coefficient and minimization of the drag co-efficient respectively, normalized by their respective datum values. The datum values, ($C_L = 1.46444, C_D = 0.0305108$), are the coefficient values of the unmodified NACA 0012 airfoil section.

$$F(C_L) = -\frac{C_L}{C_{L,\text{ datum}}} \tag{4.7}$$

$$F(C_D) = \frac{C_D}{C_{D,\text{ datum}}} \tag{4.8}$$

Ganesh is set to perform 6,000 function evaluations in each run with a population size of 120 and being allowed to run for 50 generations, as was performed by Kipouros *et al.* (2012). For all cases here, Ganesh is set to allow no duplicates. The probability of crossover pC is initially set to 0.9 and that of mutation pM to 0.5 for each member of the initial population, and their respective polynomial indices ηC to 10 and ηM to 20, as was the case for NSGA-II, but in the succeeding generations these values self-adapt. The probabilities may self-adapt in the interval [0, 1] while the polynomial indices may self-adapt in the interval [1, 100], noting that for the latter, larger values cause smaller perturbations in the original gene values, and vice-versa.

The problem definition also defines the range by which the design vector is allowed to be modified, and thus the degree by which the geometry of the airfoil may change. The range applies independently to the horizontal and vertical components of each FFD control point, where the positive or negative value specifies the direction of the change, up or down, or left or right. See Figure 4.2. The FFD implementation treats each input design vector component as a dimensionless number specifying the amount of movement in the appropriate direction.

A given solution will therefore have each of its design vector components with values in the interval $[-n, +n]$ of the range $\pm n$, where 0 means naturally no change and therefore a vector of {0,0,0,0,0,0,0,0} defines the datum design of the NACA 0012 airfoil. Design vectors of narrower ranges are thus also possible in the wider ranges.

A given run of the optimisation uses one range, and the corresponding results of Ganesh are compared with those of MOTS and NSGA-II, for the same range.

The ranges used are ±0.3, ±0.4, ±0.6 and ±1.0, and are chosen as having been used in other studies (Kipouros *et al.*, 2012), where the lower ranges have been shown to be tractable, while the higher ones have been shown to be problematic (and more time consuming). In tables of results, the ± is omitted for reasons of clarity and space (thus giving the ranges as 0.3 and so on).

4.5 Results

4.5.1 Comparing algorithms

Although 80 runs (20 per range) of the GA had been performed for the original work (Oliver *et al.*, 2013), automatic removal of unfeasible designs had not been possible, therefore a statistical analysis of the results would have provided little value and may have been potentially misleading.

These final results are the consequence of integrating the MOOEA with the new version of Xfoil, furthermore, the other algorithms, MOTS and NSGA-II, have also been re-run with the new Xfoil, enabling a thorough statistical analysis and comparison of them all to be performed, as is presented here. As before, another set of 20 runs per range per algorithm is performed, giving a total of 80 runs per algorithm and therefore 240 runs overall.

Figures 4.7 to 4.12 show scatter plots of non-dominated solutions in the objective space obtained by Ganesh and the other algorithms, in which OF1 and OF2 give the normalized values of C_L and C_D, plotted along the x and y axes respectively, as previously described. All solutions found are considered feasible designs and none have been removed, and dominated solutions are not shown. The values of C_L are shown as negative since it is being maximized and the GA is constructed internally to assume minimization. All results are for generation 50 (numbered as 0 to 49) unless stated otherwise in the figure caption, to provide a direct comparison with the MOTS and NSGA-II results also obtained. Each plot shows the obtained results for all 20 runs of the given range, for the given algorithm, at the last generation.

The number of generations for which the algorithms are permitted to run for is a limitation set in the scenario as a basis for comparison; Ganesh may well have (even, probably) not finished converging, thus if it had been permitted to run longer, even better results would possibly have been obtained. This is likely to be true of the other algorithms too.

4.5.1.1 Statistical analysis

The PISA package (Bleuler *et al.*, 2003) was used with the results obtained to produce standard metrics for hypervolume indicator (Zitzler and Thiele, 1998) and ε-indicator (Zitzler *et al.*, 2003), to understand the performance better through its statistical analysis and performance package. Table 4.1 gives the results of Kruskal-Wallis nonparametric one-tailed tests comparing each algorithm's 20 samples results, for a given range, against each of the others, in which the null hypothesis, $H0$, is that any variation seen between any two algorithm performances is merely due to random fluctuation within normal bounds, assuming a significance level of 5% (alpha value $\alpha = 0.05$).

The Kruskal-Wallis test (Hollander *et al.*, 2014) is similar to that of Mann-Whitney, but is used when there are $K \geq 3$ sets of random samples from K respective populations, and it is the nonparametric alternative to one-way analysis of variance. The test ranks the samples values from 1 to N where N is the sum of the sample counts from all populations, finds the mean of the ranks by sample set, and calculates the discrepancies between the mean ranks to give the Kruskal-Wallis statistic (H). $H0$ is therefore that the population mean ranks are actually equal, thus the associated p-value is the probability that as large a value of H would be seen when $H0$ is true.

When $H0$ is not rejected, one algorithm cannot be said to out-perform the other, conversely when $H0$ *is* rejected, the test suggests that the first algorithm does outperform the second, for the indicator under consideration. Table 4.1 gives the test results showing p-values for the Kruskal-Wallis test, thus where a p-value is less than 0.05, it is reasonable to reject $H0$ and say the first algorithm does out-perform the other. Where the p-value is greater than 0.05, $H0$ is not rejected and the algorithms probably have similar performance.

The tests use the *hypervolume* indicator (Zitzler and Thiele, 1999),(Zitzler and Thiele, 1998), which was originally described as “the size of the space covered”, and the (unary) ε-indicator (Zitzler *et al.*, 2003), for which the means and standard deviations of the values used in the tests are given in tables 4.3 to 4.5. These were chosen following the recommendations given by Zitzler *et al.* (2003) & Fonseca *et al.* (2005), since they work both well for comparison, and are inherently related to Pareto dominance. As no absolute optimum is known for this problem, no measurement against the *true* Pareto set is possible, thus indicators of this type are not useful, nor are indicators which only measure diversity. Other indicators were of potential interest, such as the R indicators of Hansen and Jaszkiewicz (1998) which can compare approximation sets, but it was felt that the two chosen were sufficient by themselves and were readily comprehendible as comparators.

The hypervolume indicator is a measure of how much area or (hyper-)volume, depending on the dimensionality of the objective space, is covered by an approximation front, thus it is not only an indicator of convergence but also of breadth of front. The (unary epsilon) ε-indicator is a measure of the minimum distance of translation needed to move every solution in the discovered front, so that the front weakly dominates the most converged front found, thus is an intuitive measure of Pareto dominance. So for a given range, the most converged front from all 60 samples is chosen, by PISA, as the reference set against which the ε-indicator is measured.

The Kruskal-Wallis tests show that for these set of results, for both indicators across all ranges, Ganesh seems to out-perform MOTS in terms of convergence and has a similar breadth of front, although at the greater ranges, not as dense as MOTS. As can be seen in the scatter plot for the 1.0 range, MOTS does achieve several very good non-dominated points. The tests also show that Ganesh out-performs NSGA-II for *hypervolume* at 0.3, both indicators at 0.6 and for ε-indicator at 0.8, but neither at 1.0. NSGA-II on the other hand does not seem to out-perform Ganesh for any indicator at any range, while it also seems to outperform MOTS across indicators and ranges.

The means and standard deviations for the indicators given in tables 4.3, 4.4 and 4.5 show that Ganesh tends to have a higher variation in indicator value than NSGA-II, and this seems to be borne out by the scatter plots, although Ganesh does seem to get solutions to the front edge of the Pareto plots. NSGA-II seems to achieve wider fronts at 1.0, while not seeming as well converged, which seems to be the reason that neither can be said to out-perform the other in all respects as shown by the Kruskal-Wallis tests.

Figure 4.7 has the results of NSGA-II plotted last, hence this set somewhat obscures the underlying results of MOTS and Ganesh, so the latter two are shown together in a separately plot, in Figure 4.8.

Figure 4.12 is a repeat of figure 4.11 but with a much later generation of a

Table 4.1: Kruskal-Wallis tests results for 3 independent data sets, comparing 20 sample runs per range for each of Ganesh (G), MOTS (M) & NSGA-II (N), showing p-values for $\alpha = 0.05$. See also continuation table 4.2.

Range	Ind	G >M	G >N	N >M	N >G	M >G	M >N
0.3	eps	2.76E-05	0.618462	9.64E-06	0.381538	0.999972	0.999999
0.3	hyp	8.99E-15	0.002448	5.56E-10	0.997552	1	1
0.6	eps	3.20E-17	0.000742	6.17E-12	0.999258	1	1
0.6	hyp	2.90E-18	0.000087	5.61E-12	0.999913	1	1
0.8	eps	1.44E-12	0.034613	1.70E-09	0.965387	1	1
0.8	hyp	1.24E-12	0.072964	3.43E-10	0.927036	1	1
1.0	eps	7.74E-08	0.263348	8.31E-07	0.736652	1	0.999999
1.0	hyp	4.33E-07	0.476934	5.37E-07	0.523066	1	0.999999

Table 4.2: Continuation of table 4.1, for Ganesh, MOTS, & NSGA-II, giving the interpretation of their relative performance.

Range	Ind	Best	2nd Best
0.3	eps	Ganesh/Nsga-II	Mots
0.3	hyp	Ganesh	Nsga-II
0.6	eps	Ganesh	Nsga-II
0.6	hyp	Ganesh	Nsga-II
0.8	eps	Ganesh	Nsga-II
0.8	hyp	Ganesh/Nsga-II	Mots
1.0	eps	Ganesh/Nsga-II	Mots
1.0	hyp	Ganesh/Nsga-II	Mots

Table 4.3: Means of the ε- and *hypervolume* indicators provided by PISA for Ganesh results.

Ganesh				
	Epsilon		Hypervolume	
Range	Mean	SD	Mean	SD
0.3	0.075457	0.039082	0.050667	0.023536
0.6	0.080692	0.033401	0.075564	0.032801
0.8	0.111012	0.032667	0.131357	0.044737
1.0	0.146081	0.056356	0.190044	0.101289

Table 4.4: Means of the ε- and *hypervolume* indicators provided by PISA for MOTS results.

MOTS				
	Epsilon		Hypervolume	
Range	Mean	SD	Mean	SD
0.3	0.155269	0.066874	0.122358	0.03185
0.6	0.317691	0.055242	0.401264	0.037391
0.8	0.251978	0.058163	0.387173	0.085911
1.0	0.234526	0.044423	0.351916	0.094109

Table 4.5: Means of the ε- and *hypervolume* indicators provided by PISA for NSGA-II results.

NSGA-II				
	Epsilon		Hypervolume	
Range	Mean	SD	Mean	SD
0.3	0.069424	0.026199	0.069150	0.015704
0.6	0.102902	0.021668	0.115390	0.029101
0.8	0.128359	0.028219	0.144089	0.042932
1.0	0.154388	0.037153	0.180101	0.046149

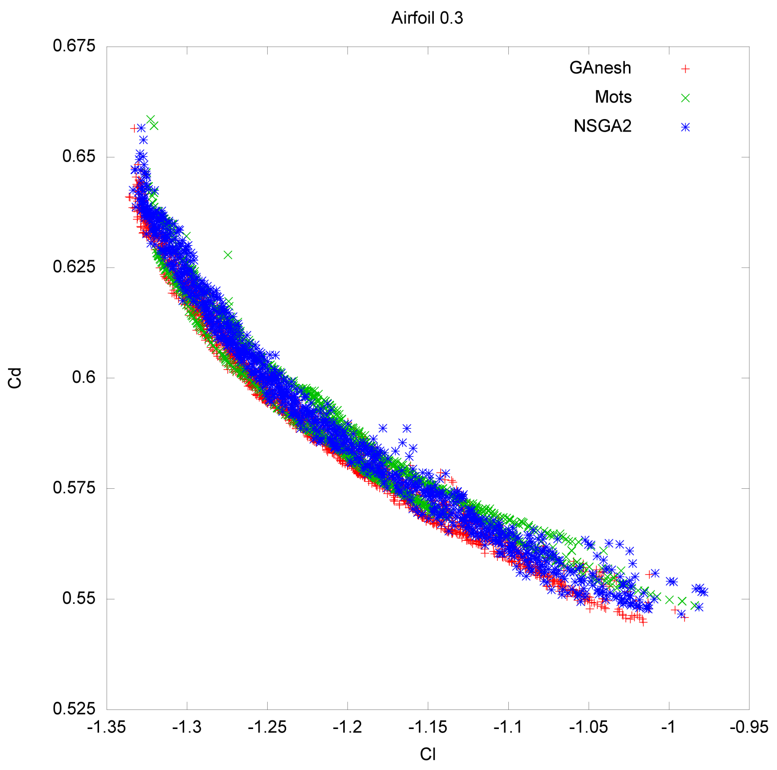


Figure 4.7: Results for range ±0.3. showing all samples of Ganesh, MOTS & NSGA-II.

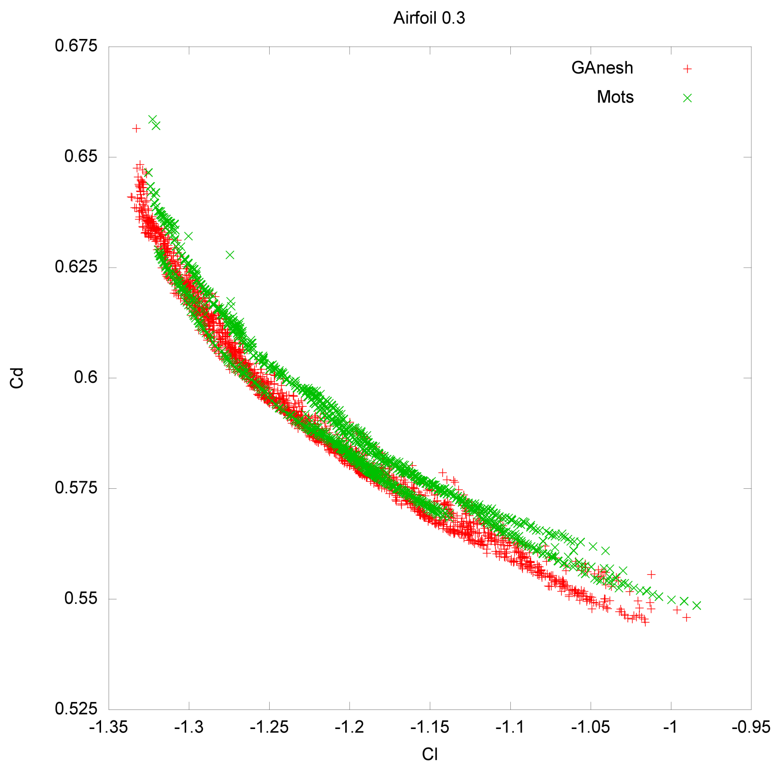


FIGURE 4.8: Results for range ±0.3. showing all samples of Ganesh & MOTS only.

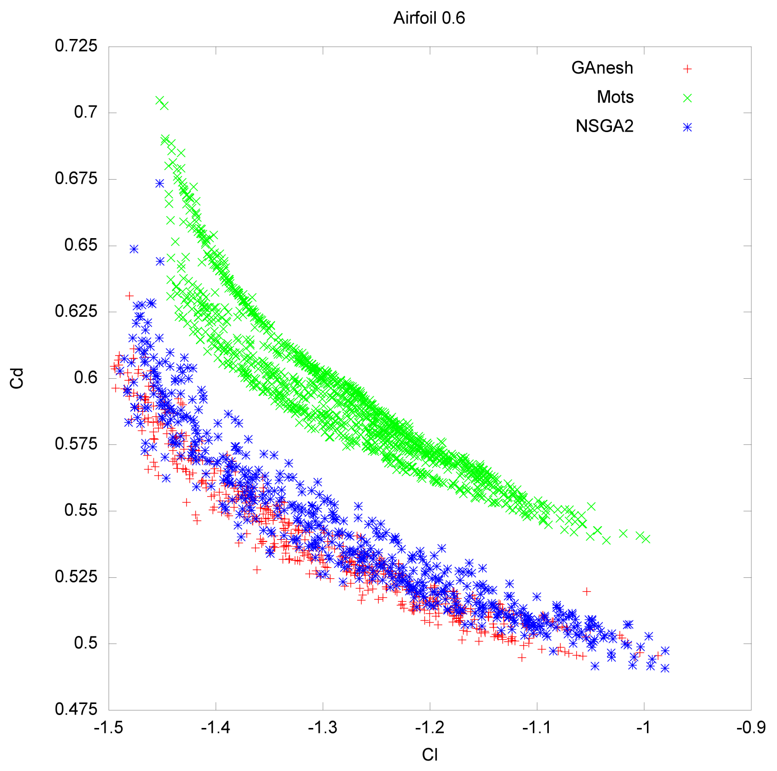


FIGURE 4.9: Results for range ±0.6. showing all samples of Ganesh, MOTS & NSGA-II.

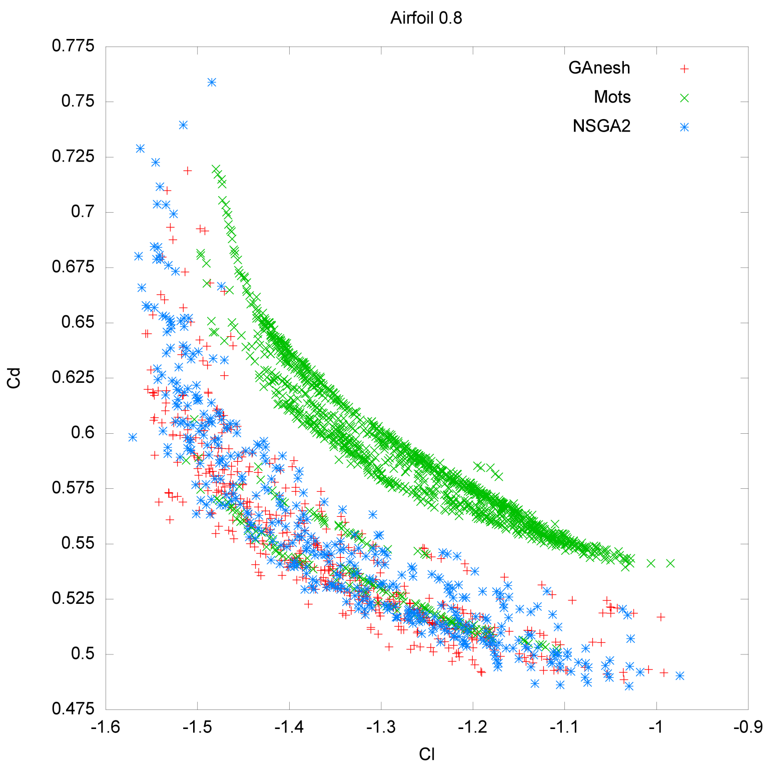


FIGURE 4.10: Results for range ±0.8. showing all samples of Ganesh, MOTS & NSGA-II.

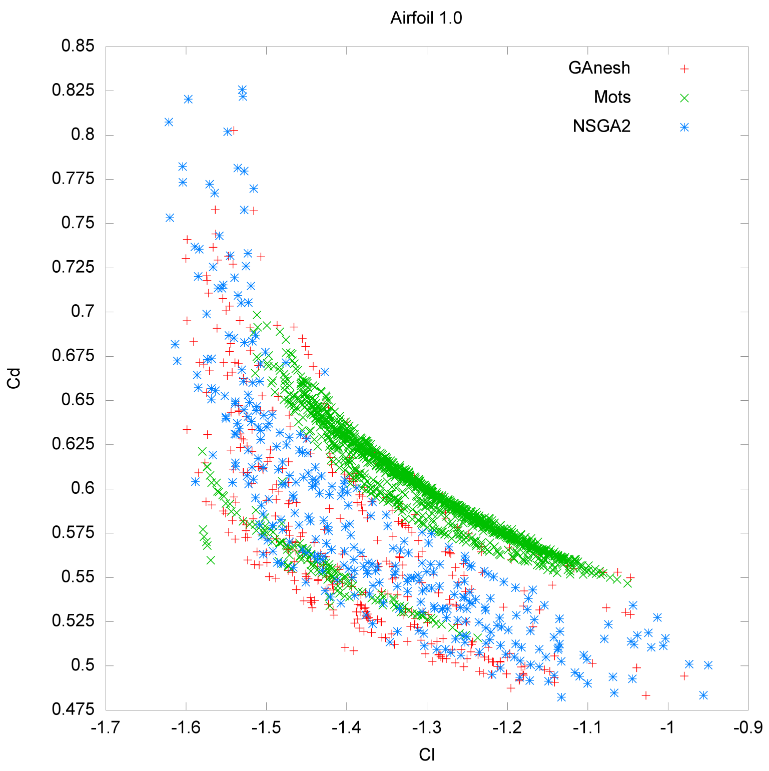


FIGURE 4.11: Results for range ±1.0. showing all samples of Ganesh, MOTS & NSGA-II.

Ganesh run: generation 863, which was the first generation in which all 120 solutions of the population of that generation were non-dominated (*after* the non-domination sorting and re-insertion of elite solutions, if any, which may have replaced solutions from the new child population). Prior to that point, every preceding generation had at least one dominated solution in it. As one would expect, this front dominates the others, but it also has quite a dense and wide front. It is reasonable to suppose that even this late generation is not yet the best performance that it might achieve, given that it has only just eliminated the last dominated solution(s).

A selection of the optimised airfoils found are shown in Figure 4.13, which is the Figure 4.12 re-scaled in order to show the original NACA 0012 airfoil relative to the optimised ones. The airfoil geometries are shown in relation to the position of their solution design vector in the Pareto approximation front, while the NACA 0012 datum design is shown as the red dot at the top right, in its correct relative position.

The compromise airfoil 'ffd-2867' from generation 863 is shown in figure 4.14, which is near the middle of the Pareto front approximation. This represents a compromise design having good lift and good drag coefficient values, which seems to result from a less significant boundary layer separation than the datum design. The graph above the airfoil section shows pressure coefficient distributions for the airfoil surfaces and boundary conditions. The parallel coordinates plot of 4.21 shows the design parameters for the selected approximate $\mathrm{C_L}$ of this airfoil (as explained further below in 4.5.1.3), while the original NACA 012 airfoil is plotted in figure 4.15 for comparison with that of the 'ffd-2867' airfoil, and Table 4.6 gives the airfoil solution, consisting of the free-form deformation parameters and their $\mathrm{C_L}$ and $\mathrm{C_D}$ objective function values.

In Figures 4.16 & 4.17 the means of the GA control parameters are plotted since each of the 120 solutions in each generation has its own value for each of these parameters. It can be seen that as the GA progresses through its generations, both pM and pC become smaller, hence the disturbance to good solutions is lessened,

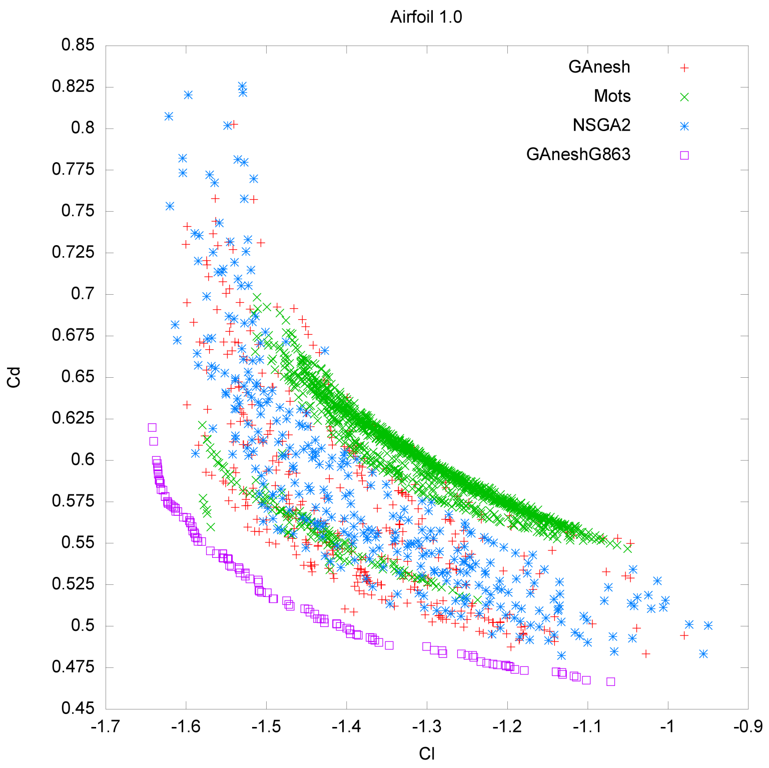


FIGURE 4.12: Results for range ±1.0. showing Ganesh, MOTS & NSGA-II, and GaneshG863 which is the first generation of Ganesh having only non-dominated solutions (in generation 863).

TABLE 4.6: The design vectors, consisting of the free-form deformation parameters (rounded to 5 decimal places), defining the airfoil geometries, and OFs, for the selected airfoils shown in Figure 4.13.

Airfoil	p1	p2	p3	p4	p5	p6	p7	p8	C_L	C_D
ffd-2867	-0.37958	0.08431	-0.09464	1.00000	-0.22016	0.78611	-0.42390	0.43285	-1.54727	0.54033
ffd-2753	-0.34021	0.27989	-0.07222	1.00000	0.25528	-0.51989	-0.40346	-0.18635	-1.13951	0.47252
ffd-2966	-0.33806	0.21757	-0.09718	0.99327	-0.35148	1.00000	-0.41564	0.90794	-1.63570	0.59800

while their respective polynomial distribution indices (ηM & ηC) become larger, which decreases the perturbation to the section geometry, thus at the start the GA is better at exploring the search space while towards the end it is better at converging to good solutions.

4.5.1.2 Control parameter trends

The figures of 4.18 and 4.19 show how the standard deviations of the control parameters vary across the generations. As can be seen, they start off low, since the parameters are set to the same value for every solution of the initial population

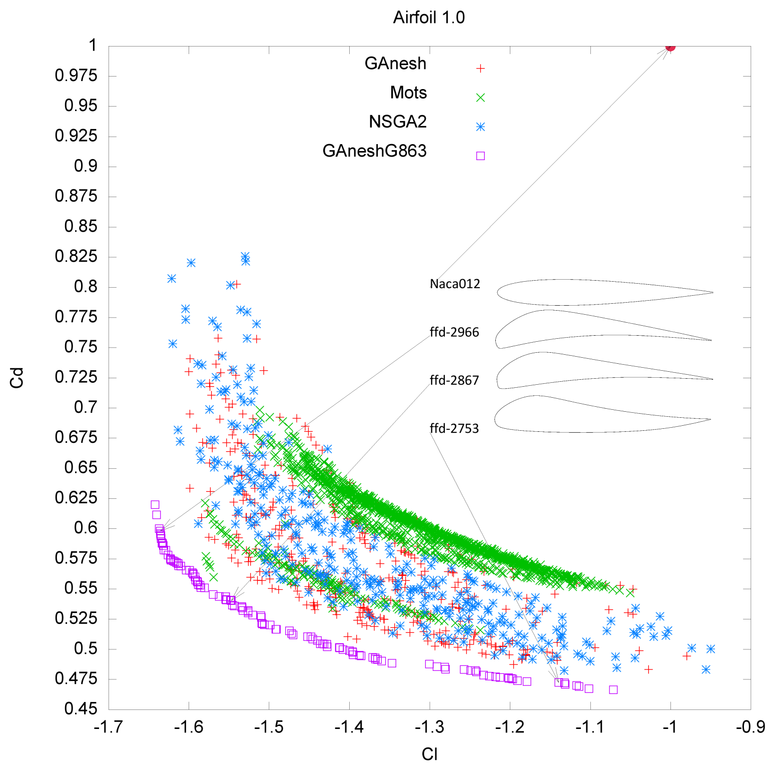


FIGURE 4.13: Results for range ±1.0. showing Ganesh, MOTS & NSGA-II, and GaneshG863 which is the first generation of Ganesh having only non-dominated solutions (in generation 863), with airfoils depicted.

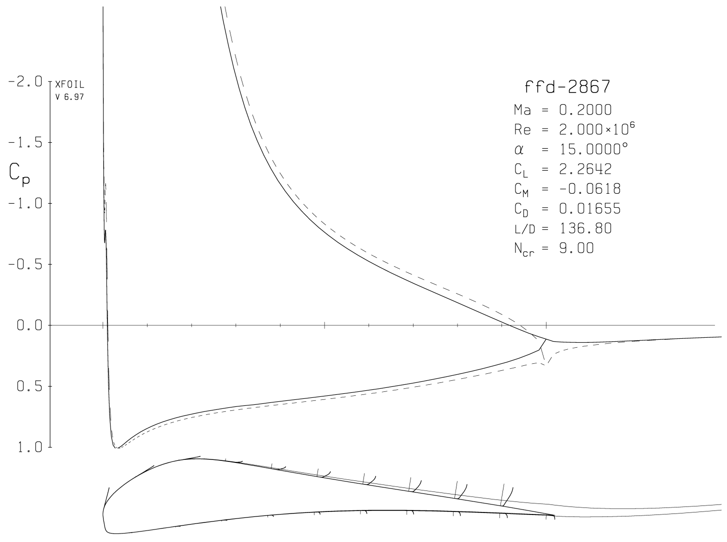


FIGURE 4.14: An Xfoil plot of an optimised airfoil ('ffd-2867') found by Ganesh at range ±1.0. in generation 863, having normalised coefficients $C_L = 1.547$, $C_D = 0.540$ (the values in the figure itself are not normalised), showing analysis of aerodynamic performance. The airfoil is shown underneath the graph of the pressure curve whose width corresponds to points along the airfoil from leading to trailing edges, and the computed boundary layer separation is shown.

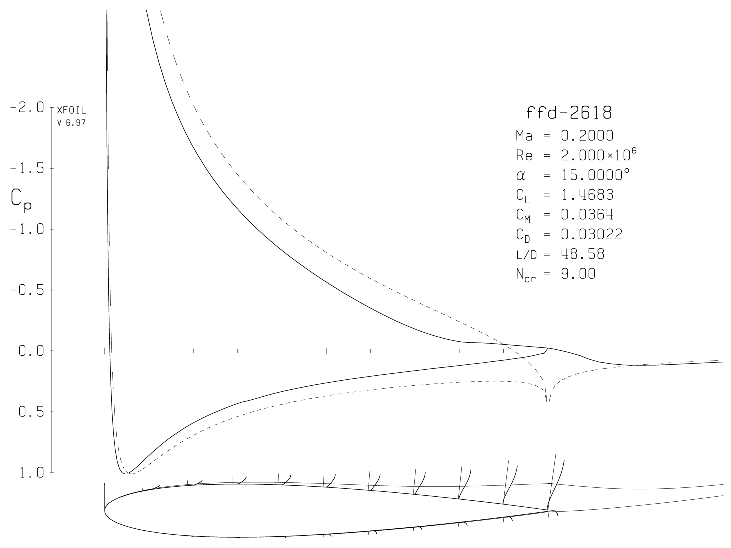


FIGURE 4.15: An Xfoil plot of the unoptimised NACA 012 airfoil, showing analysis of aerodynamic performance. The airfoil is shown underneath the graph of the pressure curve whose width corresponds to points along the airfoil from leading to trailing edges, and the computed boundary layer separation is shown.

at the start of the run, then increase rapidly at the early stages, as random chance determines whether they mutate or recombine. At about half way through the rate of change starts to decrease until towards the end of the run it seems the maximum deviation has been reached and they level out, probably because most of the solutions now have different values for these parameters as they are real numbers and the values are spread, yet the spread is also governed by the relevant η control.

The trends of the controls (pC & pM) themselves do however seem to indicate that overall the solutions are evolving over time towards an exploitative mode, in that the probability of mutation or crossover occurring is seen to decrease, and that also the η controls are increasing which means the degree of perturbation caused by the operator acting is decreasing too. It should be noted that this response is not pre-determined or programmed in, it is an emergent behaviour of the system over time.

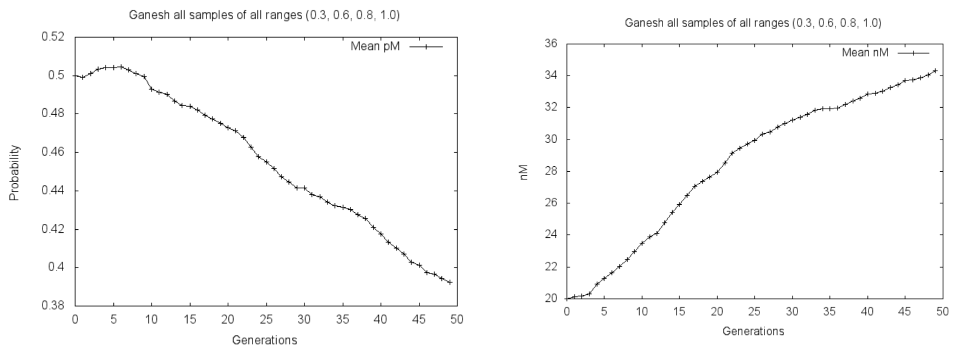


FIGURE 4.16: Trends of the means of the pM & ηM control parameters against generation number for all ranges for all samples for Ganesh.

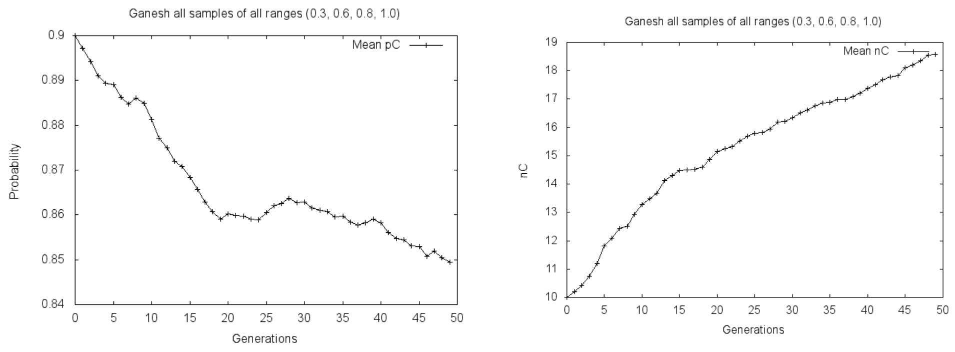


FIGURE 4.17: Trends of the means of the pC & ηC control parameters against generation number for all ranges for all samples for Ganesh.

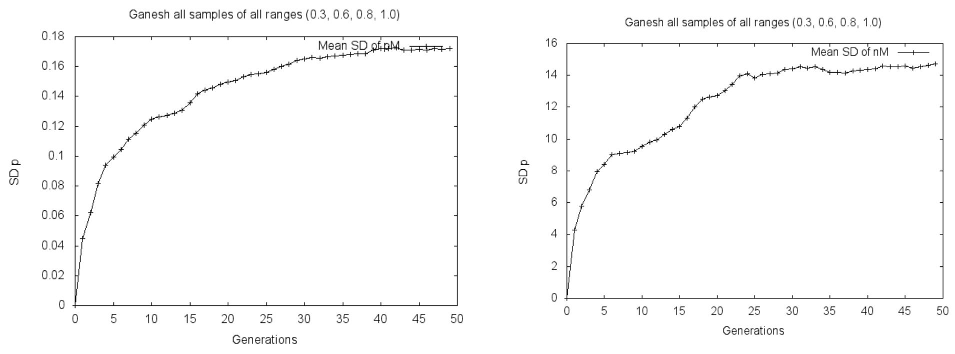


FIGURE 4.18: Trends of the standard deviations of pM & ηM control parameters against generation number for all ranges for all samples for Ganesh.

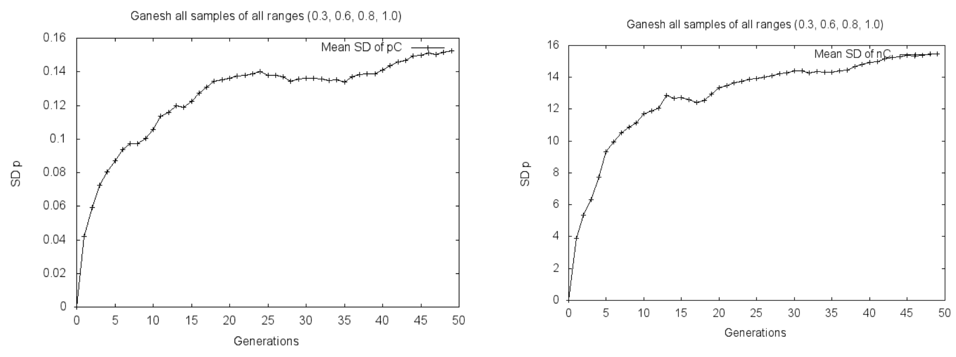


FIGURE 4.19: Trends of the standard deviations of pC & ηC control parameters against generation number for all ranges for all samples for Ganesh.

4.5.1.3 Visualising results with parallel coordinates

Here, results are presented as plots in parallel coordinates (||-coords), the technique devised by Inselberg (2009) and later used in the field of optimisation by Fleming *et al.* (2005), Siirtola and Räihä (2006), Siirtola (2000), and in engineering design by Kipouros *et al.* (2008) and Kipouros *et al.* (2013), in which each dimension is oriented parallel to the others, thus transforming an n-dimensional point into a 2-dimensional polygonal line that relates the values in each dimension. This approach enables highly multi-dimensional data to be plotted uniquely and without loss of information, and here the entire design space of each solution, 8 variables and 2 objective function results, are plotted together. The plots were produced using the Parallax tool (Avidan and Avidan, 1999). Such plots have the advantage over tables of numbers since the inherent human visual pattern recognition ability is leveraged, enabling the relationships in the data to be far more obvious. Such plots, that summarise large amounts of data as they do, also have the virtue of brevity.

Thus figure 4.20 is a ||-coords plot that shows the eight parameters and two objective functions (C_L and C_D) of the design vector of all solutions in generation 863 of a run for range ±1.0, as shown in figure 4.12. This shows parameter 6 has been selected for value 1, the value that seems to achieve the greatest lift, while the lowest drag has also been selected, showing that the least drag corresponds to the least lift (as lift here is a negative amount as explained previously). As might be expected, the opposite value of parameter 6 is selected for least drag, while high values of p5 and p3 are also selected, both of which are antagonistic for lift. Interestingly, p4 seems to have high values for both lift and drag.

Figure 4.21 shows the same data plotted in Figure 4.20 again as a ||-coords plot, but with the data of the selected optimal airfoil of 4.14 highlighted as shown by the selection markers with the data line shown in magenta. The selected airfoil is of course a compromise between high lift and low drag and is chosen from a region of the PFapprox front which is more biased towards the higher lift end of the continuum, as can be seen by the position of the markers in the C_L axis.

The mainly parallel lines seen between parameters (p3 & p4) and (p4 & p5) indicate that those parameters, for certain values, tend to be positively correlated, while conversely, the lines crossing between the axes of (p5 & p6) and (p8 & C_L) show that these are negatively correlated. That parameters p4 and to a lesser extent p5, have most of their points passing through a relatively narrow band of values, indicates that they are important and sensitive.

The inter-dependency between certain components of the decision vector, shown by the correlation mentioned above, suggests that an alternative crossover operator could be of benefit, for example a multi-point crossover or perhaps even better, a selective crossover (Vekaria and Clack, 1998), rather than the uniform one currently used, since the uniform version is more likely to disrupt the co-variance (known as 'epistasis').

The plot of Figure 4.22 shows the data for all samples of range ± 0.3 of Ganesh, in which the best results for C_L are selected by the highest values of parameter p8 and given in green; the best values for C_D are selected by the lowest values of parameter p8 and given in magenta, and some compromise solutions roughly mid-way for both C_L and C_D are selected on the C_L axis and given in blue. There are noticeable regions showing horizontal lines between adjacent axes, between p4, p5 & p6 for blue, and between p6, p7 & p8 for magenta. This arises because the solutions have evolved values for these parameters at the maximum extent of their permitted ranges, in this case ± 0.3. This is indicative that the range is too

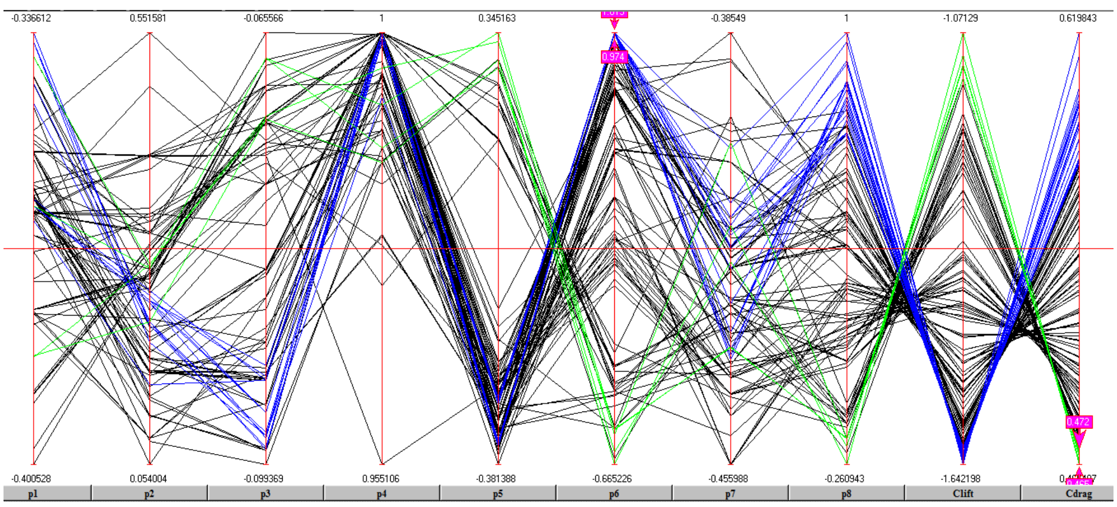


FIGURE 4.20: ||-coords plot showing Ganesh results for range ±1.0. at generation 863, which is the first generation having only non-dominated solutions. The solution with the lowest C_D has been selected on the C_D axis with its data shown in green, and the maximum of p6 has been selected with its data shown in blue.

restrictive and that a wider range may produce solutions that are more converged towards a global optimum. It can be seen that the blue region, representing some compromises, has wide ranges of values around mid-way of p8, but also in p5, p3, p2 and p1, which suggests that although p8 is an important parameter, the sensitive interaction with those others parameters determines the compromise solutions which are more likely to be of value, hence the inter-dependence of these parameters is important. It should be borne in mind that all the solutions shown are of the best rank and are all non-dominated, thus each one represents a reasonable possible choice for the airfoil, depending on where the decision maker wishes to trade-off between lift and drag performance, which in turn depends upon the intended purpose of the aircraft.

4.5.1.4 Visualising combined data sets with parallel coordinates

In order to gain a qualitative understanding of how the three algorithms under consideration work in practice, from a data perspective and relative to each other, ||-coords plots were used in a novel fashion to visualise all the data sets together.

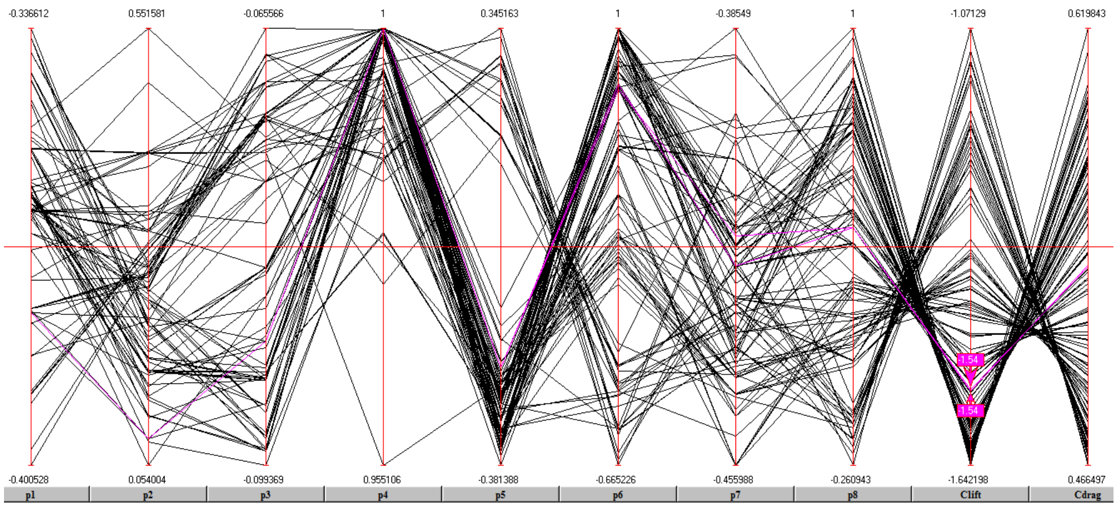


FIGURE 4.21: ||-coords plot showing Ganesh results for range ±1.0. at generation 863, with the optimal airfoil of Fig. 4.14, selected by C_L as shown by the markers.

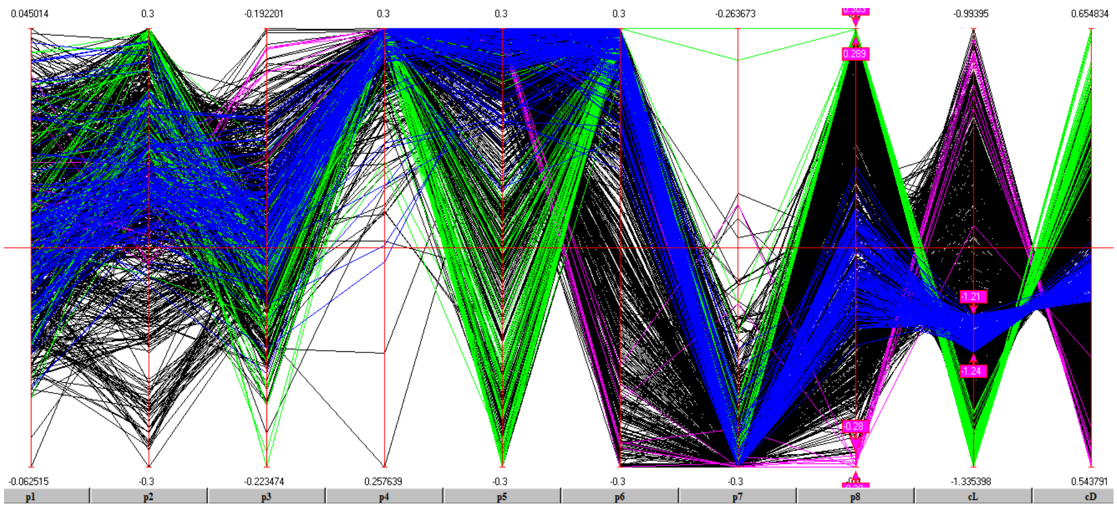


FIGURE 4.22: ||-coords plot showing Ganesh results for range ±0.3, for all 20 samples, in which the best C_L solutions are selected for in green, the best C_D solutions are selected for in magenta and some compromise solutions are shown in blue.

Here, the data sets are treated with an R (R Development Core Team, 2014) script to gather the data from all samples of all ranges from each algorithm, and to *min-max* normalise the 8 parameters and 2 objective function results, which is also known as *feature scaling*. The values are normalised into the interval [1,0] except for the coefficient of lift, which is scaled into [-1,0] to keep it consistent with the underlying data negative value, and Equation 4.9 gives the normalisation into the interval$[l, h]$.

$$y = \frac{x - min(x)}{max(x) - min(x)} \cdot (h - l) + l \tag{4.9}$$

The R script normalises all the data as a combined set but also maintains them as separate matrices, and the output data has appended a key for the algorithm and gives the range for each matrix, enabling the data to be separated when viewing also.

Using the Parallax tool, the plot axes are then scaled to the same intervals, thus enabling the data from these disparate sources to be viewed together, and the following figures in this section all show ||-coords plots of the normalised and scaled data.

Figure 4.23 shows the ||-coords plot for all samples of all ranges of all algorithms, normalised and scaled as set out above. In addition to the 8 airfoil parameters and the 2 objective function values, the first parallel axis, on the left, shows a discrete number that is a key for the algorithm, with 0 for Ganesh, 1 for MOTS, and 2 for NSGA-II, while the next parallel axis shows the number for the range: 0.3, 0.6, 0.8 and 1.0. These axes enable the 20 samples of a range for an algorithm to be selected for highlighting independently of the others while retaining the other data as a backdrop for visual comparison.

Similarly, Figure 4.24 shows the same data as Figure 4.23, but here the Ganesh data for range ± 0.3 is highlighted in magenta, by selecting the appropriate values from the Alg and Rng axes as described above. The magenta region is therefore effectively a zoomed-out equivalent of the plot shown in Figure 4.22, enabling the cut-off zones of the limit of the ± 0.3 range to be clearly seen as the horizontal delineation between the parameters as given above.

Continuing in this theme, Figure 4.25 is similar to Figure 4.24, but shows Ganesh data for range ± 1.0, in which it can be seen that now the parameters are allowed to extend further thus eliminating the horizontal lines, although parameters p2, p4, p6 and p8 all have values at the maximum permitted. Allowing greater extension in the parameters has enabled the objective functions to converge more,

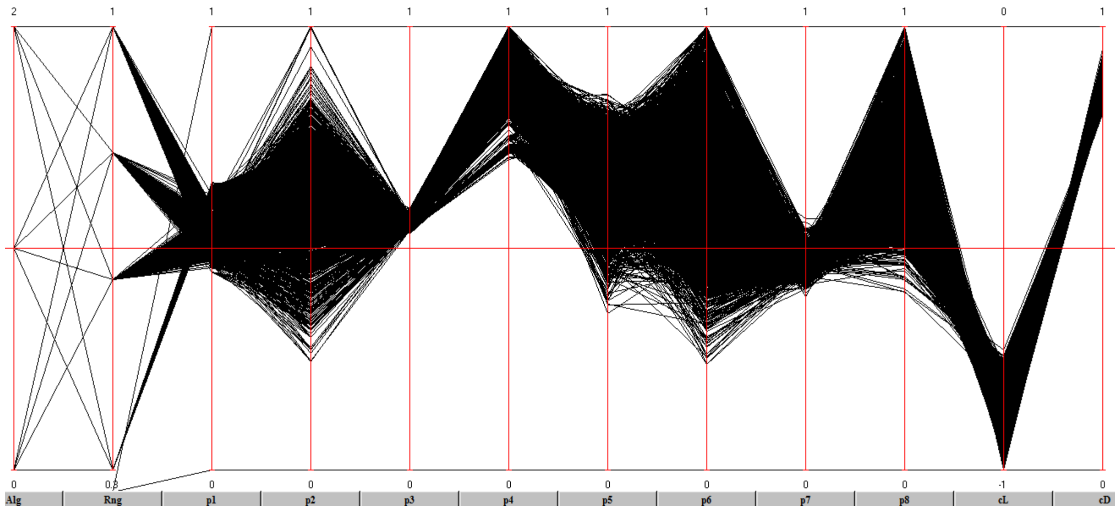


FIGURE 4.23: ||-coords plot showing normalised and scaled data from all samples of all ranges of all algorithms. The left-most axis gives the algorithm key (0:Ganesh, 1: MOTS, 2: NSGA-II) and next to it the range is given (±0.3, ±0.6, ±0.8, ±1.00).

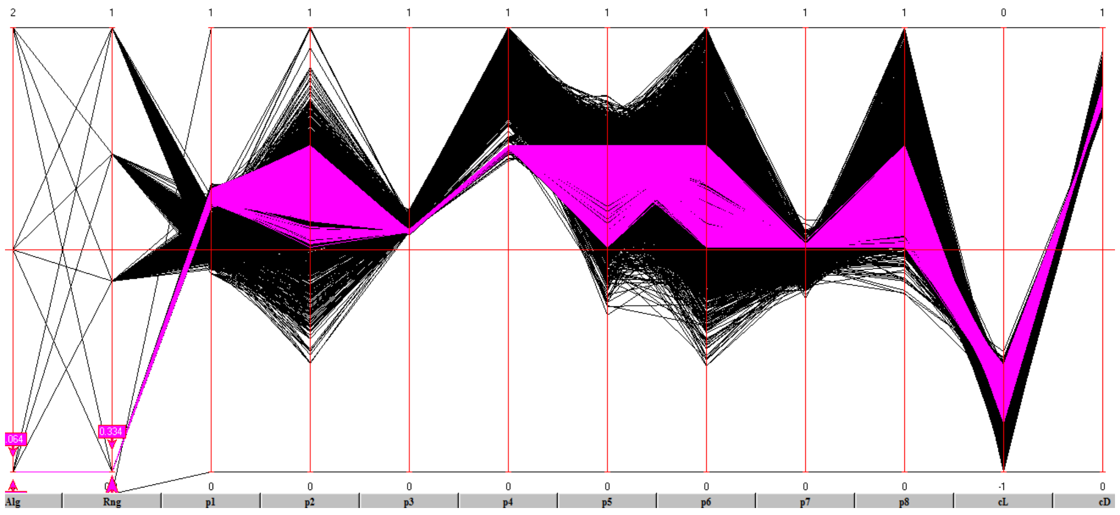


FIGURE 4.24: ||-coords plot showing normalised and scaled data from all samples of all ranges of all algorithms, as in Figure 4.23, with the Ganesh data for range ±0.3 highlighted in magenta.

thus the axes for C_L and C_D are now coloured magenta at their best value regions, and it can be seen that much more of the search space is coloured magenta too. The Ganesh data for ranges ±0.6 and ±0.8 are naturally in between the ±0.3 and ±1.0 plots and are not particularly instructive to show, as they do not illustrate any additional features.

It is interesting to note that parameter p3 seen at this scale, has a narrow band of values even at the ±1.0 range. This parameter specifies the horizontal

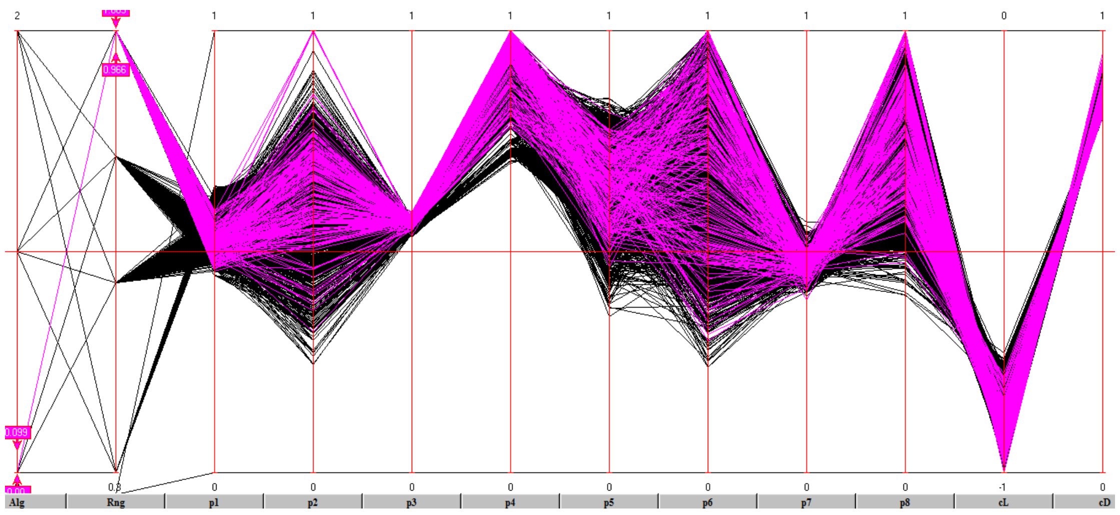


FIGURE 4.25: ||-coords plot showing normalised and scaled data from all samples of all ranges of all algorithms, as in Figure 4.24, but with the Ganesh data for range ±1.0 highlighted in magenta.

displacement of the control point on the top left of the FFD hull, and so is the main control of the shape of the upper surface of the leading edge of the airfoil, which obviously exerts great influence over the aerodynamic performance for both lift and drag. Similarly, but to not quite such great an extent, parameter p7 has a narrow band of values too and is the horizontal displacement parameter at the top right of the FFD hull thus controlling the shape of the upper surface of the trailing edge.

In Figure 4.26, the data for all ranges of MOTS is highlighted in green and superimposed on the region highlighted in Figure 4.25, thus MOTS overlays the Ganesh range ±1.0 data. It is apparent that the MOTS exploration is more regular than the Ganesh GA strategy, by the appearance of the regular gaps between MOTS points on parameter axes, and it is noticeable that it has explored less of the parameters p2 and p6 spaces than Ganesh. Figure 4.27 highlights, in grey, the parameter 6 space covered by Ganesh but not by MOTS, by selection on the parameter 6 axis, showing that it is this region that has helped Ganesh out-perform MOTS in the best coefficient of drag region.

Continuing from Figure 4.27, Figure 4.28, highlights in blue the parameter 6 space covered by NSGA-II but by neither Ganesh nor MOTS, which shows that

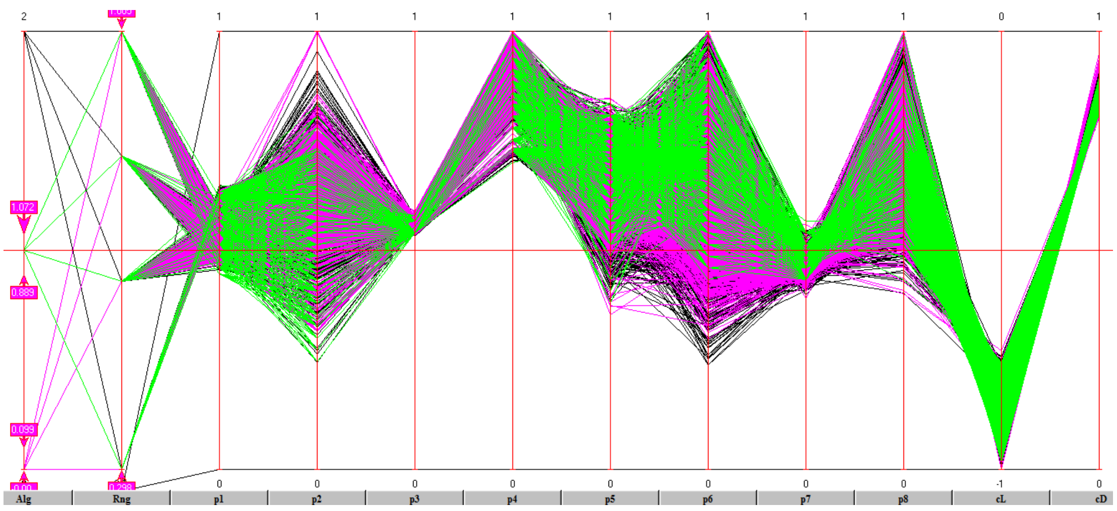


FIGURE 4.26: ||-coords plot as in Figure 4.25, but with the MOTS data for all ranges highlighted in green and superimposed.

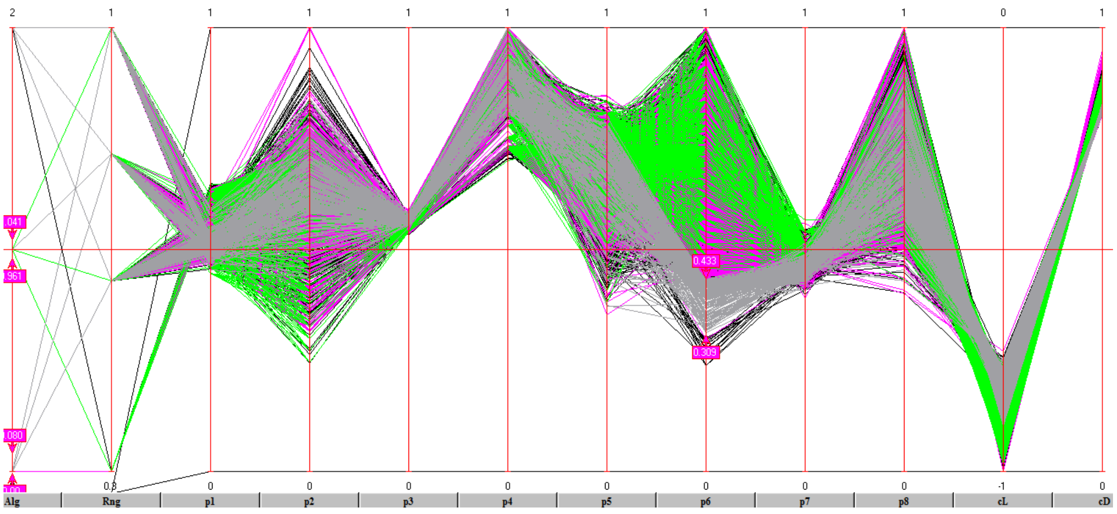


FIGURE 4.27: ||-coords plot as in Figure 4.26, highlighting the parameter 6 space covered by Ganesh but not by MOTS, in grey.

this region exists inside the grey area defined by Figure 4.27, thus enabling NSGA-II to perform better than MOTS but not quite as well as Ganesh for the lowest C_D values influenced by this region of p6. Figure 4.29 is a zoom-in plot of Figure 4.28, showing just the detail of the C_D axis, in which the effects of the above data selection are more easily seen, and in which the black area is due to other NSGA-II data not selected so far.

Figure 4.30 follows up the exploration of the best C_D region shown in Figures 4.28 & 4.29, changing the colours of Ganesh and MOTS data to grey and dark green respectively, leaving NSGA-II in black, to make it easier to see the selection

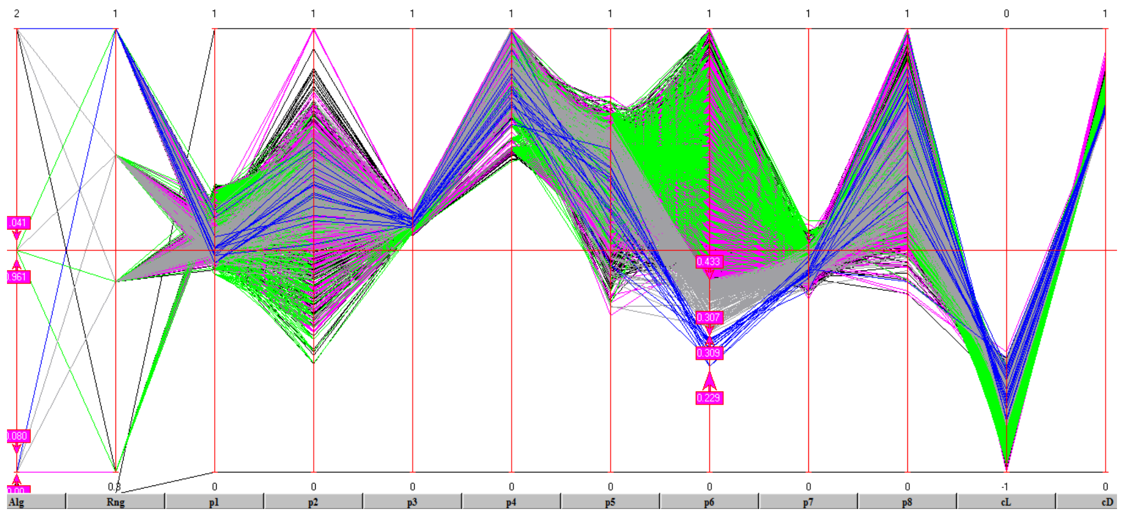


FIGURE 4.28: ||-coords plot as in Figure 4.27, highlighting in blue the parameter 6 space covered by NSGA-II but not by Ganesh or MOTS.

FIGURE 4.29: ||-coords plot zoom in of Figure 4.28 showing the C_D axis detail.

of the best C_D values shown in yellow. This makes it clear that all 3 algorithms cover, in broad terms, the search spaces of each parameter's values for the selected good performing region for C_D, thus each algorithm could in theory have attained the best C_D found. However, for MOTS, the relevant region of p8 is at the extreme end of the values found and the points are more sparse here than is the case for Ganesh or NSGA-II, and this is true for p4 also.

The Figure 4.30 also indicates that for C_D at least, it is the combination of non-extreme parameter values that produce a good result. The search behaviour of MOTS, put simply, is to use regular step sizes by which to vary a parameter, and this regularity can be seen. The step size is an important factor in determining the likelihood of finding a good value of a given parameter, and in the case here, it seems the step size may have been too large, for some of the duration of the run, at least. Depending upon the step size chosen, the starting position could be important too, as it may create a bias to the number of odd or even values being found. MOTS does adjust its step size during the run, but apparently has not made the best choices here. Ganesh uses the 64-bit floating point data type for parameters and objective function values (as does NSGA-II), which coupled with its stochastic crossover and mutation operators, leads to values of far greater precision (having more places after the decimal point) and thus greater exploration potential in this regard, than MOTS. Nevertheless, as seen in Figure 4.11, MOTS did find some good compromise solutions.

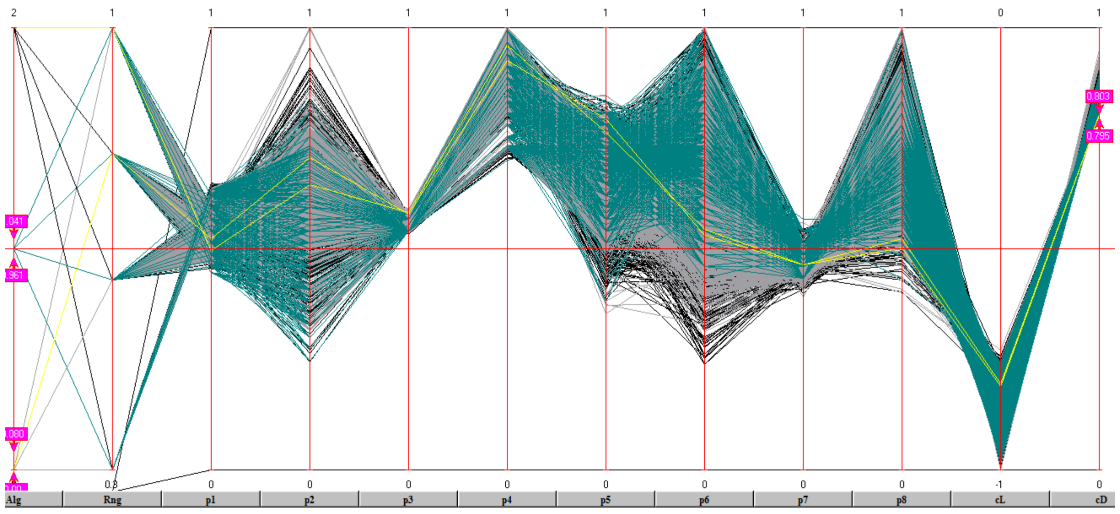


FIGURE 4.30: ‖-coords plot for all samples of all ranges of all algorithms, but with Ganesh data colour shown in grey and MOTS data in dark green to make it easier to see the best C_D values selected and highlighted in yellow, while the NSGA-II data is black.

4.5.2 Alternative crossover operator

As described above in discussing Figures 4.21 and 4.20, it was thought that possibly a multi-point crossover operator might out-perform the uniform crossover operator used for the airfoil optimisation problem.

In order to test this, a new problem class was derived from the existing one, thus inheriting all the problem component definitions, but which overrides the crossover operator, instead specifying the self-adaptive multi-point swap crossover (SAMPSC).

The SAMPSC is a novel operator that is similar to and derived from the self-adaptive uniform blend crossover (SAUBC) defined in Section 3.2.2 and depicted in Figure 3.2, however in SAMPSC whole genes are swapped between the parents, not blended as with the SBX gene operator. In the case defined here, SAMPSC is used with two randomly chosen chromosome recombination loci generated for each crossover operation event, since it has been shown (De Jong and Spears, 1992) that with two loci, a crossover operator has a reasonable balance between productivity (a measure of its creating diversity in the population) and disruption.

It is recognised (Deb, 2001) that a real gene non-blending swap has less search power than a blending one, but also that this can be offset by a higher mutation rate, thus the self-adaptivity of the operator should manage to overcome this limitation with the proviso that it has enough time to run.

The same scenario was run again in the same manner as before, with 20 independent runs with randomly chosen initial populations, for the same four ranges for parameter modifications, in order to generate data to compare with the data of the previous set of Ganesh.

4.5.2.1 Statistical analysis

The results obtained by these new sample Ganesh runs were compared, using PISA, with the original Ganesh results, thus comparing the effects of the two different crossover operators, and the PISA comparison results are shown in Table 4.7. In this case, the Mann-Whitney two-tailed test (Hollander *et al.*, 2014) is used to compare the result sets; this test is similar to that of Kruskal-Wallis, but is for specifically two sample groups.

Here, as before, the significance level is taken as $\alpha = 0.05$ and the test is a non-parametric one, so compares ranks rather than raw data, and the test is for mean ranks, where $H0$ is that any difference seen is within normal bounds for randomly fluctuating values and that therefore the sample groups can be considered to be from the same population. This means that neither operator can be thought of as 'better' than the other. Conversely, if $H0$ is rejected, this means that, for this problem, the relevant operator can be thought of as providing better performance than the other.

Table 4.7 shows that for all ranges, except that of 1.0, $H0$ is not rejected, however, at range 1.0 $H0$ *is* rejected and the SAMPSC operator seems to improve performance for both the ϵ-indicator and the hypervolume indicator. It is worthy of note that the range 1.0 is the most difficult to optimise, so it may be that SAMPSC is helpful in this regard since it is less disruptive of good solutions.

Tables 4.8 & 4.9 give the means and standard deviations of both indicators for the new and original results.

It seems that for this problem, SAMPSC would be a good choice since it behaves no worse than the uniform crossover (SAUBC), yet performs better for the most difficult range. The analysis by ||-coords of the original results was instrumental in encouraging this investigation and thus a major factor in arriving at this conclusion.

Table 4.7: Mann-Whitney test results for 2 independent sample groups, comparing 20 sample runs per range for the new Ganesh (G2) results and the original Ganesh (G1) results, showing p-values for $\alpha = 0.05$.

Range	Ind	G2 > G1	G1 > G2	Best
0.3	eps	0.767475	0.232525	$H0$
0.3	hyp	0.705799	0.294201	$H0$
0.6	eps	0.061467	0.938533	$H0$
0.6	hyp	0.456898	0.543102	$H0$
0.8	eps	0.941708	0.058292	$H0$
0.8	hyp	0.705791	0.294209	$H0$
1.0	eps	0.008566	0.991434	G2
1.0	hyp	0.009911	0.990089	G2

Table 4.8: Means of the ε- and *hypervolume* indicators provided by PISA for Ganesh new SAMPSC results.

	Ganesh			
	Epsilon		Hypervolume	
Range	Mean	SD	Mean	SD
0.3	0.926929	0.001036	0.913583	0.000675
0.6	0.086694	0.017940	0.104108	0.027463
0.8	0.099363	0.029108	0.134173	0.056580
1.0	0.177642	0.072984	0.214561	0.107224

Table 4.9: Means of the ε- and *hypervolume* indicators provided by PISA for Ganesh original uniform crossover results.

Ganesh				
	Epsilon		Hypervolume	
Range	Mean	SD	Mean	SD
0.3	0.927084	0.000854	0.913568	0.000428
0.6	0.078459	0.014635	0.099121	0.031876
0.8	0.114916	0.039931	0.145594	0.071164
1.0	0.134460	0.044184	0.154040	0.063444

4.6 Airfoil comparison with other work

Having compared the results of airfoil optimisation by Ganesh with optimisations by the MOTS and NSGA-II algorithms as set out earlier in this chapter, this section compares the optimisation results of this chapter with other work in which airfoil optimisations have been performed, where there is a meaningful basis of comparison.

Results from other work are compared with the Ganesh results shown in Figure 4.32, previously described earlier in this chapter. Here, this figure denormalises C_L & C_D (they were originally normalised by dividing by the NACA 0012 datum values) and makes C_L positive. The figure gives C_L & C_D at an angle of attack (α) of $\alpha = 15°$ as this was used in the original optimisation. Since the values of C_L & C_D depend upon α, it is necessary to use Xfoil again on the Ganesh airfoils, to produce values of C_L & C_D for them, at the α used by the other works, in order to make comparison possible.

Table 4.13 provides the FFD parameters of the selected Ganesh airfoils annotated in Figure 4.32. The Ganesh optimisation, as stated previously, places constraints on the minimum airfoil thickness as it is intended to be used as a wing cross section (see Figures 4.4 & 4.5) thus space is preserved for strengthening spars. NACA 0012 has been used in a number of aircraft wings as either root or tip (Lednicer, 2010) and is shown in Figure 4.1.

Ram *et al.* (2014) performed an optimisation of the NACA 0012 airfoil using MATLAB to provide an implementation of a genetic algorithm, Xfoil to provide coefficients of lift (C_L), drag (C_D), and pressure (C_P), and ANSYS Fluent, after the optimisation, for computational fluid dynamics (CFD) calculations of the airfoil performance to validate the Xfoil results. Since they used a low airflow velocity of 20 m/s for their Fluent simulation, it seems they were assuming an application of a wind turbine or similar, rather than for aircraft wing application, and the references they cite reinforces this assessment. They modify the airfoil geometry (though it is not quite clear how) and use Xfoil for analysis, which in their case means evaluating lift and drag coefficients at two different angles of attack (α), of $\alpha = 0°$ and $\alpha = 3°$.

They used a single objective function to calculate a weighted sum of, C_D at both $\alpha = 0°$ and $\alpha = 3°$, and the maximum pressure coefficient (C_P) at $\alpha = 3°$, in which the first two are given equal weighting and the latter with diminished weighting. Although their stated aim is to maximise lift and minimise drag, there seems to be no mechanism through which an effect on C_L would be influenced, thus the process seems to concentrate on changing C_D. Their results bear this out. After the optimisation, their Fluent results gave a slightly improved estimation of C_L and a slightly worse C_D of the selected airfoil.

Table 4.10 shows the Ram *et al.* (2014) airfoil together with a Ganesh airfoil (ffd-2268) of nearest performance, for $\alpha = 3°$. Figure 4.33 shows an Xfoil plot of the airfoil and its section. It can be seen that the Ram airfoil has slightly worse lift than the selected Ganesh one, but better drag and thus a better lift/drag ratio. This is likely to be because Ram's optimisation is drag oriented, but also possibly because their airfoil is allowed to be thinner than the Ganesh constraint allows. Ram used 1,000 iterations of their optimiser, but they do not define the population size used.

Studies by both Selvan (2015) and Gardner and Selig (2003) use an angle of attack of $\alpha = 5°$ so a number of Ganesh airfoils were reassessed at this angle and ones of comparable performance are shown with their values for C_L & C_D in Table 4.11, along with the results from the other works.

Selvan (2015) was a study that set out to compare shape parameterisation methods as used in the derivation of airfoils in optimisation processes. Airfoil geometries were created from the reference NACA 0012 by using Latin Hypercube sampling from a Design of Experiments based on parameters, rather than random generation. They use their own genetic algorithm as optimiser but it uses results from a surrogate model as objective functions in an inner iterative loop, and Xfoil is used to increase accuracy in an outer iterative loop. Their single objective was to find an airfoil, within the design space they constrain, having a maximised C_L

for the same C_D as their NACA 0012 reference, and they use a single objective function which uses a weighted approach. Their results in Table 4.11 are given by the shape technique employed, given under 'Airfoil Id'.

Ganesh airfoils ffd-3301 and ffd-3337 were found to be the closest ones, in terms of their C_L & C_D values for the same angle of attack, with ffd-3337 having higher lift and higher drag compared to Selvan's results around $C_L \approx 0.6$, with a slightly worse C_L/C_D ratio of 71.897. The Ganesh airfoil ffd-3337 was closest to the Selvan results around $C_L \approx 0.7$, again having higher lift and drag but having a better C_L/C_D ratio of 82.975. Ganesh has therefore performed quite well in comparison. It was not clear how their optimisation was specified exactly but presumably the use of the surrogate model meant they could have many cheaper function evaluations, compared with Ganesh.

The work of Gardner and Selig (2003) used an airfoil shape generator called Profoil, and hybrid genetic algorithm with local search to optimise airfoil designs by generating airfoils using an 'inverse method from velocity distribution parameters'. The local search method employed a conjugate gradient method they say has been established to decrease convergence times by a third in airfoil applications. They also compared this approach with a number of others including a simple GA (SGA). Xfoil was used to analyse the airfoil performance characteristics as in the other works.

Their work used the Eppler 168 (E168) airfoil as the datum in their 'cambered airfoil design' optimisation, which set out to try to find a cambered airfoil that outperformed the state-of-the-art, being a 10% thick cambered airfoil with a maximum L/D ratio of 104 with a $C_L = 0.92$ (for a Reynolds number of 300,000). The optimisation therefore consisted of a fixed C_L with minimisation of C_D under the above conditions. The Reynolds number used is equivalent to a speed of around 22mph at sea level.

Ganesh airfoils from the existing result set were examined and relevant ones chosen (ffd-3705, ffd-3431 & ffd-3649) for comparison in consideration. Figure 4.31 shows the state of the art boundary for airfoil performance at the time their work

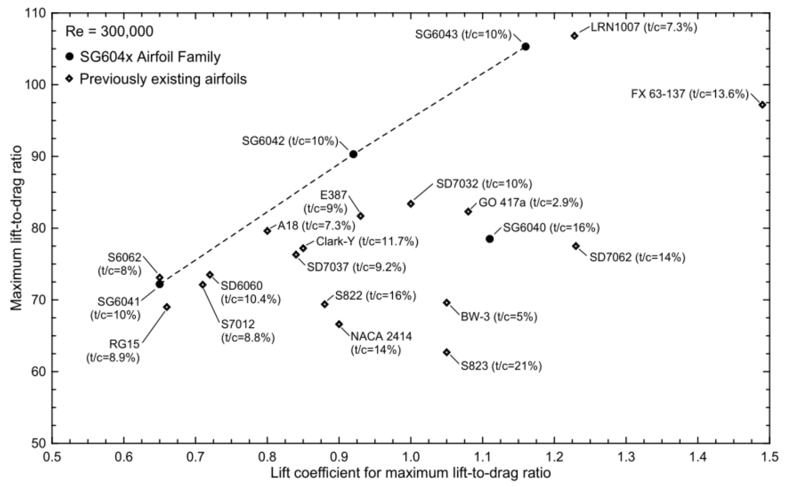


FIGURE 4.31: State of the art airfoil performance (Gardner and Selig, 2003).

was published, and their result ('Gardner') of Table 4.11 shows that they achieved their aim.

From Table 4.11 it can be seen that the Ganesh airfoil ffd-3705 has a C_L just under their figure of 0.92 and a higher C_D achieving an L/D ratio of 98.8, while ffd-3431 has a C_L just over their at 0.9298 with an increased C_D but with a slightly better L/D ratio of 99.98. The Ganesh airfoil ffd-3649 with a C_L of 1.02 and C_D of 0.00966 achieves an L/D ratio of 105.8 which compared with the performance boundary given in Figure 4.31 shows it to be above the high performance line for that C_L.

However, the original Ganesh optimisation used Xfoil working with a Reynolds number of 2×10^6, a substantial difference from the 3×10^5 used in the work being compared. Therefore the Ganesh airfoils were re-appraised at the lower Reynolds number, using Xfoil interactively, and the performance obtained rendered the previous results inappropriate for comparison.

Therefore a trial optimisation was carried out to investigate whether the Ganesh scenario could be appropriate for comparison at all, with Xfoil set to the same angle of attack ($\alpha = 5°$) and Reynolds number, but still using NACA 0012 as a reference datum. Two inequality constraints were also put in place, $0.915 \leq C_L \leq 0.92$, to approximate the C_L used in the comparison work. The newly obtained results are shown in Table 4.12, and Xfoil plots for these airfoils are shown in Figures 4.34 through 4.36.

Although it can now be see that the new trial optimisation produces results not as good as before, they are at least able to be compared, and represent an improvement of the lift/drag ratio of the original NACA 0012 airfoil (42.646) under similar conditions of approximately 60%, for this scenario. The results trend could be seen to be still improving, albeit slowly, in terms of C_D, thus a longer run would produce better results. More runs with varying starting populations would of course be necessary for a full appraisal, though it is not expected to see results approaching the work being compared as they carried out a parameter tuning exercise and were using the hybrid local search method known to be good for airfoil optimisations.

TABLE 4.10: $\alpha = 3°$. Selected airfoils from G863 of Ganesh for range ±1.0 with denormalised C_L & C_D with positive C_L, together with other airfoils from other work. *Ram* indicates (Ram *et al.*, 2014).

Source	Airfoil ID	C_L	C_D	C_L/C_D	Description
Ram		0.3536	0.00657	53.820	single objective, weighted sum
Ganesh	ffd-2268	0.3606	0.00872	41.353	Max C_L, Min C_D

TABLE 4.11: $\alpha = 5°$. Selected airfoils from G863 of Ganesh for range ±1.0 with denormalised C_L & C_D with positive C_L, together with other airfoils from other work. *Selvan* indicates (Selvan, 2015), *Gardner* indicates (Gardner and Selig, 2003).

Source	Airfoil ID	C_L	C_D	C_L/C_D	Description
Selvan	Class shape	0.7087	0.00860	82.407	Max C_L, Fixed C_D
Selvan	Hicks-Henne	0.7272	0.00860	84.558	Max C_L, Fixed C_D
Selvan	Bezier	0.6352	0.00860	73.860	Max C_L, Fixed C_D
Selvan	Polynomial	0.6263	0.00860	72.441	Max C_L, Fixed C_D
Gardner	E168	0.9200	0.00875	105.143	Fixed C_L, Min C_D
Ganesh	ffd-2268	0.6059	0.00933	64.941	Max C_L, Min C_D
Ganesh	ffd-3301	0.6593	0.00917	71.897	Max C_L, Min C_D
Ganesh	ffd-3337	0.7725	0.00931	82.975	Max C_L, Min C_D
Ganesh	ffd-3705	0.9158	0.00927	98.792	Max C_L, Min C_D
Ganesh	ffd-3431	0.9298	0.00930	99.978	Max C_L, Min C_D
Ganesh	ffd-3649	1.0218	0.00966	105.776	Max C_L, Min C_D

TABLE 4.12: Ganesh airfoil performance at $\alpha = 5°$ for Reynolds number 3×10^5, showing selected best performing airfoils (most lift, least drag, best L/D ratio). Other aspects as Table 4.11.

Source	Airfoil ID	C_L	C_D	C_L/C_D	Description
Ganesh	ffd-3308	0.9200	0.01350	68.172	Max C_L, Min C_D
Ganesh	ffd-3328	0.9151	0.01346	67.975	Max C_L, Min C_D
Ganesh	ffd-3351	0.9199	0.01349	68.173	Max C_L, Min C_D

TABLE 4.13: Referenced airfoils from G863 of Ganesh for range ±1.0 shown with their FFD parameters defining the airfoil geometries.

Airfoil	p1	p2	p3	p4	p5	p6	p7	p8
ffd-2268	-0.39017	0.29187	-0.07249	1.00000	0.16376	-0.28464	-0.42998	-0.10630
ffd-3301	-0.36908	0.33321	-0.07018	0.99695	-0.28794	-0.40392	-0.43133	0.04220
ffd-3337	-0.37769	0.27114	-0.06820	1.00000	-0.23434	0.02339	-0.44459	0.02170
ffd-3705	-0.34317	0.23027	-0.07738	0.99847	-0.33135	0.36669	-0.42917	0.16262
ffd-3431	-0.36329	0.48893	-0.07746	0.98977	-0.26614	0.20491	-0.42883	0.26825
ffd-3649	-0.35254	0.32081	-0.08409	0.99938	-0.33563	0.54770	-0.40672	0.29094
ffd-3308	-0.17340	0.81986	-0.01364	0.65003	-0.50948	0.06608	-0.19046	0.38163
ffd-3328	-0.17339	0.81757	-0.01368	0.64984	-0.52178	0.06608	-0.20723	0.37854
ffd-3351	-0.17339	0.81985	-0.01372	0.65000	-0.51014	0.06608	-0.19059	0.38165

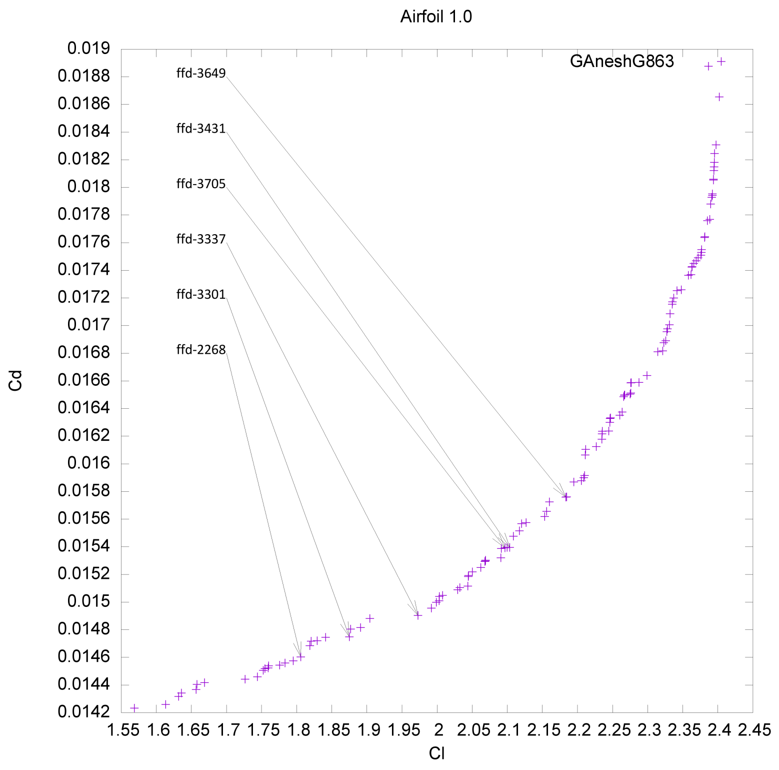
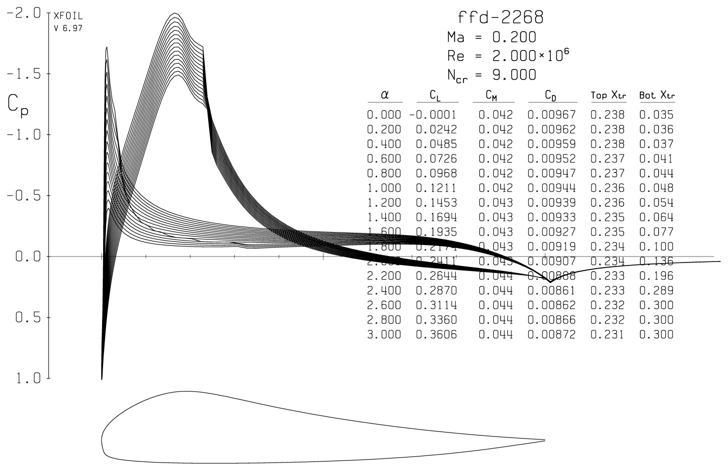


FIGURE 4.32: Ganesh results for range ±1.0, at $\alpha = 15°$, $Re = 2 \times 10^6$, at generation 863 with with selected airfoils annotated. Here, C_L & C_D have been denormalised with C_L returned to the positive, for $\alpha = 15°$.



ffd-2268

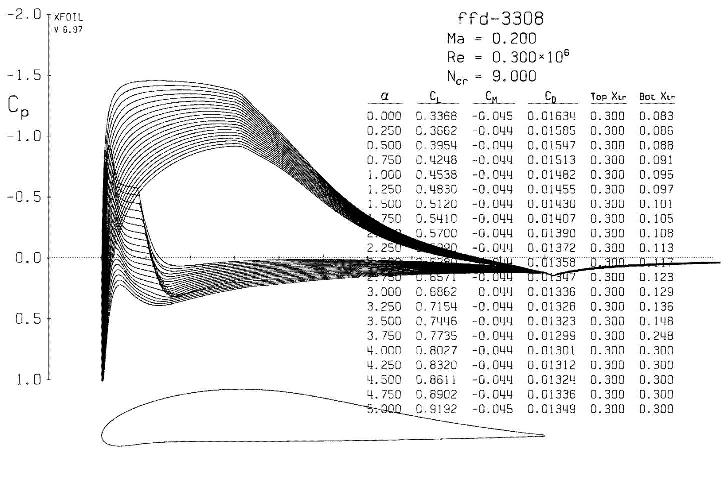
Ma = 0.200

Re = 2.000×10^6

N_{cr} = 9.000

α	C_L	C_M	C_D	Top Xtr	Bot Xtr
0.000	-0.0001	0.042	0.00967	0.238	0.035
0.200	0.0242	0.042	0.00962	0.238	0.036
0.400	0.0485	0.042	0.00959	0.238	0.037
0.600	0.0726	0.042	0.00952	0.237	0.041
0.800	0.0968	0.042	0.00947	0.237	0.044
1.000	0.1211	0.042	0.00944	0.236	0.048
1.200	0.1453	0.043	0.00939	0.236	0.054
1.400	0.1694	0.043	0.00933	0.235	0.064
1.600	0.1935	0.043	0.00927	0.235	0.077
1.800	0.2174	0.043	0.00919	0.234	0.100
2.000	0.2411	0.043	0.00907	0.234	0.136
2.200	0.2644	0.044	0.00888	0.233	0.196
2.400	0.2870	0.044	0.00861	0.233	0.289
2.600	0.3114	0.044	0.00862	0.232	0.300
2.800	0.3360	0.044	0.00866	0.232	0.300
3.000	0.3606	0.044	0.00872	0.231	0.300

FIGURE 4.33: An Xfoil plot of an optimised airfoil ('ffd-2268') found by Ganesh at range ±1.0, in generation 863, having denormalised coefficients of $C_L = 0.3606$, $C_D = 0.00872$, for $\alpha = 3°$. The airfoil is shown underneath the graph of the pressure curve whose width corresponds to points along the airfoil from leading to trailing edges.



ffd-3308

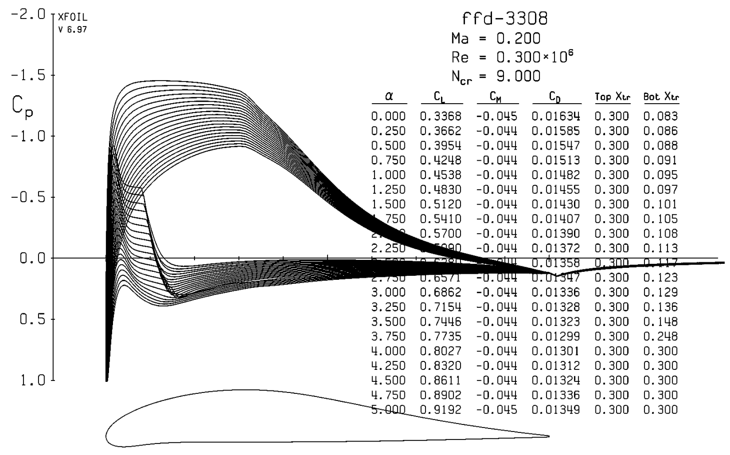
Ma = 0.200

Re = 0.300×10^6

N_{cr} = 9.000

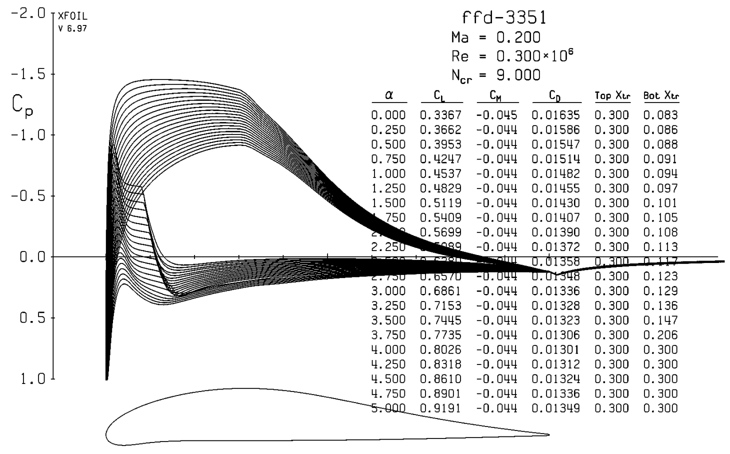
α	C_L	C_M	C_D	Top Xtr	Bot Xtr
0.000	0.3368	-0.045	0.01634	0.300	0.083
0.250	0.3662	-0.044	0.01585	0.300	0.086
0.500	0.3954	-0.044	0.01547	0.300	0.088
0.750	0.4248	-0.044	0.01513	0.300	0.091
1.000	0.4538	-0.044	0.01482	0.300	0.095
1.250	0.4830	-0.044	0.01455	0.300	0.097
1.500	0.5120	-0.044	0.01430	0.300	0.101
1.750	0.5410	-0.044	0.01407	0.300	0.105
2.000	0.5700	-0.044	0.01390	0.300	0.108
2.250	0.5990	-0.044	0.01372	0.300	0.113
2.500	0.6280	-0.044	0.01358	0.300	0.117
2.750	0.6571	-0.044	0.01347	0.300	0.123
3.000	0.6862	-0.044	0.01336	0.300	0.129
3.250	0.7154	-0.044	0.01328	0.300	0.136
3.500	0.7446	-0.044	0.01323	0.300	0.148
3.750	0.7735	-0.044	0.01299	0.300	0.248
4.000	0.8027	-0.044	0.01301	0.300	0.300
4.250	0.8320	-0.044	0.01312	0.300	0.300
4.500	0.8611	-0.044	0.01324	0.300	0.300
4.750	0.8902	-0.044	0.01336	0.300	0.300
5.000	0.9192	-0.045	0.01349	0.300	0.300

FIGURE 4.34: An Xfoil plot of airfoil ('ffd-3308') found by Ganesh for $\alpha = 5°$ and Reynolds number $Re = 3 \times 10^5$. See also Figure 4.33.



```
        ffd-3308
       Ma  = 0.200
       Re  = 0.300×10⁶
       Ncr = 9.000
  α      CL      CM       CD      Top Xtr  Bot Xtr
0.000  0.3368  -0.045  0.01634  0.300  0.083
0.250  0.3662  -0.044  0.01585  0.300  0.086
0.500  0.3954  -0.044  0.01547  0.300  0.088
0.750  0.4248  -0.044  0.01513  0.300  0.091
1.000  0.4538  -0.044  0.01482  0.300  0.095
1.250  0.4830  -0.044  0.01455  0.300  0.097
1.500  0.5120  -0.044  0.01430  0.300  0.101
1.750  0.5410  -0.044  0.01407  0.300  0.105
2.000  0.5700  -0.044  0.01390  0.300  0.108
2.250  0.5990  -0.044  0.01372  0.300  0.113
2.500  0.6280  -0.044  0.01358  0.300  0.117
2.750  0.6571  -0.044  0.01347  0.300  0.123
3.000  0.6862  -0.044  0.01336  0.300  0.129
3.250  0.7154  -0.044  0.01328  0.300  0.136
3.500  0.7446  -0.044  0.01323  0.300  0.148
3.750  0.7735  -0.044  0.01299  0.300  0.248
4.000  0.8027  -0.044  0.01301  0.300  0.300
4.250  0.8320  -0.044  0.01312  0.300  0.300
4.500  0.8611  -0.044  0.01324  0.300  0.300
4.750  0.8902  -0.044  0.01336  0.300  0.300
5.000  0.9192  -0.045  0.01349  0.300  0.300
```

FIGURE 4.35: An Xfoil plot of airfoil ('ffd-3328') found by Ganesh for $\alpha = 5°$ and Reynolds number $Re = 3 \times 10^5$. See also Figure 4.33.



```
        ffd-3351
       Ma  = 0.200
       Re  = 0.300×10⁶
       Ncr = 9.000
  α      CL      CM       CD      Top Xtr  Bot Xtr
0.000  0.3367  -0.045  0.01635  0.300  0.083
0.250  0.3662  -0.044  0.01586  0.300  0.086
0.500  0.3953  -0.044  0.01547  0.300  0.088
0.750  0.4247  -0.044  0.01514  0.300  0.091
1.000  0.4537  -0.044  0.01482  0.300  0.094
1.250  0.4829  -0.044  0.01455  0.300  0.097
1.500  0.5119  -0.044  0.01430  0.300  0.101
1.750  0.5409  -0.044  0.01407  0.300  0.105
2.000  0.5699  -0.044  0.01390  0.300  0.108
2.250  0.5989  -0.044  0.01372  0.300  0.113
2.500  0.6280  -0.044  0.01358  0.300  0.117
2.750  0.6570  -0.044  0.01348  0.300  0.123
3.000  0.6861  -0.044  0.01336  0.300  0.129
3.250  0.7153  -0.044  0.01328  0.300  0.136
3.500  0.7445  -0.044  0.01323  0.300  0.147
3.750  0.7735  -0.044  0.01306  0.300  0.206
4.000  0.8026  -0.044  0.01301  0.300  0.300
4.250  0.8318  -0.044  0.01312  0.300  0.300
4.500  0.8610  -0.044  0.01324  0.300  0.300
4.750  0.8901  -0.044  0.01336  0.300  0.300
5.000  0.9191  -0.044  0.01349  0.300  0.300
```

FIGURE 4.36: An Xfoil plot of airfoil ('ffd-3351') found by Ganesh for $\alpha = 5°$ and Reynolds number $Re = 3 \times 10^5$. See also Figure 4.33.

4.6.1 Comparison summary

The original Ganesh results were produced by an optimisation of Naca 0012 focused on an angle of attack of 15°, which is often around the point of maximum lift for an aircraft wing, and a Reynolds number of 2×10^6 which is appropriate for a flight speed of around 152 mph at 2,000 ft. The airfoil geometry was given xed positions of the leading and trailing edges in order to x the chord length and angle of attack, and the thickness was constrained at 25% and 50% along the chord in order to provide room for strengthening spars. The α and Re used has a tendency, it seems, to favour increase in lift over decrease in drag, relative to optimisations working at a fixed lower angle of attack.

The results from Ganesh stand up well in comparison to the first two other works (Ram *et al.*, 2014) & (Selvan, 2015), being of approximately equal quality or better in some regard, even though of a more general case. These others had fixed drag and were attempting to maximise lift in a single optimisation.

The third study (Gardner and Selig, 2003) achieved better results, but this had a fixed lift and was attempting to minimise drag, starting with a relatively thin airfoil and using a hybrid GA with a tuned local search. They note that “few symmetric airfoils are capable of operating eciently at a lift coecient of 0.92 at the specied Reynolds number”. Ganesh was able to acquire the same lift, starting from a datum of 0.57, but had to sacrifice some drag performance to get there. Drag was more slowly improving at the time of termination. The thickness of the airfoil constraint may be inhibiting performance in this regard, as may the constraint around the start of the leading and trailing edges, all of which prevent the airfoil from thinning especially at the trailing edge.

4.7 Summary

Chapter 4 set out the multi-objective optimisation problem of optimising a standard airfoil by changing its geometry with free-form deformation, and assessing the resultant aerodynamic performance of the new airfoil shape. The performance, in terms of the coefficients of lift and drag of the airfoil, were used as the objective function values by which the MOOEA would select the candidate airfoils for inclusion for breeding in the next generation of the MOOEA, thus evolving 'better' solutions over time. The MOOEA used was Ganesh, created as part of this work as set out in Chapter 3.

The XFoil software tool, comprising mainly Fortran but some C codes, used here to assess airfoil aerodynamic performance in batch mode (rather than interactively as was its original purpose), was modified to improve its batch mode notification capability of convergence failing, enabling improvement both in the quality of airfoil solutions discovered, and in the optimisation performance. Previous work carried out by this author and by others ((Kipouros *et al.*, 2012), (Lattarulo *et al.*, 2013)) using untreated XFoil for batch use in a similar way, had experienced this limitation that caused compromised quality outcomes, therefore this remedial modification should prove advantageous in future work exploring other algorithms using XFoil for airfoil optimisation.

Considering the results, it was apparent that as the range increased, the MOOEA was able to find attainment surfaces that were better approximations of the Pareto-optimal front, as do the other algorithms, as it is intrinsically enabled to explore wider areas of the search space at earlier times. XFoil can take longer to run with larger variations in range as it may find it harder to converge successfully and indeed may fail to converge, since the shape of airfoil is relatively more deformed from the datum design. However, the modifications to XFoil now trap these convergence failures thoroughly, enabling Ganesh to correctly rate the desirability of the retention of their solutions.

The trends of the control parameters are shown in the plots of Figures 4.16 to 4.19 and these show, over time, the probability of genes changing (by both crossover and mutation) decreasing and the degree by which they change also decreasing (since the η controls are increasing and there is an inverse relationship). It should be noted that this response is not pre-determined or programmed in, it is an emergent behaviour of the system. This response means that the system moves from a more exploratory behaviour to a more exploitative one, relatively speaking, although the degree of this response is not large.

Although each new self-adaptive parameter can also be thought of as a factor increasing the decision search space, the increase is by a relatively small percent; moreover, the number of generations allowed in these experiments can be thought of as quite low, and at greater generations, the impact of self-adaptivity would be expected to be greater. The inclusion of generation 836 of an independent run, shown in Figure 4.12, shows that Ganesh is able to produce a well-formed Pareto approximation front for this problem, given enough time, thus providing evidence that a conclusion of one of the other studies (Kipouros *et al.*, 2012), "it is very difficult for any Genetic Algorithms type optimisation method to perform competitively in shape aerodynamic optimisation cases" need not be necessarily true in all cases.

Specifying that zero duplicates are permitted is beneficial as it prevents the MOOEA from prematurely converging to just a few solutions having many copies, as can be the case, and although it is limited to intra-generational checking, as each preceding generation also has zero duplicates, it seems to perform well, even though it does not prevent previously rejected solutions from re-appearing in subsequent generations, as they might do if they had been rejected previously only because they were too near to another in objective space. Nevertheless, not having to save every solution ever produced can be a significant memory saving, especially for long runs having a great many generations, and it can save CPU time too since the searching for a new solution in the cache of old solutions, is obviated.

This self-adaptive GA has been shown to work well on a benchmark 2D aerodynamic design problem, as a real-world engineering example, and to provide better convergence than MOTS and in some cases NSGA-II while not being worse than either overall. It does not always provide as good a density of solutions in the Pareto-optimal front as MOTS, though this can be viewed as a trade-off between being better at exploring the search space widely but doing less well at exploiting solutions found locally.

Chapter 5

Multi-objective optimisation of an electrical power network

5.1 Electrical power networks

This work explores a possible method of addressing the configuration of large-scale electrical power networks, such as a national grid, using an approach based on evolutionary computing, which has been used previously in complex systems research such as emergent computation (Mitchell, 1999) and dynamics of complex networks (Aguilar-Hidalgo *et al.*, 2012), and also directly in optimal power flow research (Pandya and Joshi, 2008). Evolutionary algorithms and their applicability to real-world engineering problems has been discussed in Chapter 2, furthermore Allen *et al.* (2010) conclude that consideration of systems exhibiting complexity entails the construction of synergies between the studies of systems and their structures, and the ideas of neo-Darwinian evolutionary processes. Having shown in Chapter 4 that the MOOEA of this work, Ganesh, is able to cope with real-world engineering multi-objective optimisation problems, this chapter applies Ganesh to the optimisation of electrical power networks.

Concerns about the environmental impacts of power generation include not only the issue of global climate change and the effect that emissions, such as CO_2, have upon the climate, but also matters of harmful pollution and the availability of power for an ever increasing world population coupled with dwindling natural fossil fuel resources. For these reasons alone, the optimisation of power generation and the inclusion of renewable energy sources into electrical power grids, is already important now and this can only increase in the foreseeable future.

The essential problem in the architecture of national grid networks is that of power flow and optimal power flow (OPF) calculations of alternating current (AC) power, and these calculations are at the centre of Independent System Operator (ISO) power markets (Cain *et al.*, 2013) in which AC OPF is solved over a number of different orders of magnitude of time-scales, from minutes via hours, to annually and multi-year horizons, where the latter is for planning and investment while the former are for ensuring demand is met and for spot market pricing. For the ISOs, the central planning issue is the Unit Commitment Problem (UCP), which stated simply, is the scheduling of power generators to produce a contracted power output for a forecast load for an agreed duration, bearing in mind ramp-up and shut-down times, with the concomitant cost and profit implications.

An ISO, although having a precise definition which is country-specific, nevertheless is more generally understood to be an organisation having the responsibility for the scheduling of power production and its dissemination at a regional or national level over large-scale fixed infrastructure. The ISO produces and acquires load forecasts, receives offers of power from generating companies acting within a competitive auction market, and produces generation schedules consisting of required power units and a price, to meet demand within the constraints of the grid and generators.

The power flow computation (Glover *et al.*, 2012) consists of calculating values for voltage and angle of phase at the buses in the grid for stable and balanced three-phase conditions. These results enable the further calculation of real and reactive power flows in the connected components such as switchgear, transformers and transmission lines, and also power losses, which are of particular relevance in the latter.

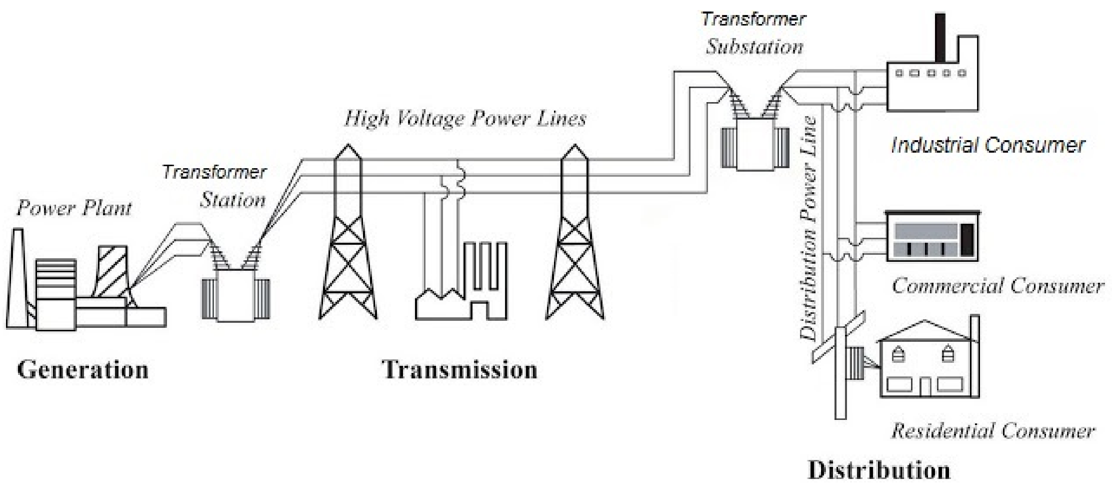


FIGURE 5.1: A schematic diagram of a general large scale grid network (Blume, 2007).

5.2 Distributed generation

Electrical power networks, broadly depicted in Figure 5.1, can be improved both technically and economically through the inclusion of distributed generation (DG) which may include renewable energy sources. DG units are lower output generators that provide incremental capacity at specific geographical locations, thus enhancing voltage support and improving network reliability while also acting economically as a hedge against a high price of centrally produced power, through locational marginal pricing (LMP). The operation of power grids by ISOs as unbundled auction wholesale spot power markets that support real-time pricing, provides a further incentive to roll-out DG, which in turn brings about a need to define the type, number and location of any extra DG units (Gautam and Mithulananthan, 2007).

The DG units considered here are renewable energy sources consisting of solar photovoltaic (PV) panels, micro wind turbine, and micro gas turbine, all of which share the incremental characteristic, making them ideal for low-end power generation that is nevertheless able to be scaled up through either individually more powerful units, or by array assembly as shown in Figure 5.2.

The work presented here addresses the composition of an AC electrical power network, based upon the IEEE 30 Bus Test Case, which in turn represents a section

FIGURE 5.2: A giant photovoltaic array, Nellis, Nevada USA.

of the American Electric Power System (in the Midwestern US) in December 1961 (Christie, 1993), to which distributed generation is added. Figure 5.3 depicts the schematic of the original grid section.

The model IEEE 30 Bus grid, as shown in the single line diagram (Figure 5.4), is configured to have six central fixed large-scale open cycle gas turbine (OCGT) electrical power stations, and twenty four nodes whose buses enable the input from each of the three types of variable distributed generators given above. This model is the basis for the OPF calculations (Saadat, 1999, p. 48).

It can be seen that in the model used, there is a one-to-one relationship between a node and bus, thus a given bus or node number can be thought of as referring to the same thing. The term *node* is used in the context of existing in a network, while *bus* is used in its electrical context.

This work, described in part in (Oliver *et al.*, 2014) and Oliver *et al.* (June 2015) , has the following aims:

1. To determine the composition of the power network in terms of the type, number and location of the DG units, with the goal of finding the cheapest configuration (capital cost), of meeting demand for power while keeping over- and under-production of power as low as possible, and of minimizing the

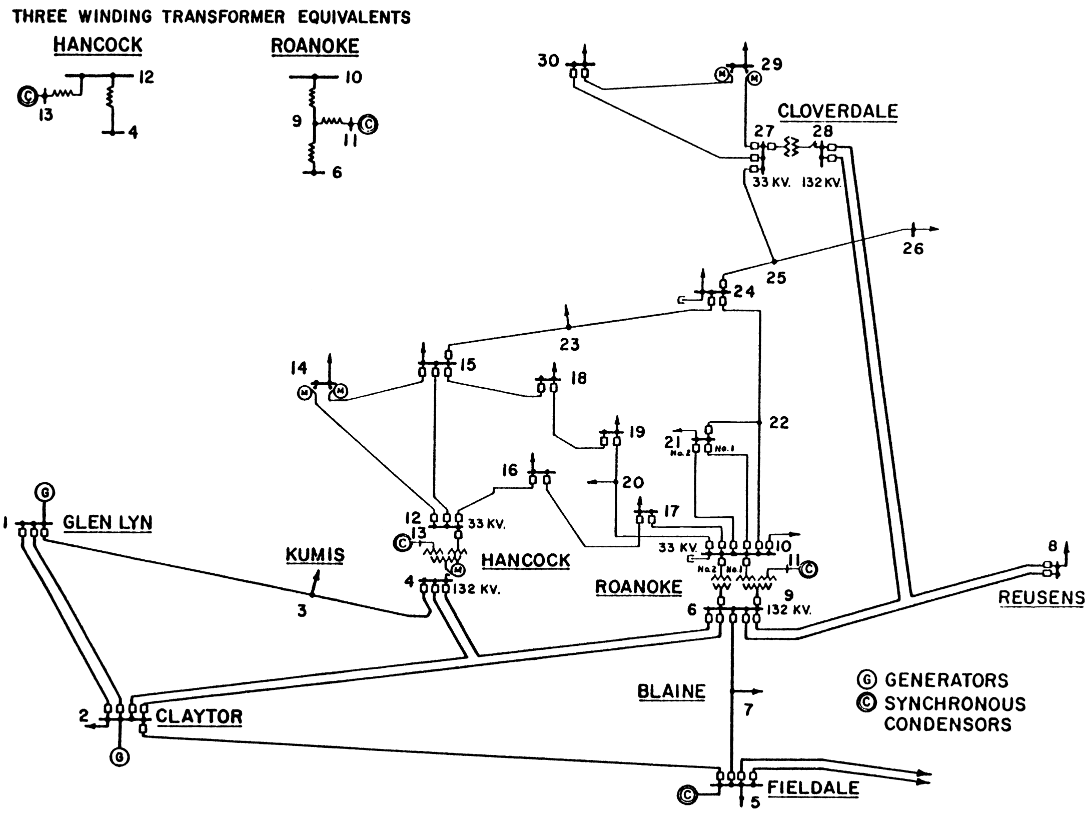


FIGURE 5.3: A section of the American Electric Power System, Midwestern US, December 1961 (Christie, 1993).

spot price and CO_2 emissions, thus determining the best, or at least high-performing candidate network solutions.

2. To analyse the multi-dimensional results of the evolutionary computation component in order to reveal relationships between the network's design vector elements, by means of most influential nodes and type of technology.

3. To enable the optimisation of the power grid connections, by incorporating the power capacity of the transmission lines as further variables in each candidate solution.

4. To explore the possibility of identifying tipping points in the candidate design solutions through their influence on system behaviour.

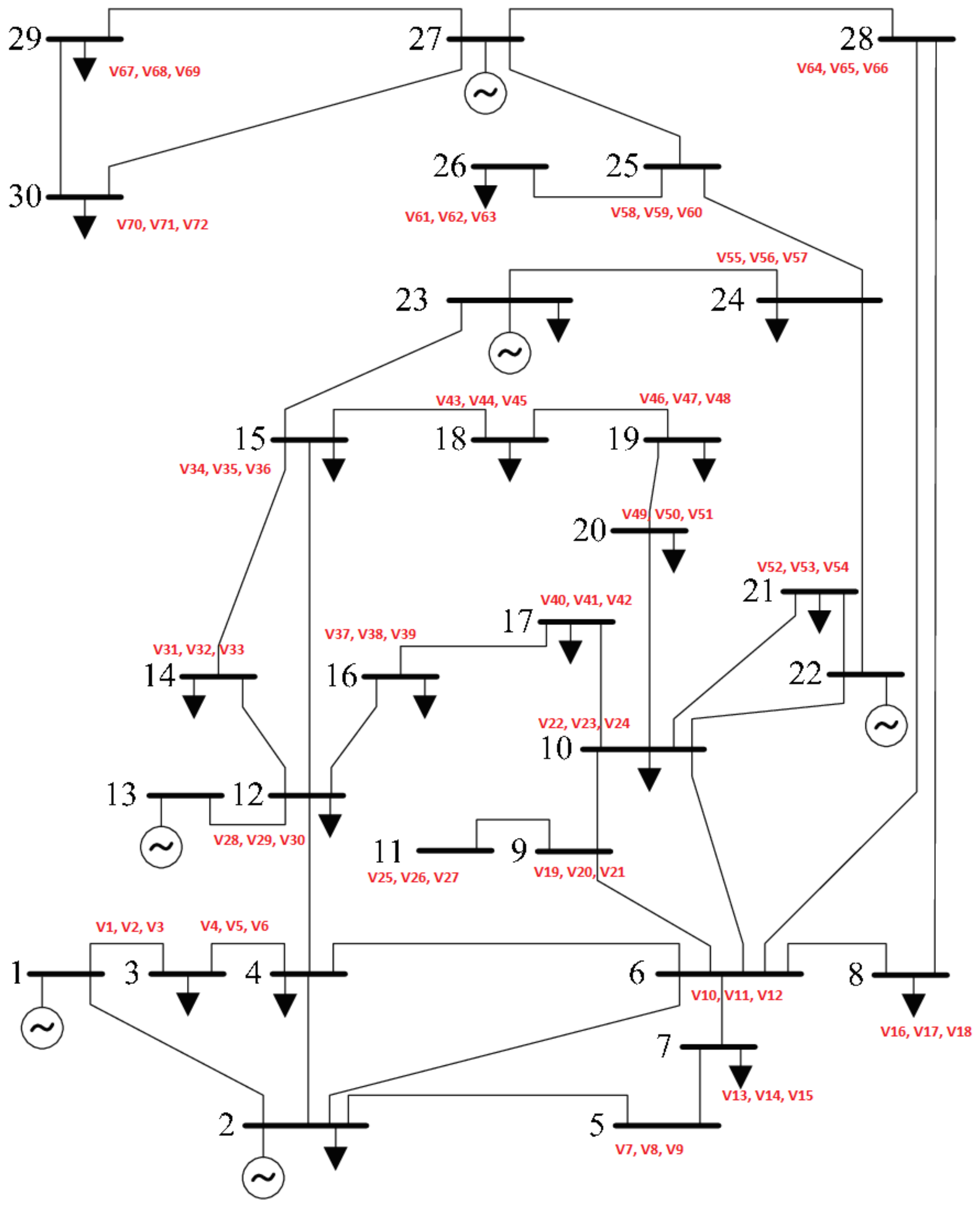


FIGURE 5.4: The IEEE 30-bus test system in single line diagram style (Dharamjit, 2012) derived from Figure 5.3, showing the location of DG units by bus, to which has been added the variable for the number of units of the given DG type at that bus, shown by the V-number. The symbol of tilde in a circle indicates a large central generator input, and the down arrow indicates output from the bus to a load. See also Table 5.1.

5.3 Integrating with power market simulation

The Plexos tool (Energy Exemplar Pty Ltd (2013)) provides both OPF and financial market simulations, in particular providing unit commitment (which generators should be used, bearing in mind their operating characteristics such as ramp-up time as well as power output and running costs), economic dispatch (which generators to use to meet demand from a cost viewpoint), transmission analyses (losses, congestion), and spot market operation. It also provides estimations of CO_2 emissions. The volume of lost load (VoLL) is the threshold price above which loads prefer to switch off, while the dump energy price is that below which generators prefer to switch off, and these along with market auctions also contribute to the ratio of power generated to power consumed. Transmission losses are also taken into account within Plexos through sequential linear programming.

Plexos is integrated with Ganesh, the self-adaptive multi-objective optimizing evolutionary algorithm (MOOEA) set out in Chapter 3, thus establishing an optimization feedback loop, since Plexos gives optimal unit commitment for a given set of DG units, while the MOOEA is used to determine the optimal set of generators for the given demand profile and weather pattern. A MOOEA is used as they have a history (Haupt and Haupt, 2004, p. 174) of tackling non-linear (Nicolis, 1995) multi-objective and multi-dimensional optimization problems successfully, and since OPF for AC power is a non-linear problem (Glover *et al.*, 2012, pp. 325-334) while power markets require multi-part non-linear pricing.

Figure 5.5 shows the integration of Plexos with Ganesh, with Plexos inputs arriving from Ganesh via a third-party interface (provided by colleagues as acknowledged in 6.4), while the output is acquired by Ganesh directly. This figure is for the DG unit optimisation scenario, while that of 5.6 shows the additional interface component of Xml through which changes to the line capacities defined in the model are provided to Plexos.

This work uses historical data of weather (in the form of actual solar PV and wind power generation), central power generation, and electrical energy demands, from Australia of 2010, thus providing a realistic simulation environment for both power demand and renewable generation. The data was provided by colleagues in Australia, as acknowledged (in section 6.4). The generator characteristics are given in Tables C.4 and C.5 of Appendix C.

5.4 Defining the optimisation

The optimisation problem is defined at a high level as a process to find the set of potential DG units and their locations, in terms of buses, which enable the system to produce required power in the most efficient way in terms of cost and emissions. The problem is non-linear (see 5.3 above) and multi-dimensional in both its design vector and its objective functions.

The DG units are defined as (i) micro-gas turbine (ii) Wind turbine and (iii) Solar photovoltaic, where a unit of value 0 means the generator is not present at the location. The scenario allows for up to 5 units of each type to be located at any of the nodes defined as variable in the network diagram (Figure 5.4), which means any node except for the nodes 1, 2, 13, 22, 23 and 27, as these are the large fixed central OCGT power stations. The simulation within Plexos has an horizon of one calendar year, represented as 365 steps of 1 day increments with a resolution to 30 minutes, from 01-Jan-2010.

Each transmission line between any two buses has a maximum flow capacity stated in megawatts (MW). The transmission line capacities are amended in the Plexos Xml model file which are sent to Plexos for each solution run. The labels shown as Vn at the given nodes indicate the design variable number that defines the number of units of the given generator types at that bus, and as can be seen, each of the 3 variable types can be present potentially.

Each DG type can be present with the number of units of the type in the interval [0,5]. As there are 24 nodes at which variable DG units can be located, and 3 types of generator, the design vector of each candidate solution therefore consists of 72 variables: $\mathbf{v} = (x_1, x_2, \ldots, x_n)$, $n = 72$. This configuration allows a candidate solution to have from 0 DG units up to a theoretical 360 (being 5 units of each of 3 DG types at the 24 nodes). Table 5.1 below shows the allocation of DG units by type to nodes, cross-referenced to its variable number (as shown in Figure 5.4), with the assumption that a given generator feeds in to one associated node only.

The candidate solutions chosen by the MOOEA, using the results from Plexos, are thus selected due to the effect their chosen DG units have on the electrical network due to their operating characteristics and where they feed into the network, defined in the topology as shown in Figure 5.4.

The MOOEA allows each new experiment to override its default initializer, which creates an initial population of candidate solutions by generating random[1] variable values, u, within their defined ranges, in this case $0 \leq u \leq 5$. The initializer used instead generates solutions that meet the hard constraint, by selecting for each solution a random value for the sum of the vector variables, between 0 and the chosen constraint (see below), and using this as the limit for that candidate solution. Each variable of that solution is then selected randomly, and is allocated a random value within its range, until the solution's own limit is reached. In this way, solutions in the initial population will vary between 0 DG units and the chosen constraint.

In subsequent generations, solutions may evolve that break the hard constraint, due to mutation and recombination operators acting on 'fit' parent solutions selected for breeding, and in this case the solutions will be retained in the population but repaired. Repairing in this context means that a failing solution's vector of DG variables is changed until it falls within the constraint, by randomly choosing one of the variables, decrementing its DG unit count (when it has $u \geq 1$), and then repeating the process until the total falls within the constraint.

[1]In this thesis, *Random* should always be taken to mean pseudo-random as generated by algorithm, unless specified explicitly otherwise, likewise a uniform distribution should be assumed.

There is a fixed population of size 30, allowing 0 duplicate solutions in any single generation, with initial crossover and mutation probabilities of 0.9 and 0.01389 (1/(72)) respectively. Each evaluation of a solution is an independent run of Plexos, which performs its own optimal power flow calculations, thus providing the data which the MOOEA uses as its objective function values. The MOOEA is allowed to run for 2,000 function evaluations (67 generations), with each generation taking approximately 3.5 to 5 hours elapsed time (on a desktop PC with quad-core and hyper-threading). The population size is a compromise, chosen for a minimum size which would likely be useful, based on Grefenstette (1986) referenced in Mitchell (1999), and which would not have an over long elapsed time per generation. Even so, each run takes between 1.5 and 2 weeks to perform.

There are three independent optimisations (and one related) defined in the following sections, which share the above overall characteristics, but differ in the first two cases by the definition of their first objective function (of four), and in the last case by the components of its decision vector. There is a separate results section, the sub-sections of which give the outcomes for each of the problems.

TABLE 5.1: The nodes (buses), their generator types, and associated variable number in which the quantity of assigned DG units of that generator type is given.

	Gas			Wind			Solar PV	
Node	DG	Var	Node	DG	Var	Node	DG	Var
n03	g02	V01	n03	g09	V02	n03	g10	V03
n04	g02	V04	n04	g09	V05	n04	g10	V06
n05	g02	V07	n05	g09	V08	n05	g10	V09
n06	g02	V10	n06	g09	V11	n06	g10	V12
n07	g02	V13	n07	g09	V14	n07	g10	V15
n08	g02	V16	n08	g09	V17	n08	g10	V18
n09	g02	V19	n09	g09	V20	n09	g10	V21
n10	g02	V22	n10	g09	V23	n10	g10	V24
n11	g02	V25	n11	g09	V26	n11	g10	V27
n12	g02	V28	n12	g09	V29	n12	g10	V30
n14	g02	V31	n14	g09	V32	n14	g10	V33
n15	g02	V34	n15	g09	V35	n15	g10	V36
n16	g02	V37	n16	g09	V38	n16	g10	V39
n17	g02	V40	n17	g09	V41	n17	g10	V42
n18	g02	V43	n18	g09	V44	n18	g10	V45
n19	g02	V46	n19	g09	V47	n19	g10	V48
n20	g02	V49	n20	g09	V50	n20	g10	V51
n21	g02	V52	n21	g09	V53	n21	g10	V54
n24	g02	V55	n24	g09	V56	n24	g10	V57
n25	g02	V58	n25	g09	V59	n25	g10	V60
n26	g02	V61	n26	g09	V62	n26	g10	V63
n28	g02	V64	n28	g09	V65	n28	g10	V66
n29	g02	V67	n29	g09	V68	n29	g10	V69
n30	g02	V70	n30	g09	V71	n30	g10	V72

5.4.1 Optimising for DG allocation by generation cost

The title of this optimisation, as given by the heading of this section, is derived from the definition of objective function 1 as given in Equation 5.1.

There are 4 objective functions defined, all of which are to be minimised simultaneously and the values for all of which come from Plexos, these being:

$$\min F(genCost) = genCost \tag{5.1}$$

$$\min F(useDump) = |useDump| \tag{5.2}$$

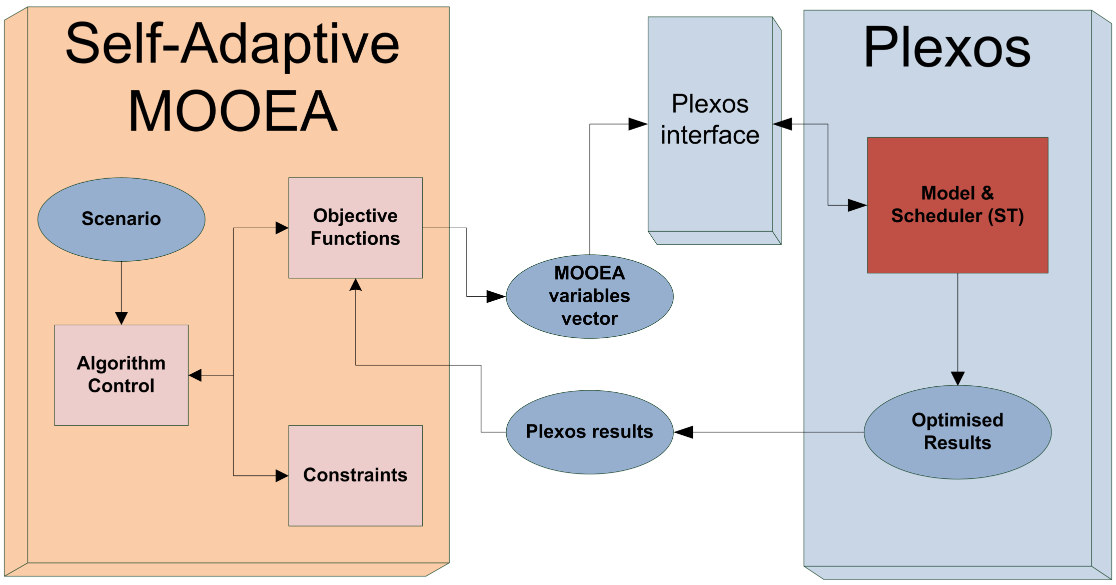


FIGURE 5.5: The integration of Plexos with the self-adaptive multi-objective optimisation algorithm. The Plexos interface, a .Net program, is called with the parameters in the MOOEA variables vector.

$$\min F(spotPrice) = spotPrice \tag{5.3}$$

$$\min F(CO_2) = CO_2 \tag{5.4}$$

in which the values represent respectively:

1. The generation cost (in currency, e.g. \$)
2. The USE/DUMP energy (MWh)
3. Spot Price (\$/MWh)
4. CO_2 emissions (Kg)

Considering the values above, useDump, depending whether it is negative or positive, is either the un-served amount of energy due to under-production or the dump energy due to over-production, relative to demand. The spot price is the mean price achieved in the simulated market auctions over the course of the simulation in Plexos.

A hard constraint on the total number of DG units deployed, u, is applied in Equation 5.5, in order to investigate how the system transforms itself. Without such a constraint, which can be viewed as a limit to financial resources available as investment into DG, we would perhaps expect the system to maximize DG deployment since they provide a known benefit and where cost is the only downside, and this would hide the effects that placement may have when otherwise.

$$\sum_{i=1}^{72} u_i \leq 70 \tag{5.5}$$

The hard constraint is changed in other runs (equations 5.6 and 5.7), to see what effect a different number of allowed DG units may have:

$$\sum_{i=1}^{72} u_i \leq 200 \tag{5.6}$$

$$\sum_{i=1}^{72} u_i \leq 35 \tag{5.7}$$

There is a fixed population of size 30, allowing 0 duplicate solutions in any single generation, with initial crossover and mutation probabilities of 0.9 and 0.01389 (1/72) respectively.

5.4.2 Optimising for DG allocation by sum of DG units

The title of this optimisation, as given by the heading of this section, is derived from the definition of objective function one as given in Equation 5.8. There are two optimisations given in this section, the first and a follow-up as given below.

5.4.2.1 Part one: Original line capacity

Here, the original objective of equation 5.1 (genCost) is replaced by the new definition of equation 5.8, in which the summed total of the DG units assigned (sumU) is to be minimised. The remaining objectives are the same as previously, and the integration with Plexos is extended by the amendment of the model which is defined in the Xml file, as in Figure 5.6.

$$\min F(sumU) = \sum_{i=1}^{72} u_i \tag{5.8}$$

$$\min F(useDump) = |useDump| \tag{5.9}$$

$$\min F(spotPrice) = spotPrice \tag{5.10}$$

$$\min F(CO_2) = CO_2 \tag{5.11}$$

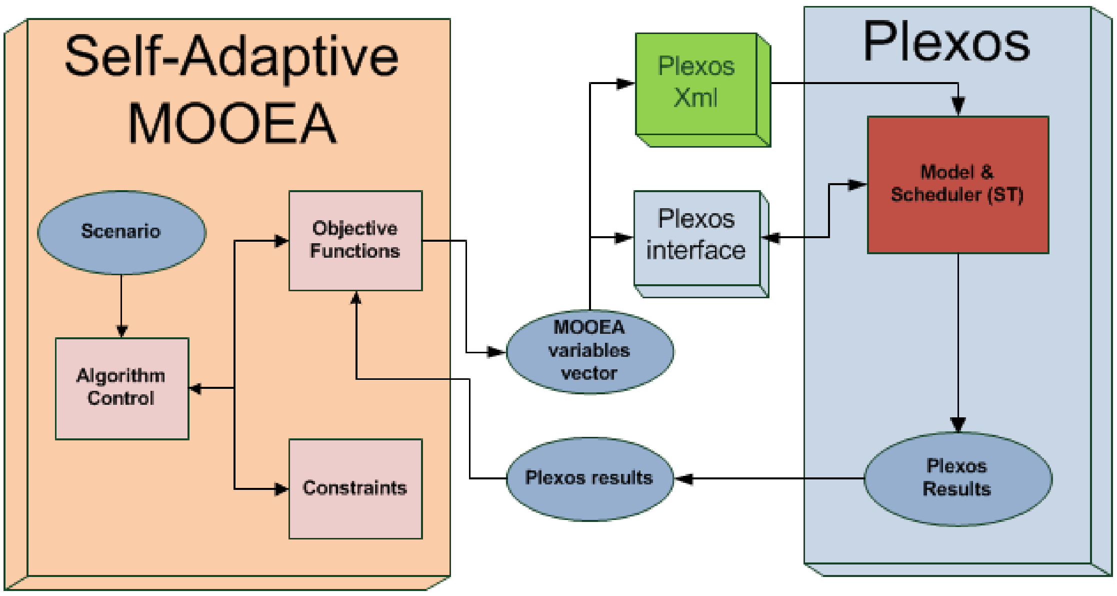


FIGURE 5.6: The integration of Plexos with the self-adaptive multi-objective optimisation algorithm. Here the interface is extended through changes to the line capacities defined in the model in the Xml file.

The optimisation is performed with the same settings given in the overview of section 5.4.

A new hard constraint on the total number of DG units deployed, u, is applied in Equation 5.12. It is the number of DG units (and their placement) that is particularly of interest in these studies, and having the objective function for the total DG units is important as it ensures diversity in sumU, enabling plots such as Figure 5.18 to be possible. The intention of this rather low constraint for this case is to encourage the optimisation to find the best locations for the extra DG units, rather than simply adding more units overall, to better illustrate the potential of the method.

$$\sum_{i=1}^{72} u_i \leq 35 \tag{5.12}$$

5.4.2.2 Part two: Change line capacity

In a follow-up optimisation, the problem is further modified from the genCost one above, to test the effect of the change of a line's maximum flow capacity on the outcome of a given solutions DG unit assignment.

The line capacity, or 'Max Flow', sets the maximum allowable flow on the line, which is also known as the thermal limit. Power in a line may flow in either direction (from or to a node's bus) with a negative value indicating an opposite flow to the direction specified in the model, since a line has to be designated as an import or export line.

The maximum flow capacity of just one line is altered in the model (the Xml file) and the results compared with a previous run in which all aspects are the same, including the seed for the pseudo-random number generator, except for the line capacity. In this case, line 11 is chosen, being that between the most highly connected bus, node 6, and node 9 which has less than half the connections, and for which the line capacity is a lowish 65 MW. The line's capacity is doubled to 130 MW, a value used by other transmission lines in the network, in the new network definition. This scenario is chosen with the intention that it is most likely to show a difference if there is one to be found.

Again, the optimisation is performed with the same settings given in the overview of section 5.4.

5.4.3 Optimising for DG allocation by sum of DG units and line capacities

The title of this optimisation, as given by the heading of this section, is derived from the addition of decision vector components to contain line capacities.

Here, the optimisation problem of section 5.4.2 (Optimising for DG allocation by sum of DG units) is modified to include all lines' maximum flow capacities as part of the design solution, hence the network and not just the allocation of DG units becomes part of the solution to the problem. The objective functions remain the same as before.

The MOOEA is configured as before, to have a mixed chromosome consisting of a vector of 72 integers, for the DG genes, one per bus, with the self-adaptive control parameters encoded as real numbers. In addition, another 41 new genes are included, for each to contain the line maximum flow capacity (LC), in MW, of a given transmission line, and all 41 line capacities are thus enabled to evolve along with the DG unit assignments. The decision vector is thus:

$$v = ((x_1, x_2, \cdots, x_{72}), (y_1, y_2, \cdots, y_{41}))$$

In this case, the 41 line capacity (LC) genes were initialised following a Gaussian distribution, using the mean and standard deviation of line capacities in the original model definition, with limits applied for a minimum of 4 MW and a maximum of 300 MW. This is to enable the line capacities to be defined as a realistic set, yet be arrived at stochastically in order to avoid bias in the initial population. An additional hard constraint was applied on the total flow capacity, being equal to the original plus 20%. A Gaussian distribution has in theory no minimum or maximum value, so the above values are used to cap the distribution at either end. A value of 4 MW was chosen for the minimum as it is both small but still usable, given that the DG units are rated at 1 MW each, however the permitted range of the design parameter (allele) is from 0, so a solution may still be able to evolve which has effectively no line (with a capacity of 0 MW) between a given pair of nodes. An upper value of 300 MW is chosen as it represents a power load between the 'typical' and 'maximum' range of a 275 kV line in the UK National Grid, which is a second level transmission line (400 kV being the first level), and hence a realistic and reasonable value (National Grid, 2014).

The optimisation is performed with the same settings given in the overview of section 5.4, except for the mutation rate which is changed to 0.008850 ($= 1/(72 + 41)$), and with a hard constraint of $HC = 35$ for the maximum number of DG units allowed.

Table C.1 of Appendix C gives the limits of the lines defined in the system, as also depicted in Figure 5.4, where the LineLimits are the thermal limits and the LowLimits are a level that the line can be de-rated to, set at 25% less than the thermal limit.

Table C.2 of Appendix C gives the total flow capacity at a given node, which is simply the sum of the 'Max Flows' of all lines connected to it, and ranks the nodes in descending capacity order. The table also shows the variable number of the DG unit associated with the given node.

Table C.3 ranks nodes by the number of lines connected, and it is interesting to note that there is little overlap between this ranking, and the ranking of nodes by capacity, however the rank 1 node in both ranking schemes is the same, Node 06 which has the highest capacity and the largest number of connections. It is this reason that makes its DG unit variables, V10, V11 and V12 so influential in the performance of solutions found. Node 10 on the other hand, is ranked 2nd by number of connections, but only 6th by capacity. Clearly both factors will play a role in deciding how useful a given unit of DG is at a given location, since a greater number of lines will mean a higher probability of output being accepted theoretically, but the available capacity will naturally limit the actual flow, depending upon the prevailing condition of the system at a given time. Table C.3 gives the nodes by connection in descending rank order along with the associated DG unit variables, for easier comparison with the capacity ranking and for consideration with ||-coords plots in the results section.

5.5 Results

Result plots are given as 2D scatter plots and higher dimensional plots using the parallel coordinates (||-coords) technique (Inselberg, 2009) as previously described in the preceding chapter for Figure 4.20.

The ||-coords technique enables multivariate data to be plotted uniquely and without loss of information, together in one plot. The results shown in the ||-coords plots and related scatter plots contain all the non-dominated solutions of each generation throughout the history of the run, unless explicitly stated otherwise.

5.5.1 Optimising for DG allocation by generation cost

This section gives the results for the optimisation defined in section 5.4.1.

The ||-coords plots show the whole design space of each solution, 72 variables, which are plotted alongside their objective function results, and also with the sum of the variables, the total number of DG units (sumU), as in Equation 5.5.

The scatter plots of Figure 5.7 show the sumU plotted against the four OFs as described in Equation 5.1 through Equation 5.4, from which it can be seen that there is broadly a trade-off between the number of DG units deployed and the quality of each of the other OF values. An obvious optimal trade-off front has not yet developed in the course of the optimisation, but the trend is clear, and that is that increasing the number of DG units deployed improves (decreases) the other OF values. The ||-coords plot of Figure 5.8 emphasises this point, in that there is a narrow region between sumU and genCost in which the lines from sumU cross, itself an indication of a negative correlation, while the lines from genCost to the

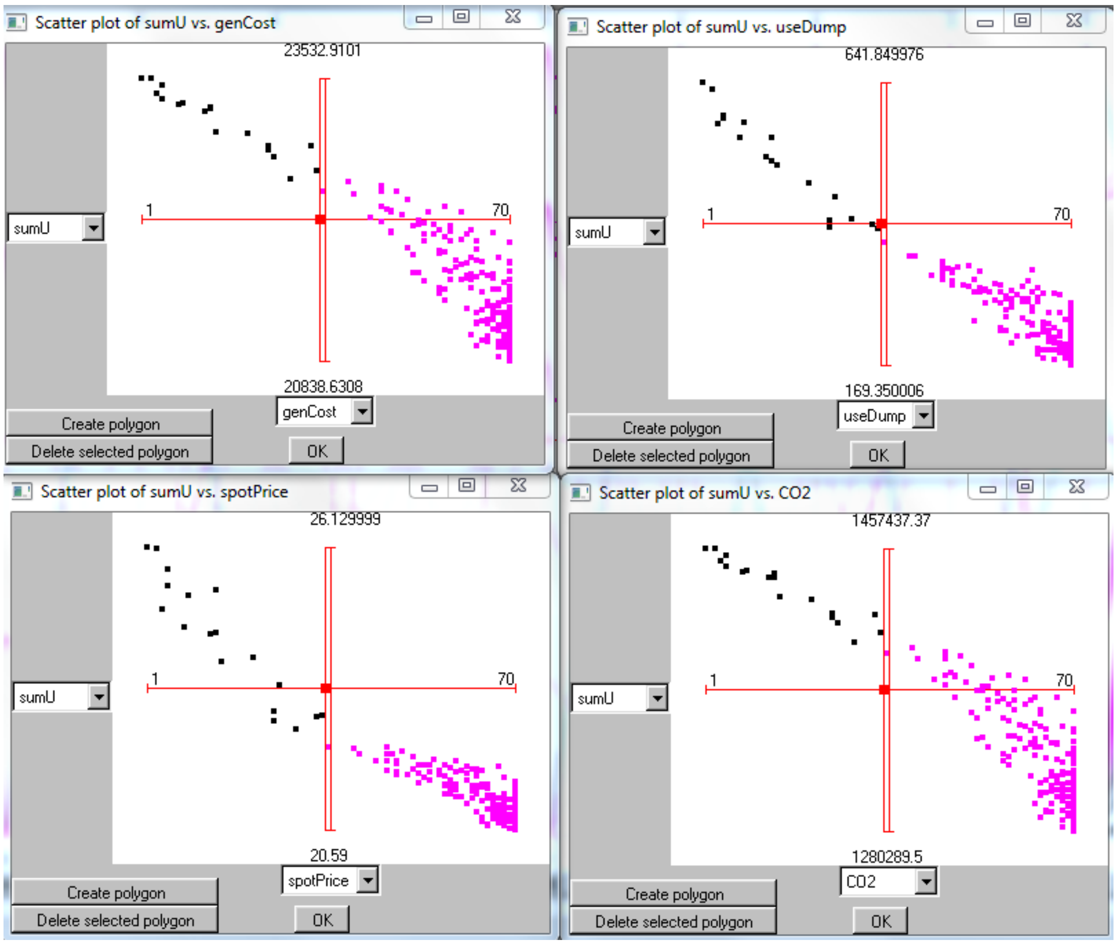


FIGURE 5.7: genCost/$HC = 70$ result. Scatter plots showing sumU on x-axis, OF values on y-axis. (a) Top left: genCost (b) Top right: useDump (c) Bottom left: spotPrice (d) Bottom right: CO_2.

other OFs to its right are mostly positively correlated. It can be seen however that not all of the lower genCost points lead to the lower useDump points.

In the ||-coords plot of Figure 5.8 that complements that of 5.7, there is also an indication that the system may have a tipping point (bifurcation) dependent upon the value of sumU at around 34 units, as the selected region shown by the two arrows has an upper boundary of 34, and the OFs relating to all these lines seem to be in the top half of the worst performers.

The ||-coords plots of Figures 5.13 and 5.14 show the best performing solution found for the genCost objective. The latter figure makes it clear that it is the wind turbine DG units (indicated by W) that are the primary contributor to the performance of the best solution for genCost, with variable V32 having the

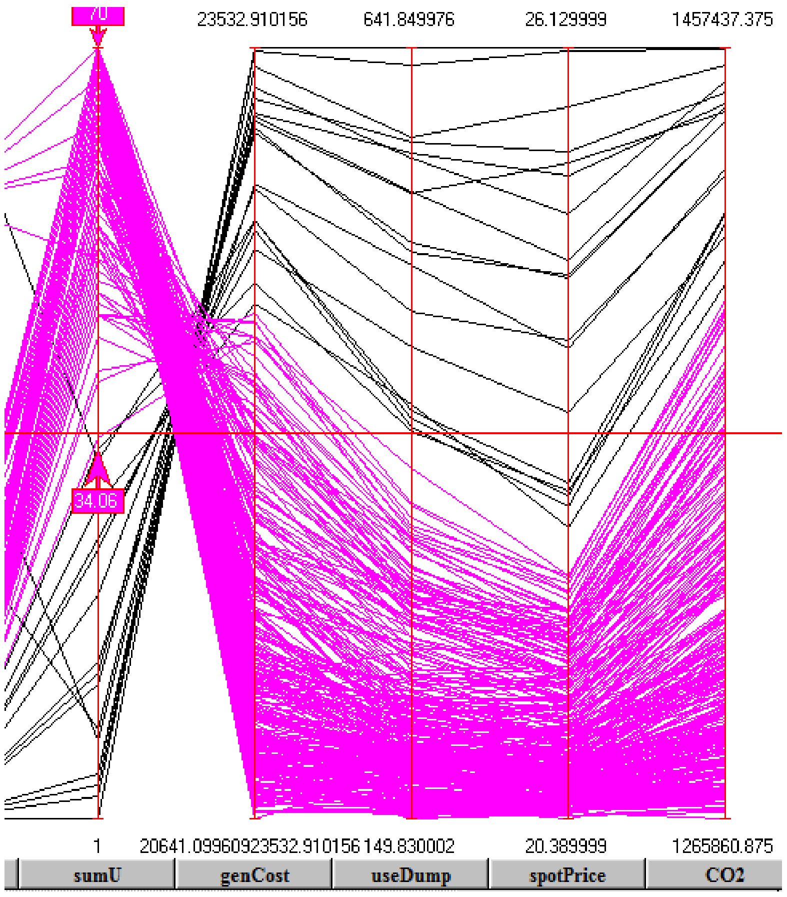


FIGURE 5.8: genCost/HC = 70 result. A ‖-coords plot showing sumU and OFs only, for clarity; a selected region of higher sumU which corresponds to lower OF values. The x-axis shows sumU followed by OFs in order: genCost, useDump, spotPrice, CO_2, and the y-axis their values.

most units allocated, and V11 being the most connected in the network (feeding into node n06). The best performing solution also has the maximum permitted ($HC = 70$) number of DG units allocated.

The scatter plots of Figure 5.9 show results with $HC = 200$. The results are similar to those for $HC = 70$ and as can be seen there is an obvious relationship: the more DG units allocated, the better the OF performance, for all four OFs. The best performing solutions are shown as selected in Figure 5.10 by the upper and lower bounding arrows. The minimum number of DG units allocated has increased well beyond the apparent tipping point seen previously, as HC has been relaxed to 200, so the OF results do not appear to be obviously split into different regions as before. The number of generations has not been sufficient for the optimisation to find the maximum number of DG units allowed by HC, but total DG has grown by around 50% from the optimisation used with the previous HC constraint.

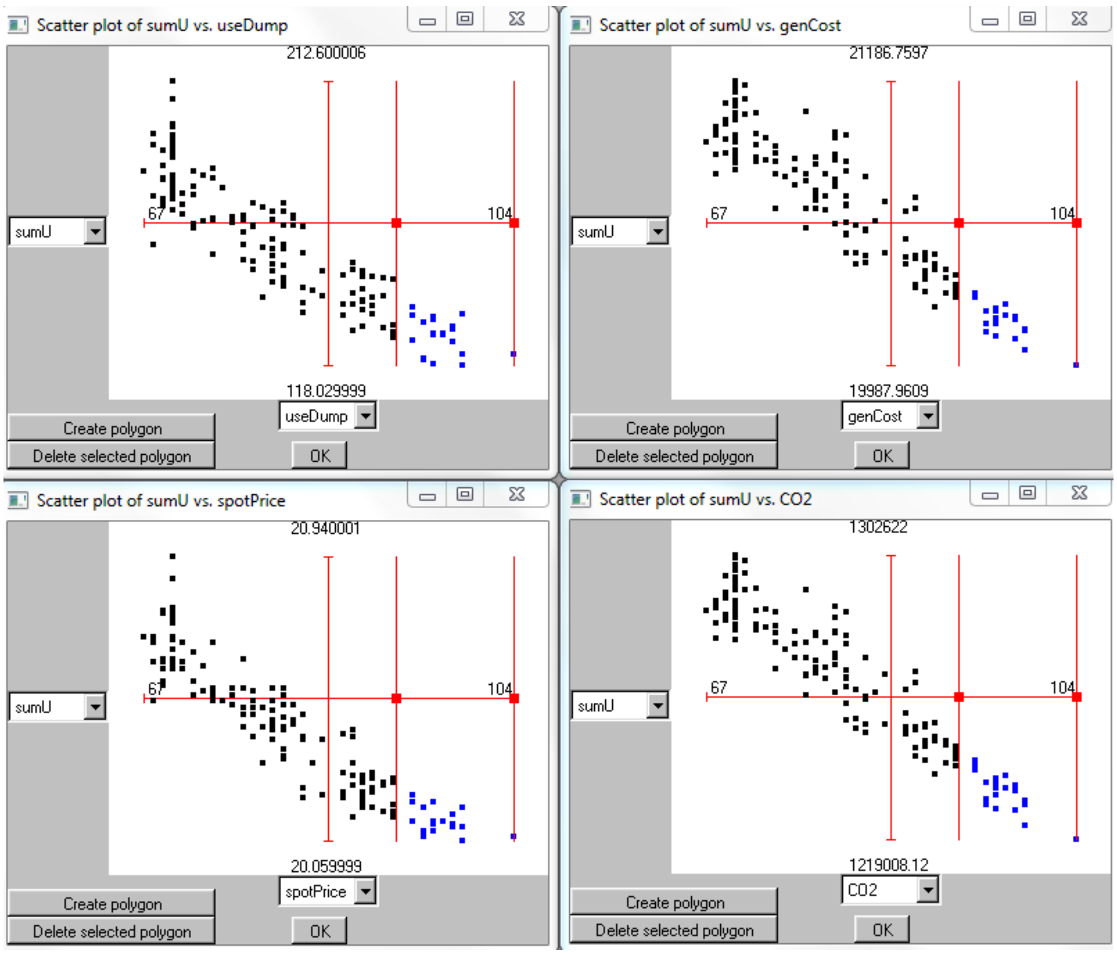


FIGURE 5.9: genCost/$HC = 200$ result, Scatter plots showing sumU on x-axis, OF values on y-axis. (a) Top left: useDump (b) Top right: genCost (c) Bottom left: spotPrice (d) Bottom right: CO_2.

The scatter plots of Figure 5.11 show results with $HC = 35$, and Figure 5.12 shows the same data plotted with ||-coords and showing the selection of certain solutions. As can be seen, the best performing solutions are those with the maximum permitted number of DG units assigned, which is to say where $sumU = 35$. The magenta solutions are those selected for performing well on generation cost, while the green set are selected for performing well on useDump. In Figure 5.12 is can be noted that a low generation cost tends to be associated with a low CO_2 value, while a low useDump value tends to be associated with a low spotPrice value. This suggests, as might be expected, that the DG units are being effective for both cost and CO_2, but when they cannot be fully utilised

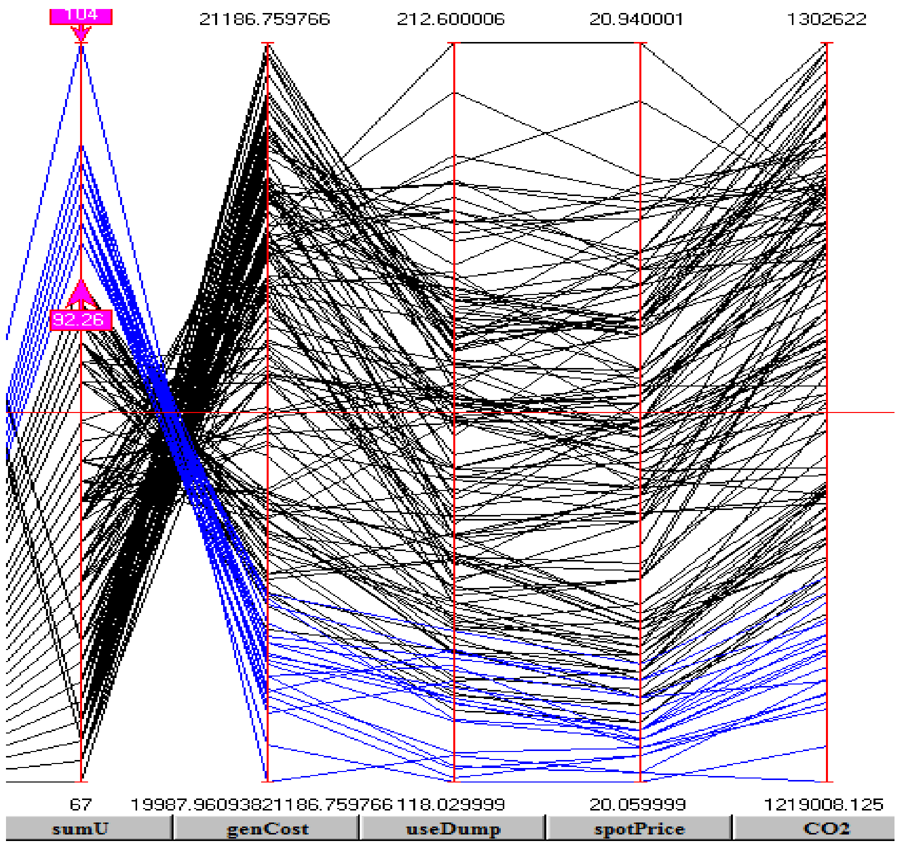


FIGURE 5.10: genCost/HC = 200 result. A ||-coords plot showing sumU and OFs only, for clarity; a selected region of higher sumU which corresponds to lower OF values. The x-axis shows sumU followed by OFs in order: genCost, useDump, spotPrice, CO_2, and the y-axis their values.

(due to weather conditions for example), other generation cuts in, which is more efficient (it can better match demand and is more easily planned for) but more costly. However, there are highlighted solutions which do well on all OFs with only a slight increase in useDump, illustrating that good and practical trade-offs are available.

The scatter plots still show, as expected, the direct relationship between increasing number of DG units and the decreasing (improving) value of the OF values. However, the scatter plots also show a range of outcomes even when $sumU$ = 35, since it is also the combination of types of DG unit comprising the total of 35 which is of importance. Moreover, it is the location of the DG unit type which is also of importance, in that, say, 3 units of micro gas turbine at node N03 may well have a different impact than the same combination at node N28. In particular, the micro wind and solar PV units produce a variable output depending upon the weather, so the precise details of the combination of these units has a corresponding effect.

The same data set, for genCost and $HC = 35$, is shown in the ||-coords plot of Figure 5.15, in which only the variables non-zero for the best performing solutions are shown, with the same selection as Figure 5.11. The solutions having lowest cost of generation (shown in magenta lines) are those with the greatest number of micro wind turbines, although it does also depend upon which node the wind turbines feed into, as the number of transmission lines connected and their capacities also partly determine the outcome.

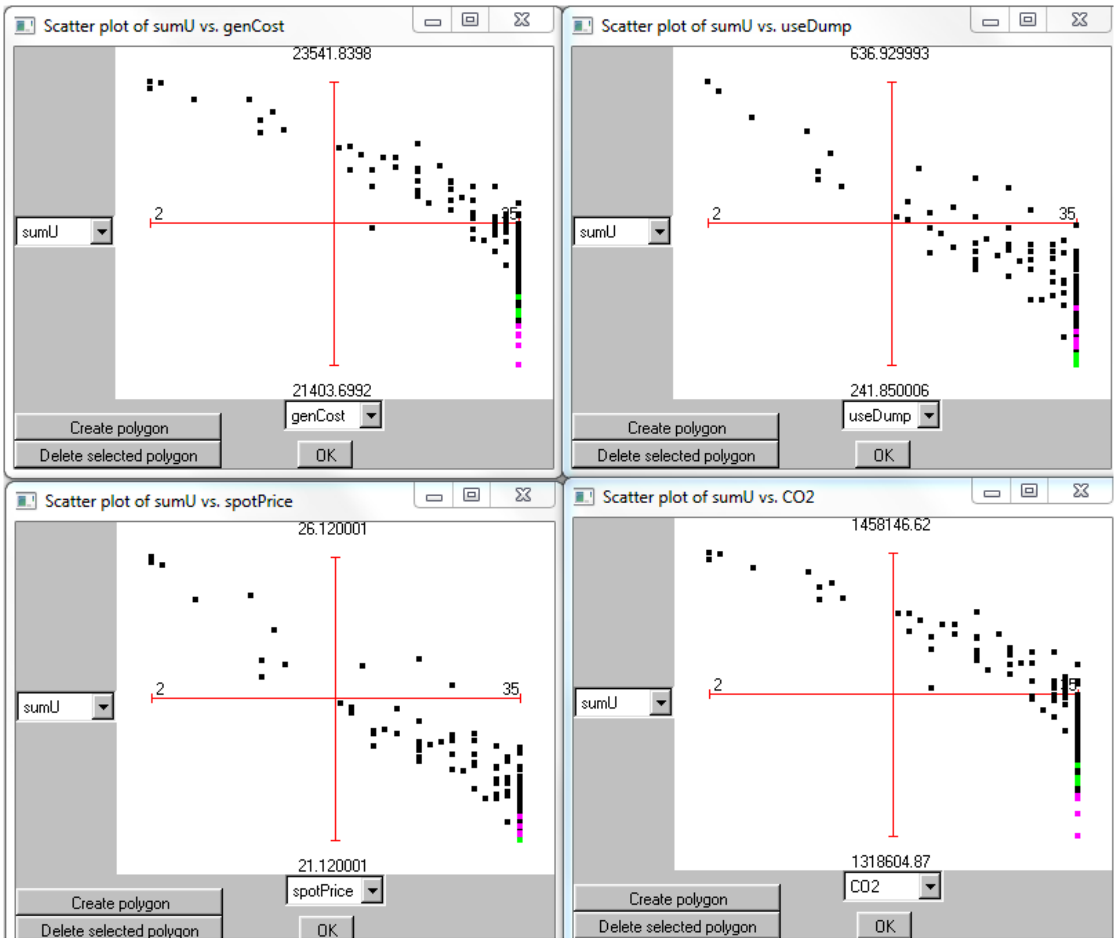


FIGURE 5.11: genCost/$HC = 35$ result. Scatter plots showing sumU on x-axis, OF values on y-axis. (a) Top left: genCost (b) Top right: useDump (c) Bottom left: spotPrice (d) Bottom right: CO_2. Best-performing solutions selected for genCost (magenta) and best for useDump (green).

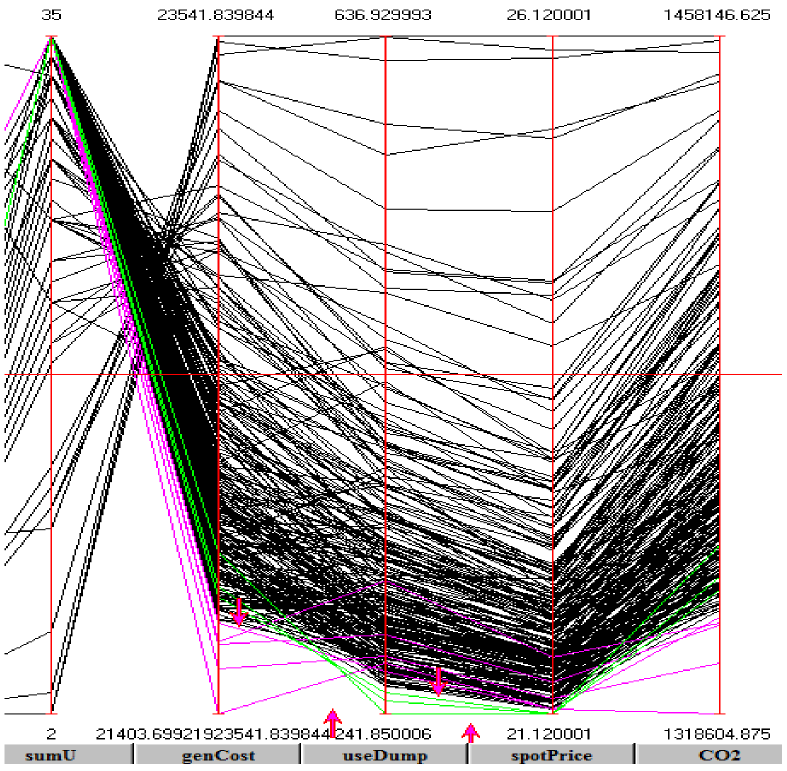


FIGURE 5.12: genCost/HC = 35 result. A $\|$-coords plot showing sumU and OFs only, for clarity. The x-axis shows sumU followed by OFs in order: genCost, useDump, spotPrice, CO_2, and the y-axis their values. Best-performing solutions selected for genCost (magenta) and best for useDump (green).

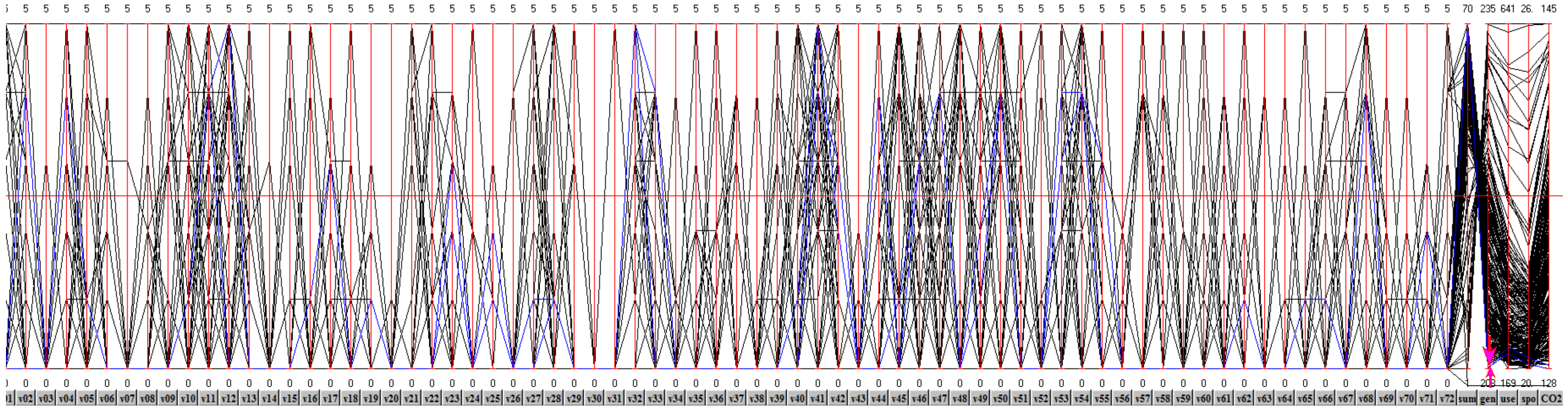


FIGURE 5.13: genCost/$HC = 70$ result. ‖-coords plot of the entire data set, showing the 72 variables, the derived sumU followed by the 4 objective functions. The best performing point of genCost is shown selected by the two arrows at the far right bottom, and its associated variables shown in blue when in colour. See Figure 5.14 for a clearer picture.

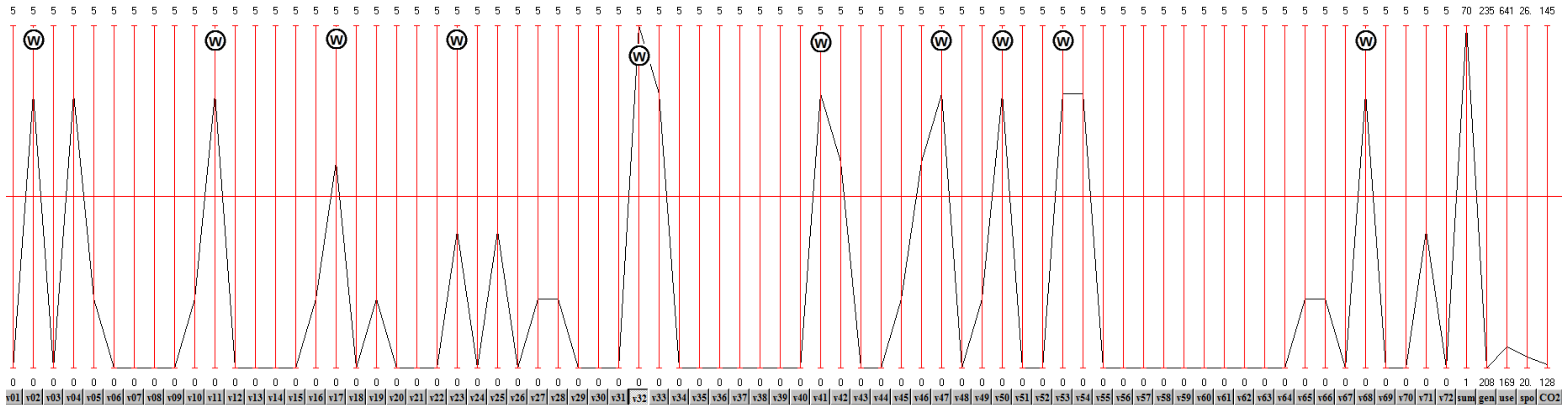


FIGURE 5.14: genCost/$HC = 70$ result. ||-coords plot of the entire data set, showing the 72 variables, the derived sumU followed by the 4 objective functions. The best performing point of genCost and its associated variables are shown, with the rest filtered out. The (W) annotation against a variable indicates that it is a Wind DG unit.

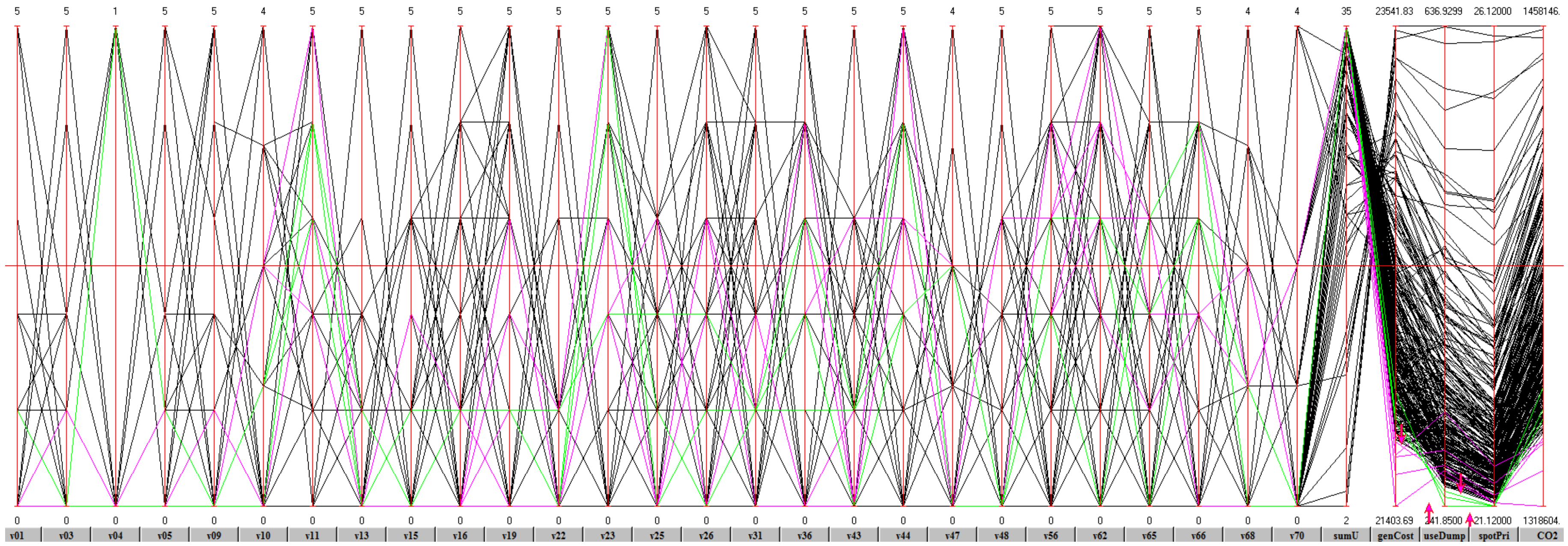


FIGURE 5.15: genCost/$HC = 35$ result. ∥-coords plot of the genCost data set, showing only those variables non-zero for the selected best performing solutions, as selected in Figure 5.11.

5.5.2 Optimising for DG allocation by sum of DG units

This section gives the results for the optimisation defined in section 5.4.2.

Here, the results set of the first optimisation of 5.4.2, with the original line capacity, is termed R003, and the second set with the higher line capacity is termed R008.

The plot in Figure 5.16 shows the entire 72-variable set and the objective functions for the R008 result set with the higher line 11 capacity. This has some variables as always 0, hence these can be said to be of no relevance to further optimisation runs, allowing them to be possibly removed in future, in order to improve optimisation performance.

The results of the objective function minimisations appear in Table 5.2, although sumU (the total number of DG units used) is not listed as this is always between 0 and 35, given the hard constraint. It can be seen that just changing the one line capacity from 65 MW to 130 MW improves each OF result.

Table 5.2: New and previous best objective function results, from any solution, optimising with OF1 = sumU

Run	useDump	spotPrice	CO_2
R003	300.73	21.67	1,348,057
R008	260.00	21.22	1,346,496

The plot in Figure 5.16 shows that the decision variable V11 (see Figure 5.4 & Table 5.1), which contains the number of units of Wind DG for node 6, when having the value 5, is on the many highly performing solutions, including the best solution of all. The R003 results shown in Figure 5.17 in a similar fashion to Figure 5.16, seem to indicate that the reasons for the improved performance in R008, is that the number of DG units for node 9 are no longer so important as variables v19 (node 9, Gas) and v20 (node 9, Wind) are no longer on the optimum path in R008, while for R003 both are at maximum (5). R008 also has fewer variables at 0, which seems to suggest the network load may be better balanced too.

The scatter plot of Figure 5.18 shows the variation of the useDump values against the total number of DG units (sumU), with the most converged points manually selected, and in Figure 5.19, the subset of those selected points in which V11 has 5 DG units, are highlighted. Manually selecting points in this way is not easy due to the shape of the front and the way the tool works, and it can be seen that a few dominated points have been included when it would be preferable to exclude them, nevertheless, the high-performing cluster of Figure 5.19 suggests that there is a tipping point caused by the number of DG units assigned to V11, which is the variable for micro wind units of node 6, the highest ranked node in both capacity and connection rankings. When $V11 = 5$, the maximum allowed in this scheme, the solutions appear in the best performing cluster, but when $V11 \leq 4$, the clusters are not longer apparent.

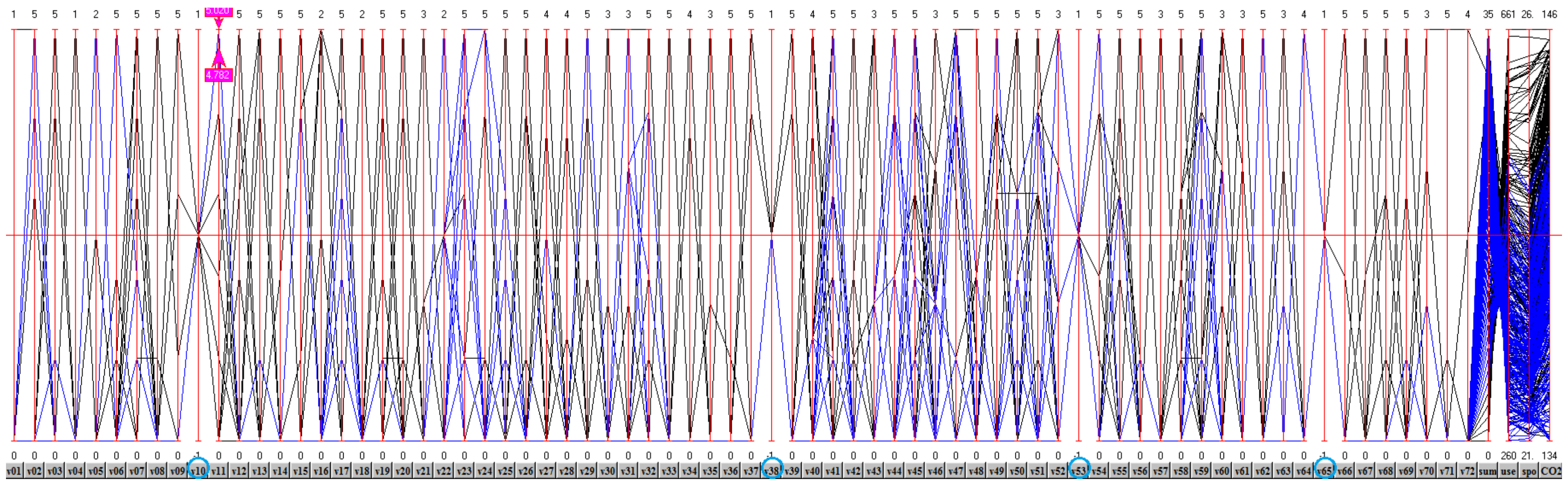


FIGURE 5.16: sumU,$HC = 35$ result. ∥-coords plot for R008 showing all 72 variables and 4 OFs, with selection of results in which V11 has 5 units, and in which V10, V38, V53 and V65 are always 0.

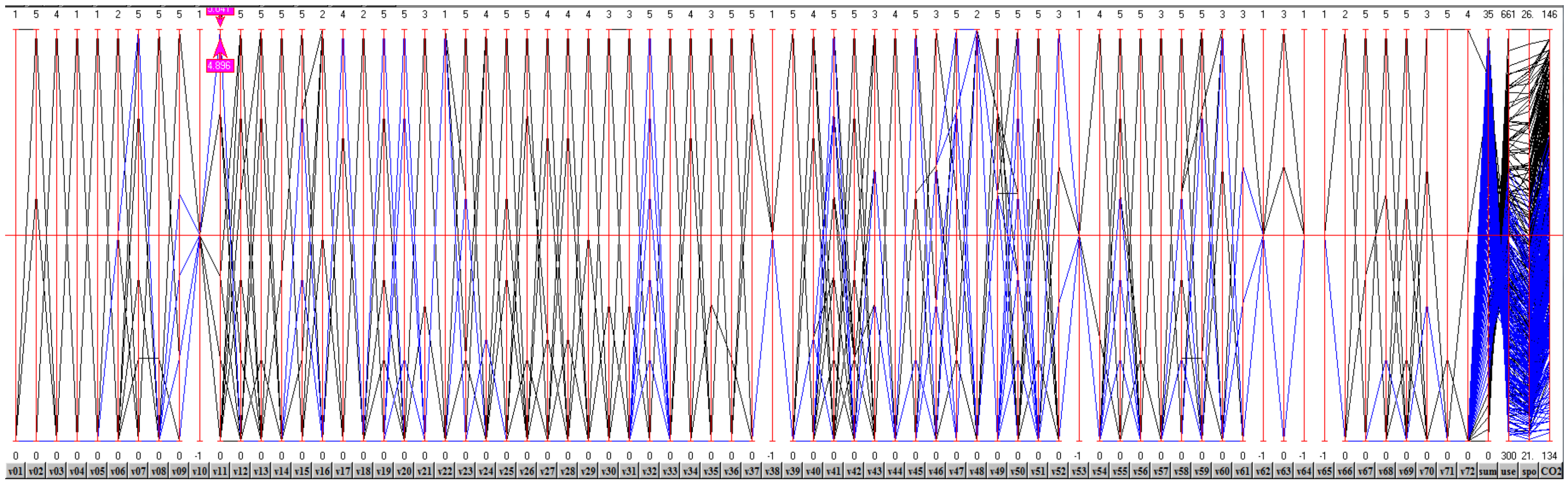


FIGURE 5.17: sumU,$HC = 35$ result. $\|$-coords plot for R003 showing all 72 variables and 4 OFs, with selection of results in which V11 has 5 units.

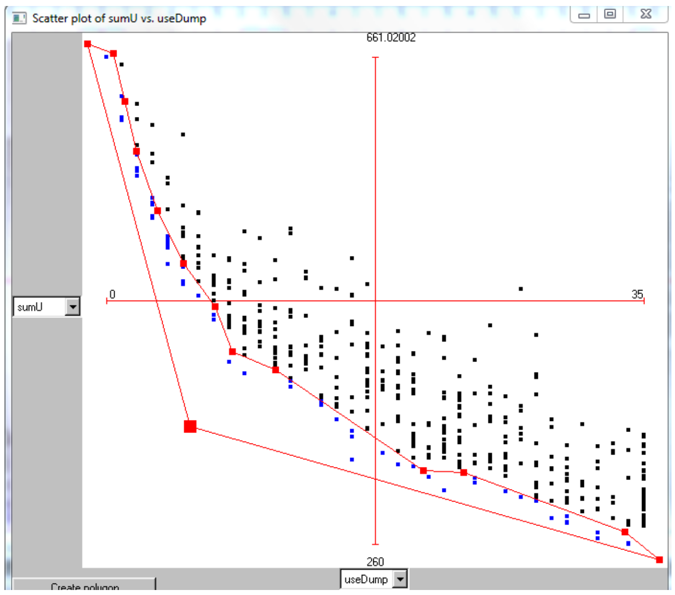


FIGURE 5.18: sumU,HC = 35. A scatter plot for R008, showing sumU on x-axis against useDump on y-axis, with the most converged points selected by hand using the polygon tool of ParallAX.

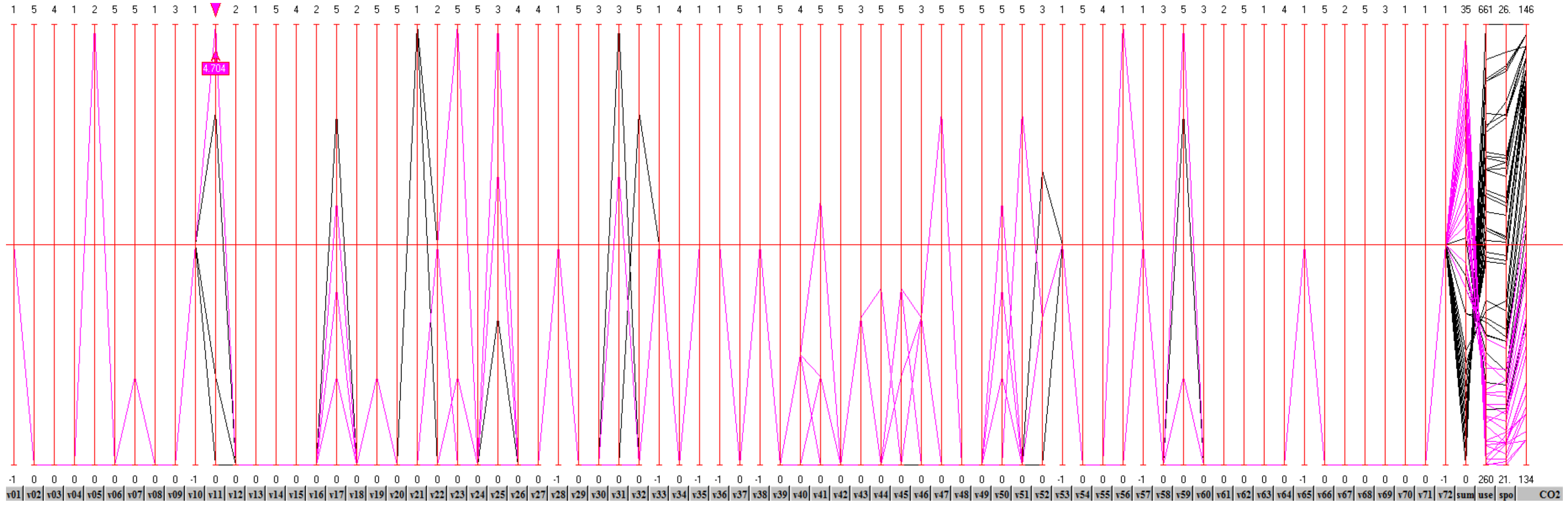


FIGURE 5.19: sumU,$HC = 35$ result. ‖-coords plot for R008; the set of points selected in Figure 5.18 are isolated and shown here, with those that have $V11 = 5$ selected (in magenta), resulting in two apparent clusters, the lower set being the best performing.

5.5.3 Optimising for DG allocation and line capacities

This section gives the results for the optimisation defined in section 5.4.3.

These results are for the case in which the line capacity of each line in the system is enabled to evolve along with the DG unit assignment for each node in the system. The design vector now comprises 72 DG genes and 41 LC genes giving 113 genes in total, and the solution includes the four OF values resultant.

In the ||-coords plots, the OFs are given in the following order: sumU, useDump, spotPrice, CO_2, as the last four axes on the right.

Figure 5.20 is a ||-coords plot that shows all non-dominated solutions of every generation in the history of an optimisation run, intended as a first view of the global picture. The solutions having $V11 = 5$ have been selected (in magenta), remembering that V11 is the variable for micro wind turbines on node 6, the highest ranking node for both connections and capacity. It can be seen that some of these solutions are among the best performers for the useDump OF, and that many other DG variables for the same solutions are 0. Some DG values are always 0: those of V06, V31, V42. It is also apparent that most LC values are in approximately the bottom quartile of the plot with only a few peaks above the mid-point, and that they all have a spread of values.

In order to clarify the global view, Figure 5.21 is a plot of just the last generation of the run, in which all DG variables valued 0 are hidden, and in which the best performing solutions of each OF have been selected and then isolated, where magenta is used for useDump, green for spotPrice and blue for CO_2. It is now apparent that most DG variables are 0, including those of node 6 and even the second ranking nodes, 10 or 4, of the connection or capacity rankings, only have one DG variable each and those are below the maximum of 5. This suggests that the DG units play a less important role than with the datum settings of the line capacities. Line 37 has converged to one value $L37 = 53$, which may be premature, but is nevertheless in the best performing solutions.

To zoom into detail, Figure 5.22 shows the solutions of the final generation, with LC variables LC01 to LC41, OFs (sumU, useDump, spotPrice), & sumCap which is the derived total capacity of all lines of the solution. The backgrounds of the cells are coloured to give a type of heat map to show the LC deviance from the datum, in which white shows no difference, $green < datum$, $red > datum$ and $black = 0$, while the OFs are also coloured white. The datum design is shown on the second row in light green. Figure 5.23, for clarity, shows just the selected best performing solutions subset, derived from those in Figure 5.22.

Figure 5.24 shows the same best performing solutions, but with their DG genes, with also previous results for comparison, with the results for this run termed R006, and previous ones as described in preceding sections, being R003 and R008. Only the DG genes which are non-zero for ≥ 1 solutions in this set are shown.

An immediate impression from Figure 5.22 is that the reds tend to be on the right and the greens to the left, which is because in the datum design, the lowest LCs are on the right and the highest on the left, indicating that the optimisation has tended to increase the lowest LCs (16 to 32) while decreasing the highest LCs (65 to 130), with an apparent tipping point at around 32 MW, as it seems less common that these increase, while it is more common that the 16s increase. Thus the optimisation seems to be eliminating spare capacity while adding capacity where it would be beneficial. While a real national grid system would need contingency capacity, it is not the purpose of the optimisation to model this.

However, this is not always the case, as some LCs of 300 MW have been added and not decreased. This is probably due to a combination of (i) the Gaussian distribution being capped at 300 and these extreme values arising naturally out of the stochastic assignment process; (ii) the optimisation is incomplete - it has run for 2,010 function evaluations (67 generations) but needs more generations to converge as far as it is able, to better eliminate inefficiencies.

Considering the CO_2 result of R006, it can be seen from the detail figures (5.23 & 5.24), that there is now a much better OF value, by an order of magnitude.

However, the corresponding spotPrice is an order of magnitude worse than that of the other result sets, thus clearly a decision-maker has a trade-off to consider, which may be helped by factoring in carbon taxes and costs of generation.

There are in fact two best solutions for CO_2, which have the same results for all OFs and the same DG unit assignments, the only difference between them being the LC value of line 22, which is 44 MW for one and 300 MW for the other, while the original datum value for the line was 16 MW. This means that 44 MW for line 22 is the maximum value needed for this combination of variables of the design vector.

It should be noted that the solution of best CO_2 with the lowest sumCap value, 1760 MW, is less than the datum sumCap of 1954 MW, while it also has fewer than 35 DG units. Thus the system has optimised to, for CO_2, a lower than original total transmission capacity.

Also of interest is the 3rd best CO_2 solution, which while having a slightly higher value of CO_2, has a lower value for useDump and an even lower total capacity, the only difference being the $LC6 = 0$, between nodes 6 and 2, while the two best solutions have $LC6 = 17$ MW. This result can only be due to the lack of power available from LC6 being compensated for by another OCGT generator, which is better able to match the demand, but causing the system to be less efficient overall, either due to unit commitment scheduling problems or transmission losses. It would be interesting to run a series of single runs of Plexos with these DG and LC settings, varying just LC6 to see if an optimum value lies above or below $LC6 = 17$ and by how much[2].

Considering the spotPrice result of R006 and figures (5.23 & 5.24), this result is lowest of all, though beating the next nearest, of R008, by just 0.01, while having a comparable CO_2 value (about 2.6% more than the best R008 result), and having a useDump value nearly half of that of R003 and R008 results.

[2]See subsequent section 5.5.3.1

Although the best solution of R006 for spotPrice has the full 35 DG units assigned, they are either not present or at low units in all variables of each node in the 4 highest ranking nodes of either ranking scheme, which suggests that the precise DG assignment is of less importance than the LC values.

The sumCap of this solution is 30% greater than the datum, having a number of lines (1,8, 12) at 300 MW. Line 1 is between nodes 1 and 2, both of which have OCGT generators, and line 12 is between nodes 6 and 10. Node 6 is the highest ranked node in both rankings, and node 10 is the below node 6 in the connection rankings. It is possible that the 300 MW capacities have space capacity, but it does suggest that much higher values than the datum are beneficial. More single runs of Plexos with these settings, varying one of those lines at a time, would establish the best values. This solution may be a reasonable compromise, given moderate CO_2, good useDump and best spotPrice.

Considering the useDump result of R006 and figures (5.23 & 5.24), the top 3 solutions found have useDump close to zero, with the best at only 0.03 which is very efficient indeed, and a CO_2 value which is less than the next best solution (from R008). However, the spotPrice at 212.6 is then an order of magnitude greater than the best spotPrice of 21.21.

This R006 solution has a sumCap greater than the datum. but only by 0.5%, and has 34 out of a possible 35 DG units assigned, and again these are not concentrated at the highest ranking nodes of either scheme. Although sumCap is more than the datum, only 18 out of 41 LCs are greater, with one LC being the same and the rest lower than the datum. With only one, $LC8 = 300$, with possibly much spare capacity, it seems that this solution may have evolved to have the line capacities where they are approaching optimal, but this would need to be confirmed by further runs varying just LC8.

It is instructive to note that the next best useDump solution of R006, having a value of just 0.8 (which although an order of magnitude more than the best, is still very good), has both an extra DG unit in total, giving the full complement of 35, a larger sumCap, and slightly lower spotPrice and CO_2 values. The extra

DG unit total actually comprises two variables which were 0 now being one (V63, V65), and one which was 1, now being zero (V3). It is not possible from this data to tell if this difference of 1 MW in total of DG would be sufficient on its own to account for the difference in spotPrice and CO_2, but the line capacities are sufficiently different to suspect that LC plays a part.

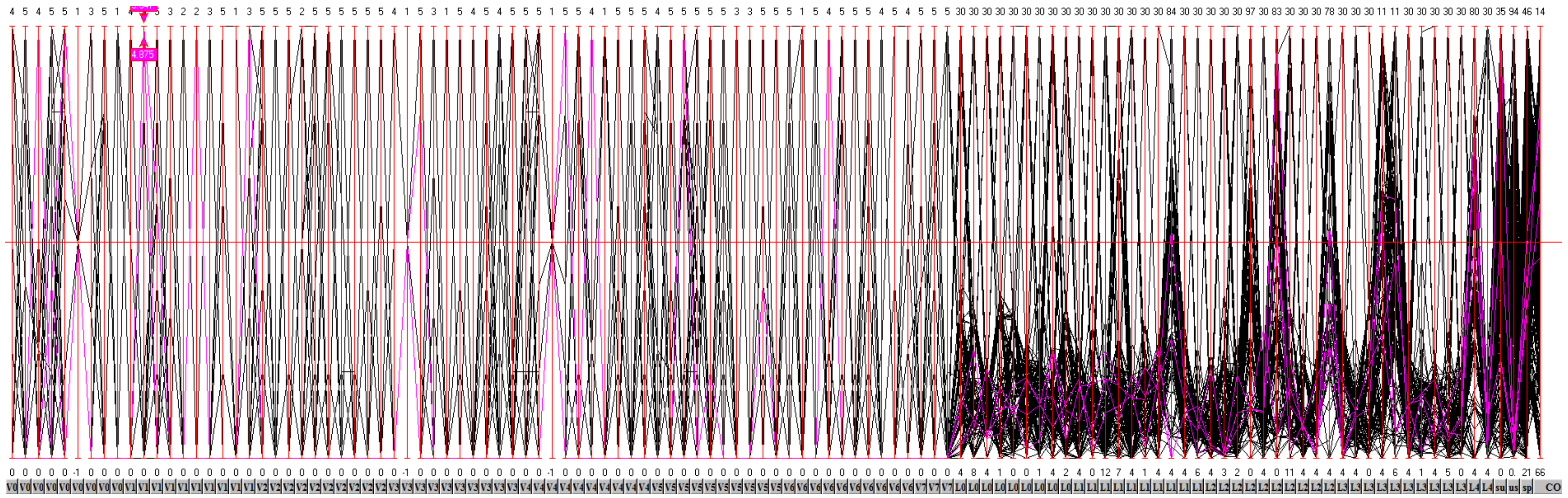


FIGURE 5.20: ‖-coords plot for LC/sumU,$HC = 35$ result. Showing the 72 DG unit values and the 41 line capacity values, along with the 4 OF results, in which the solutions having $V11 = 5$ have been selected (in magenta). The last four columns give the OFs: sumU, useDump, spotPrice.

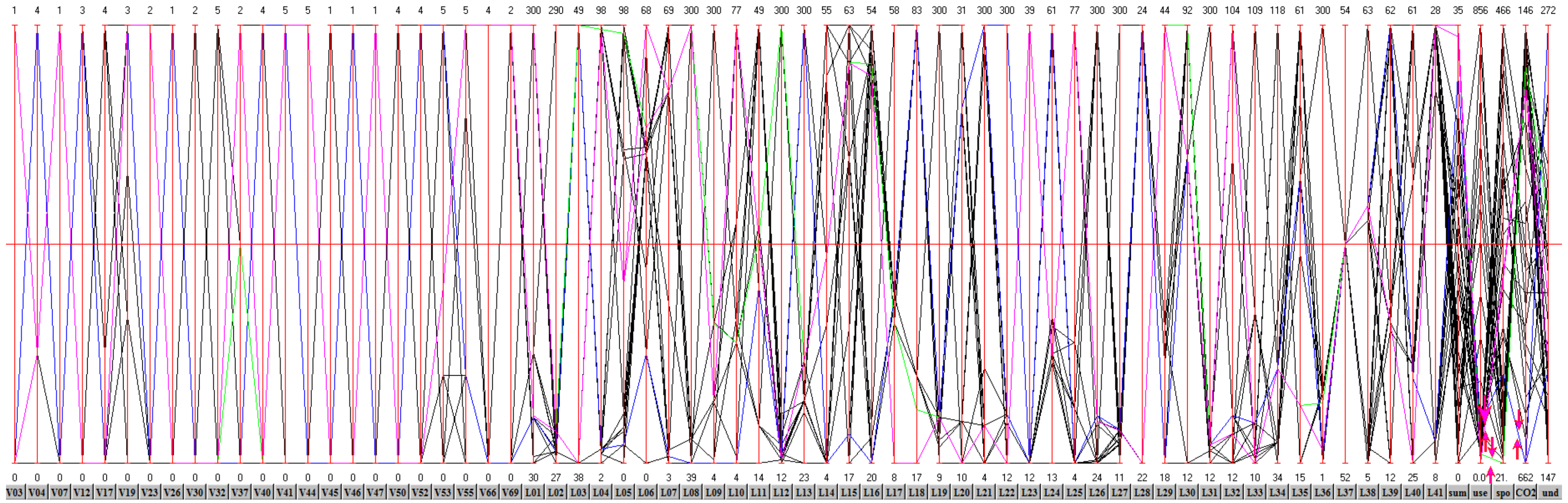
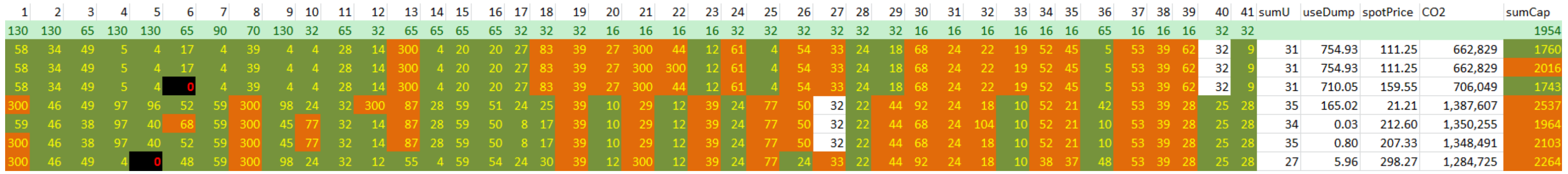


FIGURE 5.21: ‖-coords plot for LC,sumU,$HC = 35$ result. DG variables valued 0 are hidden, and the best performing solutions of each OF have been selected and isolated, where magenta is used for useDump, green for spotPrice and blue for CO_2.

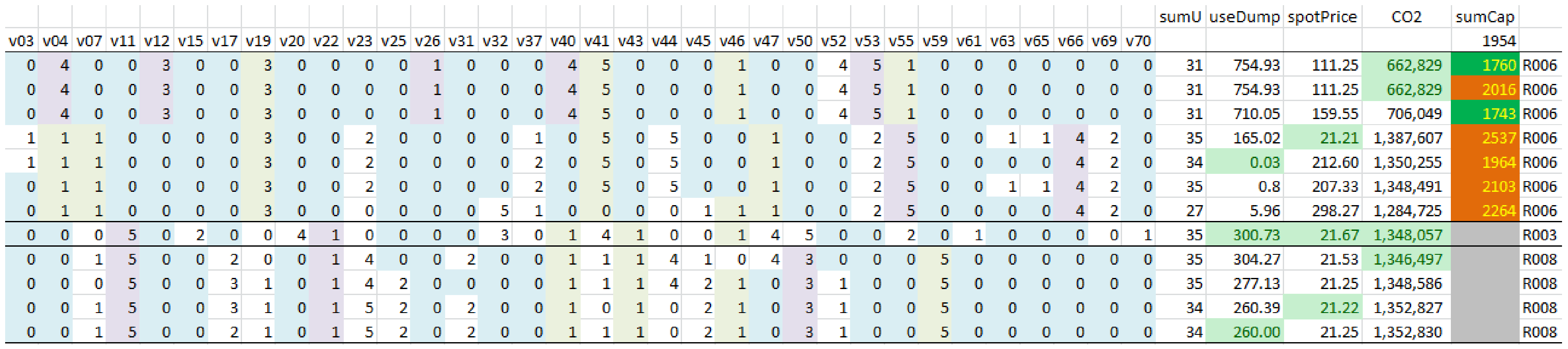
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	sumU	useDump	spotPrice	CO2	sumCap
130	130	65	130	130	65	90	70	130	32	65	32	65	65	65	65	32	32	32	16	16	16	16	32	32	32	32	32	32	16	16	16	16	16	16	65	16	16	16	32	32					1954
59	46	38	97	40	68	59	300	45	77	32	14	87	28	59	50	8	17	39	10	29	12	39	24	77	50	32	22	44	68	24	104	10	52	21	10	53	39	28	25	28	34	0.03	212.60	1,350,255	1964
300	46	38	97	40	52	59	300	45	77	32	14	87	28	59	50	8	17	39	10	29	12	39	24	77	50	32	22	44	68	24	18	10	52	21	10	53	39	28	25	28	35	0.80	207.33	1,348,491	2103
300	46	49	4	0	48	59	300	98	24	32	12	55	4	59	54	24	30	39	12	300	12	39	24	77	24	33	22	44	92	24	18	10	38	37	48	53	39	28	25	28	27	5.96	298.27	1,284,725	2264
58	27	49	5	96	48	59	39	300	27	17	18	63	4	59	20	24	30	24	31	29	12	12	25	4	26	27	24	18	68	24	18	19	52	53	5	53	5	62	32	9	12	157.03	466.07	1,200,040	1575
300	46	49	97	96	52	59	300	98	24	32	300	87	28	59	51	24	25	39	10	29	12	39	24	77	50	32	22	44	92	24	18	10	52	21	42	53	39	28	25	28	35	165.02	21.21	1,387,607	2537
300	43	49	4	11	48	69	300	99	24	17	14	62	4	59	54	24	30	39	12	300	12	12	29	4	24	33	24	18	92	28	76	12	38	45	48	53	37	62	49	28	9	208.61	269.29	1,099,240	2286
30	43	49	5	7	48	5	53	98	24	33	18	62	55	63	54	24	83	39	12	300	44	12	61	4	24	33	24	18	68	20	12	44	38	61	48	53	5	27	33	28	15	256.79	401.78	1,011,647	1762
300	43	49	4	8	48	69	300	99	24	17	14	62	48	17	54	24	30	39	12	300	12	12	29	13	300	23	24	18	12	20	104	44	38	61	48	53	37	28	33	28	8	264.67	422.81	1,049,206	2498
58	43	49	98	70	30	68	53	6	28	49	30	87	4	20	50	27	30	300	10	4	44	12	28	24	24	300	24	26	68	24	22	11	38	61	48	53	39	62	25	9	16	302.13	252.68	1,137,950	2056
30	34	38	2	4	48	27	53	46	44	49	18	62	4	20	52	27	30	39	27	29	12	12	29	4	24	33	24	18	68	300	22	13	38	61	48	53	7	62	32	25	20	337.42	223.49	972,797	1568
300	43	49	5	8	48	68	53	98	24	49	24	62	48	17	22	26	83	24	12	4	44	12	61	4	48	23	24	18	12	20	18	44	38	45	48	53	5	46	33	28	17	361.52	448.53	946,514	1691
34	34	38	5	4	48	4	300	45	24	17	18	62	55	63	50	27	30	39	12	29	12	12	25	4	24	23	24	18	68	300	22	19	38	61	48	53	7	62	61	8	14	490.72	350.02	880,022	1827
58	27	49	5	96	17	59	39	4	4	28	14	63	4	59	20	24	30	24	27	300	44	12	61	4	26	33	24	18	68	24	22	19	52	45	5	53	39	62	32	9	23	505.40	197.39	912,311	1603
59	46	49	5	4	17	3	39	4	77	28	14	300	28	59	20	27	83	39	27	29	44	12	24	77	50	33	24	18	68	24	22	19	52	53	5	53	39	60	25	9	28	535.29	313.65	848,750	1668
98	34	49	98	98	48	68	53	98	24	17	300	62	55	63	52	27	30	19	12	300	12	12	25	24	24	11	24	24	92	12	18	19	38	61	48	53	7	30	61	28	0	562.62	24.24	1,460,511	2228
58	35	49	5	4	17	4	39	45	5	49	18	300	4	49	20	24	30	24	27	29	44	12	60	4	54	23	24	18	68	24	22	19	52	53	5	53	5	62	32	9	25	625.92	324.94	782,586	1478
58	27	38	98	11	64	4	39	4	4	14	14	62	20	49	23	58	83	9	10	68	44	12	24	4	25	23	24	18	68	24	12	109	52	45	300	53	39	62	25	9	19	663.20	150.15	751,819	1729
58	34	49	5	4	0	4	39	4	4	28	14	300	4	20	20	27	83	39	27	300	44	12	61	4	54	33	24	18	68	24	22	19	52	45	5	53	39	62	32	9	31	710.05	159.55	706,049	1743
58	34	49	5	4	17	4	39	4	4	28	14	300	4	20	20	27	83	39	27	300	44	12	61	4	54	33	24	18	68	24	22	19	52	45	5	53	39	62	32	9	31	754.93	111.25	662,829	1760
58	34	49	5	4	17	4	39	4	4	28	14	300	4	20	20	27	83	39	27	300	300	12	61	4	54	33	24	18	68	24	22	19	52	45	5	53	39	62	32	9	31	754.93	111.25	662,829	2016
300	34	49	98	68	48	68	53	98	24	17	300	62	4	63	54	24	30	19	12	300	12	12	25	13	24	11	24	26	92	12	18	10	52	61	48	53	7	12	61	28	6	1601.25	25.38	1,450,958	2326
98	34	49	98	68	48	68	53	98	24	17	24	62	4	63	52	27	30	24	27	4	38	12	25	24	300	11	24	26	92	12	12	19	38	45	48	53	7	30	61	28	0	2441.26	29.21	1,453,166	1877
300	43	49	4	68	48	69	300	99	24	17	14	62	55	59	54	24	30	39	12	300	12	12	29	13	300	33	24	26	92	24	104	10	62	61	48	53	37	28	61	28	4	3275.71	63.79	1,372,698	2727
300	27	49	98	68	48	68	53	98	24	17	14	62	4	59	54	24	30	19	12	285	12	12	24	13	24	33	24	26	68	24	104	10	52	61	1	53	37	12	61	28	7	4347.32	30.50	1,431,031	2092
98	290	49	98	98	49	68	53	98	44	49	44	62	55	63	20	27	30	19	12	4	12	12	13	4	24	33	24	24	68	12	12	13	118	15	48	53	5	62	32	28	10	4987.80	44.81	1,407,529	1942
30	43	49	98	70	49	68	53	98	24	49	18	23	4	20	54	27	30	300	31	4	12	12	25	24	24	27	24	26	12	12	12	12	38	61	48	53	39	62	61	28	2	5523.89	45.57	1,430,614	1754
30	43	49	98	98	49	59	53	98	44	49	16	62	4	31	54	27	30	24	12	4	12	12	13	4	24	24	24	26	12	20	12	12	118	15	48	53	39	12	61	28	4	6237.88	47.08	1,415,393	1503
98	35	49	4	98	48	68	53	98	24	17	300	62	55	49	52	24	30	19	12	300	12	12	25	24	24	11	24	24	92	12	18	19	38	61	48	53	63	30	61	28	1	7151.97	93.25	1,370,470	2174
98	43	49	4	98	48	68	53	98	24	49	24	62	49	63	54	27	30	24	12	300	38	12	24	4	300	11	24	26	92	20	18	44	34	61	48	53	5	30	48	28	1	8111.28	74.75	1,371,580	2197
30	34	49	98	98	49	59	53	98	44	49	14	23	4	20	20	27	30	39	27	4	12	12	13	4	54	33	24	26	68	20	12	13	118	15	51	53	5	62	32	25	21	8568.47	55.34	1,364,290	1521

FIGURE 5.22: LC,sumU,$HC = 35$, final generation, showing LC01 to LC41, OFs (sumU, useDump, spotPrice), & sumCap (the total capacity for all lines). The top row gives the Line number and other column headings, the second row gives the datum value for each Line capacity, and subsequent rows give the solutions' LCs. Cells are coloured thus: white for $LC = datum$; green for $LC < datum$; red for $LC > datum$; black for $LC = 0$. OFs are white also.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	sumU	useDump	spotPrice	CO2	sumCap
130	130	65	130	130	65	90	70	130	32	65	32	65	65	65	65	32	32	32	16	16	16	16	32	32	32	32	32	32	16	16	16	16	16	16	65	16	16	16	32	32					1954
58	34	49	5	4	17	4	39	4	4	28	14	300	4	20	20	27	83	39	27	300	44	12	61	4	54	33	24	18	68	24	22	19	52	45	5	53	39	62	32	9	31	754.93	111.25	662,829	1760
58	34	49	5	4	17	4	39	4	4	28	14	300	4	20	20	27	83	39	27	300	300	12	61	4	54	33	24	18	68	24	22	19	52	45	5	53	39	62	32	9	31	754.93	111.25	662,829	2016
58	34	49	5	4	0	4	39	4	4	28	14	300	4	20	20	27	83	39	27	300	44	12	61	4	54	33	24	18	68	24	22	19	52	45	5	53	39	62	32	9	31	710.05	159.55	706,049	1743
300	46	49	97	96	52	59	300	98	24	32	300	87	28	59	51	24	25	39	10	29	12	39	24	77	50	32	22	44	92	24	18	10	52	21	42	53	39	28	25	28	35	165.02	21.21	1,387,607	2537
59	46	38	97	40	68	59	300	45	77	32	14	87	28	59	50	8	17	39	10	29	12	39	24	77	50	32	22	44	68	24	104	10	52	21	10	53	39	28	25	28	34	0.03	212.60	1,350,255	1964
300	46	38	97	40	52	59	300	45	77	32	14	87	28	59	50	8	17	39	10	29	12	39	24	77	50	32	22	44	68	24	18	10	52	21	10	53	39	28	25	28	35	0.80	207.33	1,348,491	2103
300	46	49	4	0	48	59	300	98	24	32	12	55	4	59	54	24	30	39	12	300	12	39	24	77	24	33	22	44	92	24	18	10	38	37	48	53	39	28	25	28	27	5.96	298.27	1,284,725	2264

FIGURE 5.23: LC,sumU,HC = 35, best performing solutions of final generation, showing LC01 to LC41, OFs (sumU, useDump, spotPrice), & sumCap (the total capacity for all lines). The top row gives the Line number and other column headings, the second row gives the datum value for each Line capacity, and subsequent rows give the solutions' LCs. Cells are coloured thus: white for $LC = datum$; green for $LC < datum$; red for $LC > datum$; black for $LC = 0$. OFs are white also.



																																		sumU	useDump	spotPrice	CO2	sumCap	
v03	v04	v07	v11	v12	v15	v17	v19	v20	v22	v23	v25	v26	v31	v32	v37	v40	v41	v43	v44	v45	v46	v47	v50	v52	v53	v55	v59	v61	v63	v65	v66	v69	v70					1954	
0	4	0	0	3	0	0	3	0	0	0	0	1	0	0	0	4	5	0	0	0	1	0	0	4	5	1	0	0	0	0	0	0	0	31	754.93	111.25	662,829	1760	R006
0	4	0	0	3	0	0	3	0	0	0	0	1	0	0	0	4	5	0	0	0	1	0	0	4	5	1	0	0	0	0	0	0	0	31	754.93	111.25	662,829	2016	R006
0	4	0	0	3	0	0	3	0	0	0	0	1	0	0	0	4	5	0	0	0	1	0	0	4	5	1	0	0	0	0	0	0	0	31	710.05	159.55	706,049	1743	R006
1	1	1	0	0	0	0	3	0	0	2	0	0	0	0	1	0	5	0	5	0	0	1	0	0	2	5	0	0	1	1	4	2	0	35	165.02	21.21	1,387,607	2537	R006
1	1	1	0	0	0	0	3	0	0	2	0	0	0	0	2	0	5	0	5	0	0	1	0	0	2	5	0	0	0	0	4	2	0	34	0.03	212.60	1,350,255	1964	R006
0	1	1	0	0	0	0	3	0	0	2	0	0	0	0	2	0	5	0	5	0	0	1	0	0	2	5	0	0	1	1	4	2	0	35	0.8	207.33	1,348,491	2103	R006
0	1	1	0	0	0	0	3	0	0	0	0	0	0	5	1	0	0	0	0	1	1	1	0	0	2	5	0	0	0	0	4	2	0	27	5.96	298.27	1,284,725	2264	R006
0	0	0	5	0	2	0	0	4	1	0	0	0	0	3	0	1	4	1	0	0	1	4	5	0	0	2	0	1	0	0	0	0	1	35	300.73	21.67	1,348,057		R003
0	0	1	5	0	0	2	0	0	1	4	0	0	2	0	0	1	1	1	4	1	0	4	3	0	0	0	5	0	0	0	0	0	0	35	304.27	21.53	1,346,497		R008
0	0	0	5	0	0	3	1	0	1	4	2	0	0	0	0	1	1	1	4	2	1	0	3	1	0	0	5	0	0	0	0	0	0	35	277.13	21.25	1,348,586		R008
0	0	1	5	0	0	3	1	0	1	5	2	0	2	0	0	1	0	1	0	2	1	0	3	1	0	0	5	0	0	0	0	0	0	34	260.39	21.22	1,352,827		R008
0	0	1	5	0	0	2	1	0	1	5	2	0	2	0	0	1	1	1	0	2	1	0	3	1	0	0	5	0	0	0	0	0	0	34	260.00	21.25	1,352,830		R008

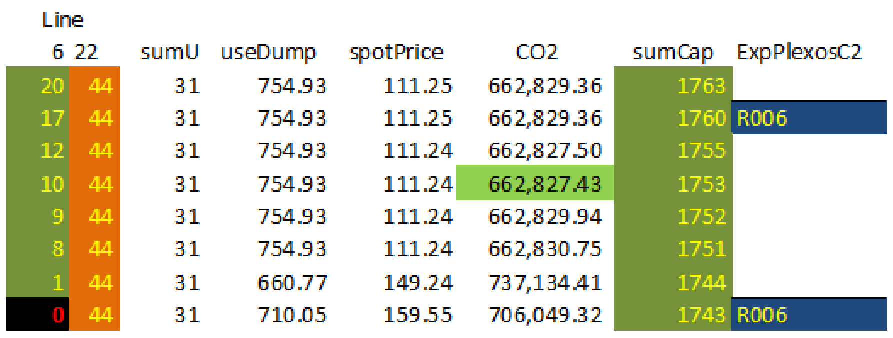
FIGURE 5.24: LC,sumU,HC = 35, best performing solutions, showing non-zero DG genes only, headings giving the Variable number, together with previous results for comparison. Best OFs in a result set are in light green, the sumCap is as per Figure 5.21, and the colour in the DG genes is merely to show similar values by Variable column, to make it easier to differentiate.

5.5.3.1 Supplementary Results

This section gives some results obtained in taking up one or two of the suggestions above, to do some runs of GA/Plexos having just one solution in the population, in order to elicit further information about the behaviour of the system. By using the ability of Ganesh to resume from a previous run's log file, Ganesh can resume from an edited log file having just one solution which has its variable vector set as required, thus running an individual test case. The runs performed here relate to the best solutions of the last generation as given in Figure 5.23.

The following results relate in particular to the three solutions given for CO_2 (the first 3 solutions of Figure 5.23), all of which have the same DG values and mostly the same LC values, with the exception of line 22 in the 2nd solution, and line 6 in the third solution, as described in the previous section.

Here, the values for the DG and LC variables of the best solution for CO_2 are used, with line 22 set to 44 MW, and allowing line 6 to vary from a baseline of 17 MW as seen in the best solution, remembering that line 6 is between node 6 (the highest ranked node by capacity and connection) and node 2 (ranked 2 and 4, respectively). The results are shown in Figure 5.25.



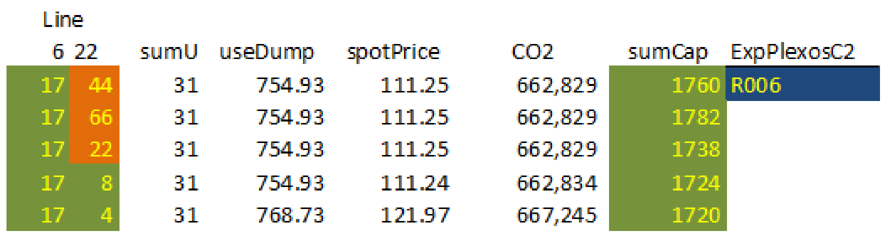
Line							
6	22	sumU	useDump	spotPrice	CO2	sumCap	ExpPlexosC2
20	44	31	754.93	111.25	662,829.36	1763	
17	44	31	754.93	111.25	662,829.36	1760	R006
12	44	31	754.93	111.24	662,827.50	1755	
10	44	31	754.93	111.24	662,827.43	1753	
9	44	31	754.93	111.24	662,829.94	1752	
8	44	31	754.93	111.24	662,830.75	1751	
1	44	31	660.77	149.24	737,134.41	1744	
0	44	31	710.05	159.55	706,049.32	1743	R006

FIGURE 5.25: Similar to Figure 5.23, and using the DG and LC settings of the first CO_2 solution therefrom, varying line 6 and showing the OF results. R006 are the original results, for comparison.

The results in Figure 5.25 show that increasing line 6 from 17 MW to 20 MW has no effect, while lowering it to 8 MW slightly worsens (increases) the OF result for CO_2 but very slightly improves the spotPrice, and leave useDump unaffected. Decreasing line 6 further to 1 MW worsens both CO_2 and spotPrice by a larger margin, but useDump is improved. This shows that the optimum value of line 6 for this design vector, for CO_2, lies in the range 8 MW $<$ optimum $<$ 17 MW, and further single case runs are needed to explore this range. When line 6 is at 8 MW, it suggests that an OCGT central generator (presumably other than that at node 2) is supplying a little more power (since the DG units are unchanged), thus increasing CO_2. Further runs show that line 6 at 10 MW is the best capacity for CO_2 with these settings, however it is also seen that a trade-off for useDump exists for line 6 at 1 MW.

It is interesting to note that when line 6 allows 0 MW, the CO_2 result is better than that when line 6 allows 1 MW, although spotPrice is worse and use-Dump too is slightly worse but still better than the other results. This shows the complexity of the response of the system, and indicates that to understand the system response, the power output (and start up time and ramp-up duration) of each OCGT generator and the power actually transmitted over each line needs to be known. The scheduling of the OCGT generators may cause thresholds of efficiency, considering the useDump figures of these results.

Another set of single case runs were performed, this time keeping line 6 fixed at 17 MW and varying line 22, with the DG units and other LC variables being the same as the previous cases, and the results are shown in Figure 5.26.



Line							
6	22	sumU	useDump	spotPrice	CO2	sumCap	ExpPlexosC2
17	44	31	754.93	111.25	662,829	1760	R006
17	66	31	754.93	111.25	662,829	1782	
17	22	31	754.93	111.25	662,829	1738	
17	8	31	754.93	111.24	662,834	1724	
17	4	31	768.73	121.97	667,245	1720	

FIGURE 5.26: Similar to Figure 5.23, and using the DG and LC settings of the first CO_2 solution therefrom, varying line 22 and showing the OF results. R006 are the original results, for comparison.

The results in Figure 5.26 show that increasing line 22 from 44 MW to 66 MW has no effect, and that decreasing it to 22 MW has no effect either. This indicates that for these settings, line 22 does not need a capacity of more than 22 MW. Decreasing it further to 8 MW slightly worsened the CO_2 result and very slightly improved the spotPrice. Decreasing it yet further to 4 MW worsened all OF results. These results are inconclusive; it may be that a value between 22 MW and 8 MW may be better for CO_2, but not necessarily, as it may be that the best CO_2 result for these settings has been reached. It does show that below 8 MW the value is too low. Further runs are therefore necessary to explore between 9 and 21 MW, starting at 15 MW and from those results see whether to explore up or down from that value, halving the relative difference each time. Of course, using further results from Figure 5.25, setting line 6 to 10 MW and re-running these cases may show different results anyway.

5.6 Comparison with random search

Further work was carried out to compare the performance of Ganesh and a random search algorithm on problems which have some of the characteristics of the optimal power flow problems. Due to the non-availability of a Plexos version licence , it was not possible to run direct comparisons on the optimisations of this chapter. Section 3.4 provides the detail and results.

5.7 Summary

It has been shown that this method, combining an evolutionary multi-objective optimisation with an optimal power flow, can indicate not only the number of DG units, but also their type and their network location, in order to gain high performance when used with an appropriate OPF tool such as Plexos. Moreover, using the MOOEA with Plexos and examining the results with a multi-dimensional visualisation, can be used to assist in the design of network topologies from the perspective of transmission line maximum power flow capacities, by allowing the optimisation process to determine the maximum flow capacities along with the types and locations of DG units.

It also shows that this method could be used to assist in the determination of network topology, working within the limitations of geography and socio-economic factors, from a bus-to-bus connection perspective, through elucidation of at least best and worst lines for transmission and therefore connectivity. It should be remembered that these results relate to particular weather patterns for a region in which this model power grid is imposed, and that the DG unit placement is realistic in that regard, considering micro-wind turbines and solar pv units.

For this network, considered with the datum line capacities, and the weather seen in the stated time period, it appears that wind-turbines may be the most important DG technology to deploy, with the proviso that they work in conjunction with the other DG types and the inter-dependencies could be further explored.

However, the results obtained when the line capacities are enabled to evolve along with the DG unit assignment, show that there is much scope to optimise the grid by deploying transmission lines with the appropriate maximum flow capacity. There may well be a cut-off point where the power supplied by DG units, by 'farms' of wind and solar pv arrays, would be sufficiently high that they would outweigh the influence of realistic transmission line capacities, but at the levels of DG used here, LC is very influential.

The key points arising from this work are as follows:

1. The minimisation of the sumU OF, which is the total number of DG units deployed in the system, causes solutions to evolve a spread of results from 0 up to the maximum imposed by the given hard constraint. DG is shown to be desirable, since in the runs where HC is 35, 70 or 200, solutions evolve having sumU at the HC level, despite the minimisation of it.
2. Both the total capacity of a node and its number of connections have been shown to be important indicators of its contribution to the performance of the system, as seen by the node rankings in Tables C.2 & C.3.
3. Neither node capacity nor node connection can be said to be more important than the other, it depends on the network configuration and the conditions

pertaining to the system at the operating time. In either case, more is better, but there is a cost trade-off.

4. In the preceding section discussing particular results of R006, some suggestions were made to run specific experiments. These can be performed using the resume capability of Ganesh, by copying and editing the log file, so that a resume with one solution is used, with its parameters set as required. Thus Ganesh can facilitate a single run of Plexos, without needing to manually configure Plexos via the GUI, which will be useful when needing to run a series of runs varying one of the 113 parameters at a time.

5. Further experiments could be designed to confirm whether it is the number of lines connected or total node capacity (as defined in this work as total of connected lines' capacities) that is the most influential ranking factor in this model, or to determine what the tipping points are (if any) in flipping from one to the other, as could be the case.

6. The Line capacities in the original definition of this model can be improved by optimisation.

7. It is not the total line capacity, but the targeted line capacity which is the most important LC factor, from an optimisation viewpoint.

8. Total LC is as important as targeted DG as seen in this work, however it is expected that this may change with higher DG power input either through more powerful individual units or arrays of lesser units. Further experiments could be designed to confirm this.

9. Optimisation by Ganesh or other EA, bearing in mind the above points, can be expected to produce a reasonable range of trade-off options to assist in the design of electrical power grids.

10. In further optimisation work, the sumU OF could be replaced by a cost factor to be minimised, for example when applied to a network of national proportions, for better optimisation performance and for realistic scenarios.

The sumU OF showed that there is a trade-off to be made between the number of DG units and the other OFs, but this could be transformed into a cost, such as cost of production and deployment of DG units, which could be added to the cost of power generation (from Plexos), to give an overall cost function to be minimised. This would give an almost unique OF (unlike sumU) which would benefit optimisation performance since rank 1 solutions would appear later in the run, and also be a realistic and more useful metric.

Chapter 6

Conclusions and recommendations

This chapter considers the work of this thesis as a whole, and sets out the meeting of research objectives while describing its limitations, offering conclusions, and positing the possibilities of further work that would be of relevance in following up the work described in this thesis.

6.1 Contributions

The novel Ganesh algorithm and framework was produced, demonstrated to function correctly and usefully applied to optimisation of real-world complex engineering systems, that of the airfoil and an electrical power network.

Self-adaptivity was shown to work, at least in the context of the work performed, with pM,pC,ηM, ηC control parameters defined in the chromosome, even for the novel recombination (crossover) operators, rather than at the level of the population.

Features of the framework having real-world usefulness were:

- The use of the plugin architecture, to separate problem definitions from the optimisation algorithm, was shown to work and to enable new problems to be simply added without the need to change the existing code base.
- The facility enabling a plugin to have its own specific parameter file proved useful in running concurrent airfoil optimisations.
- The ability to resume from interrupted lengthy optimisation runs, proved invaluable in running the power network optimisation problems.

The XFoil tool was improved to make it function better in a batch environment, enabling the airfoil optimisation to be successful.

It was shown that, in this case at least, genetic algorithms can be useful for airfoil optimisation, as this had been thought (Kipouros *et al.*, 2012) to be not the case.

The optimisation of electrical power networks was shown to be feasible using evolutionary algorithms, specifically the Ganesh GA, and that using weather patterns with appropriate DG units would be of benefit as would their specific placement. Enabling transmission line capacities to evolve as part of the solutions produced was also shown to be beneficial and therefore has applicability in real-world network design. It was shown that parallel coordinates may be able to help identify tipping points in system design and that applying ||-coords to multi-dimensional analysis, for both parameters and objective functions, is useful in understanding optimisation problem dynamics and results.

6.2 Limitations

The research of this thesis does have limitations and these are set out in this section.

Ganesh uses self-adaptivity, but there are further ways that self-adaptivity could be utilised and these could be explored by additions to the Ganesh code base. Currently, Ganesh uses the tournament selection mechanism, at the binary level (choosing between two candidate solutions); this could be generalised from the 2-way choice to an n-way choice, in which n is a self-adaptive parameter held at the chromosome level, with a similar method to the crossover operator of choosing which chromosome's value to use, rather than holding it at the population level. There could also be choices between which operators to use and which selection mechanism to use, all held similarly at the chromosome level.

A further limitation is the use of un-directed mutation operators; this is discussed further in the recommendations for further work, below.

In the power optimisation work, there were limitations in the number of generations of the problems that could be run due to time constraints, licensing issues, and run interruptions, despite the resume capability. More lengthy runs, meaning more generations, would give GA/Plexos a better chance to converge towards global optima and possibly lead to more firm hypotheses regarding the analysis of the system.

Also in the power optimisation work, for the LC optimisation, further ideas were set out for runs of GA/Plexos, to investigate particular aspects of the grid behaviour for particular settings revealed by the optimisation runs. It was hoped to perform at least some of these within the scope of this work, but this proved to be not possible in the time available.

6.3 Recommendations for further work

6.3.1 Power optimisation

To run some of the optimisation problems for extended periods of time, to overcome limitations as given above. Using the resume capability of Ganesh, this could be achieved by extending runs already performed, rather than necessarily starting again from generation 0.

To undertake the further runs of GA/Plexos described at the end of section 5.5.3 to confirm (or deny) suspected impacts of DG and LC settings and to better understand how network design could be influenced positively by DG and LC and the node ranking schemes. The summary section also suggests the design of experiments to determine whether it is the number of lines connected to a node, or the node capacity (being the sum of connected lines' capacities) that is most important in ranking nodes, which drives the benefits of DG placement.

In scaling up to tackling networks of national dimensions, using the alternative OF discussed in the power summary section, 5.7, for better performance and applicability.

6.3.2 Directed mutation

In explaining the mechanisms, and debunking the misconceptions, of natural selection, Gregory (2009) notes that mutation of gene alleles is *undirected*, which is to say random, with respect to the advantage or disadvantage that the effect of the mutation may have upon the individual's likely reproductive outcome. For *in silico* systems that in some way use evolution induced by artificial rather than natural selection, it ought to be advantageous to make mutation *directed*, in other words to make mutation non-random, driven in such ways to produce beneficial effects, through choosing which genes to mutate and by how much and in which direction.

By keeping track of solution ancestry, which would also enable full life-cycle duplication control, it would be possible to calculate covariance of genes to establish correlation and anti-correlation between them, as shown in the parallel coordinates visualisation in Chapters 4 and 5 (for example in section 4.5.1.4), and to in turn relate the direction of gene change to the direction of the objective function value changes, and thereby arrive at a mutation function driven by a vector parameter of {gene, ±weight} change pairs, in which each gene in the chromosome has a co-existing weight parameter driving its possible mutations.

6.3.3 Hybridise with a genetic program to act as a surrogate model

Real-world optimisation problems often have the characteristic that their objective functions are computationally expensive to determine, leading to long-running execution times. In order to decrease run durations, which may mean that a non-feasible problem becomes feasible to tackle, the following hybrid algorithm is suggested, at least in the first instance as something to investigate.

The idea is to use a GP as a surrogate model, and use the surrogate as a replacement for some GA generations, occasionally refreshing the surrogate with new data from subsequent GA generations, alternating between GA generations & the surrogate. Each GA OF would be replaced by a polynomial approximation from the GP, arrived at through symbolic regression.

The objective of the GP is thus to evolve a solution whose polynomial function most nearly approximates the mapping of decision vectors to objective function values for every GA solution existing in a given GA generation. For MOOPs, the GP would need to evolve n functions, for n OFS of the GA, whose fitnesses are the minimum differences to the objective function values.

In this way, the GA/GP hybrid could proceed very much faster when the polynomial approximations are used instead of the GA OFs, where the GA results act as supervised learning inputs and the GP acts as a surrogate model produced by supervised learning (Gathercole, 1998).

6.4 Acknowledgements

This work was performed with the help of a grant from the EPSRC (EPSRC, 2014) and as such is required to state that the data and methods used are held on secure media within the Power and Propulsion Group of the School of Engineering at Cranfield University (contact Professor Mark Savill).

The author is grateful to the colleagues and collaborators of the optimisation of electrical power networks work of Chapter 5, Mr Blair Bethwaite and Dr Ariel Liebman (Monash University), Mr Leighton Brough, Dr Liam Wagner and Professor David Abramson (University of Queensland), for their provision of the Australian weather data and their interface to Plexos.

6.5 Concluding remarks

This work resulted in the production of a multi-objective framework and algorithm that has shown itself to be of true value in tackling real-world complex engineering systems. It can be built upon to improve it further, by the addition of new components either derived from existing classes, such as new self-adaptive crossover operators, and the addition of other self-adaptive mechanisms such as adaptive n-way tournament selection. It could have alternative dominance schemes or ranking schemes added, to help it tackle optimisation problems of four or more dimensions. It remains a practical and useful optimisation tool.

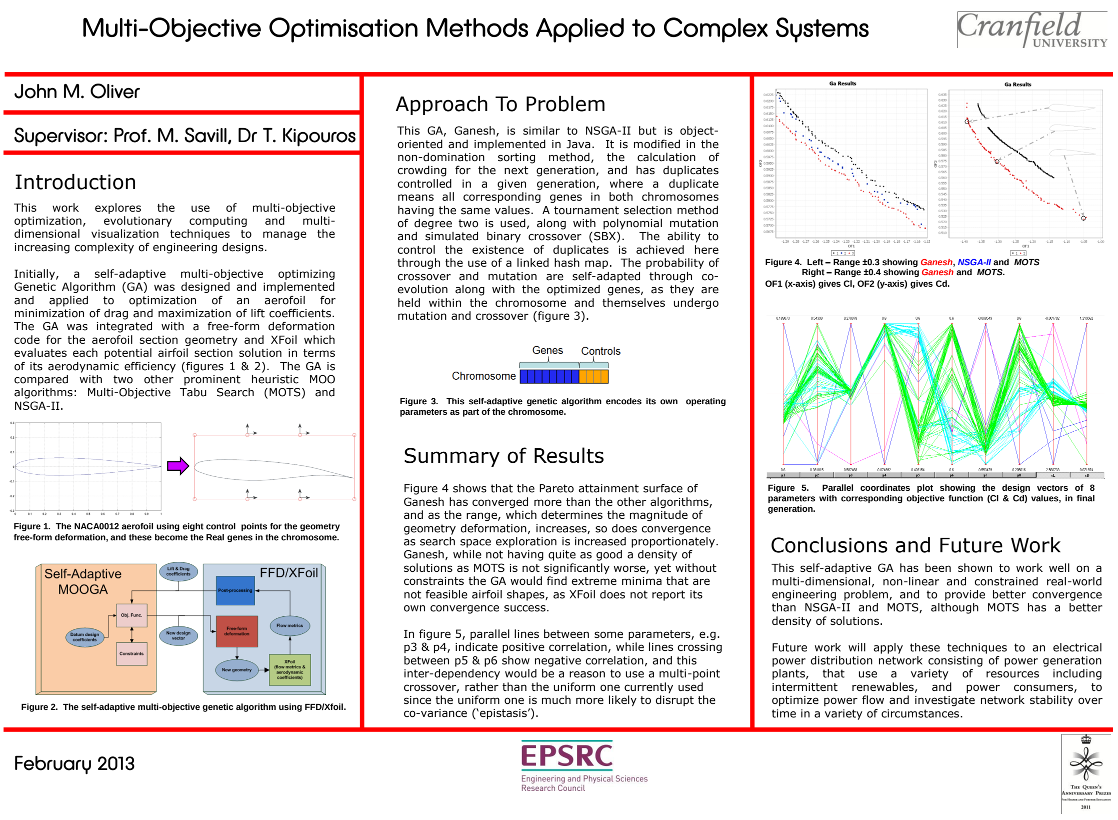
This research work showed that Genetic Algorithms could in fact be usefully applied to airfoil optimisation problems, in some circumstances at least. The XFoil tool was improved in its batch running capability and has been passed back to the Engineering Design Centre at Cambridge University.

The optimisation of electrical power networks by evolutionary algorithm, using both distributed generation and line capacity specification, has been shown to be feasible and useful.

Appendix A

Poster

A poster showing some earlier results prior to the modifications I made to XFoil.



Multi-Objective Optimisation Methods Applied to Complex Systems

Cranfield UNIVERSITY

John M. Oliver

Supervisor: Prof. M. Savill, Dr T. Kipouros

Introduction

This work explores the use of multi-objective optimization, evolutionary computing and multi-dimensional visualization techniques to manage the increasing complexity of engineering designs.

Initially, a self-adaptive multi-objective optimizing Genetic Algorithm (GA) was designed and implemented and applied to optimization of an aerofoil for minimization of drag and maximization of lift coefficients. The GA was integrated with a free-form deformation code for the aerofoil section geometry and XFoil which evaluates each potential airfoil section solution in terms of its aerodynamic efficiency (figures 1 & 2). The GA is compared with two other prominent heuristic MOO algorithms: Multi-Objective Tabu Search (MOTS) and NSGA-II.

Figure 1. The NACA0012 aerofoil using eight control points for the geometry free-form deformation, and these become the Real genes in the chromosome.

Figure 2. The self-adaptive multi-objective genetic algorithm using FFD/Xfoil.

Approach To Problem

This GA, Ganesh, is similar to NSGA-II but is object-oriented and implemented in Java. It is modified in the non-domination sorting method, the calculation of crowding for the next generation, and has duplicates controlled in a given generation, where a duplicate means all corresponding genes in both chromosomes having the same values. A tournament selection method of degree two is used, along with polynomial mutation and simulated binary crossover (SBX). The ability to control the existence of duplicates is achieved here through the use of a linked hash map. The probability of crossover and mutation are self-adapted through co-evolution along with the optimized genes, as they are held within the chromosome and themselves undergo mutation and crossover (figure 3).

Figure 3. This self-adaptive genetic algorithm encodes its own operating parameters as part of the chromosome.

Summary of Results

Figure 4 shows that the Pareto attainment surface of Ganesh has converged more than the other algorithms, and as the range, which determines the magnitude of geometry deformation, increases, so does convergence as search space exploration is increased proportionately. Ganesh, while not having quite as good a density of solutions as MOTS is not significantly worse, yet without constraints the GA would find extreme minima that are not feasible airfoil shapes, as XFoil does not report its own convergence success.

In figure 5, parallel lines between some parameters, e.g. p3 & p4, indicate positive correlation, while lines crossing between p5 & p6 show negative correlation, and this inter-dependency would be a reason to use a multi-point crossover, rather than the uniform one currently used since the uniform one is much more likely to disrupt the co-variance ('epistasis').

Figure 4. Left – Range ±0.3 showing *Ganesh*, *NSGA-II* and *MOTS*
Right – Range ±0.4 showing *Ganesh* and *MOTS*.
OF1 (x-axis) gives Cl, OF2 (y-axis) gives Cd.

Figure 5. Parallel coordinates plot showing the design vectors of 8 parameters with corresponding objective function (Cl & Cd) values, in final generation.

Conclusions and Future Work

This self-adaptive GA has been shown to work well on a multi-dimensional, non-linear and constrained real-world engineering problem, and to provide better convergence than NSGA-II and MOTS, although MOTS has a better density of solutions.

Future work will apply these techniques to an electrical power distribution network consisting of power generation plants, that use a variety of resources including intermittent renewables, and power consumers, to optimize power flow and investigate network stability over time in a variety of circumstances.

February 2013

EPSRC
Engineering and Physical Sciences Research Council

Figure A.1: Poster for airfoil optimisation

Appendix B

Software information

B.1 Ganesh

B.1.1 Ganesh Parameters

This section gives the run-time parameters of Ganesh that may be supplied to it as part of the execution command, and is obtained by entering the following command at the system prompt, which is given generically as “prompt> ”:

```
prompt> java -jar bin\GA.jar -?
Ganesh version: Rel: n.n.n, rev: x, branch: default, node: <node>
Options are as follows:
-? or ?: Show this help
-f  <s>: Output filename & path
-j  <s>: Class name of experiment plugin (*mandatory*)
-p  <n>: Initial Population Size
-g  <n>: Maximum number of Generations
-mp <n>: Mutation Probability
-cp <n>: Crossover Probability
-nd <n>: Number of duplicate Organisms allowed (Integer.MAX_VALUE)
```

```
-s  <n>: Seed for random number generator (int, must be > 0)
-nc <n>: Crossover polynomial distribution idx (default in plugin)
-nm <n>: Mutation  polynomial distribution idx (default in plugin)
-r1    : Keep going until all solutions are of Rank 1 (give -m)
-m  <n>: Maximum number of minutes to run
-e  <s>: plugin parameter filename & path
-lr <s>: Resume an old run using this log file: filename & path
-or <s>: Resume from old EvalFile: filename & path
-q     : Save EvalFile (partial evaluation logfile) to use with -or
-a2    : Operator order as nsga2: (crossover,mutation)
-b1    : Binary last generation (false)
-b2    : Binary last generation only - no text log (false)
```

The parameters above take precedence over those that may be defined in the plugin, given by the -j parameter.

B.2 Utility programs produced

This section provides details of the utilities that were also produced.

B.2.1 GADataCollect

This takes GA results, either singularly or collectively and processes them for various reasons, including:

1. Formatting GA results for input into the PISA tool-set.
2. Formatting GA results for input into the Parallax tool.
3. Obtaining aggregate information.

The run-time parameters are obtained by entering the following command at the system prompt, which is given generically as “prompt> ”:

```
prompt> java -jar bin\GADataCollect.jar -?
GADataCollect: Process Ganesh output - default to aggregate
GADataCollect Usage:
?               : Help
-d  <path>      : The input directory path
-f  <regex>     : A regex for the files to be read
-o  <filepath> : Path and filename of the output file (null)
-g  <gen>       : Get only generation <gen> (default all)
-r  <rn>        : Get rank <rn> only (default all)
-i1             : No aggregation - each read file to 1 output file
-p              : PISA, many in to 1 out, no aggregation
-v              : Get Params as well as Obj Funcs
-hv             : Calculate hypervolume result
-s              : Get statistics for aggregated data
-so             : Get statistics only (for aggregated data)
```

B.2.2 GAResultPlot

This plots objective functions as 2D scatter plots, either for the solutions of one generation or for a sub-set (including the entire set) of the generations in the run. Various options enable selection of the solutions to plot and also enable print quality resolutions to be chosen for the output images.

The run-time parameters are obtained by entering the following command at the system prompt, which is given generically as “prompt> ”:

```
prompt> java -jar bin\GAResultPlot.jar ?
Usage:
? or -? : Show these options
-f <s>  : Data file filename (& path)
-p <s>  : Filename prefix
-h <n>  : Output pic height (pixels, default 600)
```

```
-w <n>  : Output pic width  (pixels, default 600)
-gs <n> : Increase Tick label glyph size in font points
-vf     : Do not display chart
-v1     : view specified generations in the same plot
-va     : view all generations in the file
-d      : The output directory path
-r1     : Only use Rank 1 (non-dom)
-lx <s> : X axis label (default OF1)
-ly <s> : Y axis label (default OF2)
-gdl <n>: Graph domain (x) low  end (default auto or 0
-gdh <n>: Graph domain (x) high end (default auto or 1
-grl <n>: Graph range  (y) low  end (default auto or 0
-grh <n>: Graph range  (y) high end (default auto or 1
-s <n>  : size of plotted point shapes (pixels, default 4)
-t <s>  : Title for plot (default none)
-i <n>  : Image resolution in file in dpi (default 300)
-of1n   : OF1 negate (e.g. min to max)
-of2n   : OF2 negate (e.g. min to max)
-a      : Alternative delimited file
-n <n[,n...]>: Generation(s) to report on (default last)
```

B.2.3 GA-EvalFile

This takes as input the binary output file produced by Ganesh (when the relevant option was specified) and translates it into an output text file. The run-time parameters are obtained by entering the following command at the system prompt, which is given generically as “prompt> ”:

```
prompt> java -jar bin\GA-EvalFile.jar ?
```

```
GAEvalFile options are as follows:
```

```
-? or ?: Show this help
-i  <s>: Input EvalFile path and name
-o  <s>: Output text file path and name default null
-f  <s>: Format of EvalFile: C36C1BSILFD (Char,Byte,Short,Int,Long,
         Float,Double)
-f specifies the format of the contents of the EvalFile, defined:
  <data type of gene><number of instances> - repeated, e.g.
  Always C36C1 - for the ID and the Remove flag
  followed by the design vector, then the objective functions,
  then the optional adaptive controls.
  e.g. ExpPlexosC2 plugin needs: C36C1B72I41D4D4
```

B.2.4 PXmlTest

This acts both as a test of the Xml interface to Plexos, but is also the method of generating the Sql to create the data definition (tables and columns) and to populate the tables with data from the Xml model, when a new Plexos model arises. The run-time parameters are obtained by entering the following command at the system prompt, which is given generically as "prompt> ":

```
prompt> java -cp plugins\lib\pXmlTest.jar;bin\lib\UtilLib.jar
         plexosIF.PXmlTest ?
```

B.2.5 Rastrigin's function script

This section gives the R language script used to produce the plots of Rastrigin's function given in Figures 2.4 & 2.5.

```
# Initialise - clear objects and screen
rm(list=ls())
graphics.off()
```

```
rastrigin <- function(x,y) {
  # Rastrigin 2D function & plot
  res <- 10 * 2 + x ^ 2 - 10 * cos(2 * pi * x) +
                y ^ 2 - 10 * cos(2 * pi * y)
  return(res)
}
# vectors of values for x & Y
x <- seq(-5.12, 5.12, length=40)
y <- x
# create matrix Z as outer product
Z <- outer(x,y,rastrigin) #2D
# d=10 gives perspective amount
persp(x,y,Z,theta=30,phi=35, expand=0.6,
      col='lightgreen', shade=0.1, ltheta=120,
      ticktype="detailed")
contour(x,y,Z,xlab="x",ylab="y")
filled.contour(x,y,Z,xlab="x",ylab="y",nlevels=15)
```

Appendix C

Tables for power optimisation

This appendix contains supplementary information relating to the work of optimisation of power networks, that is set out in Chapter 5.

TABLE C.1: Line limits of Plexos model

Line	Node From	Node To	Property	Limits	Capacity	Units	Uid	Data_id
01. 1-2	1	2	Max Flow	LineLimits	130	MW	35148	2455
01. 1-2	1	2	Min Flow	LineLimits	-130	MW	35149	2458
01. 1-2	1	2	Max Flow	LowLimits	97.5	MW	1023105156	2454
01. 1-2	1	2	Min Flow	LowLimits	-97.5	MW	1023105154	2459
02. 1-3	1	3	Max Flow	LineLimits	130	MW	35152	2463
02. 1-3	1	3	Min Flow	LineLimits	-130	MW	35153	2466
02. 1-3	1	3	Max Flow	LowLimits	97.5	MW	1023105157	2462
02. 1-3	1	3	Min Flow	LowLimits	-97.5	MW	1023105161	2467
03. 2-4	2	4	Max Flow	LineLimits	65	MW	35156	2471
03. 2-4	2	4	Min Flow	LineLimits	-65	MW	35157	2474
03. 2-4	2	4	Max Flow	LowLimits	48.75	MW	1023105158	2470
03. 2-4	2	4	Min Flow	LowLimits	-48.75	MW	1023105159	2475
04. 3-4	3	4	Max Flow	LineLimits	130	MW	35160	2479
04. 3-4	3	4	Min Flow	LineLimits	-130	MW	35161	2482
04. 3-4	3	4	Max Flow	LowLimits	97.5	MW	1023105144	2478

Line	Node From	Node To	Property	Limits	Capacity	Units	Uid	Data_id
04. 3-4	3	4	Min Flow	LowLimits	-97.5	MW	1023105142	2483
05. 2-5	2	5	Max Flow	LineLimits	130	MW	35164	2487
05. 2-5	2	5	Min Flow	LineLimits	-130	MW	35165	2490
05. 2-5	2	5	Max Flow	LowLimits	97.5	MW	1023105145	2486
05. 2-5	2	5	Min Flow	LowLimits	-97.5	MW	1023105149	2491
06. 2-6	2	6	Max Flow	LineLimits	65	MW	35168	2495
06. 2-6	2	6	Min Flow	LineLimits	-65	MW	35169	2498
06. 2-6	2	6	Max Flow	LowLimits	48.75	MW	1023105146	2494
06. 2-6	2	6	Min Flow	LowLimits	-48.75	MW	1023105147	2499
07. 4-6	4	6	Max Flow	LineLimits	90	MW	35172	2503
07. 4-6	4	6	Min Flow	LineLimits	-90	MW	35173	2506
07. 4-6	4	6	Max Flow	LowLimits	67.5	MW	1023105168	2502
07. 4-6	4	6	Min Flow	LowLimits	-67.5	MW	1023105166	2507
08. 5-7	5	7	Max Flow	LineLimits	70	MW	35176	2511
08. 5-7	5	7	Min Flow	LineLimits	-70	MW	35177	2514
08. 5-7	5	7	Max Flow	LowLimits	52.5	MW	1023105172	2510
08. 5-7	5	7	Min Flow	LowLimits	-52.5	MW	1023105167	2515
09. 6-7	6	7	Max Flow	LineLimits	130	MW	35180	2519
09. 6-7	6	7	Min Flow	LineLimits	-130	MW	35181	2522
09. 6-7	6	7	Max Flow	LowLimits	97.5	MW	1023105106	2518
09. 6-7	6	7	Min Flow	LowLimits	-97.5	MW	1023105107	2523
10. 6-8	6	8	Max Flow	LineLimits	32	MW	35184	2527
10. 6-8	6	8	Min Flow	LineLimits	-32	MW	35185	2530
10. 6-8	6	8	Max Flow	LowLimits	24	MW	1023105104	2526
10. 6-8	6	8	Min Flow	LowLimits	-24	MW	1023105105	2531
11. 6-9	6	9	Max Flow	LineLimits	65	MW	35188	2535
11. 6-9	6	9	Min Flow	LineLimits	-65	MW	35189	2538
11. 6-9	6	9	Max Flow	LowLimits	48.75	MW	1023105114	2534
11. 6-9	6	9	Min Flow	LowLimits	-48.75	MW	1023105109	2539
12. 6-10	6	10	Max Flow	LineLimits	32	MW	35192	2543
12. 6-10	6	10	Min Flow	LineLimits	-32	MW	35193	2546

Line	Node From	Node To	Property	Limits	Capacity	Units	Uid	Data_id
12. 6-10	6	10	Max Flow	LowLimits	24	MW	1023105094	2542
12. 6-10	6	10	Min Flow	LowLimits	-24	MW	1023105095	2547
13. 9-11	9	11	Max Flow	LineLimits	65	MW	35196	2551
13. 9-11	9	11	Min Flow	LineLimits	-65	MW	35197	2554
13. 9-11	9	11	Max Flow	LowLimits	48.75	MW	1023105092	2550
13. 9-11	9	11	Min Flow	LowLimits	-48.75	MW	1023105093	2555
14. 9-10	9	10	Max Flow	LineLimits	65	MW	35200	2559
14. 9-10	9	10	Min Flow	LineLimits	-65	MW	35201	2562
14. 9-10	9	10	Max Flow	LowLimits	48.75	MW	1023105102	2558
14. 9-10	9	10	Min Flow	LowLimits	-48.75	MW	1023105097	2563
15. 4-12	4	12	Max Flow	LineLimits	65	MW	35204	2567
15. 4-12	4	12	Min Flow	LineLimits	-65	MW	35205	2570
15. 4-12	4	12	Max Flow	LowLimits	48.75	MW	1023105115	2566
15. 4-12	4	12	Min Flow	LowLimits	-48.75	MW	1023105131	2571
16. 12-13	12	13	Max Flow	LineLimits	65	MW	35208	2575
16. 12-13	12	13	Min Flow	LineLimits	-65	MW	35209	2578
16. 12-13	12	13	Max Flow	LowLimits	48.75	MW	1023105128	2574
16. 12-13	12	13	Min Flow	LowLimits	-48.75	MW	1023105129	2579
17. 12-14	12	14	Max Flow	LineLimits	32	MW	35212	2583
17. 12-14	12	14	Min Flow	LineLimits	-32	MW	35213	2586
17. 12-14	12	14	Max Flow	LowLimits	24	MW	1023105138	2582
17. 12-14	12	14	Min Flow	LowLimits	-24	MW	1023105139	2587
18. 12-15	12	15	Max Flow	LineLimits	32	MW	35216	2591
18. 12-15	12	15	Min Flow	LineLimits	-32	MW	35217	2594
18. 12-15	12	15	Max Flow	LowLimits	24	MW	1023105136	2590
18. 12-15	12	15	Min Flow	LowLimits	-24	MW	1023105119	2595
19. 12-16	12	16	Max Flow	LineLimits	32	MW	35220	2599
19. 12-16	12	16	Min Flow	LineLimits	-32	MW	35221	2602
19. 12-16	12	16	Max Flow	LowLimits	24	MW	1023105116	2598
19. 12-16	12	16	Min Flow	LowLimits	-24	MW	1023105117	2603
20. 14-15	14	15	Max Flow	LineLimits	16	MW	35224	2607

Line	Node From	Node To	Property	Limits	Capacity	Units	Uid	Data_id
20. 14-15	14	15	Min Flow	LineLimits	-16	MW	35225	2610
20. 14-15	14	15	Max Flow	LowLimits	12	MW	1023105126	2606
20. 14-15	14	15	Min Flow	LowLimits	-12	MW	1023105127	2611
21. 16-17	16	17	Max Flow	LineLimits	16	MW	35228	2615
21. 16-17	16	17	Min Flow	LineLimits	-16	MW	35229	2618
21. 16-17	16	17	Max Flow	LowLimits	12	MW	1023105124	2614
21. 16-17	16	17	Min Flow	LowLimits	-12	MW	1023105123	2619
22. 15-18	15	18	Max Flow	LineLimits	16	MW	35232	2623
22. 15-18	15	18	Min Flow	LineLimits	-16	MW	35233	2626
22. 15-18	15	18	Max Flow	LowLimits	12	MW	1023105122	2622
22. 15-18	15	18	Min Flow	LowLimits	-12	MW	1023105125	2627
23. 18-19	18	19	Max Flow	LineLimits	16	MW	35236	2631
23. 18-19	18	19	Min Flow	LineLimits	-16	MW	35237	2634
23. 18-19	18	19	Max Flow	LowLimits	12	MW	1023105118	2630
23. 18-19	18	19	Min Flow	LowLimits	-12	MW	1023105121	2635
24. 19-20	19	20	Max Flow	LineLimits	32	MW	35240	2639
24. 19-20	19	20	Min Flow	LineLimits	-32	MW	35241	2642
24. 19-20	19	20	Max Flow	LowLimits	24	MW	1023105120	2638
24. 19-20	19	20	Min Flow	LowLimits	-24	MW	1023105135	2643
25. 10-20	10	20	Max Flow	LineLimits	32	MW	35244	2647
25. 10-20	10	20	Min Flow	LineLimits	-32	MW	35245	2650
25. 10-20	10	20	Max Flow	LowLimits	24	MW	1023105134	2646
25. 10-20	10	20	Min Flow	LowLimits	-24	MW	1023105137	2651
26. 10-17	10	17	Max Flow	LineLimits	32	MW	35248	2655
26. 10-17	10	17	Min Flow	LineLimits	-32	MW	35249	2658
26. 10-17	10	17	Max Flow	LowLimits	24	MW	1023105130	2654
26. 10-17	10	17	Min Flow	LowLimits	-24	MW	1023105133	2659
27. 10-21	10	21	Max Flow	LineLimits	32	MW	35252	2663
27. 10-21	10	21	Min Flow	LineLimits	-32	MW	35253	2666
27. 10-21	10	21	Max Flow	LowLimits	24	MW	1023105132	2662
27. 10-21	10	21	Min Flow	LowLimits	-24	MW	1023105099	2667

Line	Node From	Node To	Property	Limits	Capacity	Units	Uid	Data_id
28. 10-22	10	22	Max Flow	LineLimits	32	MW	35256	2671
28. 10-22	10	22	Min Flow	LineLimits	-32	MW	35257	2674
28. 10-22	10	22	Max Flow	LowLimits	24	MW	1023105098	2670
28. 10-22	10	22	Min Flow	LowLimits	-24	MW	1023105101	2675
29. 21-22	21	22	Max Flow	LineLimits	32	MW	35260	2679
29. 21-22	21	22	Min Flow	LineLimits	-32	MW	35261	2682
29. 21-22	21	22	Max Flow	LowLimits	24	MW	1023105100	2678
29. 21-22	21	22	Min Flow	LowLimits	-24	MW	1023105091	2683
30. 15-23	15	23	Max Flow	LineLimits	16	MW	35264	2687
30. 15-23	15	23	Min Flow	LineLimits	-16	MW	35265	2690
30. 15-23	15	23	Max Flow	LowLimits	12	MW	1023105096	2686
30. 15-23	15	23	Min Flow	LowLimits	-12	MW	1023105111	2691
31. 22-24	22	24	Max Flow	LineLimits	16	MW	35268	2695
31. 22-24	22	24	Min Flow	LineLimits	-16	MW	35269	2698
31. 22-24	22	24	Max Flow	LowLimits	12	MW	1023105110	2694
31. 22-24	22	24	Min Flow	LowLimits	-12	MW	1023105113	2699
32. 23-24	23	24	Max Flow	LineLimits	16	MW	35272	2703
32. 23-24	23	24	Min Flow	LineLimits	-16	MW	35273	2706
32. 23-24	23	24	Max Flow	LowLimits	12	MW	1023105112	2702
32. 23-24	23	24	Min Flow	LowLimits	-12	MW	1023105103	2707
33. 24-25	24	25	Max Flow	LineLimits	16	MW	35276	2711
33. 24-25	24	25	Min Flow	LineLimits	-16	MW	35277	2714
33. 24-25	24	25	Max Flow	LowLimits	12	MW	1023105108	2710
33. 24-25	24	25	Min Flow	LowLimits	-12	MW	1023105170	2715
34. 25-26	25	26	Max Flow	LineLimits	16	MW	35280	2719
34. 25-26	25	26	Min Flow	LineLimits	-16	MW	35281	2722
34. 25-26	25	26	Max Flow	LowLimits	12	MW	1023105164	2718
34. 25-26	25	26	Min Flow	LowLimits	-12	MW	1023105169	2723
35. 25-27	25	27	Max Flow	LineLimits	16	MW	35284	2727
35. 25-27	25	27	Min Flow	LineLimits	-16	MW	35285	2730
35. 25-27	25	27	Max Flow	LowLimits	12	MW	1023105165	2726

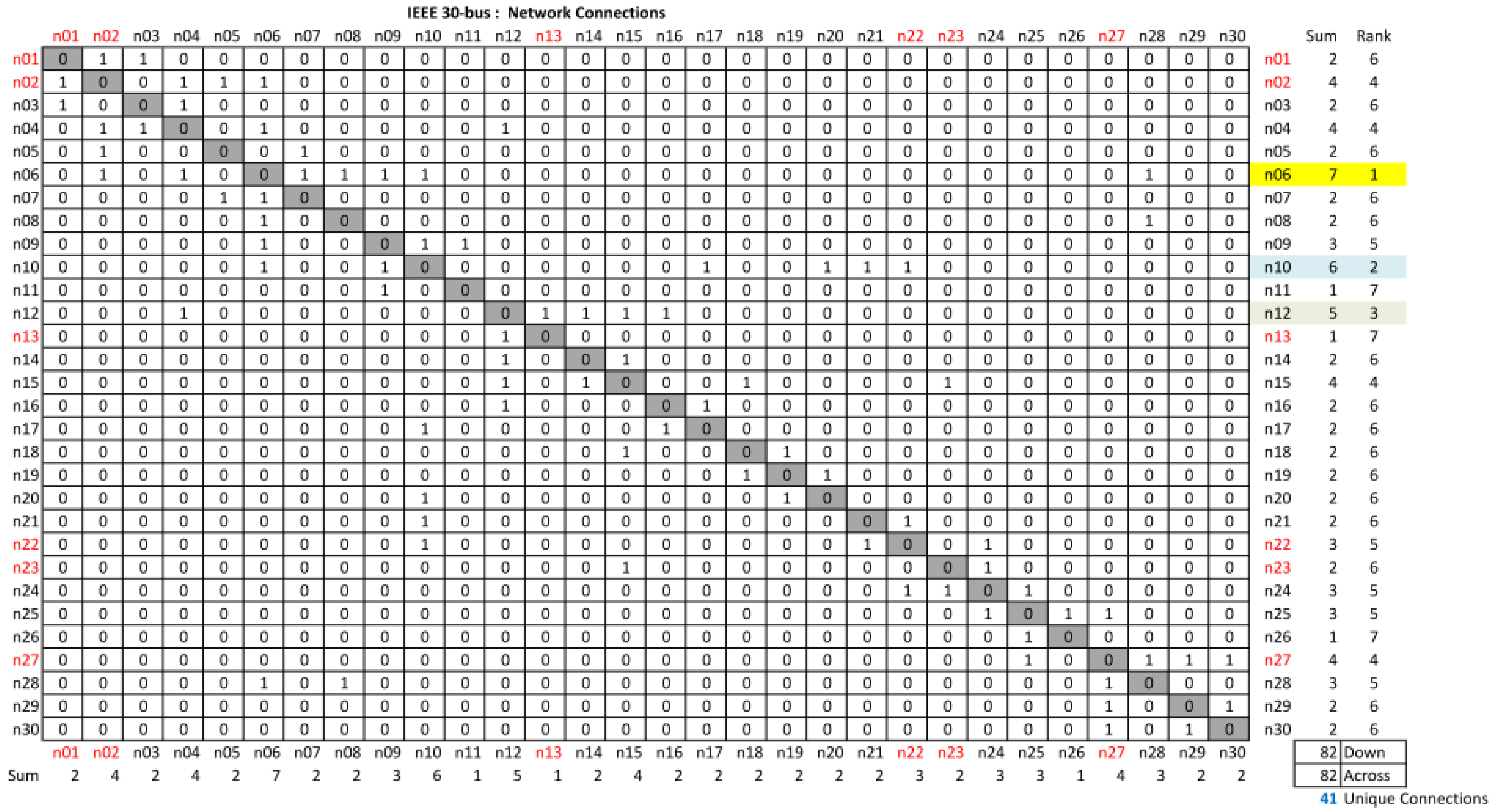
Line	Node From	Node To	Property	Limits	Capacity	Units	Uid	Data_id
35. 25-27	25	27	Min Flow	LowLimits	-12	MW	1023105163	2731
36. 28-27	28	27	Max Flow	LineLimits	65	MW	35288	2735
36. 28-27	28	27	Min Flow	LineLimits	-65	MW	35289	2738
36. 28-27	28	27	Max Flow	LowLimits	48.75	MW	1023105171	2734
36. 28-27	28	27	Min Flow	LowLimits	-48.75	MW	1023105148	2739
37. 27-29	27	29	Max Flow	LineLimits	16	MW	35292	2743
37. 27-29	27	29	Min Flow	LineLimits	-16	MW	35293	2746
37. 27-29	27	29	Max Flow	LowLimits	12	MW	1023105150	2742
37. 27-29	27	29	Min Flow	LowLimits	-12	MW	1023105141	2747
38. 27-30	27	30	Max Flow	LineLimits	16	MW	35296	2751
38. 27-30	27	30	Min Flow	LineLimits	-16	MW	35297	2754
38. 27-30	27	30	Max Flow	LowLimits	12	MW	1023105140	2750
38. 27-30	27	30	Min Flow	LowLimits	-12	MW	1023105143	2755
39. 29-30	29	30	Max Flow	LineLimits	16	MW	35300	2759
39. 29-30	29	30	Min Flow	LineLimits	-16	MW	35301	2762
39. 29-30	29	30	Max Flow	LowLimits	12	MW	1023105151	2758
39. 29-30	29	30	Min Flow	LowLimits	-12	MW	1023105160	2763
40. 8-28	8	28	Max Flow	LineLimits	32	MW	35304	2767
40. 8-28	8	28	Min Flow	LineLimits	-32	MW	35305	2770
40. 8-28	8	28	Max Flow	LowLimits	24	MW	1023105162	2766
40. 8-28	8	28	Min Flow	LowLimits	-24	MW	1023105153	2771
41. 6-28	6	28	Max Flow	LineLimits	32	MW	35308	2775
41. 6-28	6	28	Min Flow	LineLimits	-32	MW	35309	2778
41. 6-28	6	28	Max Flow	LowLimits	24	MW	1023105152	2774
41. 6-28	6	28	Min Flow	LowLimits	-24	MW	1023105155	2779

TABLE C.2: The nodes and their total capacities determined by summing their connected line capacities, in descending capacity order. The line capacity is counted at each node it is attached to, and the LowLimits have been rounded to the nearest integer. G,W & S are the DG unit types, heading their variables.

Node	LowLimits Capacity	LineLimits Capacity	G	W	S
06	336	446	V10	V11	V12
02	294	390	OCGT		
04	264	350	V04	V05	V06
01	196	260	OCGT		
03	196	260	V01	V02	V03
12	170	226	V28	V29	V30
10	169	225	V22	V23	V24
05	151	200	V07	V08	V09
07	151	200	V13	V14	V15
09	147	195	V19	V20	V21
28	97	129	V64	V65	V66
27	85	113	OCGT		
15	60	80	V34	V35	V36
22	60	80	OCGT		
11	49	65	V25	V26	V27
13	49	65	OCGT		
08	48	64	V16	V17	V18
20	48	64	V49	V50	V51
21	48	64	V52	V53	V54
14	36	48	V31	V32	V33
16	36	48	V37	V38	V39
17	36	48	V40	V41	V42
19	36	48	V46	V47	V48
24	36	48	V55	V56	V57
25	36	48	V58	V59	V60
18	24	32	V43	V44	V45
23	24	32	OCGT		
29	24	32	V67	V68	V69
30	24	32	V70	V71	V72
26	12	16	V61	V62	V63

Table C.3: Node connections, ranked by number of connections per node, in descending rank order, with DG variables.

Node	Lines	Rank	G	W	S
06	7	1	V10	V11	V12
10	6	2	V22	V23	V24
12	5	3	V28	V29	V30
02	4	4	OCGT		
04	4	4	V4	V5	V6
15	4	4	V34	V35	V36
27	4	4	OCGT		
09	3	5	V19	V20	V21
22	3	5	OCGT		
24	3	5	V55	V56	V57
25	3	5	V58	V59	V60
28	3	5	V64	V65	V66
01	2	6	OCGT		
03	2	6	V1	V2	V3
05	2	6	V7	V8	V9
07	2	6	V13	V14	V15
08	2	6	V16	V17	V18
14	2	6	V31	V32	V33
16	2	6	V37	V38	V39
17	2	6	V40	V41	V42
18	2	6	V43	V44	V45
19	2	6	V46	V47	V48
20	2	6	V49	V50	V51
21	2	6	V52	V53	V54
23	2	6	OCGT		
29	2	6	V67	V68	V69
30	2	6	V70	V71	V72
11	1	7	V25	V26	V27
13	1	7	OCGT		
26	1	7	V61	V62	V63



IEEE 30-bus : Network Connections

	n01	n02	n03	n04	n05	n06	n07	n08	n09	n10	n11	n12	n13	n14	n15	n16	n17	n18	n19	n20	n21	n22	n23	n24	n25	n26	n27	n28	n29	n30		Sum	Rank
n01	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n01	2	6
n02	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n02	4	4
n03	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n03	2	6
n04	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n04	4	4
n05	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n05	2	6
n06	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	n06	7	1
n07	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n07	2	6
n08	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	n08	2	6
n09	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n09	3	5
n10	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	n10	6	2
n11	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n11	1	7
n12	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n12	5	3
n13	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n13	1	7
n14	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n14	2	6
n15	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	n15	4	4
n16	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	n16	2	6
n17	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n17	2	6
n18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	n18	2	6
n19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	n19	2	6
n20	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	n20	2	6
n21	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	n21	2	6
n22	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	n22	3	5
n23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	n23	2	6
n24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	n24	3	5
n25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	n25	3	5
n26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	n26	1	7
n27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	n27	4	4
n28	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	n28	3	5
n29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	n29	2	6
n30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	n30	2	6
	n01	n02	n03	n04	n05	n06	n07	n08	n09	n10	n11	n12	n13	n14	n15	n16	n17	n18	n19	n20	n21	n22	n23	n24	n25	n26	n27	n28	n29	n30		82	Down
Sum	2	4	2	4	2	7	2	2	3	6	1	5	1	2	4	2	2	2	2	2	2	3	2	3	3	1	4	3	2	2		82	Across

41 Unique Connections

FIGURE C.1: Node connections, ranked by number of connections per node, in node order.

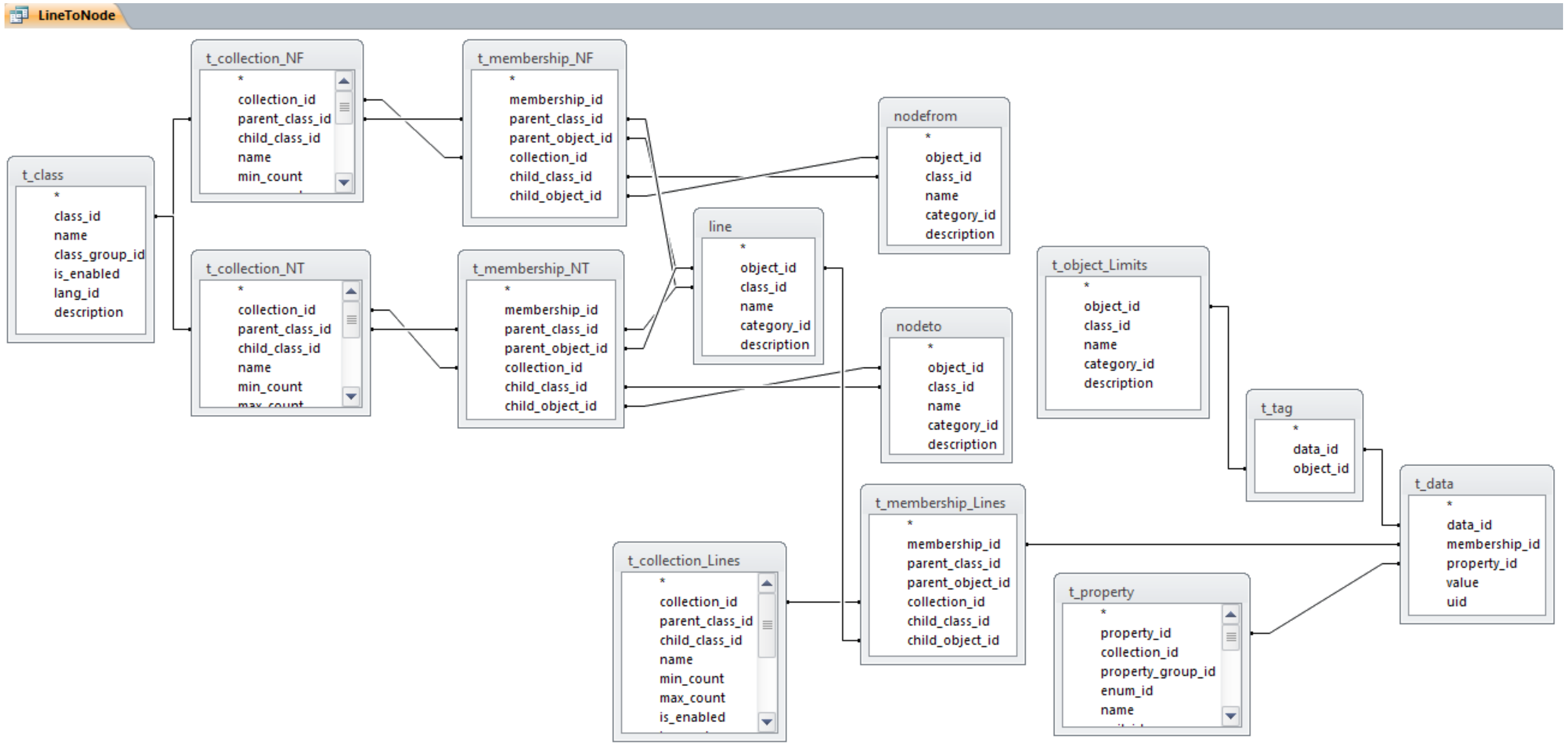


FIGURE C.2: Data model for extracting Plexos line information, showing tables derived from the Plexos XML data model (version 6.205).

Table C.4: OCGT fixed central generator characteristics.

Units	#	MW	MW	GJ/MWh	$/MWh	$	hrs	hrs	#	MW/min.	MW/min.	%	$/MWh	%
Node/Gen	Units	Max Capacity	Min Stable Level	Heat Rate	VO&M Charge	Start Cost	Min Up Time	Min Down Time	Commit	Max Ramp Up	Max Ramp Down	Aux Incr	Mark-up	Bid-Cost Mark-up
Gen 01	1	80	10	9.16	1.5	500	2		1	1.5	25	8.1	2	20
Gen 02	1	80	16	9.2	1.5	500	2		1	1.5	25	8.1	2	20
Gen 13	1	50	5	10.59	1.5	1000	3	1	1	1.5	25	8.5	2	20
Gen 22	1	30	3	9.7	1.5	1000	3	1	1	1.5	1.5	4.8	2	20
Gen 23	1	30	3	10.32	1.75	500	0.5	0.5	1	10	10	0.5	0	
Gen 27	1	55	5	10.32	1.75	400	0.5		1	15	15	0.5	0	

Table C.5: Distributed Generation (DG) generator type characteristics

	MW	GJ/MWh	$/MWh	MW	#	#	MW	$/MWh	%	%	%	#
DG type	Max Capacity	Heat Rate	VO&M Charge	Rating	Must-Run Units	Commit	Offer Quantity	Offer Price	Max Capacity Factor Day	Min Capacity Factor Day	Aux Incr	Power to Heat Ratio
G02 gas_rec_b_cg	1	8.75	7.5	1	1	1			65	65	1	1.1
G09 Wind	1		0.5	0	1		1	0				
G10 Solar	1		0.5	0			1	0				

References

Abott, I. and von Doenhoff, A., 1959. *Theory of wing sections: including a summary of airfoil data.* Dover, New York.

Aerospaceweb, 2012. Angle of attack and pitch angle. URL http://www.aerospaceweb.org/question/aerodynamics/q0165.shtml.

Aguilar-Hidalgo, D., Zurita, A.C., and Lemos, M.F.C., 2012. Complex networks evolutionary dynamics using genetic algorithms. *International Journal of Bifurcation and Chaos*, 22(07):1250156. URL http://dx.doi.org/10.1142/S0218127412501568. Doi: 10.1142/S0218127412501568; M3: doi: 10.1142/S0218127412501568; 09.

Allen, P.M., Strathern, M., and Varga, L., 2010. Complexity: The evolution of identity and diversity. In P. Cilliers and R. Preiser, editors, *Complexity, Difference and Identity*, volume 26 of *Issues in Business Ethics*, pages 41–60. Springer Netherlands. ISBN 978-90-481-9186-4. doi:10.1007/978-90-481-9187-1_3. URL http://dx.doi.org/10.1007/978-90-481-9187-1_3.

Amin, M., 2003. North America's electricity infrastructure: are we ready for more perfect storms? *Security & Privacy, IEEE*, 1(5):19–25. ID: 1.

Atallah, M. and Blanton, M., 2009. *Algorithms and Theory of Computation Handbook, Second Edition: General Concepts and Techniques.* CRC Press, Boca Raton, FL. ISBN 9781584888222.

Avidan, T. and Avidan, S., 1999. Parallax - a data mining tool based on parallel coordinates. *Computational Statistics*, 14(1):79–89. URL http://dx.doi.org/10.1007/PL00022707.

Bäck, T., 1992. Self-adaptation in genetic algorithms. In *Proceedings of the First European Conference on Artificial Life*, pages 263–271. MIT Press.

Bäck, T., 1995. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, USA. ISBN 9780195356700. URL `http://books.google.co.uk/books?id=EaN7kvl5coYC`.

Bäck, T., Eiben, A., and van der Vaart, N., 2000. An empirical study on GAs "without parameters". In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature PPSN VI*, volume 1917, pages 315–324. Springer Berlin / Heidelberg.

Bäck, T., Hammel, U., and Schwefel, H.P., 1997. Evolutionary computation: comments on the history and current state. *Evolutionary Computation, IEEE Transactions on*, 1(1):3–17. ID: 1.

Bäck, T. and Schwefel, H.P., 1993. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23. URL `http://dx.doi.org/10.1162/evco.1993.1.1.1`.

Bertsekas, D.P., 1999. *Nonlinear Programming.* Athena Scientific, Belmont, MA.

Beyer, H.G. and Schwefel, H.P., 2002. Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52. ISSN 1572-9796. doi:10.1023/A:1015059928466. URL `http://dx.doi.org/10.1023/A:1015059928466`.

Bleuler, S., Laumanns, M., Thiele, L., and Zitzler, E., 2003. Pisa - a platform and programming language independent interface for search algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, pages 494–508. Springer.

Bloebaum, C.L. and McGowan, A.M.R., 2010. Design of complex engineered systems. *Journal of Mechanical Design*, 132(12):120301–120301.

Blum, C. and Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput.Surv.*, 35(3):268–308. URL `http://doi.acm.org/10.1145/937503.937505`.

Blume, S.W., 2007. *Electric Power System Basics For the Nonelectrical Professional.* Wiley-IEEE Press, New Jersey.

Cain, M.B., O'Neill, R.P., and Castillo, A., 2013. History of optimal power flow and formulations.

Chechkin, A.V., Metzler, R., Gonchar, V.Y., and Klafter, J., 2008. Introduction to the theory of lvy flights. *Anomalous Transport: Foundations and Applications.*

Christie, R., 1993. Power systems test case archive: 30 bus power flow test case.

Coello, C.A.C., 2006. Evolutionary multi-objective optimization: a historical view of the field. *Computational Intelligence Magazine, IEEE*, 1(1):28–36. ID: 1.

Coello, C.A.C., Lamont, G.B., and Veldhuizen, D.A.V., 2006. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation).* Springer-Verlag New York, Inc, Secaucus, NJ, USA. ISBN 978-0-387-36797-2.

Cyber Dyne s.r.l, 2016. Kimeme. URL `http://www.cyberdyne.it/`.

De Jong, K. and Spears, W., 1992. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1):1–26. ISSN 1012-2443. doi:10.1007/BF01530777.

De Jong, K.A., 1975. *An analysis of the behavior of a class of genetic adaptive systems.* Ph.D. thesis, University of Michigan. AAI7609381.

Deb, K., 1999. Evolutionary algorithms for multi-criterion optimization in engineering design. In K. Miettinen, P. Neittaanmäki, M.M. Mäkelä, and J. Périaux, editors, *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*, pages 135–161. Wiley, New York.

Deb, K., 2001. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley, Chichester. ID: 2.

Deb, K., 2012. *Optimization for Engineering Design: Algorithms and Examples*. PHI Learning. ISBN 9788120346789. URL `http://books.google.co.uk/books?id=cN_kjtySMhIC`.

Deb, K., 2013. Tutorial slides: Multi-objective evolutionary algorithms.

Deb, K. and Agrawal, R.B., 1995. Simulated binary crossover for continuous search space. *Complex Systems*, pages 115–148.

Deb, K. and Agrawal, S., 1999. Understanding interactions among genetic algorithm parameters. In *in Foundations of Genetic Algorithms 5*, pages 265–286. Morgan Kaufmann.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T., 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature PPSN VI*, volume 1917, pages 849–858. Springer Berlin / Heidelberg.

Deb, K. and Goyal, M., 1996. A combined genetic adaptive search (geneas) for engineering design. *Computer Science and Informatics*, 26:30–45.

Deb, K., Horn, J., and E., D.G., 1993. Multimodal deceptive functions. *Complex Systems*, 7(2):131–153.

Deb, K. and Jain, S., 2002. Running performance metrics for evolutionary multi-objective optimization. In *4th Asia-Pacific Conference on Simulated Evolution and Learning(SEAL '02)*, pages 13–20. Nanyang Technical University, Singapore.

Deb, K., Mohan, M., and Mishra, S., 2003. A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. Research Report 2003002, Indian Institute Of Technology Kanpur.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E., 2001. Scalable test problems for evolutionary multi-objective optimization. Technical report, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH.

Dharamjit, D., 2012. Load flow analysis on ieee 30 bus system. *International Journal of Scientific and Research Publications*, 2(11):n/a. ISSN 2250-3153. URL http://www.ijsrp.org/research-paper-1112/ijsrp-p1153.pdf.

Dorigo, M. and Blum, C., 2005. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278.

Dorigo, M., Maniezzo, V., and Colorni, A., 1996. The Ant system: Optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41. ISSN 1083-4419. doi:10.1109/3477.484436.

Drela, M., 1989. Xfoil - an analysis and design system for low reynolds number airfoils. In *Low Reynolds Number Aerodynamics Conference*, pages 1–12. Notre Dame, Germany.

Durillo, J.J. and Nebro, A.J., 2015. jmetal. URL http://jmetal.sourceforge.net/index.html.

Dynardo GmbH, 2016. optiSLang. URL https://www.dynardo.de/en/software/optislang.html.

Eiben, A., Schut, M., and de Wilde, A., 2006a. Is self-adaptation of selection pressure and population size possible? a case study. In T. Runarsson, H.G. Beyer, E. Burke, J. Merelo-Guervs, L. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193, pages 900–909. Springer-Verlag, Berlin, Heidelberg.

Eiben, A.E., Schut, M.C., and Wilde, A.R.D., 2006b. Boosting genetic algorithms with self-adaptive selection. In *In Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1584–1589.

Energy Exemplar Pty Ltd, 2013. Plexos for power systems. URL http://www.energyexemplar.com.

EPSRC, 2014. Engineering and physical sciences research council. URL http://www.epsrc.ac.uk.

ESTECO SpA, 2016. modeFRONTIER. URL http://www.esteco.com/modefrontier.

Evolving Objects Team - various, 2015. Evolving objects. URL http://eodev.sourceforge.net/.

Fleming, P. and Purshouse, R., 2002. Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, 10(11):1223 – 1241. ISSN 0967-0661. doi:http://dx.doi.org/10.1016/S0967-0661(02)00081-3. URL http://www.sciencedirect.com/science/article/pii/S0967066102000813.

Fleming, P.J. and Purshouse, R.C., 2001. Genetic Algorithms In Control Systems Engineering. Research Report 789, Sheffield University.

Fleming, P.J., Purshouse, R.C., and Lygoe, R.J., 2005. Many-objective optimization: An engineering design perspective. In *EMO'05, LNCS 3410*, pages 14–32. Springer-Verlag, Berlin Heidelberg.

Fogel, D., 1994. Applying evolutionary programming to selected control problems. *Computers & Mathematics with Applications*, 27(11):89 – 104. ISSN 0898-1221. URL http://www.sciencedirect.com/science/article/pii/0898122194901007.

Fonseca, C. and Fleming, P., 1997. Multiobjective genetic algorithms. In A. Zalzala and P. Fleming, editors, *Genetic Algorithms in Engineering Systems*, pages 63–78. Institution of Engineering and Technology. URL http://digital-library.theiet.org/content/books/ce/pbce055e.

Fonseca, C., Knowles, J., Thiele, L., and Zitzler, E., 2005. A tutorial on the performance assessment of stochastic multiobjective optimizers, an invited talk presented by j. knowles at emo 2005, guanajuato, mexico, 2005.

Fonseca, C.M. and Fleming, P.J., 1993. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest, editor, *ICGA*, pages 416–423. Morgan Kaufmann. ISBN 1-55860-299-2. URL http://dblp.uni-trier.de/db/conf/icga/icga1993.html#FonsecaF93. Conf/icga/1993; 2002-09-04.

Fonseca, C.M. and Fleming, P.J., 1995a. Multiobjective genetic algorithms made easy: selection sharing and mating restriction. In *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*, pages 45–52.

Fonseca, C.M. and Fleming, P.J., 1995b. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1):1–16. URL http://dx.doi.org/10.1162/evco.1995.3.1.1. Doi: 10.1162/evco.1995.3.1.1; M3: doi: 10.1162/evco.1995.3.1.1; 04.

Fonseca, C.M. and Fleming, P.J., 1996. On the performance assessment and comparison of stochastic multiobjective optimizers. In H.M. Voigt, W. Ebeling, I. Rechenberg, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature — PPSN IV: International Conference on Evolutionary Computation — The 4th International Conference on Parallel Problem Solving from Nature Berlin, Germany, September 22–26, 1996 Proceedings*, pages 584–593. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-70668-7. doi:10.1007/3-540-61723-X_1022. URL http://dx.doi.org/10.1007/3-540-61723-X_1022.

Fonseca, C.M. and Fleming, P.J., 1998. Multiobjective optimization and multiple constraint handling with evolutionary algorithms - part i : A unified formulation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(1):26.

Forrester, A., Sobester, A., and Keane, A., 2008. *Engineering design via surrogate modelling: a practical guide.* Wiley. URL http://eprints.soton.ac.uk/64699/.

Free Software Foundation, Inc, 2016. What is free software? URL https://www.fsf.org/about/what-is-free-software/.

Fukunaga, A.S., 1998. Restart scheduling for genetic algorithms. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V: 5th International Conference Amsterdam, The Netherlands September 27–30, 1998 Proceedings*, pages 357–366. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-49672-4. doi:10.1007/BFb0056878. URL http://dx.doi.org/10.1007/BFb0056878.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J., 1995. *Design Patterns: Elements of Reusable Object-oriented Software.* Addison-Wesley Longman, Boston, MA, USA. ISBN 0-201-63361-2.

Gardner, B. and Selig, M., 2003. Airfoil Design Using a Genetic Algorithm and an Inverse Method. In *41st Aerospace Sciences Meeting and Exhibit*, Aerospace Sciences Meetings. American Institute of Aeronautics and Astronautics. doi: doi:10.2514/6.2003-43. URL http://dx.doi.org/10.2514/6.2003-43.

Garey, M.R. and Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H.Freeman, New York.

Gathercole, C., 1998. *An Investigation of Supervised Learning in Genetic Programming.* Ph.D. thesis, University of Edinburgh.

Gautam, D. and Mithulananthan, N., 2007. Optimal DG placement in deregulated electricity market. *Electric Power Systems Research*, 77(12):1627–1636.

Geeknet inc., 2014. gnuplot. URL http://www.gnuplot.info/.

Gen, M. and Cheng, R., 2000. *Genetic Algorithms & Engineering Optimization.* Wiley, New York.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549. ISSN 0305-0548. doi:http://dx.doi.org/10.1016/0305-0548(86)90048-1. URL http://www.sciencedirect.com/science/article/pii/0305054886900481.

Glover, J.D., Sarma, M.S., and Overbye, T.J., 2012. *Power System Analysis and Design*. Cengage Learning, Stamford, 5th edition.

Goldberg, D., 1991. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.*, 23(1):5–48. ISSN 0360-0300. doi:10.1145/103162.103163. URL http://doi.acm.org/10.1145/103162.103163.

Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, MA.

Goldberg, D.E. and Deb, K., 1991. A comparative analysis of selection schemes used in genetic algorithms. In G.J. E.Rawlins, editor, *Foundations of Genetic Algorithms*, volume 1, pages 69–93. Morgan Kaufmann.

Goldberg, D.E., Korb, B., and Deb, K., 1989. Messy genetic algorithms motivation, analysis, and first results. *Complex Systems*, 3:493–530.

Grand, M., 1998. *Patterns in Java Volume 1*. Wiley, New York.

Grand, M., 1999. *Patterns in Java Volume 2*. Wiley, New York.

Grefenstette, J. and Baker, J.E., 1989. How genetic algorithms work: A critical look at implicit parallelism. In J.D. Schaffer, editor, *Proc. of the Third Int. Conf. on Genetic Algorithms*, pages 20–27. Morgan Kaufmann, San Mateo, CA.

Grefenstette, J.J., 1986. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):122–128.

Gregory, T.R., 2009. Understanding natural selection: Essential concepts and common misconceptions. *Evolution: Education and Outreach*, 2(2):156–175. URL http://dx.doi.org/10.1007/s12052-009-0128-1.

Hadka, D., 2016. Moea framework. URL http://moeaframework.org/.

Hansen, M.P. and Jaszkiewicz, A., 1998. *Evaluating the quality of approximations to the non-dominated set*. IMM, Department of Mathematical Modelling, Technical Universityof Denmark.

Haupt, R. and Haupt, S.E., 2004. *Practical Genetic Algorithms*. Wiley, New Jersey, 2nd edition. ISBN 9780471671749.

Ho, C.W., Lee, K.H., and Leung, K.S., 1999. A genetic algorithm based on mutation and crossover with adaptive probabilities. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, page 775 Vol. 1.

Holland, J.H., 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Massachusetts, 2nd edition.

Hollander, M., Wolfe, D.A., and Chicken, E., 2014. *Nonparametric Statistical Methods, 3rd Ed*. Wiley, Hoboken, New Jersey, 3 edition. ISBN 978-0-470-38737-5.

Hughes, E.J., 2005. Evolutionary many-objective optimisation: many once or one many? In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 222–227 Vol.1. ISSN 1089-778X. doi:10.1109/CEC.2005.1554688.

Inselberg, A., 2009. *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*. Springer. ISBN 9780387215075. URL `http://books.google.com/books?id=Aj-xzznPcqgC`.

Ishibuchi, H., Tsukamoto, N., and Nojima, Y., 2008. Evolutionary many-objective optimization: A short review. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2419–2426.

Jacob, Christoph, R., 2016. How open is commercial scientific software? *The Journal of Physical Chemistry Letters*, 7(2):351–353. doi:10.1021/acs.jpclett.5b02609. URL `http://dx.doi.org/10.1021/acs.jpclett.5b02609`.

Jaeggi, D.M., Parks, G.T., Kipouros, T., and Clarkson, P.J., 2008. The development of a multi-objective tabu search algorithm for continuous optimisation problems. *European Journal of Operational Research*, 185(3):1192–1212. ID: 3.

Jones, D.F., Mirrazavi, S.K., and Tamiz, M., 2002. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137(1):1–9.

Kennedy, J. and Eberhart, R., 1995. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4.

Khare, V., Yao, X., and Deb, K., 2003. Performance scaling of multi-objective evolutionary algorithms. In C.M. Fonseca, P.J. Fleming, E. Zitzler, L. Thiele, and K. Deb, editors, *Evolutionary Multi-Criterion Optimization: Second International Conference, EMO 2003, Faro, Portugal, April 8–11, 2003. Proceedings*, pages 376–390. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-36970-7. doi:10.1007/3-540-36970-8_27. URL http://dx.doi.org/10.1007/3-540-36970-8_27.

Kipouros, T., Inselberg, A., Parks, G., and Savill, A.M., 2013. Parallel coordinates in computational engineering design - AIAA 2013-1750. In *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Structures, Structural Dynamics, and Materials and Co-located Conferences. American Institute of Aeronautics and Astronautics, Boston, Massachusetts. URL http://dx.doi.org/10.2514/6.2013-1750. 11; M1: 0; doi:10.2514/6.2013-1750; M3: doi:10.2514/6.2013-1750.

Kipouros, T., Mleczko, M., and Savill, M., 2008. Use of parallel coordinates for post-analyses of multi-objective aerodynamic design optimisation in turbomachinery. AIAA-2008-2138. In *4th AIAA Multi-Disciplinary Design Optimization Specialist Conference*, Structures, Structural Dynamics, and Materials and Co-located Conferences. American Institute of Aeronautics and Astronautics, Schaumburg, Illinois. URL http://dx.doi.org/10.2514/6.2008-2138. 29; M1: 0; doi:10.2514/6.2008-2138; M3: doi:10.2514/6.2008-2138.

Kipouros, T., Peachey, T., Abramson, D., and Savill, M., 2012. Enhancing and developing the practical optimisation capabilities and intelligence of automatic design software AIAA 2012-1677. In *8th AIAA Multidisciplinary Design Optimization Specialist Conference (MDO)*. American Institute of Aeronautics and Astronautics.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., 1983. Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680.

Knowles, J. and Corne, D., 2007. Quantifying the effects of objective space dimension in evolutionary multiobjective optimization. In *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*, EMO'07, pages 757–771. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-70927-5. URL http://dl.acm.org/citation.cfm?id=1762545.1762609.

Knowles, J.D. and Corne, D.W., 2000. Approximating the nondominated front using the pareto archived evolution strategy. *Evol. Comput.*, 8(2):149–172. ISSN 1063-6560. doi:10.1162/106365600568167.

Koza, J.R., 1998. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, Massachusetts, 6th edition.

Kukkonen, S. and Deb, K., 2006. Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1179–1186. ID: 1.

LaTeX project team, 2014. LaTeX - a document preparation system. URL http://latex-project.org/.

Lattarulo, V., Kipouros, T., and Parks, G., 2013. Application of the multi-objective alliance algorithm to a benchmark aerodynamic optimization problem. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 3182–3189. doi:10.1109/CEC.2013.6557959.

Laumanns, M., Rudolph, G., and Schwefel, H., 2001. Approximating the Pareto Set: Concepts, Diversity Issues, and Performance Assessment. Research Report CI-72/99, University of Dortmund, Collaborative Research Center 531.

Lednicer, D., 2010. The incomplete guide to airfoil usage. URL http://aerospace.illinois.edu/m-selig/ads/aircraft.html.

Li, M., Cai, Z., and Sun, G., 2004. An adaptive genetic algorithm with diversity-guided mutation and its global convergence property. *Journal of Central South University of Technology*, 11(3):323–327. URL http://dx.doi.org/10.1007/s11771-004-0066-6.

Lukasiewycz, M., Glass, M., Reimann, F., and Helwig, S., 2016. Opt4J. URL http://opt4j.sourceforge.net/.

Martin, R.C., 2002. *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall PTR, Upper Saddle River NJ.

Martin, R.C., 2003. *UML for Java Programmers*. Prentice Hall PTR, Upper Saddle River NJ.

Mercurial community, 2014. Mercurial source control management. URL http://mercurial.selenic.com/.

Michalewicz, Z., Deb, K., Schmidt, M., and Stidsen, T., 1999. Evolutionary algorithms for engineering applications. In K. Miettinen, P. Neittaanmäki, M.M. Mäkelä, and J. Périaux, editors, *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*, pages 135–161. Wiley, New York.

Mitchell, M., 1999. *An Introduction to Genetic Algorithms*. MIT Press, London, England.

Mühlenbein, H., 1991. Evolution in time and space - the parallel genetic algorithm. In *Foundations Of Genetic Algorithms*, pages 316–337. Morgan Kaufmann.

Mühlenbein, H., Schomisch, M., and Born, J., 1991. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17(6 - 7):619 – 632. ISSN 0167-8191. doi:http://dx.doi.org/10.1016/S0167-8191(05)80052-3. URL http://www.sciencedirect.com/science/article/pii/S0167819105800523.

National Grid, 2014. Fields from specific power lines: 275 kV. URL `http://www.emfs.info/sources/overhead/specific/275-kv/`.

Nicolis, G., 1995. *Introduction to Nonlinear Science*. Cambridge University Press, Cambridge UK.

Noesis Solutions, 2016. Optimus. URL `http://www.noesissolutions.com/our-products/optimus`.

Oliver, J.M., Kipouros, T., and Savill, A.M., 2013. A self-adaptive genetic algorithm applied to multi-objective optimization of an airfoil. In M. Emmerich, A. Deutz, O. Schuetze, T. Bäck, E. Tantar, A.A. Tantar, P. del Moral, P. Legrand, P. Bouvry, and C.A. Coello, editors, *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV*, volume 227 of *Advances in Intelligent Systems and Computing*, pages 261–276. Springer International Publishing, Cham. ISBN 978-3-319-01127-1. URL `http://dx.doi.org/10.1007/978-3-319-01128-8_17`.

Oliver, J.M., Kipouros, T., and Savill, A.M., 2014. Electrical power grid network optimisation by evolutionary computing. *Procedia Computer Science*, 29(0):1948 – 1958. ISSN 1877-0509. doi:http://dx.doi.org/10.1016/j.procs.2014.05.179. URL `http://www.sciencedirect.com/science/article/pii/S1877050914003561`. 2014 International Conference on Computational Science.

Oliver, J.M., Kipouros, T., and Savill, A.M., expected 2016. Multi-objective optimization by self-adaptive evolutionary algorithm. In M. Emmerich and A. Deutz, editors, *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation 5, Series: Studies in Computational Intelligence*. Springer International Publishing, Switzerland.

Oliver, J.M., Kipouros, T., and Savill, A.M., June 2015. An evolutionary computing-based approach to electrical power network configuration. *Emergence: Complexity and Organization*, 17.2. URL `http://`

emergentpublications.com/ECO/about_eco.aspx. ECCS'13 European Conference on Complex Systems; Satellite Workshop: Integrated Utility Services IUS'13.

OptiY GmbH, 2016. OptiY. URL http://www.optiy.eu/.

Oracle, 2014. Java platform standard edition 7 documentation. URL http://docs.oracle.com/javase/7/docs/.

Pandya, K.S. and Joshi, S.K., 2008. A survey of optimal power flow methods. *Journal of Theoretical and Applied Information Technology*, 4(5):450–458.

Pearl, J., 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN 0-201-05594-5.

Polya, G., 2004. *How to solve it - a new aspect of mathematical method*. Princeton University Press, Princeton, expanded princeton science library edition edition.

Purshouse, R.C. and Fleming, P.J., 2007. On the evolutionary optimization of many conflicting objectives. *Trans. Evol. Comp*, 11(6):770–784. ISSN 1089-778X. doi:10.1109/TEVC.2007.910138. URL http://dx.doi.org/10.1109/TEVC.2007.910138.

R Development Core Team, 2014. The r project for statistical computing. URL http://www.r-project.org/index.html.

Ram, R., Cooper, Y.N., Bhatia, V., Karthikeyan, R., and Periasamy, C., 2014. Design optimization and analysis of NACA 0012 airfoil using computational fluid dynamics and genetic algorithm. *Applied Mechanics and Materials*, 664:111 – 116. doi:10.4028/www.scientific.net/AMM.664.111. URL http://www.scientific.net/AMM.664.111.

Rangavajhala, S., Mullur, A., and Messac, A., 2007. The challenge of equality constraints in robust design optimization: examination and new approach.

Structural and Multidisciplinary Optimization, 34(5):381–401. ISSN 1615-147X. doi:10.1007/s00158-007-0104-8. URL http://dx.doi.org/10.1007/s00158-007-0104-8.

Rao, S.S., 1996. *Engineering Optimization Theory And Practice*. Wiley, New York, 4th edition.

Rosenberg, R.S., 1967. *Simulation of genetic populations with biochemical properties*. Ph.D. thesis, University of Michigan.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., 1991. *Object-oriented Modeling and Design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. ISBN 0-13-629841-9.

Rylatt, M., Gammon, R., Boait, P.J., Varga, L., Allen, P., Savill, M., Snape, R., Lemon, M., Ardestani, B., Pakka, V., Fletcher, G., Smith, S., Fan, D., and Strathern, M., 2013. Cascade: An agent based framework for modeling the dynamics of smart electricity systems. *Emergence: Complexity and Organization (Special Issue: Complexity and the Smart Electricity Grid)*, 15(2):1–13.

Saadat, H., 1999. *Power System Analysis*. McGraw-Hill, New York.

Sareni, B., Regnier, J., and Roboam, X., 2004. Recombination and self-adaptation in multi-objective genetic algorithms. In P. Liardet, P. Collet, C. Fonlupt, E. Lutton, and M. Schoenauer, editors, *Artificial Evolution*, volume 2936, pages 115–126. Springer Berlin Heidelberg.

Saxena, D.K. and Deb, K., 2007. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization: 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007. Proceedings*, pages 772–787. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-70928-2. doi:10.1007/978-3-540-70928-2_58. URL http://dx.doi.org/10.1007/978-3-540-70928-2_58.

Schaefer, G. and Nolle, L., 2006. Generic black box optimisation algorithms for colour quantisation. In A. Tiwari, R. Roy, J. Knowles, E. Avineri, and K. Dahal, editors, *Applications of Soft Computing*, volume 36 of *Advances in Intelligent and Soft Computing*, pages 15–21. Springer Berlin Heidelberg. ISBN 978-3-540-29123-7. doi:10.1007/978-3-540-36266-1_2. URL http://dx.doi.org/10.1007/978-3-540-36266-1_2.

Schaffer, J.D., 1984. Some experiments in machine learning using vector evaluated genetic algorithms.

Schwefel, H.P., 1997. Advantages (and disadvantages) of evolutionary computation over other approaches. In T. Bäck, D.B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages A1.3:1–A1.3:2. IOP Publishing, Bristol, UK, 1st edition. ISBN 0750303921.

Sederberg, T.W. and Parry, S.R., 1986. Free-form deformation of solid geometric models. *SIGGRAPH Comput.Graph.*, 20(4):151–160.

Selig, M., 2014. Naca 0012 airfoils. URL http://aerospace.illinois.edu/m-selig/ads/afplots/n0012.gif.

Selvan, K.M., 2015. On The Effect Of Shape Parameterization On Aerofoil Shape Optimization. *International Journal of Research in Engineering & Technology*, 4:123 – 133. doi:DOI:10.15623/ijret.2015.0402016.

Siirtola, H., 2000. Direct manipulation of parallel coordinates. In *Proceedings of IEEE International Conference on Information Visualization, 2000.*, pages 373–378. ISBN 1093-9547. ID: 1.

Siirtola, H. and Räihä, K.J., 2006. Interacting with parallel coordinates. *Interacting with Computers*, 18(6):1278–1309. URL http://iwc.oxfordjournals.org/content/18/6/1278.full.pdf+html.

Smith, J. and Fogarty, T., 1996. Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*, pages 318–323. ieee.

Sörensen, K., 2013. Metaheuristics - the metaphor exposed. *International Transactions in Operational Research*, pages n/a–n/a. ISSN 1475-3995. doi: 10.1111/itor.12001. URL `http://dx.doi.org/10.1111/itor.12001`.

Sörensen, K. and Glover, F., 2013. Metaheuristics, 3rd ed. In S.I. Gass and M.C. Fu, editors, *Encyclopedia of operations research*, pages 960–970. Springer, 3rd edition.

Storn, R. and Price, K., 1997. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Tan, K.C., Chiam, S.C., Mamun, A.A., and Goh, C.K., 2009. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. *European Journal of Operational Research*, 197(2):701–713.

Tan, K.C., Goh, C.K., Yang, Y.J., and Lee, T.H., 2006. Evolving better population distribution and exploration in evolutionary multi-objective optimization. *European Journal of Operational Research*, 171(2):463–495.

The MathWorks, Inc., 2016a. MATLAB. URL `https://uk.mathworks.com/products/matlab/`.

The MathWorks, Inc., 2016b. Optimization Toolbox. URL `https://uk.mathworks.com/products/optimization/`.

The Object Management Group, 2014. Unified modeling language. URL `http://www.uml.org/`.

Tusar, T. and Filipic, B., 2007. Differential evolution versus genetic algorithms in multiobjective optimization. In *Proceedings of the 4th international conference on Evolutionary multi-criterion optimization*, EMO'07, pages 257–271. Springer-Verlag, Berlin, Heidelberg.

Vekaria, K. and Clack, C., 1998. Selective Crossover in Genetic Algorithms: An Empirical Study. In *Proceedings of 5th Conference on Parallel Problem Solving from Nature*, volume LNCS 1498, pages 438–447. Springer Verlag.

Veldhuizen, D. and Lamont, G., 2000. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147. ISSN 1063-6560. doi:10.1162/106365600568158.

Wolpert, D.H., 2012. What the no free lunch theorems really mean; how to improve search algorithms. *Sante Fe Institute working paper*. URL http://www.santafe.edu/media/workingpapers/12-10-017.pdf.

Wolpert, D.H. and Macready, W.G., 1997. No free lunch theorems for optimization. *IEEE Transactions On Evolutionary Computation*, 1(1):67–82.

Zalzala, A.M.S. and Fleming, P.J., 1997. *Genetic Algorithms in Engineering Systems*. Institution of Electrical Engineers, Stevenage.

Zell, A. *et al.*, 2016. EvA2. URL http://www.ra.cs.uni-tuebingen.de/software/JavaEvA/introduction.html.

Zhang, J. and Sanderson, A.C., 2007. Jade: Self-adaptive differential evolution with fast and reliable convergence performance. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 2251–2258. ID: 1.

Zhang, J. and Sanderson, A.C., 2008. Self-adaptive multi-objective differential evolution with direction information provided by archived inferior solutions. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2801–2810. ID: 1.

Zitzler, E., Deb, K., and Thiele, L., 2000. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8:173–195.

Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C.M., and da Fonseca, V.G., 2002. Why quality assessment of multiobjective optimizers is difficult. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002*, pages 666–674.

Zitzler, E. and Thiele, L., 1998. Multiobjective optimization using evolutionary algorithms - a comparative case study. In *Parallel Problem Solving From Nature - PPSN V*, pages 292–301. Springer.

Zitzler, E. and Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, 3(4):257–271. ID: 1.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., and da Fonseca, V.G., 2003. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.