Multi-objective evolutionary design of robust controllers on the grid

SHENFIELD, Alex <http://orcid.org/0000-0002-2931-8077> and FLEMING, Peter

Multi-objective Evolutionary Design of Robust Controllers on the Grid

Alex Shenfield[a,*], Peter J. Fleming[b]

[a] *School of Engineering, Manchester Metropolitan University, Manchester, M1 5JD, UK*
[b] *Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, S1 3JD, UK*

Abstract

Coupling conventional controller design methods, model based controller synthesis and simulation, and multi-objective evolutionary optimisation methods frequently results in an extremely computationally expensive design process. However, the emerging paradigm of grid computing provides a powerful platform for the solution of such problems by providing transparent access to large-scale distributed high-performance compute resources. As well as substantially speeding up the time taken to find a single controller design satisfying a set of performance requirements this grid-enabled design process allows a designer to effectively explore the solution space of potential candidate solutions. An example of this is in the multi-objective evolutionary design of robust controllers, where each candidate controller design has to be synthesised and the resulting performance of the compensated system evaluated by computer simulation. This paper introduces a grid-enabled framework for the multi-objective optimisation of computationally expensive problems which will then be demonstrated using and example of the multi-objective evolutionary design of a robust lateral stability controller for a real-world aircraft using H_∞ loop shaping.

Keywords: Aerospace Stability Control, H_∞ Controller Design, Multiobjective Evolutionary Algorithms, High Performance Computing

1. Introduction

Modern aircraft consist of many complex subsystems, all of which require robust and reliable control. These systems are often multi-variable, consisting of multiple inputs and multiple outputs, and frequently the desired responses of a subsystem are in conflict with each other (for example, a controller design that achieves the minimum possible overshoot of the plant often requires accepting a slower rise time than might otherwise have been achieved).

Whilst conventional robust controller design methods such as H_∞ or LQG control can be effectively used to create controllers that are robust both to modelling uncertainties and to cross-coupling between channels in complex multi-variable systems, the resulting controlled system often performs unsatisfactorily. One approach to overcoming this problem is by coupling novel evolutionary multi-objective optimisation techniques with these conventional controller design methods. This provides the engineer with a set of powerful tools for addressing complex multi-variable problems with performance constraints (Fleming and Purshouse, 2002). This type of integrated multi-objective optimisation approach to the design of robust controllers has been successfully used for the design of fixed structure robust H_∞ controllers (Wang and Li, 2011), as well as forming the basis of a novel multi-objective PID controller design procedure (Reynoso-Meza et al., 2012, 2013). However, such methods are frequently computationally expensive, requiring many thousands of controller designs to be evaluated.

Grid computing offers one potential solution to the computationally expensive nature of this evolutionary controller design process. The grid computing paradigm aims to provide "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" (Foster and Kesselman, 1999). This paradigm is differentiated from traditional approaches to distributed computing by its emphasis on providing "a seamless, integrated computational and collaborative environment" (Baker et al., 2002) for the solution of complex problems by allowing coordinated resource sharing across dynamic virtual organisations (Foster et al., 2001). By coupling evolutionary multi-objective optimisation techniques with the large scale distributed high performance computing resources offered by the grid computing paradigm, engineers and designers can effectively address many complex, computationally expensive multi-variable problems - including those that require the synthesis of robust controllers as part of the evaluation process. Grid-enabled optimisation of single objective engineering design problems has been successfully integrated into both computer aided engineering workflows (Weng et al., 2012) and multi-disciplinary design workflows (Lee et al., 2009) to provide easy ac-

*Corresponding author
Email addresses: a.shenfield@mmu.ac.uk (Alex Shenfield), p.fleming@sheffield.ac.uk (Peter J. Fleming)

cess to powerful real-time analysis and optimization routines. This allows a potential reduction in both design cycle times and development costs, with a commensurate improvement in product quality.

The purpose of this paper is to describe a grid-enabled framework for evolutionary multi-objective design. This framework will then be applied to the design of a robust controller for the flight dynamics of a real-world aircraft - a complex problem with many (often conflicting) objectives to consider. The paper is organised as follows: section 2 will provide a brief introduction to evolutionary algorithms, their use in control systems engineering and their application to multi-objective optimisation problems; section 3 will describe the grid computing paradigm in detail and highlight some of the key features that are used in the development of the optimisation framework; section 4 will discuss the implementation of the grid-enabled framework for multi-objective evolutionary optimisation; section 5 will demonstrate the use of the grid-enabled optimisation framework in designing robust control systems for the lateral stability of aircraft; and section 6 will present our conclusions and outline some ideas for further work.

2. Multi-objective evolutionary algorithms

2.1. Background to evolutionary algorithms

Evolutionary Algorithms (EAs) utilise some of the concepts behind natural selection and population genetics to iteratively evolve a population of candidate solutions to a problem (Goldberg, 1989). They both explore the solution space of a problem (by using variation operators such as mutation and recombination) and exploit valuable information present in the previous generation of candidate solutions (by using a selection operator). The trade-off between exploration of undiscovered regions of the solution space and exploitation of promising areas already discovered by the algorithm is extremely important: too much exploration and the algorithm will take too long to converge on a useful solution, too much exploitation and the algorithm may converge prematurely to local optima. Obtaining a correct balance between exploration of the solution space and exploitation of promising solutions is somewhat of a "black art" (Purshouse, 2003), with little guidance available in the literature on setting the parameters that control this balance. Some promising results have been obtained using computational steering frameworks to allow these parameters to be altered during the run-time of the algorithm (Bullock et al., 2002; Shenfield et al., 2007), but this can be time-intensive and requires the engineer to have a good knowledge of both the optimisation problem and the algorithm design. Another potential solution is to use some kind of *self-adaptation* to dynamically change the balance between exploration and exploitation as the algorithm runs (Beyer, 1995; Igel et al., 2007).

One of the main reasons evolutionary algorithms are applicable across many different problem domains (including those where conventional optimisation techniques may struggle) is their use use of evaluation function information directly, rather than derivative information or other auxiliary knowledge. For many non-trivial real-world applications this evaluation function information is obtained by computer simulation of the system. For example, in the optimisation of maintenance schedules for gas turbine aero-engines (Shenfield et al., 2010), the cost information for each schedule is obtained by the computer simulation of a candidate solution over a time period of 25 years. However, this use of computer simulation to obtain evaluation function information leads to some additional problems. To ensure that the results gained from the evolutionary algorithm accurately represent the real-world system, the simulation must be complex enough to capture all the relevant dynamics of the true system. Assuming that this level of complexity is obtainable, this can often lead to the simulation becoming very computationally expensive. Since EAs are both iterative and population based, the simulation may have to be run several thousand times which increases the computational requirements (in terms of computer clock cycles) of the optimisation process significantly.

2.2. Multi-objective evolutionary algorithms

Many real-world engineering problems involve the satisfaction of several, often conflicting, objectives. The general form of a multi-objective optimisation problem can be characterised by a vector of objective functions, f, and the corresponding set of decision variables, x, as illustrated in equation 1 (note that minimisation can be assumed here with no loss of generality).

$$\min_{f}(x) = (f_1(x), \ldots, f_n(x)) \tag{1}$$

In this case it is unlikely that a single optimal solution will exist. Instead, the solution of this kind of multi-objective problem leads to a set of Pareto optimal points, where any improvement in one objective will lead to a deterioration in one or more of the other objectives.

A set of non-dominated solutions[1] generated by a multi-objective optimisation algorithm is known as an *approximation set* (Zitzler et al., 2003) and the quality of this set can be characterised by three main performance indicators (Purshouse, 2003):

- The **proximity** of the approximation set to the true Pareto front.
- The **diversity** of the distribution of solutions in the approximation set.
- The **pertinency** of the solutions in the approximation set to the decision maker.

[1] A solution in non-dominated if there exists no other solution in the set of current candidate solutions that is better in all objectives.

These concepts are illustrated graphically in Fig. 1, where it can be seen that the ideal approximation set produced by an optimiser should be both as close as possible to the true Pareto front (i.e. having good proximity) and provide a uniform spread of solutions across the region of interest of the decision maker (i.e. having a diverse set of candidate solutions that are pertinent to the decision maker).

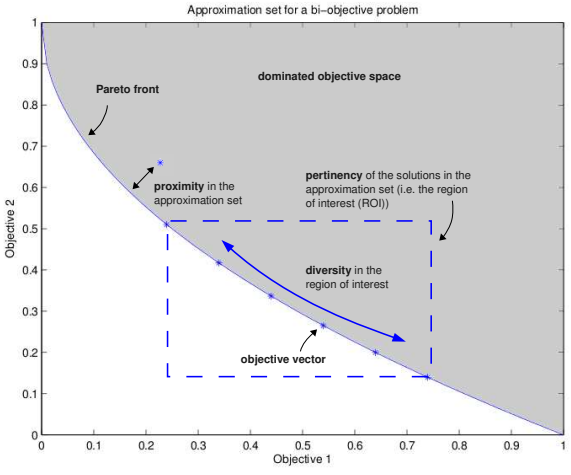

Figure 1: Characterising the approximation set for a bi-objective problem

Conventional multi-objective optimisation methods (such as the weighted sum method (Hwang and Masud, 1979) and the goal attainment method (Gembicki, 1974)) often struggle to satisfy these requirements in the optimisation of real-world engineering problems as they can find only a single point from the approximation set rather than a diverse distribution of potential solutions. This means that a decision maker cannot fully understand the shape of the trade-off space (and thus know whether the *a priori* trade-offs they have chosen are appropriate) without running the optimisation routine many times. However, since evolutionary algorithms search a population of candidate solutions in parallel, they are able to find multiple non-dominated solutions from this approximation set. This provides the decision maker with a set of potential solutions to choose from, rather than a single solution that may not meet the required performance criteria.

A further complication in the application of optimisation routines in real-world engineering design problems is that the optimiser is often required to deal with a large number of objectives. This has lead to interest amongst the research community in the area of *many-objective optimisation*[2]. The increased scale of many-objective optimisation problems means that the *pertinency* (see Fig. 1) of the candidate solutions within an approximation set is especially important so as to avoid overwhelming the decision maker with irrelevant solutions. This can be a critical issue in problems with many conflicting objectives because the global trade-off surface will contain large numbers of Pareto-optimal solutions, many of which may not be in the area of the search space that the decision maker is interested in (Purshouse, 2003). To overcome this issue many authors have suggested allowing the decision maker to focus the search on relevant areas of the solution space increases the efficiency of the optimisation process and reduces the amount of irrelevant information that has to be considered (Fleming et al., 2005).

2.3. Multi-objective evolutionary algorithms in control systems design

Multi-objective evolutionary algorithms have been successfully applied to many problems in the field of control systems engineering, from the offline design of robust controllers for a coal-fired gasification plant (Griffin et al., 2000) to model identification of nonlinear systems (Tan and Li, 2002). Whilst the majority of the applications of evolutionary multi-objective optimisation in control systems engineering have been in offline applications due to the iterative nature of the evolutionary design process, they have also been used in online applications applications such as hardware-in-the-loop tuning of a fuzzy logic based DC motor controller (Stewart et al., 2004).

The widest use of MOEAs in control systems engineering has been in controller design problems such as the tuning of robust PID controllers (Herrero et al., 2008; Zhao et al., 2011) and the design of intelligent model predictive control strategies (Garcia et al., 2011), where their robustness to noise and ability to produce a set of non-dominated candidate solutions that meet some specified performance requirements provide a powerful tool to control engineers. However, MOEAs have also been widely used in systems identification (Rodriguez-Vazquez et al., 2004) and robotics (Capi, 2008; Moshaiov and Ashram, 2009).

2.4. Parallel evolutionary algorithms

The computationally expensive nature of the evaluation process of evolutionary algorithms has led to the development of parallel evolutionary algorithms, though this parallelism often adds significant additional complexity to the algorithm design. Parallel evolutionary algorithms can be classified into two main categories (Cantú-Paz and Goldberg, 1999): those with a single panmictic population that is maintained globally, and those with multiple communicating subpopulations.

Single-population parallel evolutionary algorithms can be effectively exploited using the well established Master-Slave communication paradigm shown in Fig. 2. In this case the evaluation of the candidate solutions is typically distributed amongst the worker nodes whilst the master node applies the evolutionary operators, such as selection

[2]The phrase *many-objective* has been suggested in the Operations Research (OR) community to refer to problems with more than the standard two or three objectives (Farina and Amato, 2004).

and variation, centrally to the whole population (Fogarty and Huang, 1991). Chipperfield and Fleming (1995) also describe a similar scheme where both the evaluation of candidate solutions and the variation operators are performed by the worker nodes.

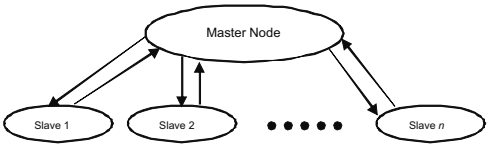

Figure 2: The master-slave communication paradigm

Parallel evolutionary algorithms based around the concept of multiple communicating subpopulations (also known as island-model or migration EAs) introduce a degree of geographical isolation into the search by dividing the population up into subpopulations (known as demes) and allowing each of these to evolve separately. Periodically migration occurs to allow an exchange of information between subpopulations (Rivera, 2001). This migration strategy is governed by the desired population topology of the algorithm and can be anything from the simple ring topology shown in Fig. 3 to a fully interconnected topology where migration occurs between each and every subpopulation.

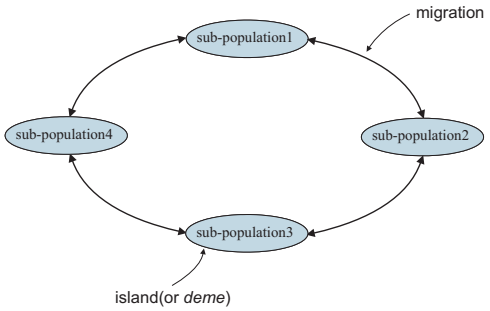

Figure 3: An island model parallel evolutionary algorithm in a ring topology

Globally parallel evolutionary algorithms with a single panmictic population provide the simplest form of parallelism (being functionally equivalent to sequential EAs). They represent an important class of parallelism since their use means that existing EA theory and design guidelines can easily be applied to the problem (Cantú-Paz and Goldberg, 1999). Whilst this type of strategy does not exploit all the parallelism inherent in the evolutionary algorithm, substantial improvements in performance can be achieved - especially in cases where the evaluation of candidate solutions is significantly more computationally expensive that the evolutionary operators themselves (Chipperfield and Fleming, 1995).

Several authors have shown that the use of multiple communicating subpopulations can improve the convergence of the algorithm for some problems (Grosso, 1985; Starkweather et al., 1991). Unfortunately, not only is it poorly understood under what conditions multiple communicating populations perform well, it is still unclear how effectively the multiple communicating populations paradigm scales to problems with more than a single objective. Whilst Deb et al. (2003) report some success applying an island model EA to optimisation problems with two and three objectives, their approach of using a guided domination strategy to calculate different parts of the Pareto-optimal front using different demes is limited to problems with a convex Pareto front. A more generally applicable approach to sharing information between subpopulations in a multi-objective optimisation problem is to use a "divide and conquer" approach such as that suggested by Hiroyasu et al. (2000), where the migration step is used to order the current set of candidate solutions between the subpopulations based how well they satisfy each objective. However, a downside to this approach is that search effort is wasted as the subpopulations evolve away from the area they have been focussed on. The scalability of both these approaches is also heavily dependent on the problem being solved.

Ultimately, the decision as to which of these forms of parallelisation to implement must consider several factors such as applicability to the problem being considered, ease of implementation and use, and the potential performance gains from parallelisation. Single-population parallel EAs are often easiest to implement and use, since experience gained with sequential EAs is directly applicable. In contrast, the implementation of parallel EAs with multiple communicating populations requires the consideration of extra design choices. For instance, the use of an island model EA requires the algorithm designer to choose the number of demes, the population topology, and the mutation rate, as well as choosing values for the standard evolutionary parameters. This substantially increases the complexity of the parallel EA since each of these parameters influences the efficiency of the algorithm and the quality of the overall solution.

3. Grid computing

The concept of grid computing is not new. As far back as 1969 Len Kleinrock suggested:

> "We will probably see the spread of 'computer utilities', which, like present electric and telephone utilities, will serve individual homes and offices across the country." Klienrock (1969)

However, it is only quite recently that technologies such as the Globus Toolkit (Foster and Kesselman, 1999) and web services have emerged to enable this kind of aggregation of compute resources. The Globus Toolkit provides an open-source, community-based set of software tools that enables multiple compute, data and other resources to be

combined to form large-scale computational grids. One of the key features of the Globus Toolkit is its ability to provide a common means of interacting with the diverse range of local resource management systems that are often found in real-world compute resources.

3.1. Web service architectures

Web services provide an interoperable, service-oriented approach to enabling application functionality using common standards-based internet protocols such as HTTP. Two main types of web services exist: those based around a transport-agnostic, standards-based approach that uses XML formatted messages exchanged between service and client (SOAP-based web services) and those using a lighter-weight style based around common HTTP operations such as GET and POST (known as representational state transfer based - or RESTful - web services). The key advantage that SOAP based web services have over RESTful web services is that their transport-agnostic nature ensures that they are well suited for distributed computing environments where the messages exchanged between service and client may go through other nodes.

Although the World Wide Web Consortium recognises both SOAP based and RESTful web services, their definition of a web service is "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages" (W3C Working Group, 2004). Three core technologies are used in the creation and specification of SOAP based web services (Chappell and Jewell, 2002):

- Simple Object Access Protocol (SOAP)
- Web Services Description Language (WSDL)
- Universal Description, Discovery, and Integration (UDDI)

Fig. 4 shows that, in a SOAP based web service, the client first queries a service registry for the contract details of the desired service (i.e. the WSDL document). This query can be done by service name, service category, or other identifier. Once this service has been located, the client uses the WSDL document to find out how to interact with the service. The communication between client and service is then carried out by sending and receiving SOAP messages over HTTP that conform to the specific XML schema found in the WSDL document.

3.2. Open grid services architecture

The Open Grid Services Architecture (OGSA) provides a service-oriented approach to grid computing that builds on key web service technologies (including SOAP and WSDL) to allow grid computing resources to be exposed as services. By following the open grid service architecture design principles and exposing grid resources such as computers, datasets and simulation software as services, the Globus Toolkit is able support the creation and management of ensembles of services maintained by virtual organisations (Foster et al., 2002). There are three main advantages to representing these resources as services:

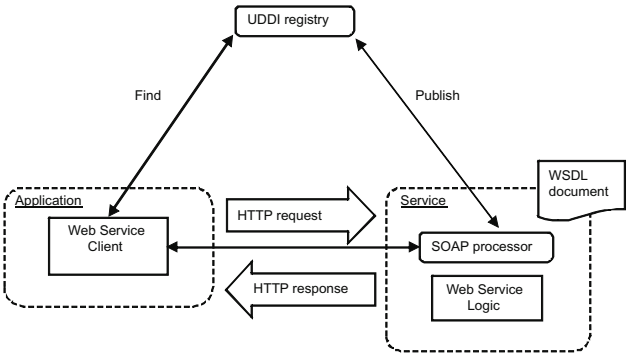

Figure 4: Interaction between Web Service Technologies

1. *It aids interoperability.* A service-oriented view addresses the need for standard service definition mechanisms, local/remote transparency, adaptation to local OS services, and uniform semantics (Foster et al., 2002).
2. *It simplifies virtualisation.* Virtualisation allows for consistent resource access across multiple heterogeneous platforms by using a common interface to hide multiple implementations (Foster et al., 2002).
3. *It enables incremental implementation of Grid functionality.* The provision of Grid functionality via services means that the application developer is free to pick and choose the services that provide the desired behaviour to their application.

4. A grid-based framework for multi-objective optimisation using evolutionary algorithms

4.1. Parallelisation of the evolutionary algorithm

In section 2.4 two main categories of parallel evolutionary algorithm were discussed: globally parallel EAs with a single population and island model EAs using multiple communicating subpopulations. Whilst the use of multiple communicating subpopulations exploits more of the inherent parallelsim within the evolutionary algorithm, it also introduces additional complexity by requiring the engineer to choose several additional parameters. Furthermore, it is still unclear how algorithms based around the multiple communicating subpopulations paradigm perform in the presence of many objectives (such as is often the case in real-world engineering design problems). In contrast, the

use of single population globally parallel evolutionary algorithms allows experience gained with sequential EAs to be directly applied.

For this reason it was decided to base the grid-enabled framework for multi-objective optimisation discussed in this paper on the single population globally parallel model using the Master-Slave communication paradigm. This model is well suited for the kind of heterogeneous grid computing environment discussed in section 3 because it keeps inter-process communication to a minimum whilst significantly accelerating the evaluation of candidate solutions (the most computationally expensive part of the application discussed in this paper).

4.2. Implementation of the parallel multi-objective evolutionary framework

The multi-objective evolutionary optimisation framework discussed in this paper was implemented in a service oriented architecture using a combination of web services and grid services. As discussed in section 3, the service oriented approach to grid computing has several advantages over creating a monolithic high performance computing architecture (such as its flexibility and the ease of interoperation between new and existing heterogeneous resources) which ensure it is well suited for the implementation of our framework. In addition to the advantages discussed in section 3, providing the functionality of the optimisation framework as services allows for both the addition of new features (such as additional evolutionary operators) and the integration of the framework into Internet portals (which can then be accessed from any device with a capable web browser).

Fig. 5 shows that the implementation of the parallel multi-objective evolutionary framework relies on two different types of service. One service type exposes the evolutionary multi-objective optimisation operators to the application client, and the other allows for the distributed evaluation of candidate solutions across the available grid resources. These services are written in Java to provide portable code that allows the components of the framework to run across multiple heterogeneous platforms and interact (as shown in the pseudo-code listed in Fig. 6) to provide a flexible grid-enabled multi-objective optimisation framework.

4.2.1. The evolutionary algorithm webservice

This service provides access to the multi-objective evolutionary operators that can be used to:

- Generate initial populations
- Perform multi-objective ranking according to either Pareto-optimality or decision maker preferences (see section 2.2)
- Select a subset of the current population for mating

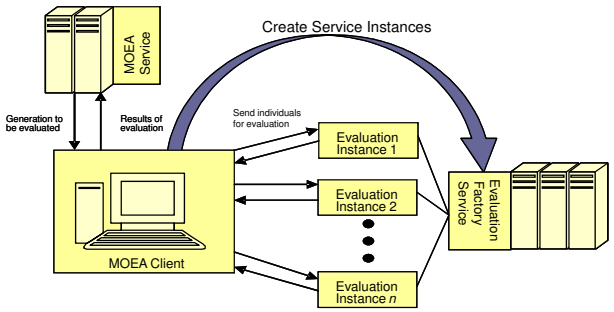

Figure 5: The implementation of the optimisation framework

```
PROCEDURE GridEnabledMOEA:

   // connect to the service
   MOEAService = connectMOEAService();
   EvalService = connectEvalService();

   // initialise resources
   resources = EvalService.findResources();

   // run the evolutionary algorithm
   MOEAService.initialise();
   EvalService.distributedSolutions(resources, solutions);
   EvalService.evaluateSolutions();
   WHILE not finished DO:
      MOEAService.selection();
      MOEAService.recombination();
      MOEAService.mutation();
      EvalService.distributedSolutions(resources, solutions);
      EvalService.evaluateSolutions();
   END

END
```

Figure 6: Pseudo-code describing the interaction between services

- Evolve the current population by applying variation operators

It is important to note that the provision of the multi-objective evolutionary methods as services allows a great deal of flexibility in both the use of the framework and in its maintenance. Providing additional functionality for the optimisation routine (such as diversity preservation measures or alternative variation methods) or improving existing functionality can be achieved by simply creating new services and integrating them into the application client.

4.2.2. The evaluation grid service

The distribution and management of computational tasks across a diverse set of geographically distributed compute resources is a difficult problem due to the highly dynamic and decentralised nature of the resources involved in grid computing. There are many resource management systems that can successfully schedule, distributed and manage tasks at a local level but the lack of centralised control in a computational grid means that there are few resource schedulers that are effective in complex large scale grid computing environments. The evaluation grid service shown in Fig. 5 addresses this problem using an *application-centric meta-scheduling approach*[3] to distribute the candidate solutions across the resources of the White Rose Grid (see section 4.3) for evaluation. This application-centric approach is similar to that taken by the AppLeS project (Berman et al., 1996). However, in this case, response time information from previously completed generations of the evolutionary algorithm is used to provide estimates of the current computational capacity of the available grid resources rather than explicitly querying the computational resources for their current state (a process that can be both complex and time intensive). Results presented in Shenfield and Fleming (2013) have shown that this approach performs extremely well in complex distributed and dynamic environments (such as computational grids).

The evaluation grid service exposes three methods to the multi-objective optimisation client (as shown in the pseudo-code in Fig. 6):

1. **findResources()** - this method queries a database to discover what grid resources are available to this application and to obtain some initial information about their states.
2. **distributeSolutions(resources, solutions)** - this method uses the application-centric meta-scheduler outlined above to calculate the optimal workload allocation (i.e. the optimal number of candidate solutions to send to each resource) with respect to the mean response time of jobs through the system, for a given set of resources. This optimal workload allocation is calculated using elements of queueing theory (see Kleinrock (1975) for more details) to minimise the mean response time for the evaluation of candidate solutions and takes into account both the usage of the grid resources and any congestion in the network (as both of these factots affect the response time of jobs through the system). It then transfers these candidate solutions to the grid resources using either SFTP (the Secure File Transfer Protocol) or GridFTP[4]. More details of this scheduling approach can be found in Shenfield and Fleming (2013).
3. **evaluateSolutions()** - this method starts a job manager daemon on the grid resources to manage the objective function evaluations. It does this by using the local resource management system (in the case of the White Rose Grid resources this is Sun Grid Engine) to run as many instances of the evaluation function as there are candidate solutions. These evaluation function instances are then queued by the local scheduler and run when appropriate compute resources become available. The results are then returned to the client.

4.3. The White Rose Grid

The White Rose Grid (The White Rose University Consortium, 2012) was established in 2002 to support e-Science research at the Universities of Leeds, Sheffield and York. It is a multi-institutional computational grid providing access to a large number of heterogeneous compute resources. The White Rose Grid currently consists of four high-performance compute nodes located at three different sites (see Fig. 7 for an overview of the network topology) providing access to approximately 700 processor cores for both local users and distributed e-Science research. Each of these compute nodes is connected via the high-speed, low latency Yorkshire and Humberside Metropolitan Area Network.

5. Robust control of aircraft flight dynamics

A control system for the flight dynamics of an aircraft must provide robust and responsive multi-variable control of the ailerons and the rudder, as well as guaranteeing stability in the presence of modelling uncertainty. Whilst H_∞ control theory offers a proven method of designing controllers that are robust to such uncertainty, the performance of the resulting system can often be unsatisfactory. The following subsections will provide an overview of the flight dynamics of an aircraft and introduce the concept of H_∞ control and loop shaping - a technique that allows the designer to 'shape' the response of a system, and thus improve performance.

[3] Meta-scheduling is an approach where jobs are submitted via local resource management systems rather than directly to the actual machines themselves.

[4] GridFTP offers potential performance benefits when dealing with large data-sets, but requires the administrators of the Grid resources to provide a GridFTP server.

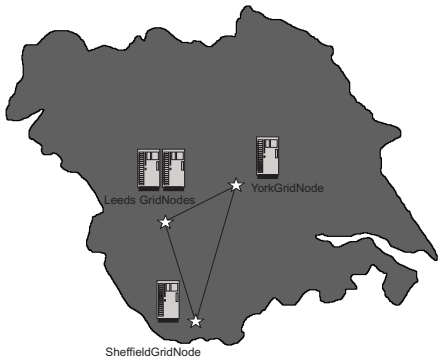

Figure 7: An overview of the network topology of the White Rose Grid

5.1. Flight dynamics

The dynamics of an aircraft in flight can be described by the rotational moments around its centre of gravity (CG) in Cartesian space. These are shown in Fig. 8 where:

- **L** is the *rolling* moment of the aircraft.
- **M** is the *pitching* moment of the aircraft.
- **N** is the *yawing* moment of the aircraft.

These flight dynamics can be separated into *longitudinal* motions, in which the wings remain level (e.g. pitch), and *lateral* motions such as roll and yaw.

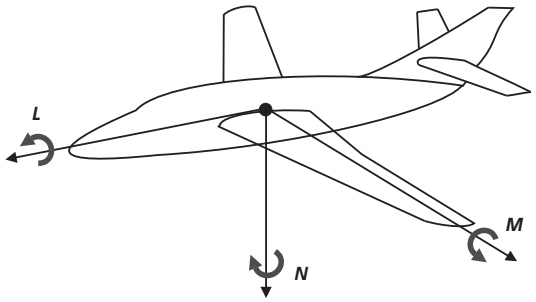

Figure 8: The rotational moments of an aircraft

The equations of motion describing these flight dynamics are non-linear; however, by applying the *small disturbance theory* (Nelson, 1998), a linearised model can be found. This linearisation process will only give a good result in cases where the motion of the aircraft can be fully described by small deviations about a steady flight condition (such as in the flight of large commercial aircraft), and therefore should not be used in cases where large amplitude motions are likely to occur.

5.2. H_∞ loop shaping controller design

H_∞ control theory was originally proposed by Zames (1981) to address the problem of uncertainty in the modelling of disturbances and plants and was further developed by Glover and Doyle (1988). H_∞ control theory provides a general framework for the design of optimal controllers, where optimal in this context refers to the minimisation of the H_∞ norm.

A drawback to robust stabilisation using H_∞ control is the inability of the designer to specify performance requirements (Skogestad and Postlethwaite, 1996), which can result in compensated systems that, whilst robust to modelling uncertainties and cross-coupling between channels, perform unsatisfactorily. To overcome this limitation, McFarlane and Glover (1990) proposed using pre- and post-compensators to 'shape' the open-loop response of the plant (see Fig. 9), and then applying robust stabilisation to the shaped system. Selecting the weighting matrices for the pre- and post-compensators is typically challenging since these choices govern the performance of the resulting system.

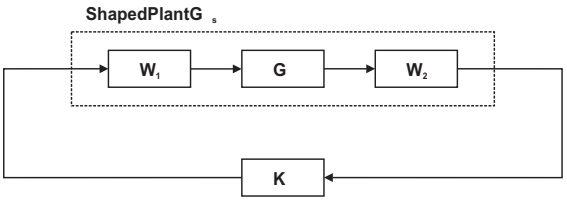

Figure 9: The shaped and compensated plant

Traditional methods for selecting these pre- and post-compensators tend to require a trial-and-error iterative approach (Skogestad and Postlethwaite, 1996), where the shaped and compensated plant is formed and its performance analysed. If this performance is unsatisfactory then the weighting matrices for the pre- and post-compensators are adjusted and the process repeats. In this controller design process the engineer is typically faced with a number of conflicting requirements that must be balanced against each other to achieve the best result possible.

This paper aims to find an optimal H_∞ loop shaping controller for the lateral stability control of a Boeing 747 aircraft using our grid-enabled framework for evolutionary multi-objective optimisation to determine the weighting matrices for the pre- and post-compensators. The Boeing 747 model used in this controller design process is a linearised multivariable system with two inputs (the control signals for the aileron and rudder) and two outputs (the roll and sideslip angles), and can be represented by the following transfer function matrix:

$$\mathbf{G} = \begin{pmatrix} \mathbf{g_{11}} & \mathbf{g_{12}} \\ \\ \mathbf{g_{21}} & \mathbf{g_{22}} \end{pmatrix}$$

where:

$$\mathbf{g_{11}} = \frac{0.1845s^2+0.04795s+0.1995}{s^4+0.6807s^3+1.049s^2+0.3373s-0.001979}$$

$$\mathbf{g_{12}} = \frac{0.06591s^2-0.12s-0.5158}{s^4+0.6807s^3+1.049s^2+0.3373s-0.001979}$$

$$\mathbf{g_{21}} = \frac{-0.01448s^2-0.01962s+0.001359}{s^4+0.6807s^3+1.049s^2+0.3373s-0.001979}$$

$$\mathbf{g_{22}} = \frac{0.005334s^3+0.4377s^2+0.1884s-0.00432}{s^4+0.6807s^3+1.049s^2+0.3373s-0.001979}$$

The pre- and post-compensators for this loop shaping controller have the following structures:

$$\mathbf{W_1} = \begin{pmatrix} \frac{s+a}{s} & 0 \\ 0 & \frac{s+b}{s} \end{pmatrix}$$

$$\mathbf{W_2} = \begin{pmatrix} c & 0 \\ 0 & d \end{pmatrix}$$

with the order of these compensators being determined by the order of the plant.

5.3. Evolutionary multi-objective controller design

The H_∞ loop shaping controller design process can be formulated as a multi-objective optimisation problem, where each performance requirement is treated as a separate objective, and thus solved using a multi-objective evolutionary algorithm (see section 2.2). The decision variables in this optimisation procedure are the weights, a, b, c, d, for the compensators. However, this controller design problem is computationally expensive since, for every candidate solution, an H_∞ controller has to be synthesized and the response of the compensated system obtained by computer simulation. The evaluation of a single candidate solution for this problem took in the order of 5.5 seconds on a Intel Core 2 Duo based PC with a clock speed of 2.60GHz.

To overcome the computationally expensive nature of this controller design problem the grid-enabled framework for evolutionary multi-objective optimisation described in this paper was used in a real-valued multi-objective genetic algorithm configuration with multi-objective ranking performed using the preference articulation operator proposed by Fonseca and Fleming (1998) to incorporate decision maker preferences. Whilst using a single compute cluster for this evaluation process would also be possible (and indeed, under reasonably light loads, is likely to provide a substantial speed up over the evaluation of candidate solutions on a single machine), the key advantage in using a computational grid for this controller design application is the dynamic and scalable nature of the computational resource pool available. This ensures that, even under fairly heavy loads, there are sufficient resources available for the evaluation process.

A real-valued representation for the compensator weights was used since Fogel and Ghoziel (1997) have shown that there is no intrinsic advantage in choosing one bijective representation over another, although particular representations may be more computationally tractable or efficient for certain problems. As a consequence of this, modern MOEA practice emphasises choosing a representation that is appropriate for the problem under consideration (Michalewicz and Fogel, 2000). Selection in our algorithm was performed using Stochastic Universal Sampling (Baker, 1987), which guarantees sampling with zero bias and minimum spread, and is generally considered superior to other selection schemes for many problems (Hancock, 1994). The extended intermediate recombination operator and BGA mutation operator (Mühlenbein and Schlierkamp-Voosen, 1993) were used to introduce variation into the population and prevent the search process from stagnating. However, it is important to note that the implementation of the multi-objective grid-enabled framework in a service-oriented architecture (see section 4.2) provides a high degree of flexibility in the choice of algorithm architecture, representation, and evolutionary operators used. This flexibility means it is simple to adapt the framework to other optimisation problems.

A set of performance requirements arising from domain specific knowledge about the problem have been specified for the response of the compensated system (see Table 1). Some of these requirements are hard constraints and others are simply desired goals. However, several of the performance requirements for this controller are in competition which makes achieving all the goals difficult. The grid-enabled MOEA implementation described in this paper uses the preference articulation operator proposed by Fonseca and Fleming (1998) to handle both the stated goals and hard constraints from Table 1.

Table 1: Performance requirements for the H_∞ controller design problem

	Requirements	Type
1	Minimise the overshoot in response to a step input	Goal (overshoot $<$ 5%)
2	Minimise the rise time	Goal ($T_r < 3$ seconds)
3	Minimise the settling time	Goal ($T_s < 4$ seconds)
4	Prevent aileron actuator saturation	Constraint (Aileron deflection $<$ 0.349 radians)
5	Prevent rudder actuator saturation	Constraint (Rudder deflection $<$ 0.52 radians)
6	Controller must be robust to 30% multiplicitive uncertainty	Constraint

5.4. Numerical results and discussion

The results shown in Fig. 10 are from 25 runs of the optimisation algorithm with no constraints on any of the

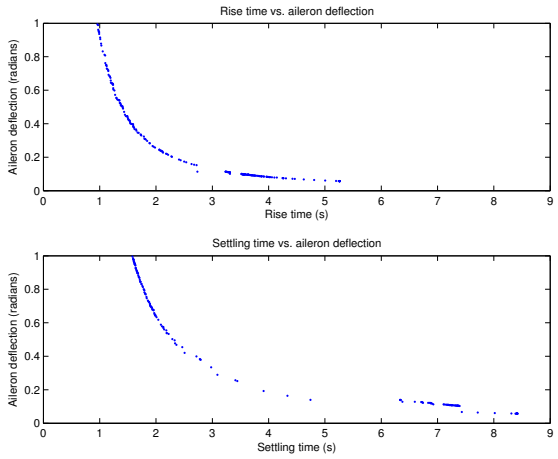

Figure 10: Trade-offs between aileron control signal and other performance requirements

objectives. It can be seen from Fig. 10 that there is a strong trade-off between the deflection of the aileron actuator and two other key performance objectives – the rise time of the compensated system and the settling time of the compensated system. Whilst the aileron actuator deflection in Fig. 10 is shown up to 1 radian, in a real-world aircraft this would not be realistically achievable. A reasonable maximum limit for a large commercial aircraft such as the Boeing 747 considered in this paper would be 0.35 radians which, as can be seen from Fig. 10, would impose performance bounds on the best achievable rise time and settling time of the compensated system of 1.75 seconds and 2.75 seconds, respectively.

5.4.1. Optimisation results

Table 2 shows the best achieved objective value, the average achieved objective value, and the variance of the achieved objective values for each objective. These values are based on offline archives of all optimal solutions from 100 independent runs of the optimisation routine[5]. It can be seen from Table 2 that all the performance requirements specified in Table 1 are satisfied, with significant improvements over all the goal values achievable.

Fig. 11 shows the performance of the chosen controller design to a 0.1 radians step change in the aileron control signal. It can be seen from this step response that the performance of the compensated system is excellent; achieving minimal overshoot and good rise time and settling time.

5.4.2. Performance assessment

A full statistical analysis of the results from 100 independent runs of the optimisation algorithm was undertaken to assess its performance. As mentioned in section

[5]An average of 2213 solutions were stored in the offline archive after each run of the optimiser. These represent all solutions in each optimisation run that met or exceeded the desired goals for the performance of the compensated system.

Table 2: Achieved performance of the compensated system

Obj.	Best Value	Average Value	Variance
1	0.687%	1.07%	0.128
2	1.71 s	1.864 s	0.006
3	2.77 s	2.98 s	0.029
4	0.211 rad	0.333 rad	0.001
5	0.0111 rad	0.185 rad	0.000
6	36.4%	32.4%	2.92

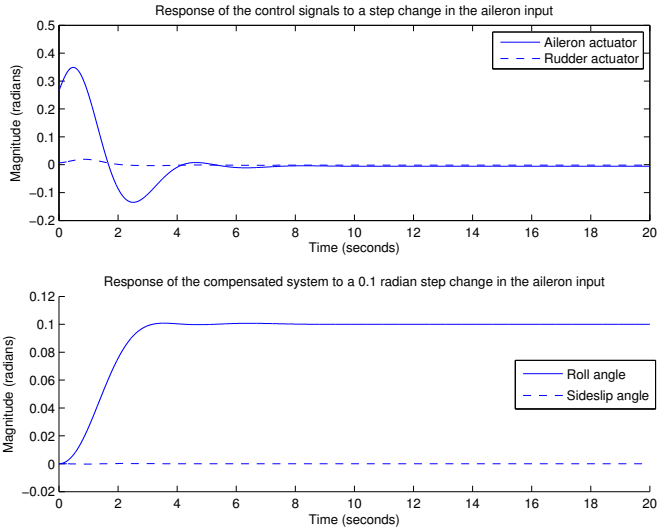

Figure 11: Step response of the compensated system to aileron deflection

2.2, the quality of the final approximation set produced a multiobjective optimisation algorithm can be characterised by:

1. The proximity of the solutions to the true Pareto front.
2. The diversity of the distribution of solutions in the approximation set.
3. The pertinency (i.e. the relevance) of the solutions in the approximation set to the decision maker.

The results presented in section 5.4.1 above have shown that the grid-based framework for multiobjective optimisation using evolutionary algorithms presented in this paper has produced final approximation sets that are highly pertinent to the decision maker meeting all the specified constraints and achieving significant improvements over all the specified goal values and so fulfils the third of these criteria.

The quality of the final approximation sets of the optimisation algorithm in terms of the first and second of the characteristics above can be assessed using three different performance metrics: generational distance (Van Veldhuizen, 1999) to assess proximity, Schott distance (Schott, 1995) to assess diversity, and the hypervolume indicator (Zitzler and Thiele, 1998) which provides a measure of both convergence and diversity in one combined metric. The results for these metrics are shown in Table 3.

Table 3: Statistical results for the performance of the grid-based framework for multiobjective optimisation using evolutionary algorithms in terms of generational distance (GD), Schott distance (SD) and the hypervolume indicator (HV)

	Mean	Variance	Maximum	Minimum
GD	0.012432	$1.7161e^{-5}$	0.029544	0.006869
SD	0.060367	$7.5941e^{-5}$	0.085421	0.042413
HV	6.4079	$3.5524e^{-3}$	6.5003	6.19

The generational distance evaluates the average distance of solutions in an approximation set to the true Pareto front. In the case of real-world optimisation problems, such as the design of a robust lateral stability controller presented in this paper, the true Pareto front may not be known. In this case, a reference set that approximates to the true Pareto front can be used. In this paper, the reference set used was produced using all the non-dominated solutions from 25 independent runs of an unconstrained multiobjective evolutionary algorithm with a population size of 1000 individuals to ensure good coverage of the search space. As Table 3 shows, the grid-based framework for multiobjective optimisation using evolutionary algorithms presented in this paper has produced final approximation sets that are close to the reference set and have minimal variance from run-to-run.

The Schott distance, or spacing metric, quantifies the spread of solutions in objective space. The more evenly these solutions are distributed, the smaller the value of this metric will be. As can be seen from Table 3, the grid-based framework for multiobjective optimisation using evolutionary algorithms presented in this paper has produced final approximation sets that are are evenly distributed across the objective space and that have minimal variance from run-to-run.

The final quality measure we will consider in this paper is the hypervolume indicator proposed in Zitzler and Thiele (1998). This metric provides a measure of both convergence and diversity by calculating the volume (in objective space) dominated by the solutions in the final approximation set and bounded by some reference point[6]. The key advantage of this metric is that it is strictly Pareto compliant (i.e. given two approximation sets, A and B, the value of the hypervolume indicator will always be better for set A if set A dominates set B). Table 3 shows that the grid-based framework for multiobjective optimisation using evolutionary algorithms presented in this paper has produced final approximation sets that cover a large volume of the objective space and that have minimal variance from run-to-run.

5.4.3. Run-time results

Table 4 shows a representative set of execution times from the evolutionary multi-objective optimisation of the controller design problem presented in this paper. These times are averaged over 25 runs for both the results obtained from a single workstation and the results obtained using the resources of the White Rose Grid. As Table 4 shows, the grid-enabled multi-objective optimisation framework discussed in this paper has significantly reduced the time taken to optimise the performance of the compensated system.

Table 4: Execution times for the optimisation of an H_∞ controller for aircraft flight dynamics

Single Workstation		Computational Grid	
50 gen.	100 gen.	50 gen.	100 gen.
28486 s	56672 s	8126 s	122622 s

6. Conclusions and further work

Table 4 has shown that significant reductions in the execution times of the evolutionary multi-objective controller design process can be achieved by using the grid-enabled optimisation framework discussed in this paper. Whilst the example presented in this paper uses a multi-objective evolutionary algorithm in the optimisation process, the service-oriented nature of the framework means it is easily extensible to other iterative optimisation algorithms such as ant-colony or particle swarm optimisation.

[6]For the hypervolume indicator results shown in Table 3 the goal values were used as the bounding reference point in objective space.

The grid-enabled framework for multi-objective optimisation described in this paper is best suited to computationally expensive evaluation functions such as the robust controller design problem presented in section 5. This is due to both the communication overheads involved in the distribution and management of the evaluation function jobs across multiple diverse resources and the high overall utilisation of the compute resources comprising the White Rose Grid. A preliminary investigation into the scale of problems for which this framework is most effective has been performed by altering the computational complexity of the objective function evaluation process and determining the time needed to evaluate 50 candidate solutions using this altered objective function. Five runs of the experiment were performed and the results averaged to obtain accurate data.

As can be seen from Fig. 12, for computationally trivial objective functions these distribution and management overheads can result in a degradation in performance compared with a sequential MOEA on a single machine; however, for objective functions that require over 0.5 seconds to evaluate a single candidate solution, substantial savings in the overall execution time of the algorithm can be achieved. It is expected that further research and development in the fields of grid-middleware, job submission services and job management services will result in a reduction in these communication overheads, allowing the framework described in this paper to provide increased performance for less computationally expensive problems. However, this framework is not intended to replace sequential MOEAs in cases where the performance of the sequential MOEA is satisfactory.

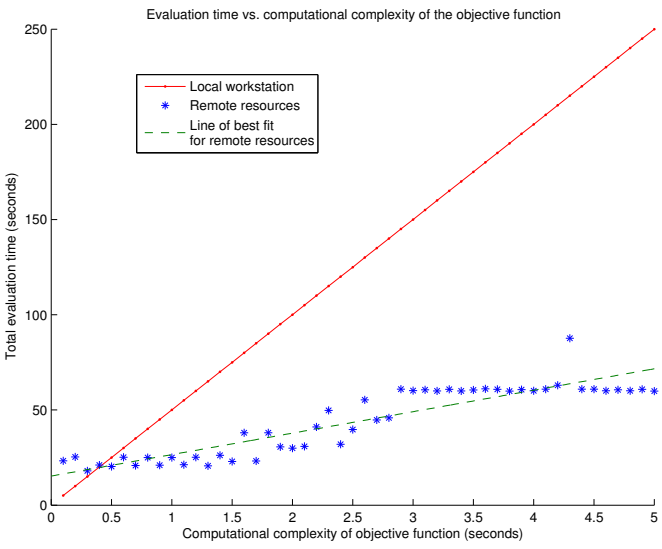

Figure 12: Time taken to evaluate 50 individuals vs. the computational complexity of the objective function

References

Baker, J. E., 1987. Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J. J. (Ed.), Proceedings of the Second International Conference on Genetic Algorithms. Lawrence Erlbaum., pp. 14–21.

Baker, M., Buyya, R., Laforenza, D., 2002. Grid and grid technologies for wide area distributed computing. Software: Practice and Experience 32 (15), 1437–1466.

Berman, F., Wolski, R., Figueira, S., Schopf, J., Shao, G., 1996. Application-level scheduling on distributed heterogeneous networks. In: Supercomputing '96.

Beyer, H. G., 1995. Toward a theory of evolution strategies: Self-adaptation. Evolutionary Computation 3 (3), 311 – 347.

Bullock, S., Cartlidge, J., Thompson, M., 2002. Prospects for computational steering in evolutionary computation. In: Bilotta, E., Groß, D., Smith, T., Lenaerts, T., Bullock, S., Lund, H. H., Bird, J., Watson, R., Pantano, P., Pagliarini, L., Abbass, H., Standish, R., Bedau, M. (Eds.), Artificial Life VIII Workshop Proceedings. MIT Press, pp. 131–137.

Cantú-Paz, E., Goldberg, D. E., 1999. On the scalability of parallel genetic algorithms. Evolutionary Computation 7 (4), 429–449.

Capi, G., 2008. Evolution of efficient neural controllers for robot multiple task performance - a multiobjective approach. In: IEEE international conference on Robotics and Automation 2008. pp. 2195 –2200.

Chappell, D. A., Jewell, T., 2002. Java Web Services. O'Reilly.

Chipperfield, A. J., Fleming, P. J., 1995. Parallel genetic algorithms. In: Zomaya, A. Y. (Ed.), Parallel And Distributed Computing Handbook. McGraw-Hill, Ch. 39, pp. 1118–1144.

Deb, K., Zope, P., Jain, A., 2003. Distributed computing of pareto optimal solutions with evolutionary algorithms. In: Proceedings of the Second International Conference on Evolutionary Multi-Criteria Optimisation (EMO2003). Springer-Verlag, pp. 534–549.

Farina, M., Amato, P., 2004. A fuzzy definition of "optimality" for many-criteria optimization problems. IEEE Transactions on System, Man and Cybernetics - Part A: Systems and Humans 34 (3), 315–326.

Fleming, P. J., Purshouse, R. C., 2002. Evolutionary algorithms in control systems engineering: a survey. Control Engineering Practice 10, 1223–1241.

Fleming, P. J., Purshouse, R. C., Lygoe, R. J., 2005. Many objective optimization: An engineering perspective. In: Coello, C. A. C., Aguirre, A. H., Zitzler, E. (Eds.), Proceedings of the International Conference on Evolutionary Multi-Objective Optimization (EMO2005). Vol. 3470 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp. 14–32.

Fogarty, T. C., Huang, R., 1991. Implementing the genetic algorithm on transputer based parallel processing systems. In: Schwefel, H.-P., Männer, R. (Eds.), Parallel Problem Solving from Nature 1. Vol. 496 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp. 145–149.

Fogel, D. B., Ghoziel, A., 1997. A note on representations and variation operators. IEEE Transactions on Evolutionary Computation 1 (2), 159–161.

Fonseca, C. M., Fleming, P. J., 1998. Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part I: A unified formulation. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 28 (1), 26–37.

Foster, I., Kesselman, C., 1999. The Globus Toolkit. In: Foster, I., Kesselman, C. (Eds.), The GRID: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, Ch. 11, pp. 259–278.

Foster, I., Kesselman, C., Nick, J. M., Tuecke, S., 2002. Grid services for distributed system integration. IEEE Computer 35 (6), 37 – 46.

Foster, I., Kesselman, C., Tuecke, S., 2001. The anatomy of the grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications 15 (3), 200–222.

Garcia, J. J. V., Garay, V. G., Gordo, E. I., Fano, F. A., Sukia, M. L., 2011. Intelligent multi-objective nonlinear model predictive control (iMO-NMPC): towards the on-line optimisation of

highly complex control problems. Expert systems with applications 39 (7), 6527–6540.

Gembicki, F. W., 1974. Vector optimization for control with performance and parameter sensitive indices. Ph.D. thesis, Case Western Reserve University, Cleveland, Ohio.

Glover, K., Doyle, J. C., 1988. State-space formulae for all stabilizing controllers that satisfy an h_∞ norm bound and relations to risk sensitivity. Systems and Control letters, 167–172.

Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

Griffin, I. A., Schroder, P., Chipperfield, A. J., Fleming, P. J., 2000. Multi-objective optimization approach to the alstom gasifier problem. Proceedings of the institute of mechanical engineers 214 (1), 453–468.

Grosso, P. B., 1985. Computer simulation of genetic adaptation: Parallel subcomponent interaction in a multilocus model. Ph.D. thesis, University of Michigan.

Hancock, P. J. B., April 1994. An empirical comparison of selection methods in evolutionary algorithms. In: Fogarty, T. C. (Ed.), Evolutionary Computing - AISB Workshop. Vol. 865 of Lecture Notes in Computer Science. Springer-Verlag, pp. 80–94.

Herrero, J. M., Blasco, X., Martinez, M., Sanchis, J., 2008. Multiobjective tuning of robust pid controllers using evolutionary algorithms. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G., Drechsler, R., Ekart, A., Esparcia-Alcazar, A., Farooq, M., Fink, A., McCormack, J., O'Neill, M., Romero, J., Rothlauf, F., Squillero, G., Uyar, A., Yang, S. (Eds.), Applications of Evolutionary Computing. Vol. 4974. pp. 515–524.

Hiroyasu, T., Miki, M., Watanabe, S., 2000. The new model of parallel genetic algorithm in multi-objective optimization problems - divided range multi-objective genetic algorithm. In: Proceedings of the Congress on Evolutionary Computation (CEC) 2000.

Hwang, C.-L., Masud, A. S. M., 1979. Multiple Objective Decision Making - Methods and Applications. Vol. 164 of Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin.

Igel, C., Hansen, N., Roth, S., 2007. Covariance matrix adaptation for multi-objective optimization. Evolutionary Computation 15 (1), 1 – 28.

Kleinrock, L., 1975. Queueing Systems Volume 1: Theory. John Wiley & Sons.

Klienrock, L., 1969. UCLA press release.
URL `http://www.lk.cs.ucla.edu/LK/Bib/REPORT/press.html`

Lee, H.-J., Lee, J.-W., Lee, J.-O., 2009. Development of a web services based multidisciplinary design optimization framework. Advancess in Engineering Software 40 (3), 176–183.

McFarlane, D., Glover, K., 1990. Robust Controller Design Using Normalized Coprime Factor Plant Descriptions. Vol. 138 of Lecture Notes in Control and Information Sciences. Springer-Verlag.

Michalewicz, Z., Fogel, D. B., 2000. How to Solve It: Modern Heuristics. Springer.

Moshaiov, A., Ashram, A., 2009. Multi-objective evolution of robot neuro-controllers. In: IEEE congress on evolutionary computation (CEC) 2009. pp. 1093–1100.

Mühlenbein, H., Schlierkamp-Voosen, D., 1993. Predictive models for the breeder genetic algorithm I: Continuous parameter optimization. Evolutionary Computation 1 (1), 25–49.

Nelson, R. C., 1998. Flight Stability and Automatic Control, 2nd Edition. McGraw-Hill.

Purshouse, R. C., 2003. On the evolutionary optimisation of many objectives. Ph.D. thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK, S1 3JD.

Reynoso-Meza, G., Blasco, X., Sanchis, J., Herrero, J. M., 2013. Multiobjective evolutionary algorithms for multivariable pi controller design. Information Sciences 221, 124–141.

Reynoso-Meza, G., Sanchis, J., Blasco, X., Herrero, J. M., 2012. Comparison of design concepts in multi-criteria decision-making using level diagrams. Expert Systems with Applications 39 (9), 7895–7907.

Rivera, W., 2001. Scalable parallel genetic algorithms. Artificial Intelligence Review 16 (2), 153–168.

Rodriguez-Vazquez, K., Fonseca, C. M., Fleming, P. J., 2004. Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming. IEEE transactions on systems, man, and cybenetics - part a: systems and humans 34 (4), 531–545.

Schott, J. R., 1995. Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Massachusetts Institute of Technology.

Shenfield, A., Fleming, P. J., 2013. A novel workload allocation strategy for batch jobs. International Journal of Computing and Network Technology 1, 1–17.

Shenfield, A., Fleming, P. J., Alkarouri, M., 2007. Computational steering of a multi-objective evolutionary algorithm for engineering design. Engineering Applications of Artificial Intelligence 20 (8), 1047–1057.

Shenfield, A., Fleming, P. J., Allan, J., Kadirkamanathan, V., 2010. Optimisation of maintenance scheduling strategies on the grid. Annals of Operations Research 180 (1), 213 – 231.

Skogestad, S., Postlethwaite, I., 1996. Multivariable feedback control - Analysis and Design. John Wiley & Sons.

Starkweather, T., Whitley, D., Mathias, K., 1991. Optimization using distributed genetic algorithms. In: Schwefel, H.-P., Männer, R. (Eds.), Parallel Problem Solving from Nature 1. Vol. 496 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp. 176–185.

Stewart, P., Stone, D. A., Fleming, P. J., 2004. Design of robust fuzzy-logic control systems by multi-objective evolutionary methods with hardware in the loop. Engineering applications of artifical intelligence 70 (3), 275–284.

Tan, K. C., Li, Y., 2002. Grey-box model identification via evolutionary computing. Control engineering practice 10, 673–684.

The White Rose University Consortium, 2012. The white rose grid website. Viewed 10 January 2012.
URL `http://www.wrgrid.org.uk/`

Van Veldhuizen, D. A., 1999. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Ph.D. thesis, Airforce Institute of Technology.

W3C Working Group, February 2004. Web services architecture. Viewed 18 October 2006.
URL `http://www.w3c.org/TR/ws-arch`

Wang, L., Li, L.-P., 2011. Fixed-structure h_∞ controller synthesis based on differential evolution with level comparison. Evolutionary Computation, IEEE Transactions on 15 (1), 120–129.

Weng, W., Wang, P., Jin, X., Cao, Y., 2012. Design and application of a platform for cae-based optimization using a grid-based environment. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 226 (6), 1601–1611.

Zames, G., 1981. Feedback and optimal sensitivity: model reference transformations, multiplicitive seminorms, and approximate inverse. IEEE Transactions on Automatic Control 26 (2), 301–320.

Zhao, S.-Z., Iruthayarajan, M. W., Suganthan, P. N., 2011. Multi-objective robust pid controller tuning using two lbests multi-objective particle swarm optimisation. Information Sciences 181 (16), 3323–3335.

Zitzler, E., Thiele, L., 1998. Multiobjective optimization using evolutionary algorithms a comparative case study. In: Eiben, A. E., Bäck, T., Schoenauer, M., Schwefel, H. P. (Eds.), Parallel Problem Solving from Nature. Springer, pp. 292–301.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., da Fonseca, V. G., 2003. Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7 (2), 117–132.