

Optimisation of Maintenance Scheduling Strategies on the Grid

A. Shenfield, P. J. Fleming, G. M. Allan and V. Kadiramanathan

Abstract—The emerging paradigm of Grid Computing provides a powerful platform for the optimisation of complex computer models, such as those used to simulate real-world logistics and supply chain operations. This paper introduces a grid-based optimisation framework that provides a powerful tool for the optimisation of such computationally intensive objective functions. This framework is then used in the optimisation of maintenance scheduling strategies for fleets of aero-engines, a computationally intensive problem with a high-degree of stochastic noise.

I. INTRODUCTION

A fundamental shift in emphasis within the aero-engine manufacturing industry is leading to the adoption of power-by-the-hour contracts, where airlines make regular fixed payments to the engine manufacturers based on the hours flown by an engine and, in return, the manufacturers of the engine retain the responsibility for servicing and maintenance. As a result of this, the accurate prediction of support costs over the life-cycle of an engine is of the utmost importance. However, aero-engines operate in a highly complex and unpredictable environment, and as such it is impossible to produce a deterministic model for these support costs. Instead, stochastic simulations can be performed to provide cost estimates. It is also important for the engine manufacturers to devise maintenance scheduling strategies to minimise support costs and thus enable more competitive pricing of these contracts.

Soft Computing techniques such as Neural Networks, Fuzzy Logic, and Evolutionary Computation have been used to solve many complex real-world engineering problems. These techniques provide the engineer with a new set of tools that often out-perform conventional methods in areas where the problem domain is noisy, stochastic or ill-defined. However, in the cases of Neural Networks and Evolutionary Computation especially, these tools can be computationally intensive.

Grid Computing offers a solution to the computationally intensive nature of these techniques. The Grid Computing paradigm is an emerging field of computer science that aims to offer “a seamless, integrated computational and collaborative environment” [1]. Ian Foster defines a computational grid as “a hardware and software infrastructure that provides

dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities” [2]. Grid Computing is differentiated from conventional distributed computing by its emphasis on co-ordinated resource sharing and problem solving in dynamic, multi-institutional virtual organisations [3]. These resources include software packages, compute resources, sensor arrays, data and many others.

The purpose of this paper is to introduce a grid enabled framework for optimisation of maintenance schedules. This framework will then be used to assist decision makers in planning maintenance scheduling strategies for aero-engines. This problem presents many challenges due to its highly stochastic nature.

Section II will introduce evolutionary algorithms and give a brief overview of their application to scheduling problems. Section III will introduce Grid Computing and the core concepts used in our optimisation framework. The MEAROS simulation package used by Rolls-Royce to model the operational life-cycle of engines will be introduced in Section IV, and the simple cost model used in this study will also be outlined. Section V will describe the implementation of our framework, and then Section VI will show its application to the planning of maintenance schedules for aero-engines. Section VII will present our conclusions and outline some ideas for further work.

II. AN INTRODUCTION TO EVOLUTIONARY ALGORITHMS

A. Evolutionary Algorithms

Evolutionary Algorithms (EAs) are an optimisation technique utilising some of the mechanisms of natural selection [4]. EAs are an iterative, population based method of optimisation that are capable of both exploring the solution space of the problem and exploiting previous generations of solutions. Exploitation of the previous generation of solutions is performed by a selection operator. This operator gives preference to those solutions which have high fitness when creating the next generation of solutions to be evaluated. Exploration of the solution space is performed by a mutation operator and a recombination operator and helps to ensure the robustness of the algorithm by preventing the algorithm from getting stuck in local optima.

Evolutionary Algorithms evaluate candidate solutions based on pay-off information from the objective function, rather than derivative information or auxiliary knowledge. This ensures that EAs are applicable to many different problem domains, including those where conventional optimisation techniques (such as hill-climbing) may fail. Evolutionary Algorithms are also robust in the presence of noise due

The authors gratefully acknowledge the financial support of the DTI funded BROADEN project.

A. Shenfield, P. J. Fleming, G. M. Allan and V. Kadiramanathan are with the Rolls-Royce University Technology Centre in Control and Systems Engineering, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, S1 3JD, UK

{a.shenfield, p.fleming, jeff.allen, visakan}@sheffield.ac.uk

to their population based nature. Because EAs maintain a population of candidate solutions, each generation contains more information about the shape of the fitness landscape than would be available to conventional, non-population based methods such as hill-climbing [5].

Evolutionary Algorithms have been used to solve problems across many different disciplines. GAs have been used in such diverse fields as Economics and Social Theory [6], Robotics [7] and Art [8]. For many non-trivial real-world applications the evaluation of the objective function is performed by computer simulation of the system. For example, in the optimisation of controller parameters for gas turbine aero-engines [9], a computer model of the engine is used to calculate the values of the objective functions for a given controller design.

The use of computer simulations to evaluate the objective function leads to some new issues. To ensure that the results gained from the evolutionary algorithm are meaningful, the simulation must be complex enough to capture all the relevant dynamics of the true system. However, assuming that this level of complexity is obtainable, the simulation may be very computationally intensive. As EAs are population based methods, the simulation must be run many times. In a typical evolutionary algorithm this could involve running the simulation 10,000 times.

B. Scheduling Applications of Evolutionary Algorithms

Finding good solutions to industrial scheduling problems is of great importance, since both production rates and plant costs are dependent on work schedules. Evolutionary algorithms have had some success in solving the canonical Job-Shop Scheduling Problem [10], [11], a problem that is representative of industrial tasks ranging from assembling cars, to scheduling aircraft maintenance. Recent focus in the EC community has been on generating robust and flexible job shop schedules [12]. Other scheduling problems solved by EAs include planning maintenance for the (UK) national grid [13] and university course timetabling [14].

C. Parallel Evolutionary Algorithms

The computationally intensive nature of the evaluation process has motivated the development of parallel evolutionary algorithms. Early proposals for the implementation of parallel EAs considered two forms of parallelisation which still apply today: multiple communicating populations, and single-population master-slave implementations [15].

The decision between which of these two types of parallelisation to implement must consider several factors, such as ease of implementation and use, and the performance gained by parallelisation. Single-population parallel EAs are often the easier to implement and use, as experience gained with sequential EAs can be easily applied to these. In contrast, the implementation and use of multiple communicating populations based parallel EAs involves choosing appropriate values for additional parameters such as size and number of populations, frequency of migration, and the number of individuals involved in migration. This increases

the complexity of the parallel EA as each of these parameters affects the efficiency of the algorithm and the quality of the overall solution.

III. GRID TECHNOLOGIES

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.” [16]

However, it is only recently that technologies such as the Globus Toolkit [2] have emerged to enable this concept to be achieved. The Globus Toolkit is an open-source, community-based set of software tools to enable the aggregation of compute, data, and other resources to form computational grids. Since version 3, the Globus Toolkit has been based on the Open Grid Services Architecture (OGSA) introduced by the Globus Project. OGSA builds on current Web Service concepts and technologies to support the creation, maintenance, and application of ensembles of services maintained by virtual organisations [17].

A. Web Services

A Web Service is defined by the W3C as “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” [18]. Web Services are accessible through standards-based internet protocols such as HTTP and are enabled by three core technologies [19]:

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

These technologies work together in an application as shown in Figure 1. The Web Service client queries a UDDI registry for the desired service. This can be done by service name, service category, or other identifier. Once this service has been located the client queries 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 that conform to the XML schema found in the WSDL document.

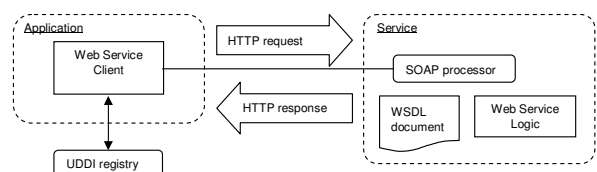


Fig. 1. Interaction between Web Service Technologies

B. Open Grid Services Architecture

The Open Grid Services Architecture forms the basis for the Globus Toolkit. OGSA represents computational resources, data resources, programs, networks and databases as services. These services utilise the Web Services technologies mentioned in Section III-A. There are three main advantages to representing these resources as services:

- 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 [17].
- 2) *It simplifies virtualisation.* Virtualisation allows for consistent resource access across multiple heterogeneous platforms by using a common interface to hide multiple implementations [17].
- 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.

IV. LIFE-CYCLE SIMULATION OF AERO-ENGINES

The Modular Engine Arisings, Repair and Overhaul Simulation (MEAROS) package was developed to enable Rolls-Royce and the Ministry of Defence to evaluate the operation, maintenance and supply of aircraft engines [20]. Although designed for the aero-engine manufacturing industry, the simulation can equally be applied to ships, land vehicles and power generation [21].

The data collected during a simulation run falls into three main categories:

- **Operations Reports** which describe the nature of the operations undergone by the engines in the simulation (such as number of hours flown).
- **Arisings Reports** which detail events that cause an engine to be taken out of service.
- **Maintenance Reports** which detail what maintenance actions were taken (such as the reconditioning or scrapping of engine modules) and the times that these occurred.

The modelling capability of the MEAROS software is extensive. The software can be used to model the operation of fleets of engines with an arbitrary number of modules [20]. Theoretically there is no limit to the size of fleets that can be modelled by the software, however in practice this is limited by the computational effort needed to model large numbers of engines.

Results produced by the simulation contain a lot of stochastic noise due to the probabilistic models used to simulate component failures. As such, the simulation has to be run multiple times and averaged to reduce the effect of this noise. Figure 2 shows that the standard deviation of the aggregate maintenance cost reduces with the number of runs of the model. It can also be seen from Figure 2 that the improvement in the standard deviation tails off significantly

after 100 passes. In practice this means that the benefit from running more than 100 passes of the model is outweighed by the additional computational cost.

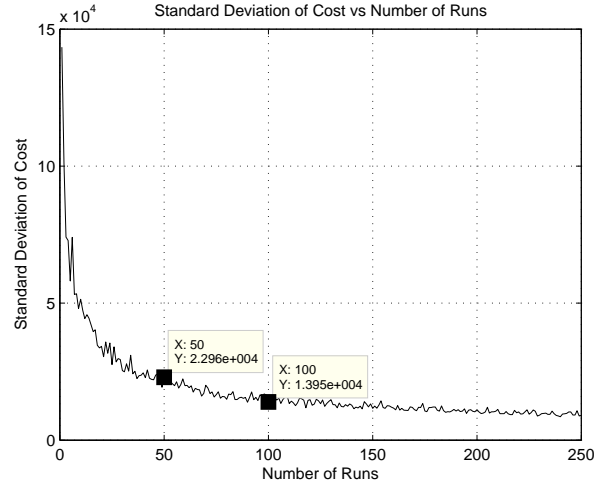


Fig. 2. Plot of the Standard Deviation of Aggregate Maintenance Cost Against Number of Runs of the Model

Originally MEAROS was used for predicting the number of spares needed to maintain a set level of operational availability. However, many of the parameters in the model are customisable (such as the failure distributions of engine modules, the stock levels, and the maintenance scheduling strategies used) and can therefore be optimised with respect to some objective (for instance the support costs or the operational availability).

In this study we have chosen to use the maintenance scheduling strategy in our optimisation because this is one of the few easily modifiable parameters affecting operational availability and support costs once an engine has gone in to service. Maintenance in the simulation is performed after an arising occurs. The main causes of arisings are either the expiry of a *hard-life*¹ or an in-service failure such as foreign object damage [20].

Once an arising occurs, the engine must be removed from the aircraft wing and the module that caused the arising must be reconditioned or replaced. However, as the removal of the engine from the wing is one of the most expensive parts of a typical maintenance shop visit, this provides the ground crew with the chance to perform opportunistic maintenance on the other modules in the engine. If one of the other modules in the engine has exceeded its *soft-life*² then it should also be reconditioned or replaced whilst the engine is removed from the wing. [22] have shown that, for relatively small engine module costs, there is likely to be an optimum value of soft-life which minimises the maintenance cost of an engine. Soft-lives that are too low result in engine modules being reconditioned or replaced during every maintenance shop

¹Hard-lives are usually assigned to safety critical components and represent the age at which that component must be replaced.

²Soft-lives represent the age whereby a component should be replaced at the next opportunity.

visit, whilst soft-lives that are too high result in cheaper but more frequent shop visits.

Table I shows a simple cost model developed in conjunction with Rolls-Royce for a hypothetical five module aero-engine. The costs given represent the costs of removal and reconditioning of the modules in the engine.

TABLE I
COST MODEL AND WEIBULL DISTRIBUTION PARAMETERS

Engine	Cost	Scale	Slope
Engine	3000	N/A	N/A
Module 1	200	1000	1
Module 2	1000	800	2.5
Module 3	900	700	3
Module 4	800	2000	2
Module 5	1200	1500	1.5

V. IMPLEMENTATION OF A GRID BASED OPTIMISATION FRAMEWORK

A. Parallelisation of the Evolutionary Algorithm

In section II-C it was found that there are two types of possible parallelisation strategies for evolutionary algorithms: multiple communicating populations, and single-population master-slave implementations. In the implementation of our grid-enabled framework for optimisation using evolutionary algorithms we have decided to parallelise our evolutionary algorithm using the single-population master-slave implementation. This is also known as distributed fitness evaluation or global parallelisation. This model uses the master-worker paradigm (see Fig. 3) from parallel computing.

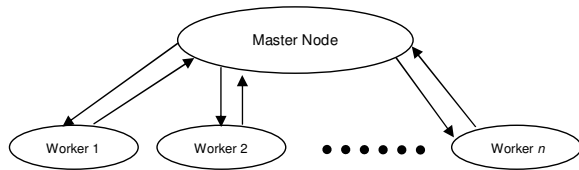


Fig. 3. The Master-Worker Programming Paradigm

A master-slave parallel evolutionary algorithm uses a single population maintained globally by the master node and parallelises the evaluation of the objective function by distributing the population amongst the worker processes. These are then assigned to the available processors for execution (in the ideal case, one individual per processor). The evolutionary operators - selection, recombination and mutation - are then applied globally by the master node to form the next generation of the population.

This model is particularly well suited for the parallelisation of EAs as the evaluation of the objective function requires only the knowledge of the candidate solution to be evaluated, and therefore there is no need for inter-communication between worker processes. Communication only occurs when the individuals are sent to the worker processes for evaluation

and when the results of those evaluations are returned to the master node.

B. Service-Oriented Architecture

We have chosen to implement our grid-enabled framework for optimisation using evolutionary algorithms in a Service-Oriented Architecture (SOA). We have implemented the framework using the Java programming language, primarily due to the portability of the code. This means that the components of the framework can easily be run across various heterogeneous platforms.

A service-oriented architecture is essentially a collection of services that communicate with each other in order to perform a complex task. SOA is an approach to building loosely-coupled, distributed systems that combine services to provide functionality to an application. IBM sees SOA as key to interoperability and flexibility requirements for its vision of an on demand business [23].

The SOA approach to grid computing is well suited to the kind of master-worker parallelism used in our optimisation framework. This service-oriented architecture view of grid computing has the client acting as the master node, and the service acting as the worker. In the implementation of the optimisation framework (see Fig. 4) there are two different types of service. One service type exposes the operations of the evolutionary algorithm to the client, and the other provides the ability to run evaluations of the objective function on the resources of the computational grid.

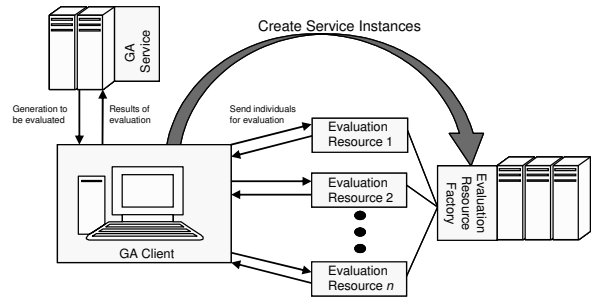


Fig. 4. The Implementation of the Optimisation Framework

This SOA approach also provides flexibility both in how the optimisation framework is used and in the maintenance of the framework. The provision of the components of the framework as services means that it is simple to add new functionality to the system, and to improve upon existing functionality, by adding new services. In the context of our optimisation framework, this functionality could be anything from the implementation of additional evolutionary operators, to the distribution and management of the objective function evaluation.

Providing the optimisation framework as services also means that the functionality can be accessed via the HTTP protocol. This means that the services can be easily integrated into an Internet portal so as to be accessible by any device with a capable web browser (such as a PDA).

This SOA approach is used in providing access to grid resources via the Globus Toolkit (see section III-B). The Globus Toolkit has become a fundamental enabling technology for grid computation, letting people carry out computations across geographically distributed resources in a secure way. The success of the Globus Project has meant that the project has become one of the driving forces in developing standards for grid computing.

C. The White Rose Grid

The White Rose Grid [24] is a multi-institutional computational grid launched in 2002 by the universities of Sheffield, York and Leeds. The main objective of the White Rose Grid project is to support e-Research by providing users with access to large amounts of heterogeneous compute resources. The White Rose Grid currently consists of five high-performance compute nodes located at three different sites (see Figure 5), and in 2003 was awarded the status of e-Science Centre of Excellence.

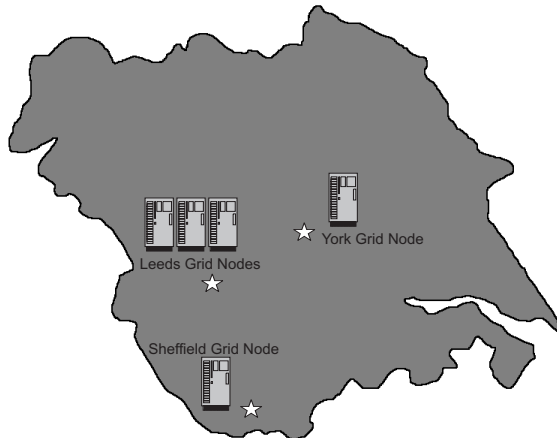


Fig. 5. The Topology of the White Rose Grid

The three participating institutions in the White Rose Grid consortium reserve 75% of their grid resources for users within their institution and allocate the remaining 25% to other users of the grid. These institutions are connected by the high bandwidth Yorkshire and Humberside Metropolitan Area Network (YHMAN).

VI. RESULTS

We have used our grid-enabled optimisation framework for the optimisation of maintenance scheduling strategies across a fleet of aero-engines. As mentioned earlier, we chose to modify the maintenance scheduling strategy in the optimisation process because it is one of the few parameters affecting the support costs and operational availability that is easily changeable. It is inexpensive to vary when compared to post-production design changes, and can be quickly implemented across an engine range [25].

The evaluation of candidate solutions in our optimisation routine was performed using the MEAROS engine life-cycle

model (see Section IV) in conjunction with the simple cost model shown in Table I. The MEAROS model was used to simulate the operation of a fleet of 25 engines over a 10 year period, and was averaged over 100 passes of the model to reduce the stochastic noise in the simulation (see Figure 2). We used a population size of 50 individuals in the EA, and the optimisation results were verified by running the evolutionary algorithm multiple times.

Figure 6 shows that the mean value of the population (the solid line in the figure) exhibits convergence after around 15-20 generations. It can also be seen from Figure 6 that the diversity of the population (each individual is shown by a dot in the figure) is significantly reduced as the search progresses.

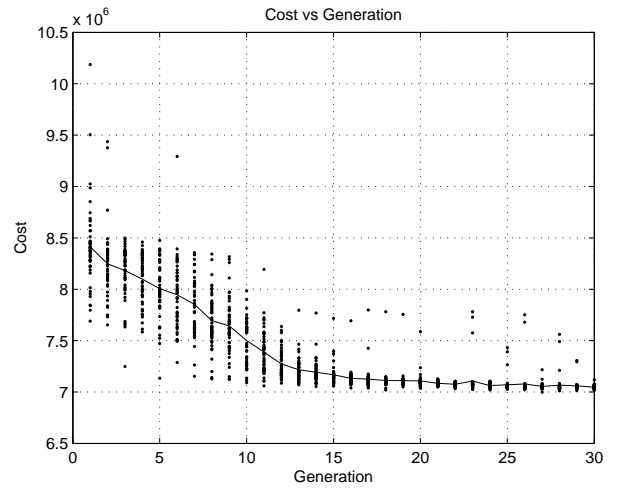


Fig. 6. Plot of Cost of Maintenance Scheduling Strategy Against Number of Generations

Our grid-based optimisation framework uses the resources of the White Rose Grid (see Section V-C) to perform the distributed evaluation of candidate solutions. Table II shows a representative set of total execution times from the optimisation of maintenance scheduling strategies for our aero-engine problem, running for 30 and 50 generations respectively. As Table II shows, the use of our grid-based optimisation framework has considerably reduced the time taken to optimise our aero-engine maintenance strategy problem.

TABLE II
EXECUTION TIMES FOR THE OPTIMISATION OF MAINTENANCE SCHEDULING STRATEGIES

Local Machine		Computational Grid	
30 generations	50 generations	30 generations	50 generations
1974 seconds	3358 seconds	873 seconds	1423 seconds

VII. CONCLUSIONS AND FURTHER WORK

Table II has shown that significant reductions in the execution times of optimisation routines can be achieved

by using our grid-based optimisation framework. Whilst the implementation of the framework described in this paper has concentrated on the application of an evolutionary algorithm to a single objective maintenance problem, our framework is easily extensible (due to the use of a service-oriented architecture approach) to both multi-objective problems and to the implementation of alternative optimisation methods such as ant-colony optimisation or particle swarm optimisation. Further work is planned to extend this optimisation framework to perform multi-objective optimisation of schedules for complex logistic and supply chain operations.

The grid-based framework described in this paper is best suited to computationally expensive objective function evaluations, such as the one described in this paper. This is due to the communication overheads involved in executing the objective function evaluations in a distributed manner, and for some computationally trivial objective functions this may result in a degradation in performance compared with a sequential EA run on a single machine. These overheads are due to the way in which job submission and management is performed in a grid computing environment. Whilst further work will be conducted into determining the scale of problems for which this framework is most effective, it is expected that further research and development of grid-middleware, job submission services, and job management services will provide a reduction in these communication overheads. This will allow our framework to provide increased performance for less computationally intensive problems. However, this framework is not intended to replace sequential EAs in cases where the performance of the sequential EA is satisfactory.

VIII. ACKNOWLEDGMENT

The authors gratefully acknowledge the input from the engineers at Rolls-Royce PLC and Data Systems & Solutions.

REFERENCES

- [1] M. Baker, R. Buyya, and D. Laforenza, "Grid and grid technologies for wide-area distributed computing," *Software: Practice and Experience*, vol. 32, no. 15, pp. 1437 – 1466, 2002.
- [2] I. Foster and C. Kesselman, "The Globus Toolkit," in *The GRID: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds. Morgan Kaufmann, 1999, ch. 11, pp. 259 – 278.
- [3] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Journal of Supercomputer Applications*, vol. 15, no. 3, pp. 200 – 222, 2001.
- [4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA.: Addison-Wesley, 1989.
- [5] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. Berlin: Springer, 2000.
- [6] R. Axelrod, "The evolution of strategies in the iterated prisoner's dilemma," in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. Morgan Kaufmann, 1987, pp. 32 – 41.
- [7] D. Pratihari, K. Deb, and A. Ghosh, "A genetic-fuzzy approach for mobile robot navigation amongst moving obstacles," *International Journal of Approximate Reasoning*, vol. 20, no. 2, pp. 145 – 172, 1999.
- [8] K. Sims, "Artificial evolution for computer graphics," *Computer Graphics*, vol. 25, no. 4, pp. 319 – 328, 1991.
- [9] P. J. Fleming, R. C. Purshouse, A. J. Chipperfield, I. A. Griffin, and H. A. Thompson, "Control systems desing with multiple objectives: An evolutionary computing approach," in *Workshops of the 15th IFAC World Congress*, 2002.
- [10] L. Davis, "Job shop scheduling with genetic algorithms," in *Proceedings of the First International Conference on Genetic Algorithms*, J. J. Grefenstette, Ed. New Jersey: Lawrence Erlbaum Associates, 1985, pp. 136 – 140.
- [11] K. Mesghouni, S. Hammadi, and P. Borne, "Evolutionary algorithms for job-shop scheduling," *International Journal of Applied Mathematics and Computer Science*, vol. 14, no. 1, pp. 91 – 103, 2004.
- [12] M. T. Jensen, "Generating robust and flexible job shop schedules using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 275 – 288, 2003.
- [13] W. B. Langdon, "Scheduling planned maintenance of the national grid," in *Evolutionary Computing - AISB Workshop*, ser. Lecture Notes in Computer Science, T. C. Fogarty, Ed., vol. 993. Berlin: Springer-Verlag, April 1995, pp. 132 – 153.
- [14] R. Lewis and B. Paechter, "Application of the grouping genetic algorithm to university course timetabling," in *Proceedings of the Fifth European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP)*, ser. Lecture Notes in Computer Science, G. R. Raidl and J. Gottlieb, Eds., vol. 3448. Berlin: Springer-Verlag, 2005, pp. 144 – 153.
- [15] E. Cant' u-Paz and D. E. Goldberg, "On the scalability of parallel genetic algorithms," *Evolutionary Computation*, vol. 7, no. 4, pp. 429 – 449, 1999.
- [16] L. Klienrock, "UCLA press release," 1969. [Online]. Available: <http://www.lk.cs.ucla.edu/LK/Bib/REPORT/press.html>
- [17] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid services for distributed system integration," *IEEE Computer*, vol. 35, no. 6, pp. 37 – 46, 2002.
- [18] W. W. Group, "Web services architecture," February 2004, viewed 18 October 2006. [Online]. Available: <http://www.w3c.org/TR/ws-arch>
- [19] D. A. Chappell and T. Jewell, *Java Web Services*. O'Reilly, 2002.
- [20] "Mearos model description version 8.31," Rolls-Royce Internal Document, 2002.
- [21] J. P. M. Argyle, "Optimisation of operational cost with application to an aerospace engine system," Ph.D. dissertation, University of Sheffield, 2006.
- [22] J. Crocker and U. D. Kumar, "Age-related maintenance versus reliability centred maintenance: A case study on aero-engines," *Reliability Engineering and Systems Safety*, vol. 67, pp. 113 – 118, 2000.
- [23] M. Colan, "Service oriented architecture expands the vision of web services: Part 1," IBM, "DeveloperWorks paper, 2004, viewed 18 October 2006. [Online]. Available: <http://www-128.ibm.com/developerworks/library/ws-soaintro.html>
- [24] "White rose grid website," viewed 18 October 2006. [Online]. Available: <http://www.wrgrid.org.uk>
- [25] J. P. M. Argyle and J. Tubby, "Integrated logistics support optimisation," Rolls-Royce PLC, Tech. Rep. RRUTC/Shef/R/02006, 2002.