A Service Oriented Architecture for Decision Making in Engineering Design

Alex Shenfield and Peter J. Fleming

Department of Automatic Control and Systems Engineering,
University of Sheffield,
Mappin Street, Sheffield, S1 3JD, United Kingdom
A.Shenfield@sheffield.ac.uk

Abstract. Decision making in engineering design can be effectively addressed by using genetic algorithms to solve multi-objective problems. These multi-objective genetic algorithms (MOGAs) are well suited to implementation in a Service Oriented Architecture. Often the evaluation process of the MOGA is compute-intensive due to the use of a complex computer model to represent the real-world system. The emerging paradigm of Grid Computing offers a potential solution to the compute-intensive nature of this objective function evaluation, by allowing access to large amounts of compute resources in a distributed manner. This paper presents a grid-enabled framework for multi-objective optimisation using genetic algorithms (MOGA-G) to aid decision making in engineering design.

1 Introduction

Soft Computing techniques such as Neural Networks, Fuzzy Logic, and Evolutionary Computation are 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 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 coordinated 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 multi-objective optimisation using genetic algorithms (MOGA-G). This framework will

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 334–343, 2005.

be presented in the context of a Service Oriented Architecture approach. This approach ties in with that taken by the Globus Project [4] to providing access to grid resources via Grid Services. Section 2 will introduce Genetic Algorithms and Multi-Objective Optimisation. Section 3 will briefly introduce the core grid concepts used in the implementation of our framework. Section 4 will outline other related work. Section 5 will provide details of the implementation of our framework, and Section 6 will draw some conclusions and present some ideas for further work.

2 Genetic Algorithms and Multi-objective Optimisation

2.1 Genetic Algorithms

Genetic Algorithms (GAs) are an optimisation technique utilising some of the mechanisms of natural selection [5]. GAs 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.

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

2.2 Multi-objective Optimisation

Many real-world engineering design problems involve the satisfaction of multiple conflicting objectives. In this case it is unlikely that a single ideal solution will be possible. Instead, the solution of a multi-objective optimisation problem will lead to a family of Pareto optimal points, where any improvement in one objective will result in the degradation of one or more of the other objectives.

Genetic Algorithms are particularly well suited to this kind of multi-objective optimisation, because they search a population of candidate solutions. This enables the GA to find multiple solutions which form the Pareto optimal set (see Fig. 1). GAs are often able to find superior solutions to real-world problems than conventional optimisation techniques (i.e. constraint satisfaction). This is due to the difficulty that conventional optimisation techniques have when searching in the noisy or discontinuous solution spaces that real-world problems often have.

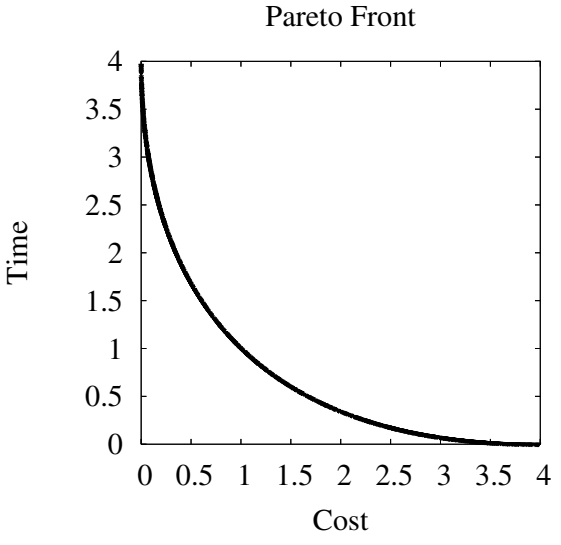

Fig. 1. The Pareto Optimal Solution Set

2.3 Applications of Genetic Algorithms

Genetic 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 genetic 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 genetic algorithms are population based methods, the simulation must be run many times. In a typical genetic algorithm this could involve running the simulation 10,000 times.

2.4 Parallel Genetic Algorithms

The computationally intensive nature of the evaluation process has motivated the development of parallel genetic algorithms. Early proposals for the implementation of parallel GAs considered two forms of parallelisation which still apply today: multiple communicating populations, and single-population master-slave implementations [10].

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 GAs are often the easier to implement and use, as experience gained with sequential GAs can be easily applied to these. In contrast, the implementation and use of multiple communicating populations based parallel GAs 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 GA as each of these parameters affects the efficiency of the algorithm and the quality of the overall solution.

3 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." [11]

However, it is only recently that technologies such as the Globus Toolkit 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 of the Globus Toolkit it 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 [12].

3.1 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" [13]. Web Services are accessible through standards-based internet protocols such as HTTP and are enabled by three core technologies [14]:

- 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 Fig. 2. 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

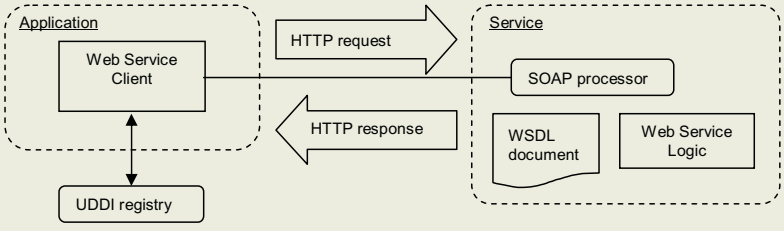

Fig. 2. Interaction between Web Service Technologies

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.

3.2 Open Grid Services Architecture

The Open Grid Services Architecture (OGSA) is the basis for the Globus Toolkit version 3. OGSA represents computational resources, data resources, programs, networks and databases as services. These services utilise the Web Services technologies mentioned in Section 3.1. 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 [12].
2. *It simplifies virtualisation.* Virtualisation allows for consistent resource access across multiple heterogeneous platforms by using a common interface to hide multiple implementations [12].
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 Related Work

In recent years the interest in using parallel genetic algorithms to solve single-objective optimisation problems has increased considerably [15]. However, there has been little research performed in applying parallel GAs to solve multi-objective optimisation problems. In [16] and [17] there is some discussion concerning multi-objective evolutionary optimisation techniques in distributed systems, but these do not implement parallel GAs in a Grid Computing environment.

A middleware system for evolutionary computation in a Grid Computing environment is proposed in [18], and then used to construct a parallel simulated annealing algorithm to solve a single objective problem. This system requires the application developer to implement a set of interfaces (comprising the middleware) and write the code for the desired evolutionary operations. Another paper that utilizes the Grid Computing concept for single-objective optimisation using genetic algorithms is [19]. This paper develops a 'Black Box Optimisation Framework' (BBOF) in C++ to optimise a computer simulation of a forest fire propagation problem from environmental science. This BBOF is executed in a Condor pool to harness the spare CPU cycles of a cluster of computers.

Our MOGA-G system differs from those proposed in [18] and [19] because it provides a concrete implementation of a *multi-objective genetic algorithm*. Like [19] we have utilised the power of computational grids to perform distributed fitness evaluation of our objectives, but we have implemented our framework in a Service Oriented Architecture using the Globus Toolkit to provide access to the resources of the grid (see section 5.2).

The power of computational grids is used to execute a distributed enumerative search in [20]. This distributed enumerative search is then used to generate the Pareto-optimal front for several benchmark test functions that are commonly used in the evaluation of the performance of multi-objective optimisation algorithms. A brief comparison with heuristic techniques is then performed.

This MOGA-G system is more computationally efficient than the distributed enumerative search described in [20]. This is because our algorithm converges on the Pareto optimal front by making intelligent choices about which points to search in the next generation, whereas the enumerative search algorithm has to evaluate every point in the search space. This approach would be impossible for a real-world engineering design problem due to the potential size of the search space.

5 Implementation

5.1 Parallelisation of the Multi-objective Genetic Algorithm

In section 2.4 we found that there are two types of possible parallelisation strategies for genetic algorithms: multiple communicating populations, and single-population master-slave implementations. In the implementation of our grid-enabled framework for multi-objective optimisation using genetic algorithms (MOGA-G) we have decided to parallelise our multi-objective genetic 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) of parallel programming.

A master-slave parallel genetic algorithm uses a single population maintained globally by the master node and parallelises the evaluation of the objective function by distributing the population to the worker processes. These are then assigned to the available processors for execution (in the ideal case, one individual

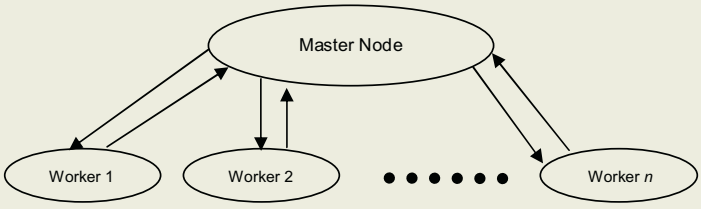

Fig. 3. The Master-Worker Programming Paradigm

per processor). The genetic operators - selection, recombination and mutation - are then applied globally by the master node to form the next generation.

This model is particularly well suited for the parallelisation of genetic algorithms as the evaluation of the objective function requires only the knowledge of the candidate solution to evaluate, 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.

5.2 Service-Oriented Architecture and the Globus Toolkit Version 3

We have chosen to implement our grid-enabled framework for multi-objective optimisation using genetic algorithms in a Service-Oriented Architecture (SOA) using the Globus Toolkit version 3 to provide access to the resources of the grid. We have implemented the MOGA-G framework using the Java programming language, primarily due to the portability of the code. This means that the components of the MOGA-G 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 [21].

The SOA approach to grid computing is well suited to the kind of master-worker parallelism used in the MOGA-G framework. This SOA view of grid computing has the client acting as the master node, and the service acting as the worker. In the implementation of the MOGA-G framework (see Fig. 4) there are two different services. One service exposes the operations of the multi-objective genetic algorithm to the client, and the other provides operations for running evaluations of the objective function on the computational grid.

This SOA approach also provides flexibility both in how the MOGA-G framework is used and in the maintenance of the framework. The provision of the components of the MOGA-G framework as services means that it is simple to

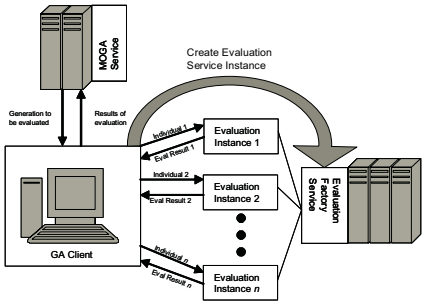

Fig. 4. The Implementation of the MOGA-G Framework

add new functionality to the system, and to improve upon existing functionality, by adding new services. In the context of the MOGA-G framework, this functionality could be anything from the implementation of the genetic algorithm operators - selection, recombination and mutation - to the distribution and management of the objective function evaluation.

Providing the MOGA-G 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 3). 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.

6 Conclusions and Further Work

This paper has described a grid-enabled framework for multi-objective optimisation using genetic algorithms (MOGA-G). This MOGA-G framework has been designed in a Service-Oriented Architecture (SOA) so as to take advantage of the flexibility that this architecture offers. In the MOGA-G framework a concrete implementation of a multi-objective genetic algorithm is provided. However, the SOA approach that we have taken allows our implementation to be easily extended to provide additional features, such as those required to construct hybrid genetic algorithms. Extending the MOGA-G framework to support additional features is an area for further investigation.

This framework is primarily suited to computationally expensive objective function evaluations, such as those performed by computer simulation, due to its distributed nature. For computationally trivial objective functions the communication overheads involved in executing the evaluations in a distributed manner

result in a decrease in performance compared to a sequential GA. This is due to the way in which job submission and management is performed. 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 GAs in cases where the performance of the sequential GA is satisfactory.

References

1. Baker, M., Buyya, R., and Laforenza, D., Grids and Grid technologies for wide-area distributed computing, Software: Practice and Experience, **32(15)**, pp. 1437–1466, 2002.
2. Foster, I., and Kesselman, C. (eds.), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999.
3. Foster, I., Kesselman, C., and Tuecke, S., The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Int. J. Supercomputer Applications, **15(3)**, 2001.
4. The Globus Project; www.globus.org
5. Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1999.
6. Axelrod, R., The evolution of strategies in the Iterated Prisoners Dilemma, in Genetic Algorithms and Simulated Annealing (L. Davies ed.), Morgan Kaufmann, pp. 32–41, 1987.
7. Pratihar, D., Deb, K., and Ghosh, A., A genetic-fuzzy approach for mobile robot navigation among moving obstacles, Int. J. Approximate Reasoning, **20(2)**, pp. 145–172, 1999.
8. Sims, K., Artificial Evolution for Computer Graphics, Computer Graphics (Proc. SIGGRAPH '91), **25(4)**, pp. 319–328, 1991.
9. Fleming, P. J., Purshouse, R. C., Chipperfield, A. J., Griffin, I. A., and Thompson, H. A., Control Systems Design with Multiple Objectives: An Evolutionary Computing Approach, Workshop in the 15th IFAC World Congress, Barcalona, 2002.
10. Cantú-Paz, E., and Goldberg, D. E., On the Scalability of Parallel Genetic Algorithms, Evolutionary Computation, **7(4)**, pp. 429–449, 1999.
11. Kleinroack, L., UCLA Press Release, July 3rd 1969.
12. Foster, I., Kesselman, C., Nick, J. M., and Tuecke, S., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Open Grid Services Infrastructure WG, Global Grid Forum, June 22nd 2002.
13. Web Services Architecture, www.w3c.org/TR/ws-arch, W3C Working Group Note February 11th 2004.
14. Chappell, D. A., and Jewell, T., Java Web Services, O'Reilly, 2002.
15. Alander, J. T., Indexed Bibliography of Distributed Genetic Algorithms Technical Report 94-1-PARA, University of Vaasa, 2003.
16. Deb, K., Zope, P., and Jain, A., Distributed Computing of Pareto-Optimal Solutions with Evolutionary Algorithms, Proc. EMO 2003, pp. 534–549, Springer-Verlag, 2003.

17. Van Veldhuizen, D. A., Zydallis, J. B., and Lamont, G. B., Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms, IEEE Trans. on Evolutionary Computation, **7(2)**, pp. 144–173, 2003.
18. Tanimura, Y., Hiroyasu, T., Miki, M., and Aoi, K., The System for Evolutionary Computing on the Computational Grid, Proc. IASTED 14th Intl. Conf. on Parallel and Distributed Computing and Systems, pp. 39–44, ACTA Press, 2002.
19. Abdalhaq, B., Cortes, A., Margalef, T., and Luque, E., Evolutionary Optimization Techniques on Computational Grids, Proc. ICCS 2002, pp. 513–522, Springer-Verlag, 2002.
20. Luna, F., Nebro, A. J., and Alba, E., A Globus-Based Distributed Enumerative Search Algorithm for Multi-Objective Optimization Technical Report LCC 2004/02, University of Malaga, 2004.
21. Colan, M., Service Oriented Architecture expands the vision of Web Services: part 1, IBM developerWorks paper, 2004.