Comparing genetic algorithms and particle swarm optimisation for an inverse problem exercise

C. R. Mouser[*] S. A. Dunn[*]

(Received 26 October 2004, revised 1 February 2005)

Abstract

We describe the performance of two population based search algorithms (genetic algorithms and particle swarm optimisation) when applied to the optimisation of a structural dynamics model. A significant difficulty arises when trying to compare the performance of such algorithms. For the two algorithms to perform at their best, several properties (for example, population size and mutation rate) need to be set. The performance of the algorithms can be highly sensitive to the choice of these parameters, and the optimisation of these leads to a search in a multi-dimensional space. This work describes how a genetic algorithm optimises the properties of a genetic algorithm and a particle swarm optimisation in order to produce algorithms

[*]Defence Science & Technology Organisation, Melbourne, Australia. mailto:carl.mouser@dsto.defence.gov.au

See http://anziamj.austms.org.au/V46/CTAC2004/Mous for this article, © Austral. Mathematical Soc. 2005. Published March 21, 2005. ISSN 1446-8735

that are optimally tuned to the particular problem being solved. The two methods are rigorously compared. This problem is implemented on a distributed computing facility spread across the Defence Science & Technology Organisation's network across four cities in south-east Australia.

Contents

1 Introduction

There is a long history of development and updating of dynamic finite element models based on measured experimental data. These models are created at the time of design of the aircraft and a significant amount of time is expended in developing them to the fidelity required. Because these models are used in assessing the flight-worthiness of the aircraft, a great importance is placed on them. Ground Vibration Tests (GVTs) are then conducted on the aircraft when it is built and the model is validated against experiment. Typically, the agreement between the model and reality is poor. A process of updating the model is then embarked upon so as to make the model a better predictor of the actual dynamic response of the aircraft. Typically, this involves the adjustment of the physical parameters in the model (such as moduli of elasticity and joint stiffness representation). This process still leaves all the assumptions that were inherent in the modelling process and the final model is typically a compromise. Also, the number of parameters that can be adjusted are typically significantly larger than the information contained in the experimental data.

Alternatively, the finite element model can be used as a starting point to produce mass and stiffness matrices that approximate the structure. Individual elements of these matrices are altered to create a model that most accurately represents the structure under consideration. This process suffers from a loss of physical causality between the model and the structure and can lead to such physical impossibilities as negative mass. It is difficult to have a very high degree of confidence in the predictions of such a model following physical modifications to a structure

Another approach, that considered here, is to build a model based on measured experimental data [1]. A relatively simple model is constructed based on the physics of the structure. This is constructed from basic finite elements such as lumped masses, beams that can support bending and torsion, springs and dampers. The properties of all these elements are assumed to be unknown and an optimisation problem is constructed to determine the values of the model parameters based on the measured data. Even though greatly simplified, the model complexity is typically such that the number of unknown parameters are in the hundreds. Traditional optimisation routines have great difficulty in solving problems of this magnitude. However, the

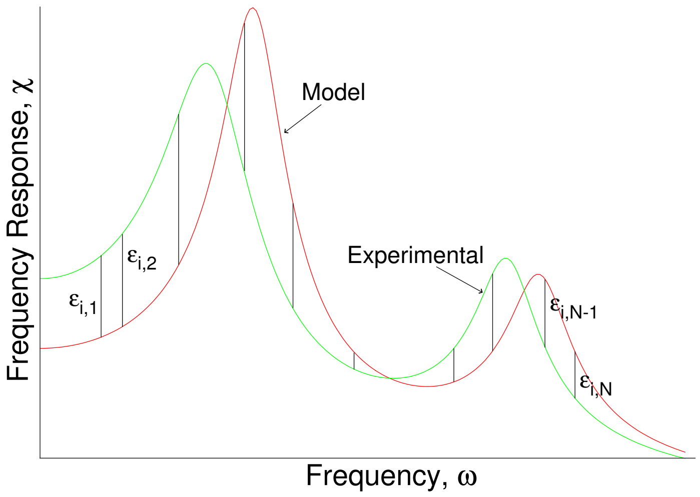

Figure 1: Sample frequency response function

field of evolutionary optimisation has given rise to a new class of optimisers to apply to such a problem. This paper will discuss two such optimisation routines, namely genetic algorithms and particle swarm optimisation.

2 Structural dynamics model

To investigate the suitability of using genetic algorithms or particle swarm optimisation for solving the class of problems being discussed here, a conceptually simple problem was contrived. This consists of a 10 degree-of-freedom mass/spring/damper system, for which the values of mass, stiffness and damping are known. From this system, theoretical frequency response functions were calculated to simulate measured experimental data. We then assume that the parameters (mass, stiffness, damping) of the model are unknown. Hence we construct an optimisation problem to minimise the difference between the "measured" frequency response functions and the calculated frequency response functions. To control the difficulty of the problem, the number of unknown masses, springs and dampers is varied to give between two and thirty unknowns. Figure 1 gives an example frequency response function, showing both the measured and experimental curves.

Based on Figure 1, a cost function is developed. Given a vector of unknowns, η, representing the unknown physical properties of the system, the objective is to minimise the error between the model predictions, $\chi(\eta)$ and the measurements $\chi'(\eta)$ for the n freedoms at the N selected comparison frequencies:

$$\min\left(\epsilon\left(\eta\right)\right) = \sum_{j=1}^{N}\sum_{i=1}^{n}\left|\chi_{i,j}\left(\eta\right) - \chi^{'}_{i,j}\left(\eta\right)\right|. \tag{1}$$

The model predictions are given by the solution to the linear equation

$$\left(-\omega^2\left[M\left(\eta\right)\right] + i\omega\left[c\left(\eta\right)\right] + \left[K\left(\eta\right)\right]\right)\chi\left(\eta\right) = \phi\,, \tag{2}$$

where M is the mass matrix, K is the stiffness matrix, c is the damping matrix and ϕ is a vector of amplitudes of a sinusoidal input force at frequency ω. Note that M, K and c are functions of the unknowns, η. This implies that the structure of these matrices is constrained by the physics of the original mass/spring/damping system matrix structure and within the optimisation procedure this structure is maintained. This ensures that the final model, while it may not be the best mathematical model for the data, is the best model for the data without compromising the physics of the problem. Consequently, the model maintains its predictive capability.

3 Optimising the structural dynamics model

Based on the structural dynamics model given above, the aim of this work is to compare the performance of two optimisation algorithms, when applied to this problem. The difficulty with the application of the optimisation algorithms is that they need to be tuned. For each algorithm there are several parameters to vary to control the algorithm, the settings of these parameters can have a significant impact on the performance of the optimisation routine. If the parameters are set poorly, the algorithm will either converge slowly, or may not converge at all. There are general guidelines given as to how these parameters should be set, but experience has shown that the optimal value of these parameters is very problem specific.

The basis for the comparison between genetic algorithms and the particle swarm optimisation is the number of cost function evaluations required to get the solution within a specified tolerance. A cost function calculates the frequency response function from the generated structural dynamic model and compares this to the simulated-measured frequency response function, returning a fitness value for the solution.

3.1 Genetic algorithm

Genetic algorithms are applied to structural dynamic problems at DSTO [1]. Briefly, the genetic algorithm consists of a population of individuals, represented by a binary string, that encode the unknowns of the optimisation problem. At each generation of the solution process, individuals are selected to breed and produce a new population of individuals. For this application, tournament selection, with uniform crossover and mutation was used. A niching strategy has also been adopted, whereby after a certain number of generations have evolved, the best solution(s) found are used to form a new population of individuals, along with randomly chosen individuals. The number of individuals to carry through, along with the number of generations to evolve before restarting are parameters that can be varied to control the performance of the algorithm.

3.2 Particle Swarm Optimisation

Particle Swarm optimisation is a relatively new optimisation technique [2]. It is inspired by the swarming behaviour of birds and insects. A population of particles exists in the n-dimensional search space that the optimisation problem lives in. Each particle has a certain amount of knowledge, and will move about the search space based on this knowledge. The particle has some inertia attributed to it and so will continue to have a component of motion in the direction it is moving. It also knows where in the search space the best solution it has encountered is, and finally it knows where the best solution any of the other particles has found is. The particle will then modify its direction such that it has additional components towards its own best position and towards the overall best position. This will provide some form of convergence to the search, while providing a degree of randomness to promote a wide coverage of the search space. For the ith particle, in the dth dimension, the algorithm is

$$\begin{aligned} v_{i,d}\left(n+1\right) &= wv_{i,d}\left(n\right) + c_1 r_{1,d}\left(p_{i,d} - x_{i,d}\left(n\right)\right) + c_2 r_{2,d}\left(p_{g,d} - x_{i,d}\left(n\right)\right) \,, \\ x_{i,d}\left(n+1\right) &= x_{i,d}\left(n\right) + v_{i,d}\left(n+1\right) \,, \end{aligned} \tag{3}$$

where, $v_{i,d}$ represents the velocity of the particle at time step $(n+1)$ and $x_{i,d}$ represents the position of the particle. The initial position and velocity of the particles are chosen randomly. The values of w, c_1 and c_2 are fixed for a given run of the particle swarm optimisation and are set by the outer genetic algorithm. $r_{1,d}$ and $r_{2,d}$ are uniformly distributed random numbers. $p_{i,d}$ and $p_{g,d}$ are the best position the ith particle has currently found and the best position found by any particle respectively.

4 Optimising the optimiser

In order to accurately compare the performance of the two optimisation algorithms, the algorithms have to be performing optimally. In order to determine the optimum properties for the optimisation routines, a genetic algorithm is used here to optimise the optimiser. This outer genetic algorithm attempts to find the inner genetic algorithm or particle swarm optimisation properties that will optimise the structural dynamics model in the minimum time (as measured by the number of cost function evaluations). The properties of the outer genetic algorithm are fixed. The properties being optimised by this outer genetic algorithm are:

Genetic algorithm	*Particle Swarm Optimisation*
• population size	• number of particles
• total number of GA runs	• inertia (w)
• proportion niched	• own-best weighting (c_1)
• pre-niche mutation rate	• global-best weighting (c_2)
• post-niche mutation rate	• maximum allowable $v_{i,d}$

For the outer genetic algorithm to operate, a cost function needs to be defined. This cost function needs to be able to return a measure of the success of the optimisation routine applied to the structural dynamics model. To be considered successful, an optimisation routine needs to meet several criteria:

1. It must be able to find an acceptable solution;
2. It should do this as quickly as possible (that is, with the minimum number of cost function evaluations); and
3. For any given run, it should have a high probability of achieving the first two goals.

This last point is an important consideration. Due to the non-deterministic nature of the optimisation routines (that is, they have a random starting point, and their evolution has an element of randomness present) one set of parameters for an optimiser may find a solution only occasionally. Such a set of parameters cannot be considered to define a good optimiser as the likelihood of the optimiser converging is too low. The defined cost function needs to incorporate all of these criteria.

For each run of the inner optimisers, a fixed number of cost function evaluations is performed (N_{cfe}). For each set of parameters, the inner optimiser is run ν_{run} times. This aids in providing a measure of the repeatability of the optimiser. The number of runs with an acceptable cost function (C_{acc}) must be greater than $\nu_{\text{run}}/2$, otherwise a strong penalty is applied. This ensures that optimisers that are not repeatable are punished and the outer genetic algorithm should move away from these solutions. Based on these considerations, the cost function for the outer genetic algorithm is described as follows:

For a given set of genetic algorithm or particle swarm optimisation parameters from the outer genetic algorithm, run the inner optimiser 10 times. For the inner genetic algorithm or particle swarm optimisation, increment the number of cost function evaluations of the inner optimiser (n_{cfe}) until $cf_{\text{inner}} < C_{\text{acc}}$ or until $n_{\text{cfe}} = N_{\text{cfe}}$. Note that if $cf_{\text{inner}} < C_{\text{acc}}$, the optimiser continues until $n_{\text{cfe}} = N_{\text{cfe}}$ but n_{cfe} is not incremented. The following algorithm was then used for the final cost function:

```
if $\nu_{\text{successful}} \geq \nu_{\text{run}}/2$
    $cf_{\text{outer}} = \sum n_{\text{cfe}}/\nu_{\text{successful}}$
else
    $cf_{\text{outer}} = 10^7 \sum cf_{\text{inner}}$
```

cf_{inner} and cf_{outer} are the value of the cost function for the inner and outer optimisers respectively. $\nu_{\text{successful}}$ is the number of successful runs of the inner optimiser.

Note that this cost function will reward parameters that show they can repetitively achieve an acceptable solution and punish those that do not. After the outer genetic algorithm has run through a prescribed number of iterations, the solutions that look promising are then run 100 times to more rigorously assess their performance. The average number of cost function evaluations from these 100 runs is then used to form the basis for comparison between the particle swarm optimisation and the genetic algorithm.

As each member of the outer population performs 10 independent optimisations, and the results from all the members only need to be merged once all the outer population members have completed their calculations, the problem is conceptually easy to run as a parallel algorithm.

5 Distributed computing

Problems of the magnitude being considered here are difficult, if not impossible, to solve on current generation desktop PCs. However, DSTO has a large number of desktop PCs that sit idle during nights, weekends and holidays. The numerical processing ability of 100s of desktop PCs is applied to the optimisation. To this end, the PCs at DSTO are used out-of-hours to perform the optimisation [3]. The system is constructed with a dedicated server providing data storage capability. Each evening, participating PCs are rebooted from a floppy disk and automatically connect to the server where they are allocated a job. Each computer then processes its allocated job and, when it is completed, submit the answer and be allocated a new job.

When an individual PC (referred to as a drone) comes on-line, it checks the job control file to determine which jobs are still to be completed. It then obtains the parameters to use for the inner optimisation routine and will run this job to completion. When finished, the drone will write the results back into a file on the server. The results written are the number of function evaluations required to optimise the structural model and the value of the cost function after performing the maximum allowed number of cost function evaluations. The drone will then request another job. If there is nothing remaining to process, the drone will take all the results currently available and use these to evaluate the fitness function for each individual of the outer genetic algorithm and advance the outer GA by one generation. This will then create a new array that needs to be filled and jobs that can be allocated to new drones.

The process has been setup such that when a drone logs on, it will be allocated a job, and if a drone "dies" before completing its allocated task (either the computer is rebooted for normal operation at the start of a working day, or there is a network or software failure) the job that drone was processing will be allocated to a new drone. Also, the nature of the system means that not all computers attached to the network are equal in terms of performance. If for one or more given generations of the outer genetic algorithm, the process is waiting for slow drones to return a result, another drone or drones will start processing the outstanding jobs with the hope of completing the job quicker. If when a drone returns a job it finds that a result has already been recorded, it will discard its result and obtain a new job. In this way, the system is robust against drones joining and leaving the network, hardware or software failures and varying drone performance capabilities. As long as the central server and at least one drone exist, the processing can continue.

6 Results

Current results indicate that the particle swarm optimisation outperforms the genetic algorithm for problems of the nature being considered here. See in Figure 2 that the number of cost function evaluations can be an order of magnitude lower for a particle swarm algorithm, when compared to the genetic algorithm. Also shown in this figure are the results from a simplex optimisation algorithm. The poor results of this algorithm highlight the

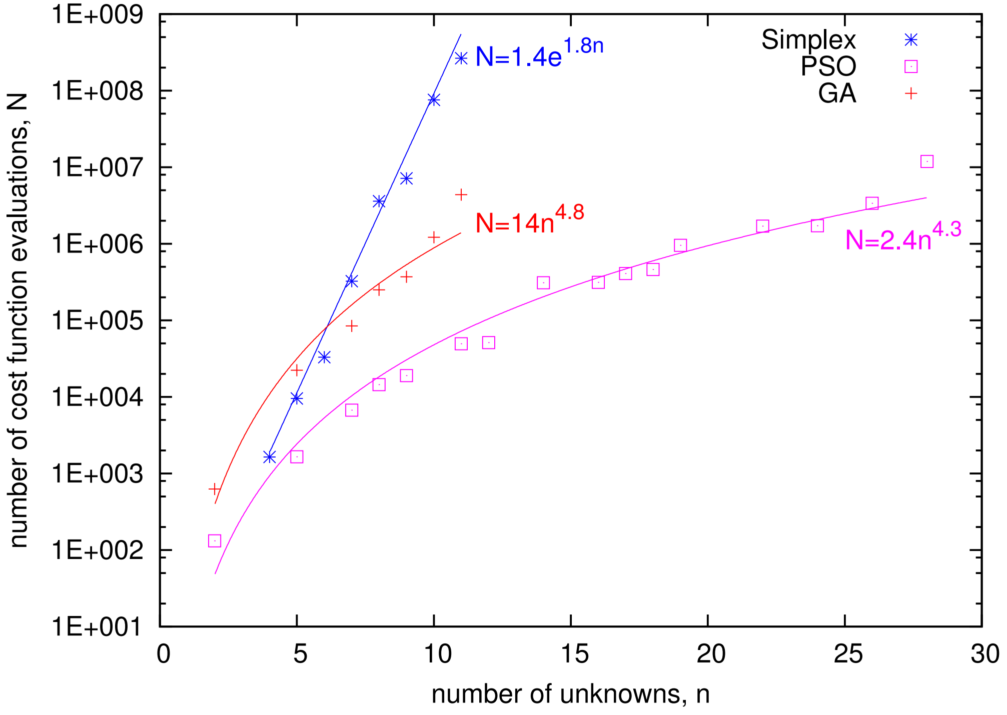

Figure 2: Number of cost function evaluations required for a given problem size

benefits of a population based optimisation, such as a genetic algorithm or particle swarm optimisation.

Note that the genetic algorithm processing has not yet advanced past 11 unknowns. Work is still underway to progress this line as far as possible. However, see that a genetic algorithm with 11 unknowns is taking a similar number of cost function evaluations to solve as a particle swarm optimisation with 25 unknowns. The amount of computing power required to advance the genetic algorithm line is significant and progress is slow.

As the number of unknowns increased, the optimal properties of the particle swarm optimisation changed in a much more predictable manner than those of the genetic algorithm. The characterisation of this change is the subject of further investigation and will be covered in a subsequent paper.

7 Conclusion

We have shown that it is possible to use evolutionary optimisation algorithms to create an optimum structural dynamics model. We also found that the selection of the optimisation algorithm has a significant effect on the suitability of the final model. For the two optimisation algorithms considered here, the particle swarm optimisation algorithm significantly outperformed the genetic algorithm. Also, the particle swarm optimisation algorithm is much easier to configure than the genetic algorithm and is more likely to produce an acceptable model.

References

[1] S. A. Dunn, Optimised Structural Dynamic Aircraft Finite Element Models Involving Mass, Stiffness and Damping Elements. *International Forum on Aeroelasticity and Structural Dynamics*, Madrid, Spain, June 2001, pp.387–396 C91, C94

[2] J. Kennedy and R. C. Eberhaurt, Particle Swarm Optimization. *Proc. IEEE Int. Conf. Neural Networks*, Perth, Australia Nov. 1995, pp.1942–1948 C95

[3] S. Dunn, S. Peucker and J. Perry, Genetic Algorithm Optimisation of Mathematical Models Using Distributed Computing. *Applied Intelligence*, in-press C98