Model Optimization and Parameter Estimation with Nimrod/O

David Abramson[1], Tom Peachey[1], and Andrew Lewis[2]

[1] Caulfield School of Information Technology,
Monash University, Melbourne, Australia
[2] Division of Information Services, Griffith University, Brisbane, Australia

Abstract. Optimization problems where the evaluation step is computationally intensive are becoming increasingly common in both engineering design and model parameter estimation. We describe a tool, Nimrod/O, that expedites the solution of such problems by performing evaluations concurrently, utilizing a range of platforms from workstations to widely distributed parallel machines. Nimrod/O offers a range of optimization algorithms adapted to take advantage of parallel batches of evaluations. We describe a selection of case studies where Nimrod/O has been successfully applied, showing the parallelism achieved by this approach.

1 Introduction

Research in optimization concentrates on search methods; the objective function is usually only mentioned with regard to how its properties affect the validity and efficiency of the algorithm. A published algorithm will typically (see for example [20]) contain lines of the form

$$\texttt{evaluate } y = f(x_1, x_2, \ldots, x_n)$$

giving the impression that the step is minor. However for a substantial class of optimization problems the execution time of the algorithm is dominated by this evaluation step.

One such set of problems involve industrial design. Increasingly, in the design of engineering machines and structures, the prototyping stage is being replaced by computer modelling. This is normally cheaper, allows exploration of a wider range of scenarios and the possibility of optimization of the design. Consider for example a design problem in mechanical engineering, that of choosing the shape of a component that meets the functional specifications and is also optimal in the sense of giving maximal fatigue life [16]. Computation of the fatigue life involves a finite element analysis of the stress field followed by computation of perturbations produced by a range of hypothetical pre-existing cracks and calculation of the growth rate of these cracks under a given load regime.

Another class of optimization problems occurs in scientific modelling where one wishes to determine the values of underlying parameters that have given

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part I, LNCS 3991, pp. 720–727, 2006.

rise to observed results. This inverse problem may be considered an optimization problem, searching through the plausible parameter space to minimize the discrepancy between the predicted and observed results. Inverse computational problems of this type are becoming common throughout many branches of science; some examples are described in a later section.

Such computational models typically take minutes or hours on a fast machine and an optimization requires that the model is executed many times. Hence it becomes attractive where possible to perform batches of these evaluations concurrently, sending the jobs to separate processors. Large clusters of processors are now commonly available for such work [1]. This paper describes a tool, Nimrod/O, that implements a variety of optimization algorithms, performing objective evaluations in concurrent batches on clusters of processors or the resources of the world computational grid.

2 The Nimrod Family of Tools

Parametric studies are explorations of the results of computational models for combinations of input parameters. Nimrod/G, [6, 4, 9], was designed to assist engineers and scientists in performing such studies using concurrent execution on a cluster of processors or the global grid. The user typically specifies a set of values for each parameter and the tasks required for a computation. Nimrod/G then generates the appropriate parameter combinations, arranges for the executions of these jobs and transfer of files to and from the cluster nodes, informing the user of progress with a graphical interface. Such an experiment may take days so failure of cluster nodes is a common problem; Nimrod/G reschedules jobs from a failed node. The number of concurrent jobs is limited only by the size of the cluster. Thus the user may achieve high concurrency without modifying the executables.

Nimrod/O [19] is a tool that provides a range of optimization algorithms and leverages Nimrod/G to perform batches of concurrent evaluations. Hence it is an efficient tool for the types of optimization problems mentioned earlier. Below we describe the operation of Nimrod/O and discuss some of the projects where it has been used. Note that Nimrod/O is not unique in offering distributed optimization. OPTIMUS Parallel [2], a commercial product, was developed at about the same time. However the focus there is narrower than that of Nimrod/O, with an emphasis on Design of Experiments and Response Surface Methods.

The Nimrod Portal [3] provides a friendly interface to the Nimrod toolset. It uses drop down menus to design and run an experiment and to select computational resources. Such resources may be added or removed as the experiment proceeds.

3 Nimrod/O

Nimrod/O requires a simple text "schedule" file that specifies the optimization task and the optimization method(s) to use. Several different algorithms and

different instances of the same algorithm (varying the algorithm settings or the starting points) may be performed concurrently. But each evaluation task is funnelled through a cache to prevent duplication of jobs. If the cache cannot find the result in its database then the job is scheduled on the computational resources available. Since the Nimrod/O cache is persistent, when an aborted experiment is restarted the cache can provide results of all jobs completed earlier and hence a rapid recapitulation of the earlier work. Nimrod/O also allows separate users to share a cache so a useful database of completed jobs may be developed.

Often the objective functions produced by computational models produce multiple local optima. Further, the noise produced by discretization of the continuum may be significant and this gives a rough landscape adding further local optima as artifacts. The ability to run multiple optimizations from different starting points often reveals these multiple optima and may indicate which is global. As they are run concurrently this may be achieved without affecting the elapsed time.

The schedule file specifies the optimization in a declarative fashion. It is however simpler than standard optimization specification languages such as GAMS or AMPL as the definition of the objective function is assumed to be hidden within an executable program. An imperative section gives the commands needed to compute that objective.

An example schedule is shown in Figure 1. The first section of this specifies the parameters, their type (floats, integers or text) and ranges. Text parameter are used for categorical data; a separate optimization is performed for each combination of text values. For float parameters the "granularity" may be specified to control the rounding of values that is applied before they are sent for evaluation of the objective. The coarser granularity will improve the chances of a cache match with previous jobs, possibly at the expense of a less accurate optimum. Integer parameters are treated as floats with a granularity of 1.

```
parameter x float range from 1 to 15
parameter y float range from 0.5 to 1.5
parameter z float range from 0.5 to 1.0
parameter w text select anyof "stt" "dynm"

constraint x >= y + 2.0^z
constraint {x > sin(pi*y)} or {x < 10}

task main
  copy * node:.
  node:substitute skeleton model.inp
  node:execute ./model.exe model.inp
  copy node:obj.dat output.$jobname
endtask
```

```
method simplex
   starts 5
    starting points random
    tolerance 0.01
   endstarts
endmethod

method bfgs
  starts 5
    starting points random
    tolerance 0.01
    line steps 8
  endstarts
endmethod
```

Fig. 1. A sample configuration file

The next section of the schedule specifies the "tasks" needed to evaluate the objective, normally to run a computational model. This includes the distribution of requisite files to the computational processors (the nodes), perhaps substitution of parameter values in input files, the execution of the evaluation programs and the return of the objective value. In this case the file `skeleton` has strings `$x`, `$y`, `$z` and `$w` replaced by their current values to form an input file `model.inp` for the computation. The executable `model.exe` performs the modelling producing a numerical result in the file `obj.dat` which is then copied back to the root node.

Finally the schedule gives the optimization method or methods to use. The example shown uses two methods, downhill simplex and BFGS. Note that several "starts" are specified, five simplex, five BFGS. This gives ten separate optimizations all of which will run concurrently if sufficient computing nodes are available.

4 Nimrod/O Algorithms

Nimrod/O is designed to solve optimization problems in engineering design and scientific modelling that typically involve a search space that is the cross product of several continuous parameters. Thus the problems are rarely of the combinatorial nature frequently encountered in operations research. Rather they require hill-climbing methods. The optimization algorithms offered by Nimrod/O reflect this requirement.

The algorithms provided fall into three categories. The first type samples the whole search space. It includes an exhaustive search with a given granularity for each numerical parameter. There is also a "subdivision search". This evaluates the space on a coarse grid then iterates, each iteration using a finer grid around the best point revealed by the previous iteration.

The second class are some traditional downhill search methods and recent variants: the direct search method of Hooke and Jeeves, the simplex method of Nelder and Mead, and some variants, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method, and simulated annealing. Finally there are population based methods. EPSOC is an evolutionary programming algorithm using the ideas of self-organised criticality [13]. Nimrod/O interfaces with external genetic algorithm implementations GENEsYs and gamut.

The design of Nimrod/O allows for co-scheduling with external optimization routines. Using library functions supplied an external program can forward batches of jobs to Nimrod/O and hence take advantage of distributed computation, caching and constraint evaluation facilities.

Traditionally, search algorithm efficiency has been judged on the number of function evaluations required. In the Nimrod/O scenario execution time is dominated by function evaluations but these are processed in concurrent batches. Assuming there are sufficient computational resources to handle the largest batch then the number of batches becomes the critical factor in execution time. In implementing the optimization algorithms we have modified standard algorithms where this gives a reduced total number of batches.

For example with the traditional simplex search each iteration evaluates the objective at four points on a line. Nimrod/O offers the alternative of a line search to find the best point along that line. It also offers variants which search along several lines. Although these modifications require more evaluations, experiments suggest, [14], that batch counts are reduced and convergence expedited.

Batches of evaluations may also be augmented with jobs that may (or may not) be required at a later stage in the algorithm. This is known as "speculative computing", [10]. The Nimrod/O simulated annealing implementation can anticipate the step after next, adding tasks that may be needed then. Again this increases the total evaluations but reduces the execution time, [15]. Note that concurrent batch processing also favours the population based methods as the members of a large population may all be assessed concurrently.

5 Nimrod/O Case Studies

Nimrod/O has been successfully applied to a wide variety of optimization problems. A sample is discussed here. Data relating to the parallelism achieved in these experiments are combined at the end of this section.

Air Quality Modelling (AQM)
The model used predicted the concentration of ozone in an airshed, given concentrations of precursor chemicals together with meteorological data for the city modelled. The task [12, 5, 7] was to minimize the ozone concentration within a range of values for N and R, the concentrations of oxides of nitrogen and reactive organics respectively. Since ozone concentration is not a monotonic function of the input concentrations, the minimum does not necessarily correspond to least N and R.

Electromagnetic Modelling (EM)
The design of a test rig for mobile telephone antennas included a ferrite bead to reduce distortion of the radiation pattern [5, 7]. The finite-difference time-domain technique was used to solve Maxwell's equations for the design. The aim was to determine the dimensions and properties of the bead that minimized the losses due to testing.

Airfoil
Flow around an airfoil was modelled, [8], using the computational fluid dynamics package FLUENT. Input parameters were the thickness, camber and angle of attack of the airfoil; the aim was to maximize the ratio of lift to drag.

Quantum Chemistry (QC)
Hybrid quantum mechanics-molecular mechanics use quantum computations for small "active" regions of a molecule and classical methods for the rest. A major problem is correct coupling of the two models. This work, [21], uses the recently developed method of inserting a "pseudobond" at the junction between the models. The method was applied to an ethane molecule using a "pseudopotential"

of the form $U(r) = A_1 \exp(-B_1 r^2) + A_2 \exp(-B_2 r^2)$. The task was to determine parameters A_1, A_2, B_1 and B_2 to minimize a least squares measure of the difference between the model properties and those of real ethane.

Plate Fatigue Life (PFL)
This work [16], modelled the fatigue life of plates containing an access hole, as occurs for example in stiffeners in airplane wings. This required finite element computation of the stress field and the Paris model of the growth of pre-existing cracks. The aim was to determine the hole profile under certain constraints that optimized this fatigue life.

Transformation Norm of an Integral Transform (TNIT)
The norm of the Generalized Stieltjes Transform is a long unsolved problem in mathematical analysis. A model was used to compute the ratio of the output norm to the input norm for a 2 parameter family of input functions and Nimrod/O optimized this ratio [18]. A novel aspect of this project was that multiple optimizations were performed for a parameter sweep of two further parameters; so multiple instances of Nimrod/O were launched by Nimrod/G.

5.1 Parallelism

For a single Nimrod/O optimization, if all evaluations required the same execution time then the ratio of the number of evaluations to the number of batches would give the parallelism attained. (When execution times vary then this overestimates the parallelism as discussed in [17], since the execution time for a batch is that of the longest job.) The case studies described above used multiple optimizations, thus increasing the effective parallelism. We assume sufficient computational resources to run all optimizations concurrently. In that situation the number of batches in the longest optimization is the main determinant of the total experiment time. Figure 2 gives n, the number of optimizations, b and e, the number of batches and of evaluations for the longest optimization, B and E, the total batches and evaluations for all optimizations. Then the ratios e/b and E/B estimate the concurrency for a single optimization provided by batch evaluation. E/b estimates the concurrency for the combined optimizations provided by both batching and concurrent optimizations.

6 Conclusion

Optimization problems where evaluation of the objective function is computationally intensive are increasingly common. Nimrod/O is a tool that can expedite such problems by providing concurrent execution of batches of evaluations and concurrent multiple searches. It uses either a cluster of processors or the resources of the world computational grid; concurrency is limited only by the number of processors available. Nimrod/O offers a range of standard search algorithms and some novel ones, and is easily extensible to new algorithms. Since the total execution time is determined by the number of batches rather than the

Experiment	Method	n	b	e	B	E	e/b	E/B	E/b
AQM	BFGS	1	10	46	10	46	4.6	4.6	4.6
EM	BFGS	10	17	95	102	581	5.6	5.7	34.2
	Simplex	10	16	42	106	286	2.6	2.7	17.9
	Simplex-L	10	21	146	124	859	7.0	6.9	40.9
	Simplex-L1	10	16	144	104	843	9.0	8.1	52.7
	RSCS	10	9	54	61	371	6.0	6.1	41.2
	RSCS-L	10	17	127	116	858	7.5	7.4	50.5
	RSCS-L1	10	12	121	89	818	10.1	9.2	68.2
	EPSOC	10	20	892	200	8616	44.6	43.1	430.8
	EPSOC	10	20	1117	200	10601	55.9	53.0	530.1
Airfoil	Simplex	10	40	160	241	908	4.0	3.8	22.7
	Simplex-L	10	70	1037	463	6753	14.8	14.6	96.5
	Simplex-L1	10	28	385	183	2657	13.8	14.5	94.9
	RSCS	10	53	525	173	1717	9.9	9.9	32.4
	RSCS-L	10	71	1042	369	5385	14.7	14.6	75.8
	RSCS-L1	10	34	530	219	3301	15.6	15.1	97.1
	EPSOC	10	20	1260	200	12472	63.0	62.4	623.6
QC	BFGS	63	300	1450	3121	14650	4.8	4.7	48.8
PFL	simplex	9	26	88	132	505	3.4	3.8	19.4
TNIT	simplex	209	93	305	3048	10602	3.3	3.5	114.0

Fig. 2. Parallelism achieved by some experiments

number of evaluations modifications to some traditional search algorithms are advantageous and have been incorporated into Nimrod/O.

References

1. `http://www.top500.org/lists/`, accessed 3 August 2005.
2. `http://www.lmsintl.com/`, accessed 3 August 2005.
3. `http://www.csse.monash.edu.au/~nimrod/nimrodportal/`, accessed 3 August 2005.
4. Abramson D. et al. The Nimrod computational workbench: A case study in desktop metacomputing. In *Australian Computer Science Conference (ACSC 97)*, pages 17 – 26, Macquarie University, Sydney Feb 1997.
5. Abramson D. A., A. Lewis, and T. Peachey. Nimrod/O: a tool for automatic design optimisation using parallel and distributed systems. In *Proceedings of the 4th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2000)*, pages 497–508, Singapore, 2000. World Scientific Publishing Co.
6. Abramson D., J. Giddy, and L. Kotler. High performance parametric modeling with Nimrod/G: Killer application for the global grid? In *International Parallel and Distributed Processing Symposium (IPDPS)*, May 2000.
7. Abramson D. A., A. Lewis, and T. Peachey. Case studies in automatic design optimisation using the P-BFGS algorithm. In A Tentner, editor, *Proceedings of the High Performance Computing Symposium - HPC 2001*, pages 22 – 26, Seattle, April 2001. The International Society for Modeling and Simulation.

8. Abramson D. A., A. Lewis, T. Peachey, and C. Fletcher. An automatic design optimization tool and its application to computational fluid dynamics. In *Supercomputing*, Denver, November 2001.
9. Abramson D., R. Buuya, and J. Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems*, 18(8), Oct 2002.
10. Burton F.W. Speculative computation, parallelism and functional programming. *IEEE Transactions on Computers, C*, 34:1190–1193, 1985.
11. Chong E.K.P. and S.H. Żak. *An Introduction to Optimization*. Wiley, 1996.
12. Lewis A., D. A. Abramson, and R. Simpson. Parallel non-linear optimization: towards the design of a decision support system for air quality management. In *IEEE Supercomputing 1997*, pages 1 – 13, California, 1997.
13. Lewis A., D. Abramson, and T. Peachey. An evolutionary programming algorithm for automatic engineering design. In *Parallel Processing and Applied Mathematics: 5th International Conference (PPAM 2003)*, Czestochowa, Poland, 2003.
14. Lewis A., D. A. Abramson, and T. Peachey. RSCS: A parallel simplex algorithm for the Nimrod/O optimization toolset. In *Proceedings of the Third International Symposium on Parallel and Distributed Computing (ISPDC 2004)*, pages 71–78, Cork, Ireland, 2004. IEEE Computer Society.
15. Peachey T. C., D. Abramson, and A. Lewis. Heuristics for parallel simulated annealing by speculation. Technical report, Monash University, 2001.
16. Peachey T., D. A. Abramson, A. Lewis, D. Kurniawan, and R. Jones. Optimization using Nimrod/O and its application to robust mechanical design. In *Proceedings of the 5th International Conference on Parallel Processing and Applied Mathematics [PPAM 2003]*, volume 3019 of *Lecture Notes in Computer Science*, pages 730–737. Springer-Verlag, 2003.
17. Peachey T. C., D. Abramson, and A. Lewis. *Parallel Line Search*. Springer, to appear.
18. Peachey T. C. and C. M. Enticott. Determination of the best constant in an inequality of Hardy, Littlewood and Polya. *Experimental Mathematics*, to appear.
19. Peachey T. C. *The Nimrod/O Users' Manual v2.6*. Monash University, http://www.csse.monash.edu.au/~nimrod/nimrodg/no.html, 2005.
20. Press W.H. et al. *Numerical Recipes in C*. Cambridge, second edition, 1993.
21. Sudholt W. et al. Applying grid computing to the parameter sweep of a group difference pseudopotential. In M Bubak et al., editor, *Proceedings of the 4th International Conference on Computa tional Science [ICCS 2004]*, volume 3036 of *Lecture Notes in Computer Science,*. Springer-Verlag, 2004.