Evolutionary Optimization Techniques on Computational Grids*

Baker Abdalhaq, Ana Cortés, Tomás Margalef and Emilio Luque
Departament d'Informàtica, E.T.S.E, Universitat Autònoma de Barcelona, 08193-Bellaterra (Barcelona) Spain

baker@aows10.uab.es
{ana.cortes,tomas.margalef,emilio.luque}@uab.es

Abstract. Optimization of complex objective functions such as environmental models is a compute-intensive task, difficult to achieve by classical optimization techniques. Evolutionary techniques such as genetic algorithms present themselves as the best alternative to solving this problem. We present a friendly optimization framework for complex objective function on a computational grid platform, which allows easy incorporation of new optimization strategies. This framework was developed using the MW library and the Condor system. The framework architecture is described, and a case study of a forest-fire propagation simulator is then analyzed.

1 Introduction

Models are increasingly being used to make predictions in many fields of environmental science Typically, simulation (a computer program that represents a given model) exhibits difficulties in exactly imitating the real behavior of the simulated system. Basically, this difficulty lies in two main considerations. On the one hand, environmental models frequently require information on a high number of input variables about which it is not usually possible to provide accurate values for all. Therefore, the results provided by the corresponding simulator usually deviate from the system's real behavior. On the other hand, models can be wrongly designed becoming the model itself the main cause of erroneous results.

Generally, model/simulator analysis is mainly focused on one of these two sources of errors. Whichever analysis goal is chosen, the degree of complexity in model analysis, which can involve model validation, verification and calibration, resulted in particularly difficult tasks in terms of computing power.
Since distributed computing systems (also called metacomputers or grid computing environments) have, over the last decade, increased as a great source of computing

* This work was supported by the CICYT under contract TIC 98-0433 and partially supported by the DGICYT (Spain).

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 513–522, 2002.

power, relevant optimization works were addressed to these environments [1][2][3]. All these works are focused on mathematical optimization techniques, which can be applied when some degree of knowledge about the function to be optimized is provided. However, there is still considerable work to do in the area of non-mathematical approaches that are more suitable for optimization processes dealing with functions about which no information is provided. Environmental models/simulators are a good example of this kind of functions. Usually, environmental simulators are provided as black-boxes, and the only known information about them consists of how they should be provided with input information, and the nature of the results that they generate. Under this assumption, mathematical optimization techniques are destined to failure, and non-mathematical schemes such as Genetic Algorithms, Simulating Annealing, Tabu Search and so on, have arisen as the best candidates to approach such problems.

The aim of this work is to provide a framework for the optimization of complex functions (considered as black-box functions) that take advantage of the computing power provided by a grid-computing environment. The BBOF (Black-Box Optimization Framework) system has been developed using the master-worker programming paradigm and uses Condor as a distributed resource management.

The rest of the paper is organized as follows. In section 2, the problem statement is reported. The architecture of the proposed optimization framework is described in section 3. Implementation details are described in section 4. Section 5 shows how BBOF has been applied to a forest-fire propagation simulator and, finally, section 6 presents the main conclusions.

2 Problem Statement

Formal optimization is associated with the specification of a mathematical objective function (called L) and a collection of parameters that should be adjusted (tuned) to optimize the objective function. This set of parameters is represented by a vector referred to as θ. Consequently, one can formulate an optimization problem as follows:

$$\text{Find } \theta^* \text{ that solves } \min_{\theta \in S} L(\theta) \qquad (1)$$

where $L : R^p \rightarrow R^1$ represents some *objective function* to be minimized (or maximized). Furthermore, θ represents the vector of adjustable parameters (where θ^* is a particular setting of θ), and $S \subseteq R^p$ represents a constrain set defining the allowable values for the θ parameters. Put simply, the optimization problem deals with the aim of defining a process to find a setting for the parameter vector θ, which provides the *best* value (minimum or maximum) for the *objective function L*. This search is carried out according to certain restrictions of the values that each parameter can take.

The whole range of possibilities that can be explored in obtaining the optimization goal is called the *search space,* which *is* referred to as *S*.

As we mentioned in the previous section, we are interested in complex model optimization regardless how the model itself works. Under this assumption, the underlying model/simulator is identified as a complex black-box function about which no information is provided. However, there is the possibility of measuring the quality of the results provided by the simulator for any input vector (θ). Consequently, the objective function (L) involves both executing the simulator and the quality function simultaneously, being the final value provided by the quality function, the value to be minimized or maximized.

Typically, the way to solve equation (1) consists of applying an existent optimization technique to an initial guess for θ in order to obtain a new set of input parameters closer to the optimal solution (θ^*). However, applying any optimization technique once alone never leads to a good solution. Therefore, the same process is repeated again by starting with a new or nearly guess.

Beside the strictly mathematical definition of the optimization problem, there are some extra issues that should be considered when dealing with geographically dispersed CPU´s as source of computing power. Some of these features include the dynamic availability of the machines, communication delays between processors, heterogeneity system and scheduling problems. Despite of all these drawbacks, these platforms are well suited to large-scale computations needed by environmental model optimization problems.

3 Distributed optimization framework

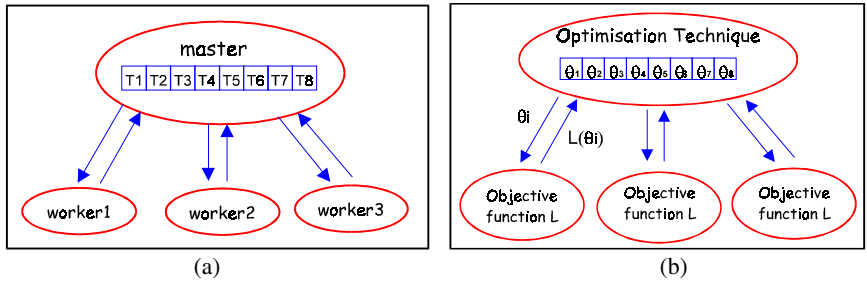
As was commented above, optimization techniques typically obtain progressive improvements in the original guess of the vector θ by consecutive executions of the optimization process. A great improvement in this way of proceeding is to consider, not only one guess at a time, but a wide set of guesses and, based on the results obtained for all of these, to automatically generate a new set of guesses and to re-evaluate the objective function for them all, and so on. For this reason, our optimization framework works in an iterative fashion, where it moves step-by-step from an initial set of guesses for θ to a final value that is expected to be closer to the true θ^* (optimal vector of parameters) than the initial guesses. This goal is achieved because, at each iteration of this process, a preset optimization technique is applied to generate a new set of guesses that should be better than the previous one.

This iterative scheme is the core of the proposed framework, which is called the Black-Box Optimization Framework (BBOF). BBOF has been implemented in a plug&play fashion. On the one hand, the optimized function can be any system

(complex simulator) that has a vector of parameters as input, and provides one or more values as output (a quality function should also be provided to determine the goodness of the results). On the other hand, any optimization technique ranging from evolutionary techniques, such as genetic algorithms or simulating annealing through strictly mathematical schemes, can easily be incorporated.

Due to its characteristics, the proposed framework fits well into the master-worker programming paradigm working in an iterative scheme. An iterative master-worker application consists of two entities: a master and multiple workers. The master is responsible for decomposing the problem into small task (and distribute these tasks among a farm of worker processes), as well as for gathering the partial results in order to produce the final result of the computation. The worker processes receive a message from the master with the next task, process the task, and send the results to the master. The master process may carry out certain computations while tasks of a given batch are being completed. After that, a new batch of tasks is assigned to the master, and this process is repeated several times until completion of the problem (after K cycles or iterations). Figure 1.a schematically shows how master and workers processes interact during one iteration of the iterative process described above, when 3 workers are considered and the number of tasks to be executed by the workers is 8.

If we analyze the above-described behavior, we can easily match each element with the main components of BBOF. In particular, since the evaluation of the black-box and quality functions for each guess of θ are independent of each other, they can be identified as the work done by the workers. Consequently, the responsibility for collecting all the results from the different workers and for generating the next set of guesses by applying a given optimization technique will be concentrated on the master process. Figure 1.b graphically illustrates how this matching is undertaken.



(a) (b)

Fig 1. Master and worker process interaction (a) and how these processes are matched to the BBOF elements (b).

Once the architecture of BBOF has been described, we now describe the manner in which the BBOF was implemented.

4 Implementation issues

We should bear in mind that our initial goal was to develop an optimization framework for solving complex problems on a computational grid. For this purpose, we have used Condor as a resource management on a grid-computing environment [4] and the MW library [5] so as to easily implement our master-worker scheme. In the following section, we briefly outline certain basic ideas on Condor and MW.

4.1 Condor

Condor is a software system that runs on a cluster of workstations in order to harness wasted CPU cycles. Condor was first developed for Unix systems, and can currently be executed on a wide range of machines. A Condor pool consists of any number of machines, possibly heterogeneous, which are connected by a network. One machine, the central manager, keeps track of all the resources and jobs in the pool. Condor allows High Throughput Computing (HTC) to be obtained, that is, the obtention of large amounts of processing capacity sustained over long time periods. The resources, i.e. hardware, middleware and software, are large, dynamic and heterogeneous.

4.2 MW

MW is a set of C++ abstract-based classes that must be re-implemented in order to fit the particular application. These classes hide difficult metacomputing issues, allowing rapid development of sophisticated scientific computing applications. Basically, there are three abstract base classes to be re-implemented. The *MWDriver* class correspondents to the master process and contains the control center for distributing tasks to workers. The *MWTask* class describes inputs and outputs –data and results – which are associated with a single unit of work. The *MWWorker* class contains the code required to initialize a worker process and to execute any tasks sent to it by the master. BBOF's implementation is based on these MW classes. As mentioned in section 3, BBOF has two main parts: the objective function (black-box and quality functions) and the optimization technique. In particular, the objective function (L) corresponds to the *MWWorker* class, whereas the optimization technique has been integrated as the *MWDriver* class. Finally, a particular instance of the *MWTask* class directly matches with a given input parameter vector. Concerning communication, there are MW versions that perform communications by using PVM, the file system and sockets. In this work, MW was used with PVM [6], where MW workers are independent jobs spawned as PVM programs. The original MW does not consider multiple iterations of the process depicted in figure 1.a. However, our optimization framework is based on an iterative way of working. For this purpose, we have developed our optimization framework based on an extended version of MW [7], which allows MW to iterate a predetermined number of times.

5 Case study: Forest-fire propagation

Forest fire propagation is considered a challenging problem in the area of simulation, due to the complexity of its physical model, the need for great amount of computation and the difficulties of providing accurate inputs to the model. Uncertainties in the input variables needed by the fire propagation models (temperature, wind, moisture, vegetation features, topographical aspects...) can have a substantial impact on result errors and must be considered. For this reason, optimization methodologies to adjust the set of input parameters of a given model would be provided in order to obtain results as close as possible to real values.

In particular, we have applied the proposed optimization framework to a fire propagation simulator called ISStest [8], which, from an initial fire front (set of points), generates the position of the fire line after a given period of time. As has just been mentioned, forest fire propagation simulators deal with a wide set of input parameters. In the experimental study reported below, we only focus on finding the proper values of the wind input parameter, which is defined as a vector composed of two components: wind speed (w_s) and wind direction (w_d).

Obviously, the expected behavior of the ISStest is that the new fire line provided after a certain preset time exactly matches the real situation of the fire line once that time has passed. Therefore, our objective function *L*, in this case, consists of executing once ISStest for a given configuration of input values, and evaluating the distance between the real fire line and the fire line obtained by simulation. For this purpose, the Hausdorff distance [9] was chosen. The Hausdorff distance measures the degree of mismatch between two sets of points by measuring the distance of a point from one set that is farthest away from any point of the other, and vice versa. Formally, the directed Hausdorff distance *h* between two set of points *M* and *I* at a specific point in *I* is:

$$h(M,I)=\max_{m\in M}\left\{\min_{i\in I}(distance\ (m,i))\right\} \quad (2)$$

Thus, the Hausdorff distance *H* is defined as: $H(M,I) = max(h(M,I), h(I,M))$. In this case, our objective resides in finding the global minimum of this function, given that our aim is that the simulated and real fire lines should be the same, and that the difference between the two should therefore be 0.

The experiments carried out considered that wind speed and wind direction remain fixed during the fire-spread simulation process. The real fire line, which was used as a reference during the optimization process, was obtained in a synthetic manner. In other words, we fixed known values for w_s and w_d parameters and, subsequently, the ISStest simulator was executed three times with a setting of the simulation time equal to15 minutes for each execution. This presupposes that the fire propagation was

spent 45 minutes. At each execution, the output fire line obtained was used as the initial fire line for the next execution. Finally, the obtained fire line was stored and treated as the real fire line. Obviously, the wind speed and wind direction used during this process (which were15km/h and 180^0 respectively) were dismissed once all this process had finished. Since for each ISStest execution we have two parameters (wind speed and wind direction), and bearing in mind that we executed the fire simulator three times, the global vector that should be guessed by the optimization technique consists of 6 elements ($\theta = (w_{s1} \quad w_{d1} \quad w_{s2} \quad w_{d2} \quad w_{s3} \quad w_{d3})$).

We now describe the experimental study carried out using the forest fire propagation simulator commented above.

5.1 Experimental platform

The experiments reported below were executed on a Linux cluster composed of 21 PC´s with Intel Celeron processor 433 MHz, each one having 32 MB RAM and connected with a Fast Ether Net 100 Mb. All the machines are configured so as to use NFS (Network File System) and the Condor system; additionally, PVM are installed on them all.

5.2 Implemented optimization techniques

In this experimental study, we considered two different optimization techniques. Genetic Algorithms (GA) were chosen as a relevant non-mathematical technique within the branch of evolutionary strategies, whereas, on behalf of mathematical optimization techniques, we have adopted a statistical/parabolic method. In the following section, we will outline the main features of each one of these techniques.

Genetic Algorithm
GAs are characterized by emulating natural evolution [10]. Therefore, under the GA scenario, in which one refers to a vector of parameters as chromosome, a gene is identified with one parameter and the population is the set of vectors (chromosome) used for one iteration of the algorithm. The way to obtain the new population consists of applying certain operators (called transmission operators) to the starting set of vectors. These operators (Elitism, Selection, Crossover and Mutation) are in charge of defining the changes to be done on the set of initial vector guesses in order to generate an improved set of vectors.

Statistical/parabolic method
As we previously mentioned, mathematical techniques are supposed to work incorrectly for the kind of problem that we are approaching. However, we implemented a statistical/parabolic method in order to compare the results provided by this scheme with those obtained with the evolutionary techniques described above. This particular

method was chosen once the behavior of the objective function had been outlined for a very simple case. In particular, we considered the simplest case in which the input vector is only composed of two parameters (wind speed and wind direction). This is the case in which the ISStest is executed only once for a fixed period of time, and for which the wind does not change during the whole simulation. Figure 2 shows the value of the objective function when wind speed is fixed to the corresponding real value and the free parameter (wind direction) varies within its corresponding search space (from 0^{o} to 360^{o}). Since a similar behavior was observed for wind direction, a good approximation to the obtained shape seemed to be the parabolic one.

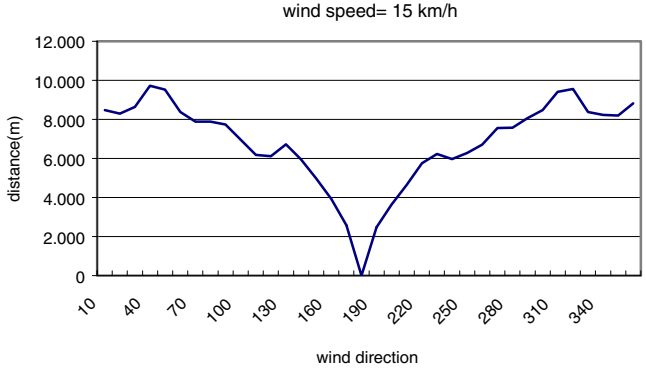


Fig.2 Objective function cross-section at optimal wind speed (15 km/h).

The formal expression to describe this kind of functions corresponds to equation 2, where *f* is the approximation of the objective function *L*.

$$f(X) = \beta_0 + \beta_{11} x_1 + \beta_{12} x_1^2 + \ldots + \beta_{1n} x_1^n + \ldots + \beta_{p1} x_p + \beta_{p2} x_p^2 \ldots + \beta_{pn} x_p^n \quad (3)$$

In order to find the optimum of this function using calculus, we need to derive it for each variable x_i and to find the zero of the derivative. The formula describing this derivation form and the equation we should solve is the following:

$$\frac{\partial f(X)}{\partial x_i} = \beta_{i1} + 2\beta_{i2} x_i + \ldots + n\beta_{i2} x_i^{n-1} = 0 \quad \forall i = 1 \ldots p \quad (4)$$

This approach can help us, if and only if, the objective function can be approximated with good precision to some function that is calculus friendly.

5.3 Experimental results

In this section, we will describe the main results of a preliminary set of experiments, which were performed with the aim of comparing both mathematical optimization techniques (parabolic method) against a new generation of strategies such as genetic algorithms. As we commented, our optimization framework was implemented by using an iterative master-worker programming scheme. In the case of the mathematical approach, the optimization process was iterated once, and the number of vectors evaluated (task to be assigned to the workers) were 200. In contrast to this, when the genetic algorithm was applied, the number of iterations of the whole optimization process were 20, whereas the number of vectors to be evaluated at each iteration were 10. Therefore, the total number of times that the objective function was evaluated is the same in both cases. Furthermore, since the initial set of guesses for both strategies was obtained in a random way, the global optimization process was performed four times; the mean values for all the results (Hausdorff distance in m.) is shown in table 1.

Table 1. Comparison between all the algorithms for homogeneous wind field

Algorithm	Genetic	Statistical/parabolic
Distance (m)	11	443
Evaluations	200	200

As we can observe, GA on average reaches a distance equal to 11m, which is one order of magnitude less than the result obtained in the case of the mathematical approach. The mathematical technique suffers both from deviation from the optimal due to the model and the rare shape of the objective function.

6 Conclusions

In this work, we have described a friendly framework for optimizing black-box functions called BBOF (Black-Boxes Optimization Framework). This framework was developed using the MW library and the Condor system. We applied BBOF to a forest-fire propagation simulator (ISStest) including two optimization techniques: the Genetic Algorithm and the Statistical/parabolic method.

A basic set of experiments were performed and the results denote the difficulty exhibited by classical optimizers in minimized complex objective function such as the one studied. In contrast to this, evolutionary optimization techniques such as genetic algorithms provide substantial improvements in results. These preliminary results have encouraged us to continue our study in this area, as they confirm our expectations.

References

1. Czyzyk J., Mesnier M.P., More J.J.: The Network-Enabled Optimization Systems (NEOS) Server. Preprint MCS-P615-1096, Argonne National Laboratory, Argone, Illinois, (1997).
2. Ferris M., Mesnier M., More J.: NEOS and Condor: Solving optimization problems over the Internet, Preprint ANL/MCS-P708-0398, available at http://www-unix.mcs.anl.gov/metaneos, (March 1998)
3. Linderoth J. and Wright S.J: Computational Grids for Stochastic Programming, Optimization. Technical Report 01-01, UW-Madison, Wisconsin-USA,(October 2001)
4. Livny M. and Raman R.: High Throughput Resource Management, Computational Grids: The Future of High-Performance Distributed Computing. Edited by Ian Foster and Carl Kesselman, published by Morgan Kaufmann (1999)
5. Goux J.-P., Kulkarni S., Linderoth J., Yoder M.: An enabling framework for master-worker applications on the computational grid. Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9), Pittsburgh, Pennsylvania, (August 2000) 43–50
6. Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R. and Sunderam V.: PVM: Parellel Virtual Machine A User's Guide and Tutorial for Networked Parallel Computing. MIT Press (1994)
7. Heymann E., Senar M.A., Luque E. and Livny M.: Evaluation of an Adaptive Scheduling Strategy for Master-Worker Applications on Clusters of Workstations. Proceedings of the 7th International Conference on High Performance Computing (HiPC'2000), LNCS series, vol. 1971 (2000) 214–227
8. Jorba J., Margalef T., Luque E., J. Campos da Silva Andre, D. X Viegas: Parallel Approah to the Simulation Of Forest Fire Propagation. Proc. 13 Internationales Symposium "Informatik fur den Umweltshutz" der Gesellshaft Fur Informatik (GI). Magdeburg (1999)
9. Reiher E., Said F., Li Y. and Suen C.Y.: Map Symbol Recognition Using Directed Hausdorff Distance and a Neural Network Classifier. Proceedings of International Congress of Photogrammetry and Remote Sensing, Vol. XXXI, Part B3, Vienna, (July 1996) 680–685
10. Thomas Baeck, Ulrich Hammel, and Hans-Paul Schwefel: Evolutionary Computation: Comments on the History and Current State. IEEE Transactions on Evolutionary Computation, Vol. 1, num.1 (April 1997) 3–17