PYTHIA: An Expert System for the Optimal Selection of PDE Solvers Based on the Exemplar Learning Approach

E.N. Houstis, J.R. Rice, P. Varodoglou, S. Weerawarana *
Department of Computer Sciences
Purdue University
West Lafayette, IN 47906, USA.

C.E. Houstis
University of Crete
Department of Computer Science
Heraklion, Greece.

November 16, 1992

1 Introduction

It is clear that accuracy and execution time are often the main performance objectives of a user of numerical simulation models based on partial differential equations (PDEs). In a PDE computation, accuracy is controlled through the refinement of the grid and the discretization scheme used, while execution time depends on the speed of the targeted machine and the efficiency of the PDE solver. For a given machine, the computation of a solution within a certain accuracy ("epsilon") and time frame ("T") requires the selection of appropriate grid, discretization and solution scheme (method) plus an appropriate machine. If the machine is parallel, then the selection of its configuration (number of processors) and the partitioning of the computation into load balanced, optimally parallel subtasks is also required. Unfortunately, the above parameters that affect the performance of a computation depend on various mathematical properties and characteristics of the specified PDE model and its unknown solution. Hence, the acquisition of knowledge about the mathematical behavior of the given PDE model and the appropriate selection of these parameters to achieve the computational objectives of the user constitute a "hard" problem. We believe that future numerical simulation systems should be capable of making these decisions for the average user while allowing knowledgeable users to be involved with the decision process.

2 PYTHIA

PYTHIA is a research project that attempts to solve the (grid, configuration) and (method, machine) selection problems for the class of "field" problems within the //ELLPACK[HRC+90] numerical simulation system by applying an expert system methodology. PYTHIA applies symbolic techniques to derive the characteristics of the PDE model or allows the user to specify them together with some weighting factors. PYTHIA knowledge base rules are derived from a database of performance data associated with a known population of PDE problems and a library of PDE solvers. Alternatively, performance data or rules may be supplied by the knowledge engineer in a predefined format. PYTHIA's inference engine tries to "match" the given problem to the available population of PDE classes and problems and then indirectly infers the appropriate selection of parameters from the known performance data of the "matched" problem or class of problems.

In this extended abstract, we provide an overview of problem features and their extraction, the rules used in PYTHIA, the inference algorithm, the architecture, and finally the results of some initial experiments.

3 Problem Features

We classify problem features in terms of the following components: (i) PDE operator, (ii) PDE equation right hand side, (iii) solution, (iv) boundary conditions, (v) initial conditions, and (vi) domain of definition. Thus, the set of characteristics of a PDE problem includes some classification information for the above components and some quantitative information about the behavior (i.e. smoothness and local variation) of the I/O functions (i.e. coefficients of the operators, right side of the operator equations, and the solution) of the PDE problem.

In the PYTHIA environment, the characteristic vector v is determined automatically through symbolic processing of the PDE problem specification and by symbolic/numeric processing of its I/O functions. Features that cannot be ascertained automatically must be provided by the user.

Problem features are represented by a data structure that contains slots for booleans quantities (for example, whether the operator is self-adjoint or not) as well as numerical quantities (for example, the smoothness and local variation of the boundary conditions). Also, if any performance data is available for the problem, then that information is kept in another slot as a list of (method, performance profile) tuples. The characteristic vector of a problem is then a vector that lists the values of all the boolean and numerical slots of a problem structure.

*Work supported in part by AFOSR grant 91-F49620, NSF grant CCR 86-19817.

4 The Rules

The knowledge base contains two classes of rules. The first are a priori rules provided by a human expert and the latter are the rules generated by PYTHIA. We expect that the number of human–given rules will be small as many immeasurable quantities such as the efficiency of the implementation also play a significant role in the performance of algorithms.

The second type of rules are generated by deliberate actions of the knowledge engineer. The knowledge engineer "trains" PYTHIA by executing many test problems then allowing PYTHIA to gather interesting performance data from them. These performance profiles are stored as facts in the knowledge base. A statistical analyzer is also used to analyze performance data in order to rank the performance of various methods on one problem and also the performance of one method on many problems of belonging to one "class." The knowledge engineer identifies a class of problems and requests PYTHIA to generate rules for that class using applicable performance profiles. Thus, there are two sub–types of rules generated by PYTHIA: The first type of rules indicate which method works best for a given class of problems at each error level while the second type of rules indicate with method works best for a given problem at each error level.

5 The Inference Algorithm

Given the user's problem, PYTHIA first attempts to find the class of problems that is "closest" by matching the properties of the problem with properties of the class. Once the closest class has been found, if class rules are available, then those can be used to provide information to the user. Otherwise, PYTHIA finds the closest exemplar within that class and then uses the rules for that problem to provide the necessary information to the user.

6 Architecture

PYTHIA is being implemented as an editor within //ELLPACK. The rule generation component uses the ELLPACK Performance Evaluation System to execute test problems, generate performance profiles and to execute the FORTRAN statistical analyzer. These profiles are then converted in to knowledge base facts by PERL scripts. The facts themselves are represented as Common Lisp structures. The expert system component is written in OPS5. The user interface and the driver of the entire system is written in C.

7 Preliminary Studies

We used the class of problems studied in [HR82] to evaluate the PYTHIA. Let C be that class of problems. First, we used the first 10 problems in C (or, 25% of C) as a "training set" to make performance predictions for the remaining problems in C. Then we used the first 20 problems in C (or, 50% of C) as the training set and as before, made predictions for the remaining problems. The Table below shows the results of this test. Each percentage indicates the percentage of correct predictions for each (training set, relative error level) pair. The "Number Predicted" is the number of valid predictions made by PYTHIA.[1]

Training Set	Number Predicted	Relative Error		
		0.1%	0.01%	0.001%
$\frac{1}{4}C$	37	76%	73%	57%
$\frac{1}{2}C$	19	95%	89%	58%

In the next test, we wish to observe the effect of a larger training set on the correctness of predictions on the same set of problems. Hence, we used the same training sets as above, but, instead of making predictions for all remaining problems for each training set (i.e., from problems 11 and 21 onwards, respectively), in both cases, we made predictions for just problems 21 onwards. The table below shows these results.

Training Set	Number Predicted	Relative Error		
		0.1%	0.01%	0.001%
$\frac{1}{4}C$	7	86%	86%	100%
$\frac{1}{2}C$	12	100%	92%	33%

8 Conclusions

Although the above tests are extremely preliminary, we observe that PYTHIA does behave quite well. We hope to closely couple PYTHIA with the //ELLPACK environment in order to allow it to learn each time a problem is solved by a user. This will allow PYTHIA to improve with usage and make her predictions even more accurate.

References

[HR82] E. N. Houstis and J. R. Rice. High order methods for elliptic partial differential equations with singularities. *International Journal for Numerical Methods in Engineering*, 18:737–754, 1982.

[HRC+90] E. N. Houstis, J. R. Rice, N. P. Chrisochoides, H. C. Karathanasis, P. N. Papachiou, M. K. Samartzis, E. A. Vavalis, Ko-Yang Wang, and S. Weerawarana. //ELLPACK: A numerical simulation programming environment for parallel MIMD machines. In J. Sopka, editor, *Proceedings of Supercomputing '90*, pages 96–107. ACM Press, 1990.

[1] A prediction is not valid if other factors, such as method applicability, are not satisfied.