Recommender Systems for Problem Solving Environments

Naren Ramakrishnan, Elias N. Houstis and John R. Rice
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
{naren,enh,jrr}@cs.purdue.edu

Abstract

Traditionally, recommender systems have been studied in domains that focus on harnessing distributed information resources, collaborative filtering, information aggregation, social schemes for decision making and user interfaces. In this paper, we present recommender systems in a different context — primarily as systems that select (scientific) software appropriate to a user's needs. This application is motivated by the wide acceptance of problem solving environments (PSEs) which are high level environments for doing computational science. We give an overview of the domain, argue the need for recommender systems and describe our work in this area. The research issues in this discipline are also highlighted.

Introduction

Most research in computational science and engineering involves designing computational models of real-life scientific phenomena and simulating them with advanced computing paradigms. Problem solving environments (PSEs) are high level environments for doing computational science (Gallopoulos, Houstis, & Rice 1994). Example PSEs are MATLAB for linear algebra and ELLPACK, PELLPACK (Houstis & Rice 1992) for elliptic partial differential equations. Besides providing all the computational facilities necessary to solve a target class of problems, PSEs present a natural interface and use the vernacular of the underlying scientific domain, so users can run them without specialized knowledge of the underlying computer hardware.

Most existing PSEs, however, assume that the choice of method (algorithm) to solve a given scientific problem is fixed *a priori* (static) and that appropriate code is located, compiled and linked to yield static programs. The user (scientist) still needs to select suitable software for the problem at hand in the presence of practical constraints on accuracy, time and cost. A recommender system for a PSE serves as an intelligent front-end and guides the user from a high level description of the problem through every stage of the solution process, providing recommendations at each step.

Recommender systems for PSEs are becoming increasingly critical due to several reasons. PSEs and WWW based 'Problem Solving Services' are becoming more ubiquitous and widely accepted in scientific communities; this has led to a rapid increase in the number of online algorithms/methods that are made available to the application scientist; and recommender systems also aid (indirectly) in the performance evaluation of scientific software.

Research

The PSE group at Purdue has taken several pioneering steps in this direction. The PYTHIA system interfaces with application specific PSEs to recommend software for specific categories of problems such as partial differential equations and numerical quadrature. In addition to selecting software for these domains, it also interfaces with the GAMS (Boisvert 1996) mathematical software ontology (`http://gams.nist.gov`) to direct the user to an appropriate location from where the software can be retrieved. In other words, software recommendation is *complete* for these classes of problems. Please see `http://www.cs.purdue.edu/research/cse/pythia`.

We have also developed a kernel (Ramakrishnan 1997) that supports the rapid prototyping of recommender systems. This kernel abstracts the architecture of a recommender system as a layered system with clearly defined subsystems for problem formulation, knowledge acquisition, performance evaluation & modeling and knowledge discovery. Work is underway to create recommender systems for several domains in computational science ranging from domain decomposition to the recommendation of high performance computing architectures.

Issues

- The very nature of the recommendations in this domain (specific advice about the efficacy of software) entails interesting research questions. The software selection problem, even for specific domains in computational science, is a non-trivial one and our recommender systems use a methodology that borrows relevant information from data mining and the performance evaluation of scientific software. One of the more important research issues is the evolutionary improvement in the selection process as more information is acquired.

- The space of applicable algorithms for specific problem subclasses is inherently large, complex, ill–understood and often, intractable by brute–force means. Approximating the problem space by a representation (feature) space also introduces an intrinsic error in the modeling sense.
- Depending on the way the problem is (re)presented, the space of applicable algorithms changes; some of the best algorithms sacrifice generality for performance and have specially customized data structures and routines fine tuned for particular problems or their reformulations.
- There is an inherent uncertainty in interpreting and assessing the performance measures of a particular algorithm for a particular problem. Different implementations of an algorithm produce substantially large variations in performance measures that render relying on purely analytic estimates impractical.
- Recommender systems for PSEs need to communicate effectively with users who have little knowledge about the problems they want to solve plus very little understanding of actual algorithms or their performance. It is our experience that successful recommendations are obtained when information is obtained about either (i) features that are easy to describe or (ii) features that are easy to determine automatically by the recommender system. This has important parallels to areas of AI research such as deliberation scheduling and resource–bounded reasoning.
- The recommender system functionality can also be extended to provide software delivery in directly usable form, in addition to software recommendation. This is one of the most important challenges facing networked scientific computing.
- In a multi–agent context where each agent serves as a recommender system for a specific class of problems, learning and adaptation techniques are required that dynamically 'track' the changes in the abilities of the individual agents and also handle situations where agents appear and disappear over time.

Results

Our design of recommender systems follows a three pronged strategy: performance evaluation of scientific software, (automatic) feature identification and an 'expert' methodology. Our idea focuses on automatic methods to generate and codify performance data, modules that conduct automatic characterization of functions, boundary geometry, conditions and domain topology (and make extensive use of features so determined) and most importantly, use relational descriptions of scientific computing objects to influence software selection.

The important issues in performance evaluation are how best to conduct performance evaluation, deciding on test cases and performance criteria, deciding on the way to represent knowledge about performance data, modes of collaboration with other servers providing such information in a networked setting, etc. Pertinent topics in feature determination are the categories of mathematical entities for which we need to determine features, deciding on a good set of features, the kinds of mechanisms needed to conduct automated feature determination (symbolic/numeric/imagistic/other), is it (at all) economic to conduct automated feature identification (cost factor), elimination of redundancy, etc. Relevant issues in the knowledge methodology are deciding on the ontological entities that need to be represented in the knowledge base, what kinds of reasoning mechanisms are needed, can knowledge from one numerical domain be used for reasoning in another, etc.

As mentioned previously, one of the main goals of a PSE is to reduce the amount of specification code required to state the considered scientific problem, to identify the solver(s) required, to estimate the user defined solver parameters and to control the execution of the code in a targeted computational environment. The PYTHIA kernel supports this functionality in the Postgres RDBMS environment (Stonebraker & Rowe 1986). Fig. 1 depicts the layered architecture of this kernel for the domain of elliptic partial differential equations (PDEs). The PDE problems and methods database management subsystem (PMDM) provides storage for problem and method information, their mathematical, physical characteristics and numerical compatibility information. A parametric macro representation that uses the very high level languages supported by the PELLPACK PSE is adopted for this purpose. The PMDM also identifies classes of PDE problems and methods based on user defined features and characteristics. The knowledge acquisition subsystem (KAS) serves to define experiments involving PDE problems and methods. It generates PELLPACK code, executes them, collects performance data and logs information pertaining to a given codified set of problems. It also facilitates storage of raw performance data using the selected database schema. The performance data management subsystem (PDMS) organizes and transforms performance data and provides support for the editing and manipulation of raw performance data. The performance analysis subsystem (PAS) conducts performance rankings of methods and/or their software parts for user defined objective levels and organizes performance knowledge. The knowledge engine transforms this performance information and automatically generates rules that will be used in knowledge–driven inference. It also manages and assimilates these rules. Finally, the PYTHIA recommender system accepts input from the user at a high level of problem and constraint specification and performs algorithm selection and parameter estimation.

Experiments

We now detail results of applying our methodology to three numerical domains. Exact details of case studies, problem and algorithm populations and results are provided.

The first two case studies form the basis of two recommender systems for PDE solver selection (GAMS categories *I2b1a1a* and *I2b1a3*). Both these studies involve recommending algorithms for classes of linear, elliptic, second–order partial differential equations with rectangular domains (These equations are important in that they describe the steady state behavior of many physical systems.). Accuracy is measured as the maximum absolute error on the rectangular mesh divided by the maximum absolute value of the PDE solution. Performance studies are conducted and the amount of time required to obtain three levels of accuracy — 10^{-3}, 10^{-4} and 10^{-5} — is tabulated. We use linear least squares approximations to the profile data as this allows us to interpolate between the discrete values of the meshes to specify the size necessary to obtain a specified accuracy.

For both the PDE studies, we use the population in (Rice, Houstis, & Dyksen 1981) of 56 linear second–order elliptic partial differential equations. The primary motivation for developing this population was to aid in the performance evaluation of numerical software. Forty two of the PDEs from this population are parameterized which gives rise to a huge number of PDEs, numbering nearly 200. We use several features of these PDEs to aid in algorithm selection. The principal characteristics used are those of the operator and right side (analytic, constant coefficients, nearly singular behavior, oscillatory, jump discontinuity, singular, peaked, singular derivatives, entire), boundary conditions (as being mixed, homogeneous, Dirichlet, Neumann, constant coefficients, etc.) and those of the domain (unit square, variable square, rectangle, etc.). The entire gamut of performance data, error reports, diagnostics, feature information is generated by the PYTHIA kernel and encoded as logic predicates. For determining rules about recommendations, the following methodology was adopted. We first determine a rule that has the maximum cover for the performance data gathered. Then we remove the clauses from the database that conform to this rule. We then repeat the process until no more rules can be found. We perform this procedure for all the three levels of accuracy considered. This imposes a notion of salience on the rules which gives a priority ranking to their ordering.

Methods for Elliptic PDEs with Singularities

For this study, we use 37 PDEs from the population described in (Houstis & Rice 1982). These problems

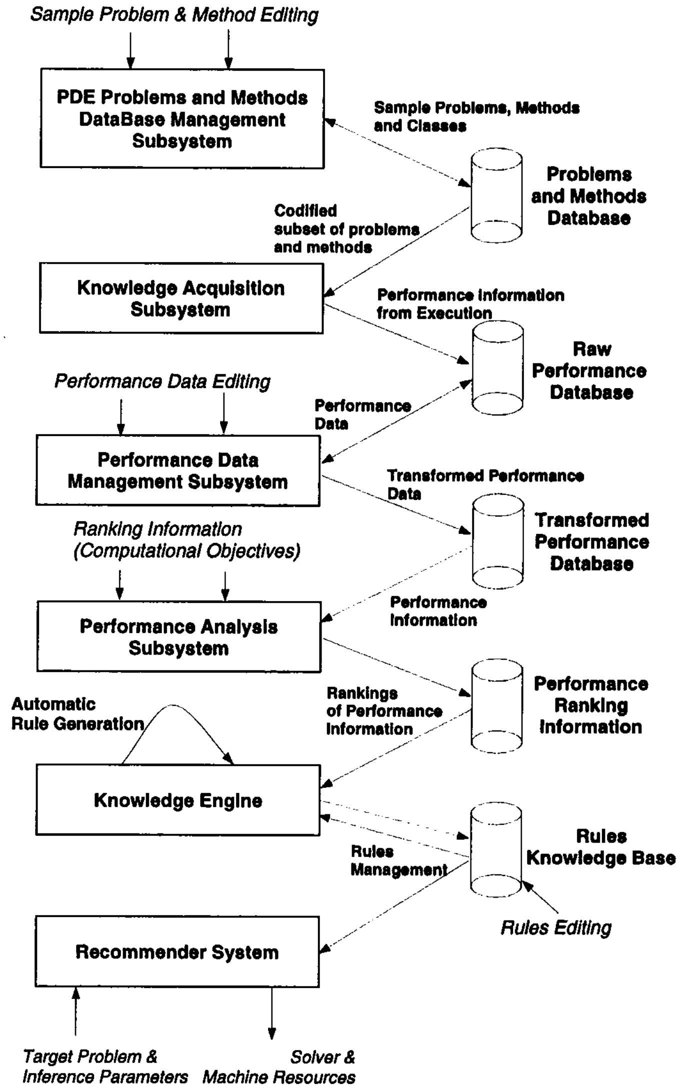
Sample Problem & Method Editing
PDE Problems and Methods DataBase Management Subsystem
Sample Problems, Methods and Classes
Problems and Methods Database
Codified subset of problems and methods
Knowledge Acquisition Subsystem
Performance Information from Execution
Raw Performance Database
Performance Data Editing
Performance Data
Performance Data Management Subsystem
Transformed Performance Data
Transformed Performance Database
Ranking Information (Computational Objectives)
Performance Information
Performance Analysis Subsystem
Rankings of Performance Information
Performance Ranking Information
Automatic Rule Generation
Knowledge Engine
Rules Management
Rules Knowledge Base
Rules Editing
Recommender System
Target Problem & Inference Parameters
Solver & Machine Resources

Figure 1: Functional organization of the PYTHIA kernel. The kernel can be viewed as an automatic recommender system generator.

are encoded as 3-1, 3-2, 7-1, etc. where the first number indicates the generic problem and the second number corresponds to one choice of parameters for the problem. The methods considered in this study are the triple modules from the ELLPACK library (combinations of discretizer, indexing scheme and linear solver) — encoded as 'PS5', 'P3C1C', 'DCG', 'DCG4', 'FFT2', 'FFT4', and 'FFT6.' PS5 is the five-point star module — a second-order finite difference scheme with 'as is' indexing and band Gauss elimination. P3C1C is a fourth-order collocation scheme with Hermite bicubics and Gauss elimination. DCG uses a second-order finite difference scheme, Dyakanov iteration with a generalized marching algorithm and a conjugate gradient method. DCG4 is the same as DCG except that it uses Richardson extrapolation to obtain a fourth-order scheme. FFT2, FFT4 and FFT6 are second, fourth and sixth order finite difference schemes with Fast Fourier Transforms.

The rules discovered confirm the statistically discovered conclusion in (Houstis & Rice 1982) that higher order methods are better for elliptic PDEs with singularities. They also confirm the general hypothesis that there is a strong correlation between the order of a method and its efficiency. More importantly, the first ten rules discovered impose an ordering of the various solvers for each of the problems considered in this study. Interestingly, this ranking corresponds almost exactly with the subjective rankings published in (Houstis & Rice 1982). This shows that very simple rules capture much of the complexity of algorithm selection in this domain. There were several other interesting inferences drawn. Whenever the DCG method is best, so is DCG4. The rule that had the maximum cover from the data was the one which stated that FFT6 is best for a PDE if the PDE has a Laplacian operator, homogeneous and Dirichlet boundary conditions and discontinuous derivatives on the right side. Some of the rules confirm this by recording the significant presence of a Laplace operator in a majority of the PDE population. Other rules also indicated when a certain method is inappropriate for a problem. The FFT6 module, for example is a 'bad' method whenever the problem has boundary conditions with variable coefficients. There are many more such interesting observations and we mention only the most interesting here. Finally, an approximate ordering was requested for the overall population. This gave rise to the ordering — FFT6, FFT4, FFT2, DCG4, DCG2, PS5. This is pertinent because this ranking corresponds most closely to that for Poisson problems which formed the bulk of our population. In overall, the rules from this study performed best algorithm recommendation for 100% of the cases.

Methods for Elliptic PDEs with Mixed Boundary Conditions For this study, we again use PDEs from (Rice, Houstis, & Dyksen 1981), but chose specifically those that have mixed boundary conditions. The main purpose of this study is to determine whether the introduction of derivative terms in the boundary conditions[1] causes any change in the relative efficacies of various solvers. We consider 5 modules, one finite element scheme and four others that use finite differences — 'HC', 'PS5', 'DCG', 'DCG4' and 'MG.' HC is a fourth order finite element discretization scheme that uses collocation with Hermite basis functions, scaled partial pivoting and band Gauss elimination. MG is a second order central finite difference scheme that uses a multigrid technique for the linear system solution. The remaining solvers are the same as those for the previous study.

Results from the methodology again resulted in the confirmation of the hypothesis in (Dyksen, Ribbens, & Rice 1988) that the introduction of the derivative in the boundary conditions does result in a degradation of performance of all the five modules. In particular, the same rules were inferred when the boundary condition was changed from Neumann to Dirichlet but the 'covers' influencing these rules was found to come down in magnitude. This is akin to a loss in the relative level of confidence for the difference between the performances of the modules as described in (Dyksen, Ribbens, & Rice 1988). For this domain, 92% of the recommendations inferred the best solving modules, 3% of them were second best for the population considered and 5% of the selections, while not optimal, still achieved the performance criteria requested.

The rules also confirm the general belief that the finite element collocation scheme (HC) is the least affected by the introduction of derivatives in the boundary conditions and the 5-point star module (PS5) is the most affected. This also explains why the multigrid method (MG) and the Hermite collocation (HC) solver are most suited for these problems.

The final study involved recommending algorithms for one-dimensional numerical integration (GAMS category *H2a*). Given a problem in numerical integration and constraints on time and accuracy, our recommender system — GAUSS — selects an efficient algorithm to solve it. It is to be emphasized that GAUSS (Ramakrishnan & Rice 1996) does not *evaluate* integrals (which is a popular bed for demonstrating concepts in AI) but only recommends algorithms to evaluate them.

The libraries from which the routines are obtained are QUADPACK, NAG, IMSL, PORT, SLATEC, JCAM and the collected algorithms of the ACM (TOMS). We have utilized 124 routines from various sites in this study.

We have used a wide variety of test integrands, most of them with special properties. The total number of test integrands used in this study is 286. The integrands are selected so that they exhibit interesting or

[1] A mixed boundary condition is one which is described using both the solution across the boundary and its outward-pointing normal derivative.

common features such as smoothness (or its absence), singularities, discontinuities, peaks, and oscillation. Some of the functions were selected so that they satisfy the special considerations on which some algorithms are designed (For example, the routine QDAWO requires that its argument contain a sine or a cosine.). Most of the functions are parameterized which generates families of integrands with similar features and characteristics – this aids in the generalization of the system. The number of experiments thus rises to a huge number (286 functions times 124 routines times 10 levels of accuracy = $354,640$). However, some routines are not applicable to quite a few integrands and results for a whole family of integrands can be quickly and automatically determined by scripting programs in GAUSS.

On examining the rules discovered by GAUSS, we observe several heuristics about the domain of numerical integration and associated algorithms. It was found, for example that the adaptive algorithms use fewer function evaluations to achieve high–accuracy results than their non–adaptive counterparts; conversely, they use more evaluations to meet low–accuracy constraints. A high accuracy adaptive algorithm has been found to be more suitable for an oscillating integrand. This could possibly be due to the fact that in an oscillating function, subdivisions are spread over the entire domain of integration and hence a smaller number of subdivisions are required to achieve a fairly high degree of accuracy. Conversely, integrands with singularities or peaks are more amenable to low and medium accuracy adaptive routines. Finally, GAUSS has helped identify 'redundant' algorithms, i.e., algorithms which perform almost exactly the same for the test functions considered in this work. For example, on examining the output generated from GAUSS, it was found that the rules selecting the algorithms DPCHIA, DCSITG and DCSPQU contained the same antecedents. On further examination, one sees that the performance data for these algorithms is nearly the same. All of these are routines specially tailored to handle tabulated data — DPCHIA evaluates the given integral using piecewise cubic Hermite functions, DCSITG evaluates the integral of a cubic spline and DCSPQU also uses spline interpolation. Thus, these three routines are mathematically very similar and incidentally, they yield the best overall performance for problems specified as a table of values. In overall, GAUSS recommended the best known algorithm for 87% of the cases, selected the second best algorithm 7% of the time and a reasonable one for 3% of the time.

Summary

The above encouraging results with the prototyping methodology indicate that recommender systems for domain specific problem solving environments are feasible. The PYTHIA kernel can be used to automate the construction of such systems for many related scientific and mathematical domains. In addition to recommending algorithms/software for the above described classes of problems, the PYTHIA system also interfaces with the GAMS ontology to provide real–time software indexing on the WWW. A more elaborate description of this facility is provided in (Ramakrishnan *et al.* 1997). We also direct the interested reader to (Ramakrishnan 1997) where we describe results that demonstrate the feasibility of tracking multiple recommenders with varying abilities.

References

Boisvert, R. F. 1996. The NIST Guide to Available Mathematical Software. URL: **http://gams.nist.gov**.

Dyksen, W.; Ribbens, C.; and Rice, J. 1988. The Performance of Numerical Methods for Elliptic Problems with Mixed Boundary Conditions. *Numerical Methods for Partial Differential Equations* Vol. 4:pages 347–361.

Gallopoulos, E.; Houstis, E.; and Rice, J. 1994. Computer as Thinker/Doer: Problem-Solving Environments for Computational Science. *IEEE Computational Science and Enginerring* Vol. 1(2):pages 11–23.

Houstis, E., and Rice, J. 1982. High Order Methods for Elliptic Partial Differential Equations with Singularities. *International Journal for Numerical Methods in Engineering* Vol. 18:pages 737–754.

Houstis, E., and Rice, J. 1992. Parallel ELLPACK: A Development and Problem Solving Environment for High Performance Computing Machines. In Gaffney, P. W., and Houstis, E. N., eds., *Programming Environments for High-Level Scientific Problem Solving.* North–Holland. 229–243.

Ramakrishnan, N., and Rice, J. 1996. GAUSS: An Automatic Algorithm Selection System for Quadrature. Technical Report CSD-TR-96-048, Department of Computer Sciences, Purdue University.

Ramakrishnan, N.; Houstis, E.; Joshi, A.; Rice, J.; and Weerawarana, S. 1997. Intelligent Networked Scientific Computing. In *Proceedings of the 15th IMACS World Congress*, 785–790. Wissensshaft and Technik Verlag.

Ramakrishnan, N. 1997. *Recommender Systems for Problem Solving Environments.* Ph.D. Dissertation, Dept. of Computer Sciences, Purdue University.

Rice, J.; Houstis, E.; and Dyksen, W. 1981. A Population of Linear, Second Order, Elliptic Partial Differential Equations on Rectangular Domains, Part I. *Mathematics of Computation* Vol. 36:pages 475–484.

Stonebraker, M., and Rowe, L. 1986. The design of POSTGRES. *SIGMOD Record* Vol. 15(2):pages 340–355.