Note on Generalization in Experimental Algorithmics

Naren Ramakrishnan
Virginia Polytechnic Institute and State University
and
Raúl E. Valdés-Pérez
Carnegie Mellon University

A recurring theme in mathematical software evaluation is the generalization of rankings of algorithms on test problems to build knowledge-based recommender systems for algorithm selection. A key issue is to *profile* algorithms in terms of the qualitative characteristics of benchmark problems. In this methodological note, we adapt a novel all-pairs algorithm for the profiling task — Given performance rankings for m algorithms on n problem instances, each described with p features, identify a (minimal) subset of p that is useful for assessing the selective superiority of an algorithm over another, for all pairs of m algorithms. We show how techniques presented in the mathematical software literature are inadequate for such profiling purposes. In conclusion, we also address various statistical issues underlying the effective application of this technique.

Categories and Subject Descriptors: G.4.2 [**Mathematical Software**]: Certification and Testing

General Terms: Experimental Algorithmics

Additional Key Words and Phrases: Profiling, Performance Evaluation, Benchmark Studies

1. INTRODUCTION

The testing and comparative analysis of numerical algorithms is a significant aspect of mathematical software evaluation. Experimental algorithmics deals with the design of methodologies for the consistent evaluation of algorithms on realistic test problems, interpreting the results of evaluations (rankings etc.) and generalizing such interpretations within the problem domain. While other studies have underscored the importance of effective design and reporting procedures [Crowder et al. 1979], our goal in this brief methodological note is to address the task:

Profiling: Given performance rankings for m algorithms on n problem

This work was partially supported by National Science Foundation grant #EIA-9974956 to Ramakrishnan and National Science Foundation grants #IRI-9421656 and #IIS-9819340 to Valdés-Pérez. Authors' addresses: Naren Ramakrishnan, Department of Computer Science, Virginia Polytechnic Institute and State University, VA 24061, USA. Raúl E. Valdés-Pérez, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213-3891, USA.

Problem	Laplace?	Square Domain?	Const. Coeff.? (Right Side)	Algorithm
p1	Y	Y	Y	FFT6
p2	Y	Y	Y	FFT6
p3	N	N	Y	DCG4
p4	N	N	N	DCG4
p5	Y	N	N	COLL
p6	Y	N	Y	COLL

Fig. 1. Best algorithm performances for 6 elliptic PDEs. FFT6, DCG4 and COLL refer to PELLPACK modules for sixth order FFT 9-point differences, Dyakunov conjugate gradient with fourth order accuracy and Collocation with band Gauss elimination, respectively.

> instances (each with p descriptive features), identify a (minimal) subset of p that discriminates the performance of an algorithm over another, for all pairs of the given algorithms

Such generalizations are important in the design of recommender systems [Ramakrishnan 1997], performance evaluation systems [Boisvert et al. 1979; LaRose 1993], problem solving environments [Houstis et al. 1998], and in the knowledge discovery of qualitative information from scientific experiments [Valdes-Perez 1999]. The traditional approach is to use a decision tree algorithm (or other machine learning technique) to learn a classifier from the problem feature space to the algorithm space, using the performance measures to identify good mappings. While such generalizations are shown to achieve high predictive accuracy on independent test sets of problems, we contend that they are inadequate for determining high-level descriptors of algorithm performance.

The rest of this note is organized as follows: In Section 2, we identify drawbacks with common generalization approaches and present situations where traditional techniques fail. The profiling algorithm is presented next, followed by a careful experimental evaluation on a testbed of elliptic PDE problems. We also address various statistical and methodological issues underlying the application of the profiling technique.

2. EXAMPLES

Typically machine learning and AI techniques have been used to construct classifiers that (i) take a problem description as input and map it to a 'best' algorithm, or (ii) take a problem description and an algorithm as input and map the combination to the expected ranking of the algorithm on the given problem instance. For example, in a PDE problem solving context, a problem description would correspond to a list of features (e.g., whether it is Laplacian, has a square domain, and constant coefficients) and the 'best' algorithm/rankings would correspond to evaluation metrics such as order-of-convergence or time required to achieve a certain level of accuracy.

Fig. 1 describes a scenario that contains data pertaining to six PDE problems, each with three symbolic features, and involving three algorithms. For simplicity assume that we are interested only in the best algorithm performance and that the performance rankings of algorithms for each problem instance are distilled into this form, as shown in Fig. 1.

(1) Consider using a decision-tree classifier to learn the data shown in Fig. 1. Deci-

sion tree approaches incrementally add nodes starting from the root, typically choosing decision criteria at each internal node that maximize an information-theoretic measure. The end result is a classifier that can be used as a discriminator for any new problems. For example, the information required (in bits) to classify the data in Fig. 1 is given by:

$$\begin{aligned} I &= -\frac{2}{6}\log\frac{2}{6} - \frac{2}{6}\log\frac{2}{6} - \frac{2}{6}\log\frac{2}{6} \\ &= 1.5850 \end{aligned}$$

where the logarithms are taken with respect to base 2. Assume that we split based on the first feature - the presence of a Laplace operator. The information required at this stage will then be:

$$\begin{aligned} I_1 &= -\frac{4}{6}(\frac{2}{4}\log\frac{2}{4} + \frac{2}{4}\log\frac{2}{4}) - \frac{2}{6}(\frac{2}{2}\log\frac{2}{2}) \\ &= 0.6667 \end{aligned}$$

If we split, instead, based on the other two features, the respective calculations are:

$$\begin{aligned} I_2 &= 0.6667 \\ I_3 &= 1.3333 \end{aligned}$$

As noted in [Lucks and Gladwell 1992], traditional decision tree algorithms (such as ID3 [Quinlan 1986; Addison et al. 1991] etc.), use a greedy approach to select relevant features. For example, the gain using each of the above attributes is given by:

$$\begin{aligned} Gain_1 &= 1.5850 - 0.6667 = 0.9183 \\ Gain_2 &= 1.5850 - 0.6667 = 0.9183 \\ Gain_3 &= 1.5850 - 1.3333 = 0.2517 \end{aligned}$$

and hence such algorithms choose the first or second attribute since the information gain is highest in its case. Continuing in this manner, the decision tree for Fig. 1 would construct a profile for the FFT6 algorithm as the conjunction 'Laplace $\wedge$ Square Domain $\wedge$ Constant Coefficients'. However, the technique fails to identify the more concise profile for the FFT6 algorithm, namely 'Laplace $\wedge$ Square Domain'. In addition, it can be observed that these two features are also enough to identify the selective superiority of all pairs of the three algorithms. Notice that while constant coefficients typically play a vital role in the success of FFT algorithms, it is not a useful feature for discriminating FFT6's performance over the other listed algorithms, for the data given in Fig. 1. Decision trees also suffer from other drawbacks — duplication of subtrees in disjunctive concepts (replication) and partitioning of data into fragments, where a high-arity attribute is tested at each node (fragmentation).

(2) Similar difficulties arise with various other learning techniques, such as neural networks [Weerawarana et al. 1996] and other restricted classes of function approximators [Lucks and Gladwell 1992]. For example, using a neural network with the data in Fig. 1 will require that PDE problem features be represented

by a characteristic-vector (for the input layer of a neural network), and that the the output layer correspond to the algorithm whose performance is expected to be best. Such attribute-value based approaches are inadequate because they treat all feature inputs as activations which can be operated upon numerically. Furthermore, it would be difficult to recover the simple profile above from a trained neural network. Thus, profiling techniques must be able to reason about issues in performance evaluation by taking into account the symbolic/categorical nature of relevant problem features (like the presence/absence of a Laplacian operator).

(3) In addition, the profiling approach should be able to isolate as many irrelevant variables (features) as possible from the induced generalization. Consider the study [Houstis and Rice 1982] which evaluates if higher order methods are better for linear elliptic PDEs whose solutions have singularities or similar difficulties. This population includes the pathological example (Problem 54 from the ELLPACK population [Rice and Boisvert 1985]):

$$\begin{aligned}
[(1+x^2)\,u_x]_x + ([1+a(y)^2]\,u_y)_y & \\
-[1+(8y-x-4)^2]u &= f(x,y),\ 0<x,y<1 \\
u &= g(x,y) \text{ for } x=0,1 \text{ and } y=0,1 \\
a(y) &= 4\,y^2+0.9 \\
b(x,y) &= max[0,(3-x/a(y))^3] \\
c(x,y) &= max[0,x-a(y)] \\
d(x,y) &= 0 \text{ if } c(x,y)<0.02 \\
d(x,y) &= e^{-b(x,y)/c(x,y)} \text{ if } c(x,y)\geq 0.02
\end{aligned}$$

where the true solution is

$$2.25\,x\,[x-a(y)]^2\,(1-d(x,y))/a(y)^3 + 1/[1+(8\,y-x-4)^2].$$

The salient feature of this problem is that it has a wildly behaving solution that has singularities. When $a(y)$ is parameterized as $a(y)=4\,y^2+\beta$, for instance, the singularity behavior is observed whenever $x-4\,y^2=\beta$ or $4\,y^2=-\beta$. As the solution has a fairly sharp ridge of considerable complexity (this is caused by the careful choice of the Dirichlet boundary condition), PDE discretizers typically exhibit a 'two-phase' phenomenon. Initially, the grid refinement is used to resolve the sharp ridge, and this phase is characterized by low accuracy and slow convergence while the latter phase has an apparently faster convergence behavior (when the ridge is resolved). As a result, orders of convergence are difficult to estimate from measured data (one accepted practice is to drop the initial, low accuracy points). The discretization phase often results in a matrix that is symmetric, positive definite, and diagonally dominant. For certain specific problem instances, an exact answer can be obtained which makes it difficult to rank algorithms on an order-of-convergence measure. Also, the two-phase phenomenon renders the estimation of convergence quite hard (and sometimes impossible). However, it is observed in experimental runs that a

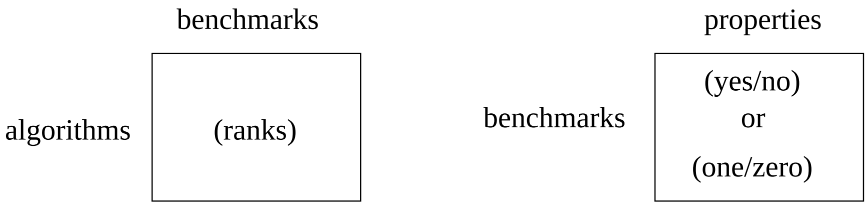

Fig. 2. Input Design for the Profiling Algorithm.

fourth order HODIE method is, in general, better than a second order method, such as the 5-point star algorithm. Given adequate coverage and a representative problem population, a profiling algorithm should be able to obtain the high-level qualitative conclusion "Higher order methods are better" (such as attained in [Houstis and Rice 1982]) despite the other behavioral aspects of individual problems such as presented above.

3. THE PROFILING ALGORITHM

We present a profiling algorithm that uses an all-pairs approach to identify the most concise profiles for a given classification. The input is assumed to consist of rankings of algorithms for each problem described by a sequence of discrete/numeric features. Fig. 2 shows two matrices that contain all the relevant data. The first matrix contains ranks (alternatively, raw measurements if the units of measurement are uniform) for each algorithm on each benchmark and expresses the empirical results. The second matrix expresses our understanding of the properties that each benchmark has. Generally, one is not interested in the performance on a benchmark *per se*, but rather is interested in a more general property that is not directly measurable, but is possessed to some degree by one or more benchmarks. In the present case, this second matrix contains only yes/no (or one/zero) entries, i.e., a benchmark either entirely has or lacks a property.

One could multiply (or join, in the relational database sense) the two matrices together and thus obtain information on the rankings of the alternative algorithms with respect to the various properties. One could further process this result into a judgment of the single best algorithm, depending on one's need for certain of the properties. However, the goal here is not to recommend a single best algorithm, but to extract some insight into the comparative strengths of each algorithm.

The overall procedure is described in Fig. 3 and has three main stages. The first stage is a brute-force computation of a CNF that conjoins feature lists for all ${}^mC_2$ combinations of algorithms. Each feature list contains a list of attributes that can contrast the given pair of algorithms. This CNF is then simplified using standard problem-reduction techniques [Valdes-Perez et al.] and then converted to a DNF that can yield smallest profiles for each of the algorithms. Since there may exist more than one such minimal profile for each algorithm, the final stage involves a further inner-loop that minimizes the use of features across profiles for all the algorithms. To tune the contrast between algorithms represented by the profiles, the algorithm allows the specification of the ceiling of overlap allowed to infer contrast. Though this algorithm is intractable in the worst case (due to the

```
Stage 1:
For each $(i, j) \in$ algorithms
    Form a disjunction of features that can contrast $i$ from $j$
    in terms of its selective superiority over the problems

Form a CNF of these disjunctions by conjoining them

Stage 2:
Simplify the CNF using term-rewriting subsumptions

Convert CNF → DNF

Use the DNF to identify smallest profiles for each algorithm

Stage 3:
For every algorithm that has multiple profiles
    Choose the profile that
    maximizes coordination across the algorithms
```

Fig. 3. The All-Pairs Profiling Algorithm.

CNF → DNF conversion, [Garey and Johnson 1979]), there are various domain-specific considerations that render it practical for experimental algorithmics. More details of the basic profiling methods are available [Valdes-Perez et al.].

To illustrate the operation of the profiling technique, consider its application to the data in Fig. 1. Stage 1 results in the CNF:

```
(and (or L S)
     (or S)
     (or L))
```

The first line, for instance indicates that the algorithms FFT6 and DCG4 can be contrasted using either the L (Laplace) or the S (Square Domain) feature. The second line indicates that only the feature S can distinguish between FFT6 and COLL, and so on. Notice now that since (or S) and (or L) subsume (or L S), the latter can be safely deleted. We thus obtain the CNF:

```
(and (or S) (or L))
```

and its equivalent DNF:

```
(or (and S L))
```

In this case, there is only one disjunction with the two features S and L. In general, this stage of the algorithm involves choosing one of the various simplest disjunctions. We now identify the profiles for each of the algorithms using these attributes. Notice that the data shown in Fig. 1 eliminates the need for Stage 3, since there are no alternative profiles for any given algorithm.

While the all-pairs technique has exponential complexity in the worst case, it is our experience that various considerations in experimental algorithmics ensure that it works very well in practice. For example, numerical algorithms are frequently designed to address special problem difficulties, hence such aspects can subsume other differences in pairwise comparisons of algorithms. For example, the Dyakunov CG algorithm is only applicable when the operator is in self-adjoint form, so inducing a more general construct will not be useful. In addition, there are various approximation algorithms, such as the set-covering heuristic that can substantially reduce the search space with graceful degradation in overall accuracy.

The goal of the profiling approach is to gain a broad qualitative understanding of algorithm performance to guide further detailed studies. In this respect, the all-pairs technique is a useful analytical tool.

4. EXPERIMENTAL STUDIES

First we describe the statistical issues involved in setting up the benchmarking data before profiling is even attempted. Then we present the profiling results and relate these to previous reports on the same data.

4.1 Statistical Issues

The first step involves ensuring that the rankings of algorithms are statistically significant to permit generalization. This requires selecting a family of statistical methods, determining a suitable level of significance, and finally performing an appropriate hypothesis test. We address each of these aspects below:

—*Selection of a Statistical Technique:* On a fundamental modeling note, the experimental algorithmicist has a choice of parametric vs. non-parametric methods. The former are typically appropriate when a distributional assumption (most often, normality) can be verified to hold, whereas the latter assume no prior knowledge on the distribution of samples. Irrespective of the method of choice, a decision needs to be made regarding the nature of the null hypothesis being tested. This is not always an easy selection because the implications of deciding in one direction might be more serious than the other. To a certain extent, it depends on the final use of the results of the statistical test. For example, consider testing the hypothesis that a new algorithm is substantially better than various others. Selecting in favor of this hypothesis produces immediate advantages for an automated recommender system for numerical algorithms. However, rejecting this hypothesis has implications for understanding the behavioral aspects of the underlying problem population! Depending on the situation, the actual hypothesis being tested might need to be reformulated. If parametric methods are chosen, the results can be sensitive to the actual values of the observations. For example, using the log of the error measured on a PDE grid instead of the actual value can adversely affect the $p-$value in a t-test. Furthermore, the literature on parametric methods demonstrates that the presence of outliers can influence the final result.

—*Determining a Level of Significance:* Statistical tests have not been originally designed for the comparative analysis of algorithms. Thus, care must be taken to ensure that they are tailored appropriately for the situation. For example, how does one select an appropriate level of significance? Assume that we are interested in pairwise tests of significance at the 0.03 level. One could conduct ${}^{m}C_2$ tests for each of the n problems in turn and use a simple paired t-test to determine if the comparisons are significant. The flaw in this line of reasoning arises from the fact that even if each of the tests has a probability of 3% of a Type I error (rejecting the hypothesis when it is actually true), it doesn't necessarily imply that together, they have the same probability of Type I error! It is easily shown that the true probability of a Type I error, instead, is actually given by $1 - (0.03)^{{}^{m}C_2}$. For large m, this figure approaches 1! The straightforward fix to this problem is to recalculate the significance level of individual tests for a given composite level of significance. As a good first approximation, each individual comparison can be tested for the required level of significance divided by ${}^{m}C_2$. This is commonly referred to as the *Bonferroni correction* and studies that do not pay close attention to this detail have earned the notorious title of 'fishing expeditions.'

—*Hypothesis Testing:* The general multi-treatment analysis technique is the distribution-free Friedman, Kendall and Babington-Smith test [Hollander and Wolfe 1973]. The goal is to test the hypothesis that the algorithms are equally effective versus the alternative that there is an ordering/ranking among them. Since this is a non-parametric method, the first step is to replace the absolute measurements by means of ranks. Within each row, the algorithms are ranked from least to highest (according to the performance measure) and assigned 'ranks'. The average of the ranks is then computed for each column. This scheme serves two useful purposes: (a) by computing ranks within each row first, it offsets any natural advantages that one problem provides over other problems, and (b) by averaging ranks across the columns it ensures that the results are invariant under monotonic transformations. It also addresses (to a certain extent) the noise and outliers issue inherent in the PDE population. A test statistic is then computed and hypothesis testing proceeds depending on the level of significance. We refer the interested reader to ([Hollander and Wolfe 1973], Chapter 7) for more details. In addition, if the various algorithms (treatments) can be ordered, then a more efficient distribution-free test (the Page test) exists that can be used to test the null hypothesis. For example, solving a PDE with a varying number of discretization nodes is one instance where the treatments have a natural ordering. The Page test is more appropriate when a particular ordering of the algorithms is being tested, since it uses a statistic that is sensitive to the orderings. In this article, we have utilized a least-squares approximation to arrive at a more high level descriptor (described below).

4.2 Results

To validate this methodology and the all-pairs technique, we considered the benchmark set involving 37 PDEs from the population described in an earlier *Transactions on Mathematical Software* article [Weerawarana et al. 1996] and eight PELLPACK modules — 'PS5', 'P3C1C', 'DCG', 'DCG4', 'HODIE', 'FFT2', 'FFT4', and 'FFT6' — used in the same study. 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, Dyakunov 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

PS5 is
- better than DCG2, DCG4
  - for problems with smo_cc
- much better than P3C1C
  - for problems with smo_e
- better than all others
  - for problems with s_vs

P3C1C is
- much better than DCG2, DCG4
- better than PS5, HODIE
  - for problems with bc_c
- much better than FFT2
- better than PS5, HODIE
  - for problems with rs_c
- much better than for PS5
  - for problems with rs_h
- better than for FFT4, FFT6
  - for problems with rs_s

FFT6 is
- much better than PS5, P3C1C, DCG2, DCG4, FFT2
  - for problems with op_laplace
- much better than FFT2, FFT4
  - for problems with rs_s
- much better than P3C1C, HODIE, FFT2, FFT4, DCG2, DCG4
  - for problems with smrhs_s

DCG2 is
- much better than P3C1C
- better than PS5
  - for problems with op_selfadj
- much better than PS5, DCG4
  - for problems with smo_dd
- better than PS5, P3C1C
- somewhat better than DCG4
- slightly better than HODIE, FFT2, FFT4, FFT6
  - for problems with s_wf

Fig. 4. Results from profiling 8 PELLPACK modules on 37 PDE problem instances. The figure demonstrates four such profiles.

fourth–order scheme. HODIE is a fourth-order finite difference method with Gauss elimination. The triple modules FFT2, FFT4, and FFT6 are second, fourth and sixth order finite difference schemes with Fast Fourier Transforms. Both DCG and DCG4 provide linear system solution by preconditioned conjugate gradient iteration. Parameters such as the maximum number of CG iterations allowed and the intial guesses are assigned to allow a fair comparison to the other modules that use direct solvers (as described in [Weerawarana et al. 1996]). Each algorithm was applied to each problem with a varying number of discretization nodes (for example, PS5 utilized 5 × 5, 9 × 9, 17 × 17, 33 × 33, and 65 × 65 grid sizes, and so on.) and a least squares fit was used to determine the time required to achieve a relative error of less than 10^{-5}. This was used as the primary measure for ranking algorithms. This, so far, is essentially the procedure as adopted in [Weerawarana et al. 1996]. To ensure that the differences between the algorithms were significant at the 95% confidence level, we used the Friedman, Kendall, and Babington-Smith test described earlier.

Once the rankings are shown to be significant, we proceed with the generalization step. The principal characteristics used in this study are those of the operator and right side (e.g. analytic, constant coefficients, nearly singular behavior, oscillatory, jump discontinuity, singular, peaked, singular derivatives, entire), boundary conditions (e.g. mixed, homogeneous, Dirichlet, Neumann, constant coefficients) and those of the domain (e.g. unit square, variable square, rectangle). The features are encoded as `smo_cc`, `rs_c` following the convention originally presented in [Rice et al. 1981] for distinguishing features of the PDE problem. The prefix identifies the nature of the feature and the suffix identifies the specific value of the feature. For example, `smo_cc` implies that the 'smoothness of the operator' is 'computationally complicated' and so on. The other prefixes `op`, `rs`, `s` correspond to the PDE operator, right side and solution properties (if known).

Since the goal of this study was to identify the selective *superiorities* of algorithms, we extended the all-pairs profiling methods in [Valdes-Perez et al.] to ensure that only comparative advantages of algorithms are reported, not deficits. Thus, a concise profile such as "this algorithm is worse than every other algorithm on problems having the property P" is rejected in favor of seeking positive statements.

We present four such profiles obtained by the all-pairs algorithm in Fig. 4. The first profile indicates that when both methods are applicable, the five-point star algorithm is contrasted from both the Dyakunov algorithms in its superior performance on problems whose operator smoothness is computationally complicated. It is better than all other algorithms for problems whose solutions have varying smoothness, and so on. Notice that this latter profile is *minimal*, since it is sufficient to contrast the PS5 algorithm from all others. The other profiles are reported as interesting, since they (a) identify features that help absolutely contrast other combinations of algorithms and/or (b) use the `smo_cc` feature which was an important factor in the design of the original PDE population.

In addition, quantitative information is easily obtainable by examining the profiles carefully. At a high level, the overlap ceiling helps determine if the performances were absolutely or partially contrasted. One could then examine the Friedman rank sums from the statistical test to make statements of the form 'For problems with feature `op_laplace`, algorithm FFT6 performed better than algorithm P3C1C by at least 4 ranks.' And finally, the examination of individual performance profiles of algorithms (either error vs. grid-size, or error vs. total time), helps transform the relative rank differences to differences in performance measures.

It should be noted that for the above data, the PYTHIA system described in [Weerawarana et al. 1996, page 465] reported only 15% valid algorithm recommendations! In other words, 85% of the recommended algorithms were not even applicable to the presented problems. The main source of difficulty comes from a PDE characteristic vector

$$\left(\vec{O}, \vec{BC}, \vec{F}, \vec{D}\right)$$

where each of the subvectors encode properties of the PDE operator, boundary conditions, functions and the domain. This design (i) gives equal emphasis to all features of the PDE problem and (ii) gives a numeric interpretation to ordinal and symbolic features, which is not appropriate. The all-pairs technique presented here can help avoid such invalid predictions. For example, the concise profiles presented in Fig. 4 can help identify the relevant features and guide further studies. Since the technique does not utilize a greedy approach (such as a forward or backward feature selection facility), all concise profiles will be determined.

5. CONCLUDING REMARKS

This article has described an all-pairs profiling technique that adapts and extends the methods in [Valdes-Perez et al.] for the purpose of extracting concise qualitative insights from experimental algorithmic studies on the comparative advantages of alternative algorithms. We applied the technique to prior data on codes for solving partial differential equations, with good results. The discovered profiles correspond to the subjective rankings first made in [Houstis and Rice 1982]. Continuous-valued feature attributes can also be accommodated in the profiling technique by (i) first sorting and bucketing the measured values of the feature and (ii) subsequently choosing a test (with a boolean/symbolic value) based on the class distribution and the subsets induced by the test.

REFERENCES

Addison, C., Enright, W., Gaffney, P., Gladwell, I., and Hanson, P. 1991. Algorithm 687: a Decision Tree for the Numerical Solution of Initial Value Ordinary Differential Equations. *ACM Transactions on Mathematical Software Vol. 17,* 1 (March), pp. 1–10.

Boisvert, R., Rice, J., and Houstis, E. 1979. A System for Performance Evaluation of Partial Differential Equations Software. *IEEE Transactions on Software Engineering Vol. SE-5,* 4 (July), pp. 418–425.

Crowder, H., Dembo, R., and Mulvey, J. 1979. On Reporting Computational Experiments with Mathematical Software. *ACM Transactions on Mathematical Software Vol. 5,* 2, pp. 193–203.

Garey, M. and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco.

Hollander, M. and Wolfe, D. 1973. *Nonparametric Statistical Methods.* John Wiley and Sons.

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*, pp. 737–754.

Houstis, E., Rice, J., Weerawarana, S., Catlin, A. C., Papachiou, P., Wang, K., and Gaitatzes, M. 1998. Parallel ELLPACK: A Problem Solving Environment for PDE Based Applications on Multicomputer Platforms. *ACM Transactions on Mathematical Software Vol. 24,* 1 (March), pp. 30–73.

LaRose, B. 1993. The Design and Implementation of a Performance Database Server. Technical Report CS-93-195 (August), University of Tennessee.

Lucks, M. and Gladwell, I. 1992. Automated Selection of Mathematical Software. *ACM Transactions on Mathematical Software Vol. 18,* 1 (March), pp. 11–34.

Quinlan, J. 1986. Induction of Decision Trees. *Machine Learning Vol. 1,* 1, pp. 81–106.

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

Rice, J. and Boisvert, R. 1985. *Solving Elliptic Problems Using ELLPACK.* Springer Verlag, New York.

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*, pp. 475–484.

Valdes-Perez, R. E. 1999. Principles of Human Computer Collaboration for Knowledge Discovery in Science. *Artificial Intelligence 107,* 2, pp.335–346.

Valdes-Perez, R. E., Pericliev, V., and Pereira, F. Concise, Intelligible, and Approximate Profiling of Numerous Classes. *International Journal of Human Computer Studies*. in press (special issue on Machine Discovery).

Weerawarana, S., Houstis, E., Rice, J., Joshi, A., and Houstis, C. 1996. PYTHIA: A Knowledge Based System to Select Scientific Algorithms. *ACM Transactions on Mathematical Software Vol. 22*, pp. 447–468.