

PARETO ANNOTATION OF THE DENDOGRAM OF PERFORMANCE DATA*

JOËL M. MALARD[†]

Abstract. The dendrogram is a fundamental data mining tool which becomes unwieldy when the number of data points exceeds several thousands. It is shown how this limitation can be alleviated through annotations based on Pareto dominance of the partitions represented by the nodes of the dendrogram. The approach is illustrated using software performance data gathered using a public domain LAPACK library.

Key words. data-mining, tree visualization, Pareto optimality

1. Introduction. Porting a software library across computers of different architectures entails more than running the existing code on the target machine. Performance can suffer much from a straight-forward port. Often not only internal parameters, such as block sizes, much be changed but algorithmic implementations must be changed as well. Thus, performance tuning can be a major operational cost.

Deep memory hierarchies ranging from registers, several levels of cache and to possibly remote memory or disk as well as other advances in computer architectures all increase the complexity of software performance tuning, see for example[3].

Agile software libraries that implement the Basic Linear Algebra Subroutines (BLAS) interface such as ATLAS [10] and PHIPAC [1] overcome this problem elegantly by calibrating some of their internal parameters during an initial training phase. Some questions still arise for more complex libraries such as LAPACK that includes over 1,000 callable subroutines, that can each operate on arguments with a wide variety of sizes. One such question is: does there exist a small representative set of library calls that could be effectively used for training? Another related question is which performance metric to use to guide the calibration? Although the ultimate goal of software performance optimization is to reduce wall-clock time the necessary calibration of subroutine parameters and source code restructuring is often guided by other metrics such as cache or page misses.

Those questions are not answered directly in this paper but a methodology is proposed that can help answer them through a novel way to look at dendrograms. The same approach can be applied to life-science or meteorological data for instance. Section 2 focuses on the exploration of structure, such as sub-clusters and outliers, within the dendrogram through the classification of hardware event measurements across multiple calls to a public domain LAPACK library. Section 3 focuses on the compression of large dendrograms and cluster selection through the classification of subroutine calls based on hardware events.

2. Regrouping Hardware Performance Counters. Advances in computer hardware and architecture have enabled the development of hardware event counters that tally on the processor chip such things as data and instruction cache misses, branch mis-prediction, etc. These hardware performance counters are becoming more widely available to application programmers through Application Programming Interfaces (API) such as PAPI, RS2HPM, Rabbit, etc. Relations between counter values

*THIS PAPER WAS SUBMITTED TO THE SIAM WORKSHOP ON DISCRETE MATHEMATICS AND DATA MINING ON JANUARY 4TH, 2002

[†]Computational Science and Applied Mathematics Division, Pacific Northwest National Laboratory, Battelle Boulevard, PO Box 999, Richland, WA 99352 (jm.malard@pnl.gov).

across various calls to a software library can lead to further insight into any performance bottleneck.

The performance data that is depicted in Figure 3.2 was gathered by running the timing program `xlintimd` that is distributed along with the public domain LAPACK library available on netlib through the URL <http://www.netlib.org>. This program was run on a SGI Octane with two 175 MHz R10K CPUs using the PAPI interface. All timed LAPACK subroutines were called once for each measurable hardware event. The resulting data was organized into a 24 by 3391 table where each row corresponds to a hardware event and each column corresponds to a single call to the LAPACK library. The data is slightly incomplete with 8 values missing. Each row of the data was scaled to have its entries in the range 0 through 1. This prevents large counts to dominate the clustering. A larger data set is being compiled on newer machines by monitoring the LAPACK library across a wider spectrum of test cases.

A hierarchical clustering algorithm was applied to the rows of the data table. Further details about clustering algorithms and dendrograms can be found for example in [6]. The most interesting output of the clustering algorithm is a weighted tree, called a dendrogram and shown in Figure 3.2. Each fork of this dendrogram corresponds to a decision to merge two rows or two clusters into a single larger cluster. Since forks are generated in increasing weight of their parent (here downward) arc, each fork in the dendrogram corresponds to a unique partition of the hardware events. Each decision to merge two objects into a cluster affects the overall quality of the resulting partition. Two conflicting goals must be juggled with. On one hand, the output clusters should be as homogeneous as possible. On the other hand, these clusters should be as distinct as possible. Several definitions of homogeneity and separation exist and are in use; the exact definitions used for this paper are not essential to the exposition. For a partition $P = (C_0, C_1, C_k)$ of the hardware events into k clusters, its separation will be denoted $S(P)$ and its homogeneity will be denoted $H(P)$. By extension, we define the separation and homogeneity of a fork F in the dendrogram to be those of the corresponding partition and we denote them $S(F)$ and $H(F)$. For clarity sake, let us assume that the higher S and H are the better.

The forks of the dendrogram can be ranked based on how well the corresponding partition fares in terms of both homogeneity and separation. This ranking makes use of the concept of Pareto dominance. A fork F is said to dominate another fork F' if 3 conditions are met: $H(F) \geq H(F')$, $S(F) \geq S(F')$ and at least one of the previous two inequalities are strict. We shall say that a dendrogram fork is dominant if no other fork in the dendrogram dominates it in the above sense. Each of the shaded ellipses in Figure 3.2 highlights the subtree corresponding to a dominant fork. Two clusters emerge from Figure 3.2. One is formed of events 0 through 6, and corresponds to various types of level 1 and 2 cache misses; call it C_c . The other large cluster, C_o consist of events 8 through 16 and 20 through 23. The events 2 and 5 that form a sub-cluster of C_c correspond respectively to the total number of level-2 data cache misses and the total number of level-2 cache misses (including the instruction cache). The level-2 instruction cache misses (event 3) are more closely related to level-1 instruction cache misses (event 1). Inspection of the bigger cluster C_o shows that the rate at which all instructions are completed (event23) and the rate at which floating points operations are completed (event 21) are more closely related to the completion of conditional store operations: failure (event 11), success (event 12) and total (event 13). The fact that events 11 and 13 form a sub-cluster of their own suggests that any performance improvements for this set of library functions in this

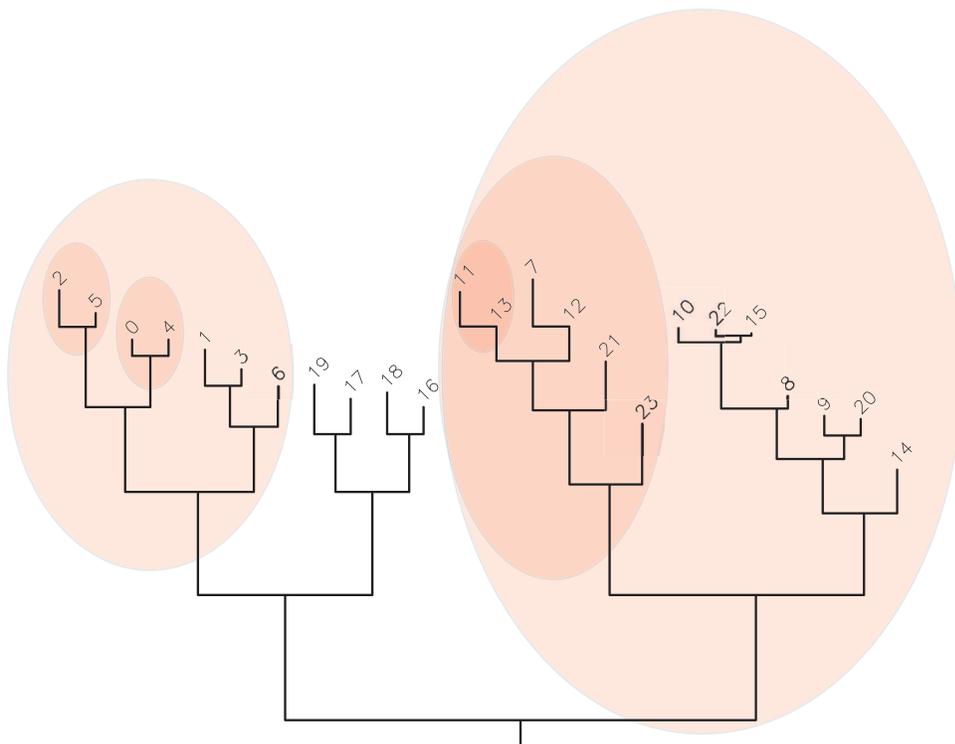


FIG. 2.1. *Annotated dendrogram shows two clear clusters of hardware events as well as sub-cluster structure*

computing environment would come from reducing the number of failed conditional store instructions.

The events 16 through 19 that belong to neither C_c nor C_o correspond respectively to total counts of instructions, floating point instructions, memory load and memory store instructions completed. Those measures do not relate well with the total number of cycles spent (event 22) (or wall clock time). The fact that these events form a cluster that does not add to the quality of the partition would be hard to detect from looking at the dendrogram alone.

3. Regrouping LAPACK Calls. Let us now turn from the classification of hardware events to the classification of library calls. Such a classification may help concentrate the performance optimization effort on a few representative cases. Many fast clustering algorithms, such as k-means, that work well with large data sets require that the user specify how many clusters will be output. Hierarchical clustering algorithms on the other hand are more flexible in this regard. Typically, each fork F of the dendrogram is assigned a figure of merit that is some increasing scalar function of $H(F)$ and $S(F)$. The user then specifies a threshold value. All forks of the dendrogram whose figure of merit exceeds the given threshold are removed. The resulting connected components form the desired clusters. Milligan and Cooper investigated 30 different such figures of merit in [8]. The more recent account by Gordon in [6], reviews the five best performing procedures in [8] and concludes that it is always advisable to compare the result of several rules.

This last advise is typical of multi-objective optimization problems where trade-off

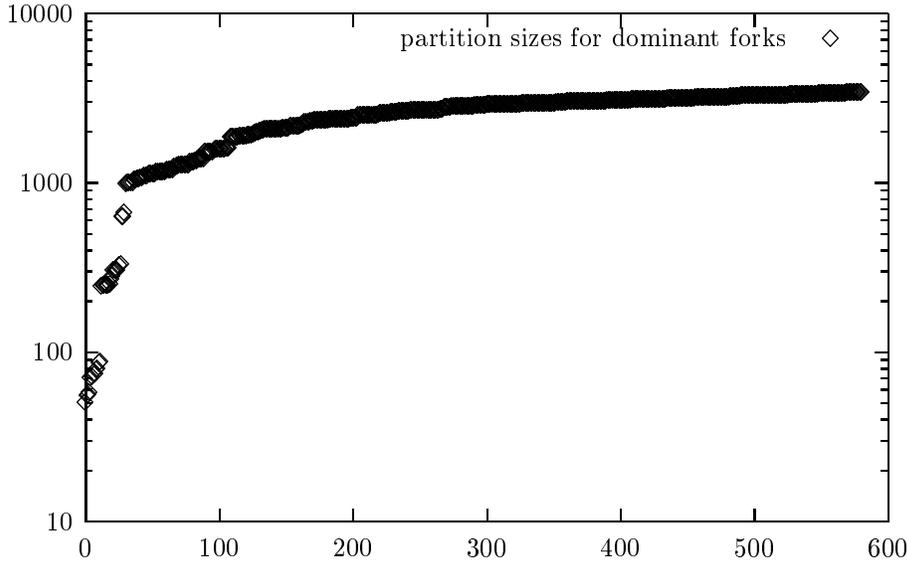


FIG. 3.1. *Number of clusters associated with each dominant fork in a dendrogram with 3390 forks.*

between conflicting goals must be resolved by human intervention. Thus, a number of output clusters can be found from the dendrogram by inspecting the set of dominant forks and the corresponding partitions. One may for instance choose the smallest partition corresponding to a dominant fork. A hierarchical clustering of the columns of our performance data resulted in a dendrogram with 3390 forks. A number that is too large for many public domain dendrogram viewers. Figure 3.1 shows the partition sizes for every one of the 581 dominant forks in the dendrogram. Based on the above decision rule one would produce 50 clusters. The next smaller dominant partitions have sizes 55 through 57, 70 and 71.

One drawback of hierarchical clustering algorithms is that dendrograms with over a few thousand leaves are hard to visualize. Much work has been done in the recent years on the subject of hierarchical data visualization, see for example [9] and [7]. One way around this limitation is to add depth to the dendrogram based on Pareto dominance. Then one can concentrate on those internal portions of the dendrogram that are nearer (in depth) to dominant forks. Several Pareto-based ranking have been used in the context of multi-objective optimization. Goldberg produced the first such ranking system in the context of genetic algorithms [5]. In essence, the ranking of an object is its distance from the set of Pareto dominant objects. In the present context all dominant forks have rank 0. The forks of rank $n + 1$ are those which would be dominant if all forks of rank at most n were disregarded. Goldberg's ranking can be implemented with $O(M^2N)$ comparisons for M objects according to N objectives using lists of dominated objects. Further details about ranking (or non-dominated sort) algorithms can be found for instance in the book by K. Deb [4]. Fonseca and Fleming [2] proposed a more expedient ranking based on the number of forks (in our context) that dominate a given fork. The rank of a fork is simply the number of forks that dominate it. The latter algorithm is an initial step of Goldberg's ranking. It does the same number of comparisons but requires no list handling.

Figure 3.2 shows a histogram of the Fonseca-Fleming ranks of each of the 3390

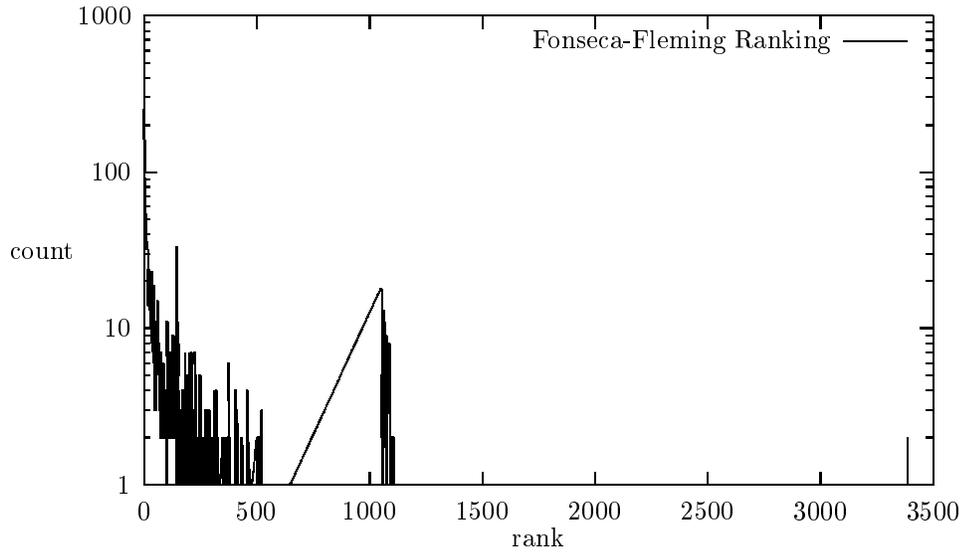


FIG. 3.2. *Goldberg ranking highlights some loose clusters.*

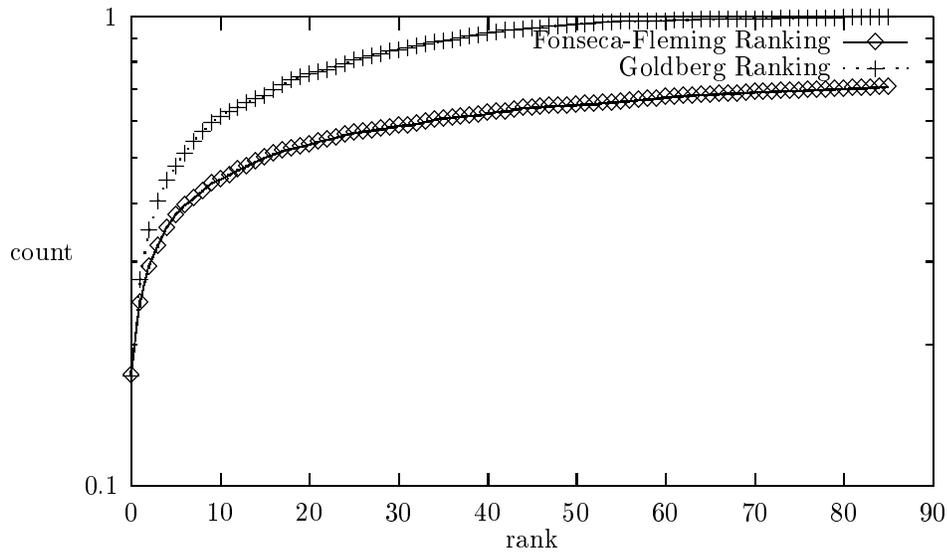


FIG. 3.3. *Layers from Goldberg's ranking are more evenly spread.*

forks in the dendogram. Although most of the ranks are within the range 0 to 500, there are clear spikes near 1000 and near 3500. Such spikes may yield information about the data but visualizing 3500 levels of depth to a dendogram would be hard. Goldberg's ranking ranges from 0 to 83 and shows no spikes. This is illustrated in Figure 3.3 that shows for each ranking, between 0 and 85, the proportion of forks with that rank.

4. Conclusions. It has been shown how concepts from multi-objective optimization can be applied to the analysis of a performance data set. The data consists of hardware event counts for standard timing subroutines of the public domain LAPACK library. The approach presented here has been shown to be effective in determining a number of output clusters, in adding depth to the dendogram (thus resulting in a 80 the number of nodes that need to be displayed at first) and in identifying meaningful subgroups. This approach can be applied to data sets from other sources and most importantly to incomplete data.

Acknowledgments. This manuscript has been authored by Battelle Memorial Institute, Pacific Northwest Division, under contracts No. DE-AC06-76RLO 1830 and 1831 with the US Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or to allow others to do so, for United States Government purposes.

REFERENCES

- [1] J. Bilmes, K. Asanonic, C.-W. Chin, and J. Demmel. Optimizing matrix multiply using PHIPAC: A portable, high-performance, ansi c coding methodology. In *1997 International Conference on Supercomputing*, Vienna, Austria, 1997.
- [2] F. C.M. and F. P.J. Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization. In S. Forrest, editor, *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–23, San Mateo, CA, 1993.
- [3] D. Culler, J. Singh, and A. Gupta. *Parallel Computer Architecture A Hardware/Software Approach*. Morgan Kaufmann, 1997.
- [4] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Systems and Optimization. John Wesley & Sons, Chichester, UK, 2001.
- [5] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.
- [6] A. Gordon. *Classification*. Number 82 in Monographs on Statistics and Applied Probability. Chapman & Hall, Boca raton, FL, 1999.
- [7] H. Kumar, C. Plaisant, and B. Shneiderman. Browsing hierarchical data with multilevel queries and pruning. Technical Report CAR-TR-772, University of Maryland, 1995.
- [8] G. Milligan and M. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–79, 1985.
- [9] S. Nestorov, J. Ullman, J. Wiener, and S. Chawathe. Representative objects: consise representation od semi-structured, hierarchical data. In *Proceedings of the International Conference on Data Engineering ICDE*, pages 79–90, 1997.
- [10] W. Whaley, A. Petitet, and J. Dongarra. Automated empirical optimization of software and the ATLAS Project. Technical Report UT-CS-00-449, University of Tennessee, Knoxville, 2000.