A Metric Space for Productivity Measurement in Software Development

Robert W. Numrich
Minnesota Supercomputing Institute
University of Minnesota
Minneapolis, MN

Lorin Hochstein
Department of Computer Science
University of Maryland
College Park, MD

Victor R. Basili
Department of Computer Science
University of Maryland
College Park, MD

ABSTRACT

We define a metric space to measure the contributions of individual programmers to a software development project. It allows us to measure the distance between the contributions of two different programmers as well as the absolute contribution of each individual programmer. Our metric is based on an action function that provides a picture of how one programmer's approach differs from another at each instance of time during the project. We apply our metric to data we collected from students taking a course in parallel programming. We display the pictures for two students who showed approximately equal contributions but who followed very different paths through the course.

1. INTRODUCTION

We define a metric space that measures the contributions of individual programmers to a software development project. This space satisfies all the mathematical requirements for a metric space and allows us to measure the distance between programmers and the absolute size of each programmer's contribution. We assign a power function, the rate of work production, for each activity performed by a programmer. The integral of the power function over time yields the work done by each programmer, and the integral of the work function yields a quantity called action in the physical sciences. After scaling and shifting each programmer's action function to the same time interval, we define a metric space with a distance function equal to the integral of the absolute difference between two action functions.

We apply our metric to data we collected observing students taking a course in parallel programming. By monitoring them during the course, we determined a set of activities performed at different times by each student and assigned a value to each activity in terms of how well it advanced the student toward a solution of the problem. In the following sections we describe how we used that data to calculate a numerical value to each student's contribution over the lifetime of the course and to calculate the difference between each pair of students.

SE-HPCS'05, May 15, 2005, St. Louis, Missouri, USA.

2. DATA COLLECTION

We collected data from students as they worked on a programming assignment for a graduate-level course on Grid Computing at the University of Maryland [2]. The assignment was to implement Conway's Game of Life [4] to run in parallel on a Beowulf Linux cluster [1]. The students used the MPI library [3] to implement the parallel program.

We collected data by instrumenting the compiler. Each time a student compiled a program, we asked two questions. First, how long have you been working before the compilation? A blank response indicated that they had been working continuously since the previous compilation. Second, what kind of work were you doing? The student selected the kind of work from a list of seven activities, which are listed in the first column of Table 1.

Table 1: Activities and Power Ratings

Activity	Power Rating
Tuning	0.9
Parallelizing	0.7
Functionality	0.6
Learning	0.5
Compile-Time Error	0.2
Run-Time Error	0.2
Other	0.1

The instrumented compiler recorded the responses along with a time stamp indicating when the compilation occurred. From this data, we computed a set of time intervals for each student along with the activity associated with that interval.

A fundamental problem in trying to define a productivity metric in software development is the definition of work. Each kind of activity in each time interval corresponds to some work that advances the student toward the solution of a problem. Some activities advance the student more quickly than others, that is, they produce work at a higher. It is sufficient, for our purposes, to know the rate at which work accumulates, the power rating, without defining the actual unit of work itself. One unit of work can be converted to any other unit of work through a suitable conversion factor. The work associated with each activity can be converted into whatever unit of work we want without changing our results.

The important quantity for our analysis is the unit of power, ρ, which we define as the maximum rate at which any programmer can perform work to finish the project. In each interval of time, each programmer, denoted by superscript i, performs some activity, denoted by subscript j, at some fraction of peak power,

$$\rho_j^i = \alpha_j^i \rho \ . \tag{1}$$

The dimensionless parameters $0 \leq \alpha_j^i \leq 1$ characterize the behavior of each programmer. Table 1, in its second column, shows the power ratings, α_j^i, assigned to each activity. We have given all programmers the same power rating for the same activity although we could, with more information, assign different ratings to each one.

These power ratings are the input parameters to our model. Their values are purely subjective at this point, and we claim no profound meaning to them. They are dimensionless quantities that represent the fraction of peak power for each activity.

3. THE SET OF PROGRAMMERS

Consider a specific software development project with some (finite) set of programmers,

$$\mathcal{P} = \{P^1, P^2, \dots\} \ , \tag{2}$$

assigned to the project. Let $t \geq 0$ represent time measured from the beginning of the project at time $t = 0$. Let T^i be the time spent on the project by programmer P^i and let

$$\mathcal{T}^i = [0, T^i] \tag{3}$$

be the corresponding time interval.

Programmers spend their time doing different things at different times during the project. To reflect this changing activity, we divide each time interval $\mathcal{T}^i$ into subintervals,

$$\mathcal{T}_j^i = [t_{j-1}^i, t_j^i] \ , \qquad j = 1, n^i \ . \tag{4}$$

Each programmer starts at

$$t_0^i = 0 \ , \tag{5}$$

and finishes at

$$t_{n^i}^i = T^i \ . \tag{6}$$

The number of intervals n^i is different for each programmer, and the total time spent T^i is different for each programmer. The width of each time interval is

$$\sigma_j^i = t_j^i - t_{j-1}^i \ , \tag{7}$$

and the activity performed in each interval is different for each programmer as represented by the constants α_j^i from equation (1) and Table 1.

4. WORK AND ACTION

In each time interval $\mathcal{T}_j^i$, programmer P^i is involved in some activity that contributes some work, $W_j^i(t)$, toward finishing the project. Some activities advance the project more than others. For each activity, the power function of equation (1) is the derivative of the work function,

$$\rho_j^i(t) = \frac{dW_j^i}{dt} \ , \tag{8}$$

the rate of work production for programmer P^i in time interval $\mathcal{T}_j^i$.

For simplicity, we assume that the power function ρ_j^i is constant in each interval so that

$$W_j^i(t) = \rho_j^i \int_{t_{j-1}^i}^{t} ds \ , \tag{9}$$

and hence work accumulates linearly in each interval,

$$W_j^i(t) = \rho_j^i (t - t_{j-1}^i) \ . \tag{10}$$

At the end of each time interval, the work accumulated over that interval is

$$W_j^i(t_j^i) = \rho_j^i \sigma_j^i \tag{11}$$

where we have used the width of the interval from equation (7).

As time increases from one interval to the next, work accumulates at different rates at different times. At time $t \in \mathcal{T}_k^i$ the total accumulated work,

$$W^i(t) = W_k^i(t) + \sum_{j=1}^{k-1} \rho_j^i \sigma_j^i \ , \tag{12}$$

is the sum of the work done during all the intervals preceding interval $\mathcal{T}_k^i$ plus the additional work done so far in interval $\mathcal{T}_k^i$.

The action generated in each time interval is the integral,

$$S_j^i(t) = 2 \int_{t_{j-1}^i}^{t} W_j^i(s) ds \ , \tag{13}$$

where we inserted the factor of two for convenience. Substituting the work function from equation (10) into the integral and evaluating the integral, we find

$$S_j^i(t) = \rho_j^i (t - t_{j-1}^i)^2 \ . \tag{14}$$

At the end of each interval, the action accumulated over that interval is

$$S_j^i(t_j^i) = \rho_j^i (\sigma_j^i)^2 \ . \tag{15}$$

The total accumulated action in interval $\mathcal{T}_k^i$ at time t is the sum,

$$S^i(t) = S_k^i(t) + \sum_{j=1}^{k-1} \rho_j^i (\sigma_j^i)^2 \ . \tag{16}$$

5. A METRIC SPACE FOR PROGRAMMERS

We make the set of programmers $\mathcal{P}$ a metric space [5] by defining a distance function based on the difference in how each programmer generates action during the project. We want this function to be a dimensionless function of a dimensionless variable such that the distance between programmers is a pure number. We also want the measure of a programmer's individual contribution to be the distance from the null programmer, a laggard that spends time on the project but produces nothing.

First we define a set of units. The unit of time, T, is the maximum time spent by any programmer in the set,

$$T = \max_i (T^i) \ . \tag{17}$$

The unit of power is ρ and the unit of action is

$$\hat{S} = \rho T^2 \ . \tag{18}$$

To put each programmer onto the same time scale [7], we define the dimensionless time variable,

$$z = 1 + (t - T^i)/T \ . \tag{19}$$

We define a dimensionless action function $s^i(z)$ in interval $\mathcal{T}_k^i$ from the sum in equation (16) evaluated at time

$$t = Tz + T^i - T \tag{20}$$

and scaled by the unit of action $\hat{S}$,

$$s^i(z) = \frac{1}{\hat{S}} \cdot \left[S_k^i(Tz + T^i - T) + \sum_{j=1}^{k-1} \rho_j^i (\sigma_j^i)^2 \right] \ . \tag{21}$$

The dimensionless time variable z spans the interval

$$1 - T^i/T \le z \le 1 \ , \tag{22}$$

and the first time interval for each programmer shifts to a new starting point,

$$z = 1 - T^i/T \ . \tag{23}$$

At this value of z, from definition (19), the time, $t = 0$, corresponds to the left end of the first interval where the action is zero. We extend the action function continuously to $z = 0$ by defining

$$s^i(z) = 0 \ , \qquad 0 \le z \le 1 - T^i/T \ . \tag{24}$$

In the variable z, every programmer ends activity at the same time,

$$z = 1 \ . \tag{25}$$

The programmer spending the longest time spans the whole interval from $z = 0$ to $z = 1$.

With these definitions, we define the distance between two programmers as the integral of the difference of the two action functions,

$$\mathrm{dist}(P^i, P^j) = \int_0^1 |s^i(z) - s^j(z)| dz \ . \tag{26}$$

The size of each programmer's contribution is the distance to the null contributor, always assumed to be in the set of programmers, such that

$$\mathrm{dist}(P^i, 0) = \int_0^1 |s^i(z)| dz \ . \tag{27}$$

6. APPLICATION TO EMPIRICAL DATA

Figure 1 shows the action functions defined by equation (21) for the set of ten programmers we considered. Each programmer is marked by a symbol at the beginning and end of the corresponding interval in the dimensionless time variable z. The size of each programmer's contribution is the area under the action function. The distance between programmers is the area under the absolute difference between action functions.

We can approximate the area under each curve by the area of the triangle determined by the end points of each curve [6]. Table 2 lists the values obtain this way in milli-action units, $\rho T^2 \times 10^{-3}$.

To interpret the information measured by our metric space, we isolate two programmers, number three and number eight in Table 2, whose contributions are approximately equal.

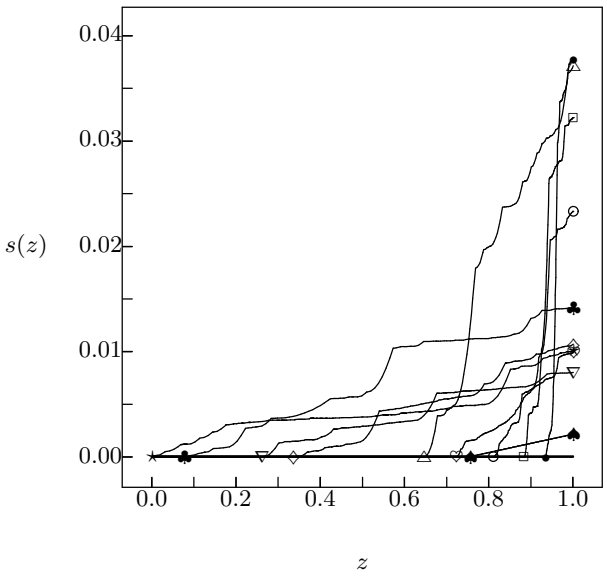

Figure 1: Action as a function of time for ten different programmers as a function of time. Each programmer is assigned a symbol that marks the beginning and ending of each curve. Time has been scaled so that the unit of time equals the longest time spent by any programmer in the set. The time for other members of the set are shifted to the right so that each programmer starts work at a different time but ends work at the same time.

Figure 2 shows the action curves for these two programmers along with two dotted triangles, determined by the end points of each curve, which we use to approximate the area under each curve. Although the area under the two curves is about the same under, indicating that the two programmers contributed about the same amount to the project, the way they contributed is quite different. One programmer took a long but steady approach while the other took a short but steep approach. The quantitative measure of the difference between the two approaches is the area between the two curves, which, from the corresponding entry in the table, equals 5.9 milli-action units.

7. SUMMARY

We have defined a metric space for a set of programmers. The distance function for that space allows us to measure the contribution of an individual programmer and to measure the difference in contributions between pairs of programmers. The metric is based on the action function for each programmer as it evolves in time. This function provides not only a measure for the total contribution, the area under the curve, but also a picture of how the approach of one programmer differs from another at each instance of time during the project. We illustrated this property by displaying the action functions for two programmers who contributed equally but followed very different paths through the project.

Table 2: Individual contributions in milli-action units, $\rho T^2 \times 10^{-3}$. The values on the diagonal are the individual contributions from equation (27). The values below the diagonal are the distances between contributions from equation (26).

1	1.2									
2	0.8	1.9								
3	5.3	4.7	6.6							
4	3.3	3.3	5.4	3.0						
5	1.4	0.9	4.4	2.8	2.2					
6	3.6	3.4	4.8	0.8	2.8	3.5				
7	1.6	1.6	5.2	1.7	1.2	2.2	1.4			
8	6.3	5.9	5.9	3.6	5.1	3.0	5.1	6.5		
9	5.2	5.0	6.2	2.1	4.4	1.6	3.7	1.6	5.0	
10	1.3	1.8	6.3	2.7	2.0	3.3	1.1	6.3	4.8	0.3
	1	2	3	4	5	6	7	8	9	10

Our model depends on several assumptions, which can be changed. We assumed that the power function is constant for each activity, independent of the time interval and independent of the programmer. This assumption results in a simple linear increase in work and a quadratic increase in action over each time interval. It is not clear that letting the power function vary over the time interval would add much to the analysis. But it might add something if we assign different constants to different programmers for the same activity. After all, some programmers are more productive than others, for example, while writing parallel code.

In the end, the distance function for our metric space takes experimental measurements of programmer activity as input and returns a dimensionless, pure number as output. By shifting and scaling the time, it accounts for the disparities in the start and stop times for different programmers. We never needed to specify the unit of work. Each activity produces work of some kind that advances the project. We subjectively judged the effectiveness of each activity by assigning a higher or a lower power rating to it. These power ratings are crucial input to the model, and the determination of what these ratings should be is the next important step we need to take to judge the utility of this model.

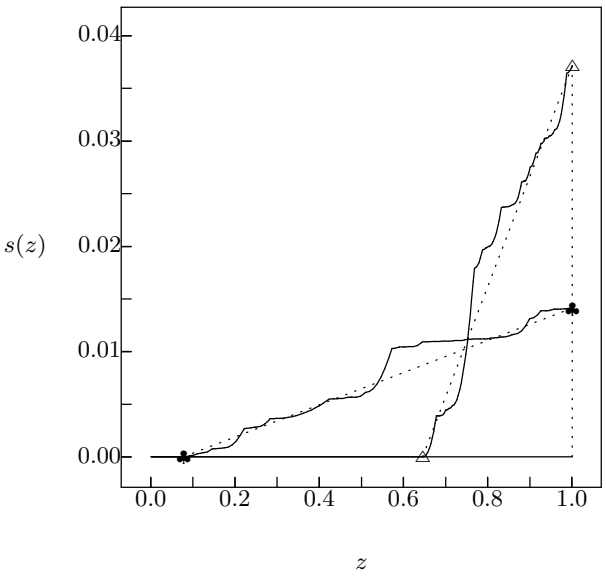

Figure 2: Action as a function of time for programmers three and eight from Table 2. The area under each curve is approximated by the triangle determined by the end points of each curve. The two programmers contributed about the same to the project, the area under the two curves is about the same, but they worked in two quite different ways to the same end.

8. ACKNOWLEDGEMENTS

This research was supported in part by Department of Energy contract DE-FG02-04ER25633 to the University of Maryland. We thank Alan Sussman for allowing us to collect data from his class on grid computing at the University of Maryland. It was also supported in part by two Department of Energy contracts to the University of Minnesota: contract DE-FC02-01ER25505, as part of the Center for Programming Models for Scalable Parallel Computing, and contract DE-FG02-04ER25629, as part of the Petascale Application Development Analysis Project, both sponsored by the Office of Science.

9. REFERENCES

[1] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer. Beowulf: A parallel workstation for scientific computation. In *Proceedings of the 1995 International Conference on Parallel Processing (ICPP)*, 1995.

[2] J. Carver, S. Asgari, V. R. Basili, L. Hochstein, J. Hollingsworth, F. Shull, and M. V. Zelkowitz. Studying code development for high performance computing: The hpcs program. In *Workshop on High Productivity Computing, Edinburgh, Scotland*, pages 32–36, May 2004.

[3] J. Dongarra, S. Otto, M. Snir, and D. Walker. A message-passing standard for MPP and workstations. *Communications of the ACM*, 39(7):84–90, 1996.

[4] M. Gardner. The fantastic combinations of John Conway's new solitaire game "Life". *Scientific American*, 223:120–123, 1970.

[5] A. N. Kolmogorov and S. V. Fomin. *Introductory Real Analysis*. Dover, revised English edition, 1970.

[6] R. W. Numrich. Performance metrics based on computational action. *International Journal of High Performance Computing Applications*, 18(4):449–458, 2004.

[7] R. W. Numrich. A dynamical approach to computer performance analysis, submitted, 2005.