Fast, but Approximate, Workflow-Runtime Estimation Using the Bell-Curve Calculus

Fast, but Approximate, Workflow-Runtime Estimation Using the Bell-Curve Calculus

Lin Yang
School of Informatics
University of Edinburgh
Email: l.yang@ed.ac.uk

Alan Bundy
School of Informatics
University of Edinburgh
Email: A.Bundy@ed.ac.uk

Conrad Hughes
School of Informatics
University of Edinburgh
Email: Conrad.Hughes@ed.ac.uk

Dave Berry
National e-Science Centre
University of Edinburgh
Email: Dave.Berry@ed.ac.uk

***Abstract*— In this paper we describe:**

- **The development of a Bell-Curve Calculus, analogous to interval arithmetic, in which normal distributions can be combined with arithmetic operations, such as addition, maximum, minimum, etc.**
- **We apply this Bell-Curve Calculus to the propagation of Quality of Service properties around e-Science workflows. In particular, we apply it to the problem of estimating the overall runtime of a workflow from estimates of the runtimes of its component services.**
- **We evaluate both the accuracy and efficiency of this Bell-Curve Calculus approach compared to alternative approaches. In particular, we show that it is much quicker than piecewise approximation approaches, but trades this off against a loss of accuracy, which nevertheless is sufficient for some kinds of application.**

I. Introduction

e-Science applications are typically represented by workflows. A workflow can be described as a graph in which the nodes stand for e-Science services and the arcs represent different ways of combining these services, labelled by data flows between them. For instance, services may be combined in the following ways.

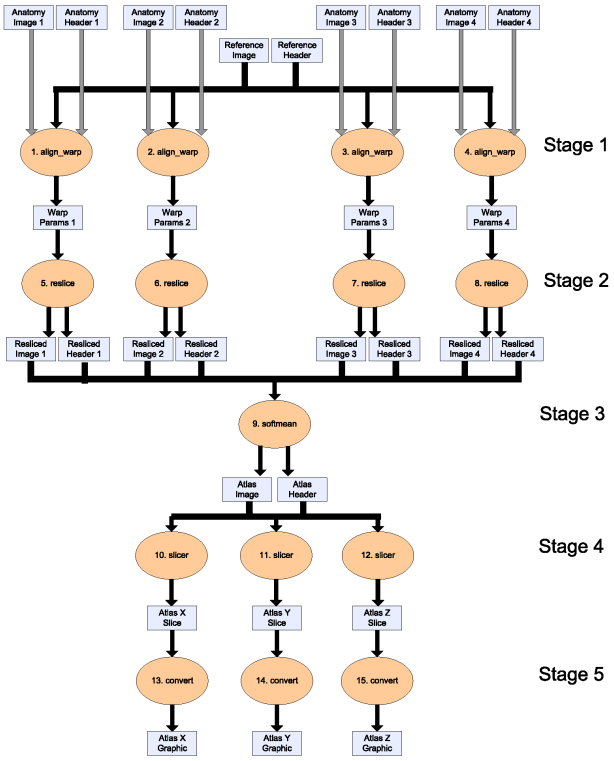
Sequential: one after another. If each node of the workflow is envisaged as applying a function to the inputs to calculate the outputs, then sequential combination provides function nesting, e.g., if x is the initial input to f and its output is input to g then $g(f(x))$ is calculated.

Parallel-All: by calling a number of services in parallel and then combining the results to provide the overall result. Parallel-All combination is typically used to calculate the arguments of a function in parallel. If $f(t_1, \ldots, t_n)$ is the function to be calculated, then each of the t_i is calculated in parallel and then the results are all input to f.

Parallel-First: by calling a number of services in parallel where the first to succeed provides the overall result. Parallel-First combination provides a form of parallel disjunction between the different services performing the same function, i.e., each of the t_i in $t_1|\ldots|t_n$ is calculated in parallel and the calculation is terminated as soon as one t_i terminates.

Disjunctional: by trying one service then another if the first fails. Disjunctional combination provides a form of sequential disjunction between two different services performing the same function, i.e., to calculate $t_1|t_2$ first t_1 is tried. If it succeeds then its result is the overall result. Otherwise, if it fails or times-out then t_2 is run.

Figure 1 gives an example.



This is an example workflow for creating population-based "brain atlases", comprised of procedures, shown as orange ovals, and data items (shown as rectangles) flowing between them. It is taken from `http://twiki.ipaw.info/bin/view/Challenge/FirstProvenanceChallenge`.

Fig. 1. An Example Workflow

In designing a workflow, it is important to estimate the quality of service it will provide. Will its results be accurate? Will it run reliably? Will it run within a reasonable time? Our

work on the Bell-Curve Calculus provides a tool to assist the workflow designer in making these estimates.

II. Estimating the Runtime of an e-Science Workflow

Consider, for instance, estimating runtime. We could associate an arithmetic function with each way of combining services, e.g., addition with sequential combination, maximum with parallel-all, minimum with parallel-first, etc. The runtime of a whole workflow can be calculated compositionally by recursion. In the base case, we use experimental evidence to estimate the typical runtimes of each individual e-Science service. In the step case, we combine the estimates of two or more sub-workflows using the arithmetic function associated with their method of combination.

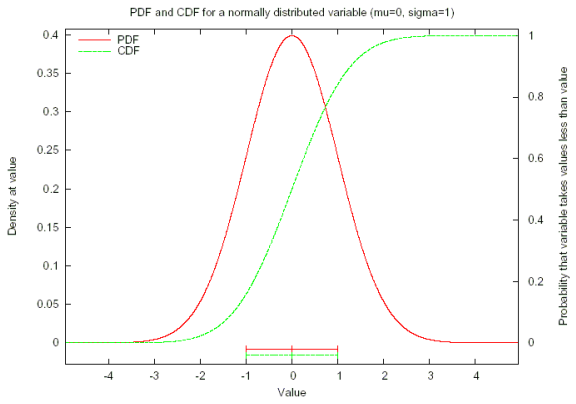
However, estimating a runtime with a single number is not good enough. Our estimates of the runtimes of e-Science services will be made by running that service several times, resulting in a range of times, even for identical or very similar conditions. *Interval Arithmetic* is a version of arithmetic over the domain of intervals of real numbers [1]. It enables us to calculate an estimated range for the whole workflow from the estimated ranges of the component services. The lower bound of this range gives the minimum estimated runtime and the upper bound gives the maximum estimate.

III. The Bell-Curve Calculus

Interval Arithmetic provides an appropriate calculus for worst-case QoS estimates, but the overall QoS range of a complex workflow is likely to be quite large and, hence, uninformative. A workflow designer is more likely to want an average-case estimate and an indication of the likelihoods of the different possible values. Not all points in a range are equally probable. If we plot the points in a range against their likelihood then we will get a curve. Often this curve will approximate a normal distribution (also called a Gaussian or a bell curve), with the points in the middle of the range being the most probable (see Figure 2). Even when the curves associated with the components of a workflow are not normal, the Central Limit Theorem[1] implies that the curve associated with the whole workflow *will be* approximately normal [2]. These observations motivated the *Bell-Curve Calculus* (BCC) — similar to Interval Arithmetic, but where the arithmetic functions are applied to bell curves, rather than intervals.

Note that some of the operations of Interval Arithmetic do not naturally return intervals. For instance, taking the union of two intervals will only output an interval if the input intervals intersect. In such cases the output sets are normalised to intervals, e.g., by forming an interval whose bounds are the minimum and maximum elements of the set. Similarly, in the BCC, it is often necessary to normalise the output curve to be a bell curve. The question then arises as to whether these normalisation steps cause unacceptable distortions of the QoS estimates. This is an empirical question that we investigated in our project and which is discussed in §VII.

[1]If the sum of the variables has a finite variance, then it will be approximately normally distributed.



The red curve is a bell-curve in which $\mu = 0$ and $\sigma = 1$. The x-axis shows the possible output values of a normally distributed, independent variable: workflow runtimes in our case. The y-axis shows the probability of each possible value of variable x. The red curve is also called the probability density function *(pdf) of x. The green curve is the cdf (cumulative density function) curve, integrated from the pdf. It gives the probability that the output will be at most x.*

In our experimental work we always normalised one of the input curves to $\mu = 0$ and $\sigma = 1$. This simplified the analysis, but is unrealistic for workflow runtimes, which will always be positive, whereas our bell curves have non-zero probability (positive y values) for negative runtime (x values). In practice, this is not a problem, since our results can be readily translated to account for unnormalised input curves. Any parts of input or output bell curves that occur to the left of the y-axis can be regarded as estimate error.

Fig. 2. A bell curve and its cdf

A bell curve can be defined by the following equation:

$$BC(\mu, \sigma) = \lambda x.\ \frac{e^{-(x-\mu)^2/2\sigma^2}}{\sigma\sqrt{2\pi}}$$

Note that each curve is defined by two parameters: the mean μ and the standard deviation σ.

The restriction of the BCC to the domain of bell curves means that it is potentially very efficient: given the parameters of two input bell curves, we need only calculate the parameters of the output bell curve. Compare this, for instance, to the piecewise approximation method of combining two curves to produce a third, described in §V. If high resolution is required, this can be a computationally expensive operation. Comparing the efficiency of these two methods of combining curves was another topic for empirical investigation in our project. This topic is also addressed in §VII.

IV. Research Questions Addressed

During our project, we worked closely with the Dependable Service-Centric Computing (DSCC) project[2]. One of the deliverables of DSCC is the *Agrajag program*, which is a tool for calculating piecewise approximations of combinations of arbitrary curves for use in QoS estimates [3]. Piecewise approximation is obtained by approximating the input curves using the union of a large number (n each, say) of uniform distributions, performing the relevant arithmetic operation pairwise on these to produce $O(n^2)$ such distributions and merging these back together in least-error pairs until we have about n of them again. The pairwise operations on the sub-distributions are usually implemented using interval arithmetic so that an operation on one pair will usually produce one result distribution. The aggregation of a large number of these pieces is what gives us an accurate result.

Agrajag and BCC provide an accuracy/complexity tradeoff: Agrajag provides higher accuracy, but BCC is more efficient. Agrajag also provides a "gold standard" against which we can measure the accuracy and efficiency of the BCC. So our research resolved around the following questions.

1) For each of the ways of combining workflows, define a function that, given BCC estimates of the runtime of the components, outputs a BCC estimate of the runtime of the combination.
2) Measure the *accuracy* of the BCC estimates compared to the piecewise approximations.
3) Measure the *efficiency* of the BCC estimates compared to the piecewise approximations.

V. Methodology

We decided to adopt an empirical methodology. It is not clear that there will always be neat, closed-form, bell-curve functions for each of the workflow combination methods, especially given the inherently approximate nature of the exercise. There is one exception to this: linear combination of bell curves, and therefore simple pairwise addition, is known to be exact and has a simple definition, which we give below..

So problem 1 can be defined as follows. Let $BC(\mu_1, \sigma_1)$ and $BC(\mu_2, \sigma_2)$ be two input bell-curves and F_c be the binary arithmetic function associated with combination method c, where c varies over $\{S, PA, PF, D\}$, standing for sequential, parallel-all, parallel-first and disjunctional combination, respectively. We call $F_c(BC(\mu_1, \sigma_1), BC(\mu_2, \sigma_2))$ the *perfect curve*. Note that, in general, the perfect curve will not be a bell-curve. Agrajag produces a piecewise approximation to the perfect curve.

For the BCC, we require to define two 4-ary functions, M_c and Σ_c, that will provide the parameters to a bell-curve that will approximate F_c, i.e.,

$$BC(M_c(\mu_1, \sigma_1, \mu_2, \sigma_2), \Sigma_c(\mu_1, \sigma_1, \mu_2, \sigma_2)) \approx F_c(BC(\mu_1, \sigma_1), BC(\mu_2, \sigma_2))$$

[2]http://digs.sourceforge.net/

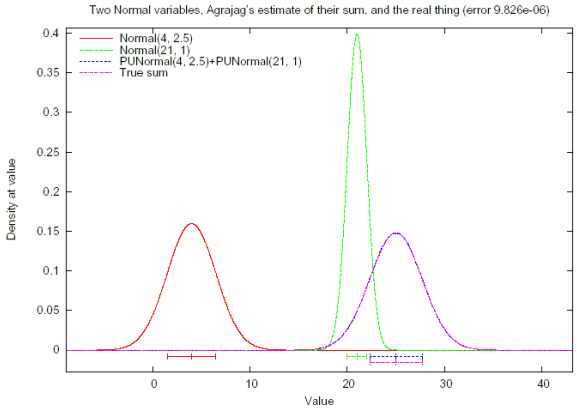
We will call the LHS of this equation the BCC *estimate*. For instance, when $c = S$ then F_c is +,

$$M_S(\mu_1, \sigma_1, \mu_2, \sigma_2) = \mu_1 + \mu_2$$
$$\Sigma_S(\mu_1, \sigma_1, \mu_2, \sigma_2) = \sqrt{\sigma_1^2 + \sigma_2^2}$$

and $\approx$ can be replaced by $=$, i.e.,

$$BC(\mu_1 + \mu_2, \sqrt{\sigma_1^2 + \sigma_2^2}) = BC(\mu_1, \sigma_1) + BC(\mu_2, \sigma_2)$$

Figure 3 illustrates this.



The red and green bell-curves are the two input curves to be summed. The mauve curve is the bell-curve estimate of this sum and the blue curve is Agrajag's piecewise approximation. The blue and mauve curves are superimposed and impossible to separate. Any small difference is due to rounding approximations in Agrajag. This is the ideal situation for BCC*, but we cannot expect such perfect answers for other bell-curve functions.*

Fig. 3. The sum of two bell curves

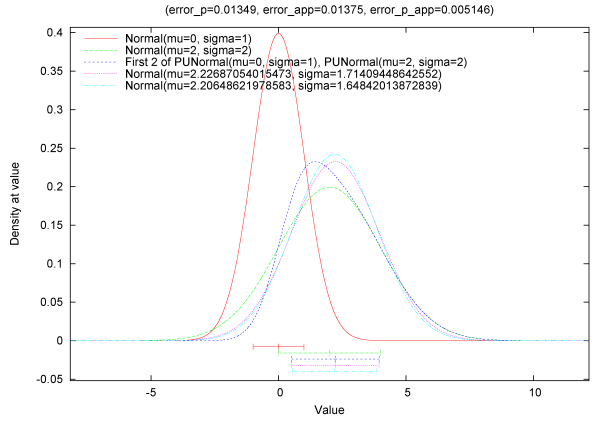
However, we also need to find M_c and Σ_c for the other three methods of workflow combination, i.e., when F_c is max, min and a function we call $cond$, that corresponds to disjunctional combination. There is no neat, exact solution for these combination methods. Our methodology for these methods was as follows.

1) Use Agrajag to make piecewise approximations for a range of different values. To simplify these calculations, without loss of generality, we set $\mu_1 = 0$ and $\sigma_1 = 1$ and vary only μ_2 and σ_2.
2) Use Agrajag to calculate the best bell-curve approximation of each value of F_c. Call this $BC(\mu_p, \sigma_p)$. We call this curve the *best bell curve*, since it is the best that can be expected of a BCC that *requires* output curves to be bell curves. Agrajag calculates μ_p and σ_p as the mean and standard deviation of the piecewise estimate.
3) Eyeball how μ_p and σ_p vary with μ_2 and σ_2. Guess some first approximations of M_c and Σ_c, and pick the ones that generated the smallest error, calculated as the

least mean square difference between the CDFs of the BCC estimate and the best bell curve.

4) Plot values of μ_2 against the difference between M_c and μ_p, and similarly with σ_2.
5) Use curve-fitting techniques to generate correction factors to the initial values of M_c and Σ_c.
6) Repeat from step 4 until the difference between the BCC estimate and the best bell curve is acceptably small.

This methodology generates rather messy but acceptable approximations to the best bell curves. Figure 4 illustrates this in the case of maximum.



The colour coding of the curves is as follows: red and green for the inputs, aqua for the BCC *estimate, mauve for the best bell curve and blue for the piecewise estimate. Note that the blue curve is not a bell curve, so some error is unavoidable. The best that the* BCC *estimate can aspire to is the best bell curve, to which it is quite close. Note that we have depicted the worst case. The error only becomes significant when the input curves overlap substantially, as in this case. Otherwise, the larger of the two input curves is almost exactly the same as the output.*

Fig. 4. The maximum of two bell curves

VI. The BCC Formulae

The formulae developed in the BCC are given below. Recall that the mean and standard deviation of the BCC estimate are given by $M_c(\mu_1, \sigma_1, \mu_2, \sigma_2)$ and $\Sigma_c(\mu_1, \sigma_1, \mu_2, \sigma_2)$, respectively, where μ_i and σ_i are the mean and standard deviation, respectively, of the two input bell curves, for $i = 1, 2$. c denotes the kind of workflow combination and takes values in the set $\{S, PA, PF, D\}$.

We have already given the formulae for M_S and Σ_S in §V, but we repeat them here for the sake of completeness. The other three formulae are quite complex. So, in the interests of brevity, we give them only for the special case when $\mu_1 = 0$, $\sigma_1 = 1$ and $\mu_2 > 0$. At the end we provide the transformations required to generate the general cases from these special cases. Also, we abbreviate the formulae using some parameters whose values are given in the Appendix. Although these parameters are functions with arguments, we omit the arguments for brevity. We also recycle the same parameter names for each of the six formulae, although each case is different. Note that the parameters for M_D and Σ_D also contain ρ, which is needed for the generalisation process given below.

Sum: Used for sequential combination of runtimes.

$$\begin{aligned} M_S(\mu_1, \sigma_1, \mu_2, \sigma_2) &= \mu_1 + \mu_2 \\ \Sigma_S(\mu_1, \sigma_1, \mu_2, \sigma_2) &= \sqrt{{\sigma_1}^2 + {\sigma_2}^2} \end{aligned}$$

Max: Used for parallel-all combination of runtimes.

$$\begin{aligned} M_{PA}(0, 1, \mu_2, \sigma_2) &= A.\mu_2 + B + C.e^{-\frac{\mu_2}{D}} \\ \Sigma_{PA}(0, 1, \mu_2, \sigma_2) &= A.\sigma_2 + B - C.e^{-\frac{\sigma_2}{D}} \end{aligned}$$

Min: Used for parallel-first combination of runtimes.

$$\begin{aligned} M_{PF}(0, 1, \mu_2, \sigma_2) &= A.\mu_2 + B - C.e^{-\frac{\mu_2}{D}} \\ \Sigma_{PF}(0, 1, \mu_2, \sigma_2) &= A.\sigma_2 + B + C.e^{-\frac{\sigma_2}{D}} \end{aligned}$$

Cond: Used for disjunctional combination of runtimes.

$$\begin{aligned} M_D(0, 1, \mu_2, \sigma_2) &= A.\mu_2 + B + C.e^{-\frac{\mu_2}{D}} \\ \Sigma_D(0, 1, \mu_2, \sigma_2) &= A.\sigma_2 + B + C.e^{-\frac{\sigma_2}{D}} \end{aligned}$$

To generate the general formulae from the above special cases, the following transformations must be applied.

If $\mu_2 \geq \mu_1$ then

$$\begin{aligned} M_c(\mu_1, \sigma_1, \mu_2, \sigma_2) &= \mu_1 + M_c(0, 1, \frac{\mu_2 - \mu_1}{\sigma_1}, \frac{\sigma_2}{\sigma_1}).\sigma_1 \\ \Sigma_c(\mu_1, \sigma_1, \mu_2, \sigma_2) &= \Sigma_c(0, 1, \frac{\mu_2 - \mu_1}{\sigma_1}, \frac{\sigma_2}{\sigma_1}).\sigma_1 \end{aligned}$$

Note that in the case of Max and Min we can assume $\mu_2 \geq \mu_1$ without loss of generality, since M_c and Σ_c are commutative with respect to these two arguments, but in the case of Cond we also need to consider the case $\mu_1 > \mu_2$.

If $\mu_1 > \mu_2$, substitute $(1 - \rho)$ for ρ then

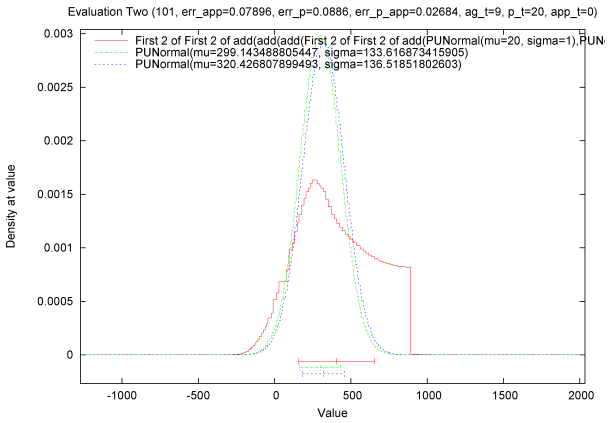
$$\begin{aligned} M_c(\mu_1, \sigma_1, \mu_2, \sigma_2) &= \mu_2 + M_c(0, 1, \frac{\mu_1 - \mu_2}{\sigma_2}, \frac{\sigma_1}{\sigma_2}).\sigma_2 \\ \Sigma_c(\mu_1, \sigma_1, \mu_2, \sigma_2) &= \Sigma_c(0, 1, \frac{\mu_1 - \mu_2}{\sigma_2}, \frac{\sigma_1}{\sigma_2}).\sigma_2 \end{aligned}$$

VII. Evaluation

Research question 2 from §IV now breaks into two sub-questions.

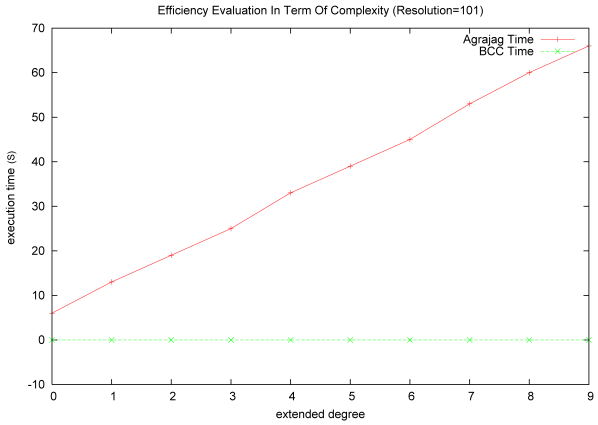
2.1 What is the error, measured as the root mean square difference, between the piecewise approximation and the best bell curve?
2.2 What is the error between the best bell curve and the BCC estimate?

We can ask these sub-questions both of individual workflow combinations and of compound workflows. The result for a complete workflow is given in Figure 5.



The accuracy of BCC *and Agrajag are compared on a typical, complete workflow. The red curve shows Agrajag's piecewise approximation and the green curve is the* BCC *estimate. The blue curve is the best bell curve, showing how closely the* BCC *estimate approximates this. For many purposes this degree of accuracy would be acceptable, e.g. estimating the cost of an expensive calculation, assessing whether the calculation could be run in a reasonable time, etc.*

Fig. 5. Comparison of accuracy of Agrajag and BCC



The efficiency of Agrajag and BCC *are compared by plotting their runtimes for an increasingly large series of workflows. The red curve shows the Agrajag runtimes and the green curve shows those for* BCC*. The workflow series was generated by systematically replacing one node after another with a complete copy of the original workflow. The x-axis enumerates the number of replacements and the y-axis gives the resulting runtimes in seconds. Note that the $y-axis$ is plotted from -10, so that the green curve can be distinguished from the x-axis. These results are robust with respect to the choice of nodes to be replaced. Both curves are linear, although the slope of the green* BCC *curve cannot be seen at this resolution and needs to be plotted with the y-axis showing milliseconds.*

Fig. 6. Comparison of efficiency of Agrajag and BCC

Re question 3, we ran both Agrajag and BCC on an example series of workflows. Some results are summarised in Figure 6. Both Agrajag's and BCC's runtime increased linearly with increasing workflow complexity. However, whereas the runtime of Agrajag was of the order of seconds, even when both the workflow and the resolution were relatively small, the corresponding BCC runtime was of the order of milliseconds.

VIII. Conclusion

We have presented the Bell-Curve Calculus, a version of arithmetic in which the domain is bell-curves rather than numbers. This calculus is inspired by Interval Arithmetic. But whereas intervals can be used for worst-case analyses, bell-curves can be used for average-case analyses. The BCC shows both the range and the likelihood of each value in the range. We have applied the BCC to estimating quality of service properties in e-Science workflows, in particular, we have focused on their runtimes. The BCC estimate is not as accurate as the piecewise approximation, but it is very efficient to calculate. This could make it the calculus of choice for making rough estimates of QoS properties of very large workflows. Further details about the BCC can be found in [4], [5].

Bell curves were chosen due to their ubiquity in experimental results. This includes e-Science workflow runtimes, but also other QoS properties, such as accuracy and reliability. So, the BCC has much wider potential application. To realise this potential, many additional bell-curve functions would have to be defined, such as multiplication, which is needed for sequential combination of QoS estimates of accuracy and reliability, for instance.

One could also envisage similar calculi for other common families of curves, e.g., log normal curves and exponential decay curves. The same methodology could be employed. However, our dream is to identify a larger family of curves that encompass, with reasonable accuracy, all these sub-families as special cases. Then the calculus could mix curves of different types while staying within the same family. To retain the benefits of the BCC each member of the family would need to be definable with a few parameters. Using a larger such curve family would both extend the range of application but also, potentially, the accuracy, since there would be a wider family of curves from which to select the best approximation.

Appendix

Parameters for M_{PA}

- $A = 0 \cdot 0058.\sigma_2 + 1 \cdot 1 - 0 \cdot 13.0 \cdot 83^{\sigma_2} - 0 \cdot 017.\sigma_2.(1 \cdot 0 - \frac{\sigma_2}{36})^{4 \cdot 0} + \frac{\sigma_2{}^{15}}{1 \cdot 5e+27}$
- $B = 0 \cdot 00020.\sigma_2{}^{2 \cdot 8} - 0 \cdot 57.e^{-\frac{(\sigma_2 - 20 \cdot)^{2 \cdot 0}}{68}}$

$+ 0 \cdot 85.e^{-\frac{(\sigma_2-41)^{2\cdot 0}}{1\cdot 0e+2}} - 0 \cdot 050.e^{-\frac{(\sigma_2-26)^{2\cdot 0}}{24}}$
$- \frac{{\sigma_2}^{15}}{8\cdot 0e+25} + 0 \cdot 052.e^{-\frac{(\sigma_2-46)^{2\cdot 0}}{20\cdot}}$

- $C = \sigma_2.e^{-\frac{{\sigma_2}^{0\cdot 90}}{20\cdot}} - 1 \cdot 0 - 1 \cdot 6.e^{-\frac{(\sigma_2-11)^{2\cdot 0}}{95}}$
$+ 1 \cdot 9.1 \cdot 3^{-1\cdot 9.\sigma_2} + 0 \cdot 071.e^{-\frac{(\sigma_2-32)^{2\cdot 0}}{50\cdot}}$
$- 0 \cdot 055.e^{-\frac{(\sigma_2-7\cdot 1)^{2\cdot 0}}{17}} + 0 \cdot 032.e^{-\frac{(\sigma_2-2\cdot 8)^{2\cdot 0}}{3\cdot 0}}$
- $D = 10 \cdot .{\sigma_2}^{0\cdot 11} + 0 \cdot 66 - (\sigma_2 + 3 \cdot 7).1 \cdot 1^{-\frac{(\sigma_2+3\cdot 7)^{1\cdot 1}}{0\cdot 39}}.3 \cdot 8$
$+ 2 \cdot 4.50^{\cdot -2\cdot 4.\sigma_2} - 0 \cdot 41.e^{-\frac{(\sigma_2-14)^{2\cdot 0}}{50\cdot}} + 0 \cdot 27.e^{-\frac{(\sigma_2-5\cdot 8)^{2\cdot 0}}{13}}$
$- 0 \cdot 39.e^{-\frac{(\sigma_2-1\cdot 7)^{2\cdot 0}}{1\cdot 7}} + 0 \cdot 075.e^{-\frac{(\sigma_2-43)^{2\cdot 0}}{1\cdot 6e+2}}$

Parameters for Σ_{PA}

- $A = (1 \cdot 3e - 05).{\mu_2}^{2\cdot 3} + 0 \cdot 58$
- $B = 0 \cdot 30.\mu_2 - (3 \cdot 3e - 05).{\mu_2}^{3\cdot 1} + 0 \cdot 35.e^{-\frac{(\mu_2-18)^{2\cdot 0}}{2\cdot 0e+2}}$
$- 0 \cdot 081.e^{-\frac{(\mu_2-44)^{2\cdot 0}}{40\cdot}} - 0 \cdot 13.e^{-\frac{(\mu_2-1\cdot 5)^{2\cdot 0}}{0\cdot 30}}$
- $C = 0 \cdot 32.\mu_2 + 0 \cdot 19 - (3 \cdot 0e - 05).{\mu_2}^{3\cdot 1}$
$- 0 \cdot 77.e^{-\frac{\mu_2}{1\cdot 3}} + 0 \cdot 14.e^{-\frac{(\mu_2-22)^{2\cdot 0}}{1\cdot 4e+2}} - 0 \cdot 055.e^{-\frac{(\mu_2-43)^{2\cdot 0}}{40\cdot}}$
- $D = 0 \cdot 44.\mu_2 + 0 \cdot 99 - (12e - 05).{\mu_2}^{2\cdot 8})$
$-0 \cdot 52.e^{-\frac{(\mu_2-39)^{2\cdot 0}}{1\cdot 5e+2}} - 0 \cdot 78.e^{-\frac{(\mu_2-1\cdot 5)^{2\cdot 0}}{0\cdot 30}} - 0 \cdot 41.e^{-\frac{(\mu_2-3\cdot 8)^{2\cdot 0}}{1\cdot 8}}$

Parameters for M_{PF}

- $A = 0 \cdot 000075.{\sigma_2}^{2\cdot 0} - 0 \cdot 021.e^{-\frac{(\sigma_2-22)^2}{1\cdot 3e+2}} - \frac{1\cdot 0}{1\cdot 5^{-\frac{\sigma_2}{1\cdot 1e+11}}}$
- $B = -0 \cdot 47.\sigma_2 + 11 - 12.0 \cdot 88^{\sigma_2}$
$- 0 \cdot 64.\sigma_2.e^{-\frac{|\sigma_2-1\cdot 5|^{1\cdot 6}}{50\cdot}} + \frac{5\cdot 5^{-\sigma_2}}{1\cdot 8}$
$- 0 \cdot 036.\sigma_2.e^{-\frac{(\sigma_2-2\cdot 9)^{2\cdot 0}}{5\cdot 5}} - 0 \cdot 047.e^{-\frac{(\sigma_2-15)^{2\cdot 0}}{10\cdot}}$
$+ 0 \cdot 17.e^{-\frac{(\sigma_2-32)^{2\cdot 0}}{1\cdot 2e+2}} - \frac{1\cdot 0}{1\cdot 3^{-\frac{\sigma_2}{6\cdot 3e+6}}}$
- $C = \sigma_2.e^{-\frac{{\sigma_2}^{0\cdot 90}}{20\cdot}} - 0 \cdot 58 - 0 \cdot 27.\sigma_2.e^{-\frac{(\sigma_2-2\cdot 9)^{2\cdot 0}}{1\cdot 2e+2}}$
$+ 2 \cdot 1^{-\sigma_2}.1 \cdot 1 + 0 \cdot 086.e^{-\frac{(\sigma_2-26)^{2\cdot 0}}{70\cdot}} - 0 \cdot 083.e^{-\frac{(\sigma_2-47)^{2\cdot 0}}{70\cdot}}$
- $D = 10 \cdot .{\sigma_2}^{0\cdot 11} - (\sigma_2 + 4 \cdot 5).1 \cdot 1^{-\frac{(\sigma_2+4\cdot 5)^{1\cdot 1}}{0\cdot 39}}.3 \cdot 5$
$+ 1 \cdot 0e + 2^{-\sigma_2}.2 \cdot 4 - 0 \cdot 14.\sigma_2.e^{-\frac{(\sigma_2-1\cdot 1)^{2\cdot 0}}{2\cdot 6}} - 0 \cdot 51.e^{-\frac{(\sigma_2-12)^{2\cdot 0}}{35}}$
$+ 0 \cdot 0039.\sigma_2.e^{-\frac{(\sigma_2-31)^{2\cdot 0}}{2\cdot 5e+2}} + 5 \cdot 0^{-\frac{\sigma_2}{6\cdot 2}} - 0 \cdot 057.e^{-\frac{(\sigma_2-19)^{2\cdot 0}}{20\cdot}}$

Parameters for Σ_{PF}

- $A = -(1 \cdot 4e - 06).{\mu_2}^{2\cdot 7} + 0 \cdot 58$
- $B = -2 \cdot 8.{\mu_2}^{0\cdot 40} + e^{-\frac{\mu_2-7\cdot 0}{7\cdot 0}}.10 \cdot$
$+ e^{-\frac{\mu_2-22}{9\cdot 0}}.e^{-e^{-\frac{\mu_2-22}{9\cdot 0}}}.3 \cdot 2 + \frac{e^{-\frac{(\mu_2-2\cdot 5)^{2\cdot 0}}{2\cdot 0}}}{2^{2^{8}}}$
$- e^{-\frac{\mu_2-8\cdot 5}{2\cdot 0}}.\frac{e^{-e^{-\frac{\mu_2-8\cdot 5}{2\cdot 0}}}}{3\cdot 1} - \frac{e^{-\frac{(\mu_2-46)^{2\cdot 0}}{1\cdot 1e+2}}}{5\cdot 1}$
- $C = {\mu_2}^{0\cdot 71} + e^{-\frac{37-\mu_2}{11}}.e^{-e^{-\frac{37-\mu_2}{11}}}.2 \cdot 2$
$- e^{-\frac{\mu_2-10\cdot}{6\cdot 5}}.e^{-e^{-\frac{\mu_2-10\cdot}{6\cdot 5}}} + 1 \cdot 5^{-\frac{\mu_2}{3\cdot 5}} - 2 \cdot 0^{\frac{\mu_2}{1\cdot 5e+16}}$
- $D = {\mu_2}^{0\cdot 67} - e^{-\frac{\mu_2-9\cdot 0}{6\cdot 0}}.e^{-e^{-\frac{\mu_2-9\cdot 0}{6\cdot 0}}}.1 \cdot 3 + 6 \cdot 0^{-\frac{\mu_2}{2\cdot 0}}.1 \cdot 3$
$+ \frac{e^{-(\mu_2-1\cdot 6)^{2\cdot 0}.5\cdot 0}}{2\cdot 2} + \frac{e^{-\frac{(\mu_2-36)^{2\cdot 0}}{1\cdot 6e+2}}}{5\cdot 2} - 1 \cdot 5^{\frac{\mu_2}{1\cdot 4e+10}}$

Parameters for M_D

- $A = \begin{cases} 0 \cdot 50.\rho + 0 \cdot 50, & 0 \cdot 0 < \sigma_2 < 9 \cdot 5; \\ \rho, & \sigma_2 \geq 9 \cdot 5. \end{cases}$
- $B = \begin{cases} (2 \cdot 3.\rho - 2 \cdot 3).\sigma_2 + 2 \cdot 4.\rho - 2 \cdot 4, & 0 \cdot 0 < \sigma_2 < 9 \cdot 5; \\ 0 \cdot 0, & \sigma_2 \geq 9 \cdot 5. \end{cases}$
- $C = \begin{cases} -2 \cdot 4.\rho + 2 \cdot 4.\sigma_2 - 5 \cdot 2.\rho + 5 \cdot 2 & \\ -(-2 \cdot 7.\rho + 2 \cdot 7).e^{-\frac{\sigma_2}{3\cdot 2}}, & 0 \cdot 0 < \sigma_2 < 9 \cdot 5; \\ 0 \cdot 0, & \sigma_2 \geq 9 \cdot 5. \end{cases}$
- $D = \begin{cases} 2 \cdot 3.\sigma_2 + 1 \cdot 9, & 0 \cdot 0 < \sigma_2 < 9 \cdot 5; \\ 27, & \sigma_2 \geq 9 \cdot 5. \end{cases}$

Parameters for Σ_D

- $A = -(-0 \cdot 018.(\rho - 0 \cdot 51)^{2\cdot 0} + 0 \cdot 0038).\mu_2 + 1 \cdot 2$
$- 2 \cdot 5^{-2\cdot 1.\rho} - (-0 \cdot 35.(\rho - 0 \cdot 67).|\rho - 0 \cdot 67|^{0\cdot 61}$
$+ 0 \cdot 074) * e^{-\frac{\mu_2}{8\cdot 5.\rho+11.\rho.2\cdot 3^{(-1\cdot 2.\rho)}+4\cdot 1}}$
- $B = (-0 \cdot 027.|\rho - 0 \cdot 27|^{1\cdot 4} + 0 \cdot 016).{\mu_2}^{0\cdot 30.\rho^{1\cdot 5}+1\cdot 7}$
- $C = (-51.(\rho - 1 \cdot 4).3 \cdot 3^{(\rho-1\cdot 4)} - 13)$
$.(\mu_2 + (-(\rho - 0 \cdot 70)^{2\cdot 0}.(1 \cdot 5e + 2) + 51 + e^{-|\rho-0\cdot 20|.40\cdot}.5 \cdot$
$4 - e^{-(\rho-0\cdot 48)^{2\cdot 0}.50\cdot})$
$.3 \cdot 6 + e^{-|\rho-0\cdot 30|.25} - e^{(\rho-0\cdot 78).55})$
$.(-\frac{(\rho-0\cdot 75)^{2\cdot 0}}{55} + 0 \cdot 99)^{(\mu_2-(\rho-0\cdot 70)^{2\cdot 0}.(1\cdot 5e+2)+51+e^{-(|\rho-0\cdot 20|).40\cdot}.5\cdot 4)}$
$.(-\frac{(\rho-0\cdot 75)^{2\cdot 0}}{55} + 0 \cdot 99)^{(-e^{-(\rho-0\cdot 48)^{2\cdot 0}.50\cdot}.3\cdot 6+e^{-|\rho-0\cdot 30|.25}-e^{(\rho-0\cdot 78).55})}$
$- (-(\rho - 0 \cdot 65)^{2\cdot 0}.(1 \cdot 6e + 2) + 59 + e^{-|\rho-0\cdot 20|.40\cdot}.3 \cdot 7 -$
$e^{(\rho-0\cdot 70).22}$
$.2 \cdot 0 - e^{-(\rho-0\cdot 42)^{2\cdot 0}.80\cdot}.3 \cdot 7 + \frac{e^{-|\rho-0\cdot 67|^{2\cdot 4}.(2\cdot 0e+2)}}{1\cdot 3})$
- $D = (e^{-e^{-(\rho-0\cdot 41).5\cdot 0}}.e^{-(\rho-0\cdot 41).5\cdot 0}.1 \cdot 5 + 0 \cdot 14)$
$.\mu_2 - (e^{-e^{-(\rho-0\cdot 40).6\cdot 3}}.e^{-(\rho-0\cdot 40).6\cdot 3}.43 - 4 \cdot 7)$
$^{-(\mu_2-51)^{2\cdot 0}/(e^{-e^{-(\rho-0\cdot 35).10\cdot}}.e^{-(\rho-0\cdot 35).10\cdot}}$
$.(1\cdot 6e+3)+(4\cdot 0e+2)$
$+e^{-|\rho-0\cdot 54|.15}.(5\cdot 2e+2)$
$-e^{-|\rho-0\cdot 40|.20\cdot}.11$
$-e^{-|\rho-0\cdot 30|.40\cdot}.5\cdot 0-e^{-|\rho-0\cdot 60|.40\cdot}.11$
$- e \quad +e^{(\rho-0\cdot 67).74}.1\cdot 4-e^{(\rho-1\cdot 5).(-23)}.(0\cdot 14e-13))$
$.(e^{-(\rho-0\cdot 43)^{2\cdot 0}}.22.18 - 3 \cdot 4 + e^{-|\rho-0\cdot 28|.28}.4 \cdot 7$
$- e^{-|\rho-0\cdot 40|.50\cdot}.1 \cdot 8 + e^{(\rho-0\cdot 69).89}.1 \cdot 3)$
$+ (e^{-(\rho-0\cdot 42)^{2\cdot 0}.30\cdot}.15 + e^{-|\rho-0\cdot 30|.(1\cdot 0e+2)}.2 \cdot 7 - 0 \cdot 90 -$
$e^{-|\rho-0\cdot 40|.60\cdot}$
$.1 \cdot 7 + e^{-(\rho-0\cdot 55)^{2\cdot 0}.(3\cdot 0e+2)}.2 \cdot 2 + e^{(\rho-0\cdot 69).56}.1 \cdot 3)$

Acknowledgements

The work reported in this paper was supported by EPSRC grant GR/S62949.

REFERENCES

[1] C. Reinsch, "A synopsis of interval arithmetic," in *The relationship between numerical computing and programminmg languages*, J. K. Reid, Ed. North Holland, 1982, pp. 85–100.

[2] H. F. Trotter, "An elementary proof of the central limit theorem," *Arch. Math.*, vol. 10, pp. 226–234, 1959.

[3] C. Hughes and J. Hillman, "QoS explorer: A tool for exploring QoS in composed services," *icws*, vol. 0, pp. 797–806, 2006.

[4] L. Yang, A. Bundy, D. Berry, and C. Hughes, "Towards a bell-curve calculus for e-science," in *UK e-Science All Hands Meeting*, September 2006.

[5] L. Yang, "Towards a bell-curve calculus and its application to e-science," MPhil Thesis, School of Informatics, University of Edinburgh, 2006, forthcoming.