How to Find Big-Oh in Your Data Set (and How Not To)

C. C. McGeoch
Department of Mathematics and Computer Science,
Amherst College, Amherst, MA 01002 `ccm@cs.amherst.edu`
D. Precup
P. R. Cohen
Department of Computer Science, University of Massachussetts
Amherst, MA 01003 `precup,cohen@cs.umass.edu`

January 31, 1997

Abstract The ***empirical curve bounding problem*** is defined as follows. Suppose data vectors X, Y are presented such that $E(Y[i]) = \bar{f}(X[i])$ where $\bar{f}(x)$ is an unknown function. The problem is to analyze X, Y and obtain complexity bounds $O(g_u(x))$ and $\Omega(g_l(x))$ on the function $\bar{f}(x)$.

As no algorithm for empirical curve bounding can be guaranteed correct, we consider heuristics. Five heuristic algorithms are presented here, together with analytical results guaranteeing correctness for certain families of functions. Experimental evaluations of the correctness and tightness of bounds obtained by the rules for several constructed functions $\bar{f}(x)$ and real datasets are described. A hybrid method is shown to have very good performance on some kinds of functions, suggesting a general, iterative refinement procedure in which diagnostic features of the results of applying particular methods can be used to select additional methods.

1 Introduction

Suppose the expected cost of algorithm A under some probabilistic model is described by an unknown exact function $\bar{f}(x)$ which belongs to some unknown class $\Theta(\bar{g}(x))$ (where x denotes problem size). An experimental study of A produces a pair of vectors X, Y such that $E(Y[i]) = \bar{f}(X[i])$. The empirical curve-bounding problem, addressed in this paper, is: ***Analyze*** (X, Y) ***and estimate complexity classes*** $O(g_u(x))$ ***and*** $\Omega(g_l(x))$ ***to which*** $\bar{f}(x)$ ***belongs.*** While a primary goal of traditional algorithm analysis is to identify complexity classes to which unknown functions belong, this empirical version of the problem appears to be new. We can find no techniques in the data analysis literature designed for finding bounds on data, although much is known about fitting curves to data (see sec. 5). Approaches to domain-independent function finding [13] might be adapted to curve bounding and some are considered here.

For any finite set of points X there are functions $\bar{f}(x)$ of arbitrarily high degree but indistinguishable from the constant c at those points. Therefore any heuristic producing an upper bound estimate can be fooled, and no curve-bounding method can be guaranteed correct. This paper presents five robust heuristics that often produce correct bound estimates (or clear indications of failure) for broad classes of functions and for functions that tend to arise in practice. We describe each rule R together with a justification that describes a class F_R: for any function $\bar{f} \in F_R$, the rule is guaranteed to find correct (sometimes exact) bounds when applied to data vectors $Y = \bar{f}(X)$. We also present empirical studies of the rules using constructed multi-parameter functions, and "typical" data sets from algorithm analysis. The experiments indicate the limitations of the rules and suggest an appropriate level of conservatism in their application. We also discovered that the rules can "diagnose" qualitative features of functions $\bar{f}$, and given these diagnoses, we can apply additional rules that are specific to functions with these features, and attain higher levels of performance.

The rules can be viewed as interactive tools or as offline algorithms. To accomodate both views, we present the algorithms with a small set of ***oracle functions*** which decide, for example, whether "residuals are concave upwards." In interactive use, a human provides the oracle result; in offline use, a simple computation is used. However, the experiment in section 5 suggests that offline versions are far more efficient, and often more effective, than interactive versions.

2 Notation and the Heuristics

The vector X contains k distinct nonnegative values arranged in increasing order. Each heuristic takes a pair of vectors (X, Y) generated according to $Y = \bar{f}(X)$ or sometimes $E(Y) = \bar{f}(X)$. The heuristic reports a class estimator $g(x)$ together with a bound type, either ***upper***, ***lower***, or ***close***. ***Upper*** signifies a claim that $\bar{f}(x) \in O(g(x))$, and ***lower*** signifies a claim that $\bar{f}(x) \in \Omega(g(x))$. A ***boundtype*** = ***close*** is returned when a data set does not meet the rule's criteria for upper or lower bound claims. An upper bound estimate $O(g(x))$ is ***correct*** if in fact $\bar{f}(x) \in O(g(x))$. A correct upper bound is ***exact*** if $g(x)$ labels the smallest correct class. Analogous definitions hold for lower bound estimates. Some heuristics generate internal ***guess functions*** $f(x)$ before reporting the estimate $g(x)$; for convenience we assume that $g(x)$ and $\bar{g}(x)$ take the standard one-term form of complexity class labels.

The following computations are required by the oracle functions:

Trend(X, Y, c_r). Returns a value indicating whether Y appears to be increasing, decreasing, or neither. The function compares the correlation coefficient r (computed on X and Y) to a cutoff parameter c_r which is 0.1 by default.

Concavity (X, Y, s). The function examines signs of smoothed residuals from a linear regression fit of X to Y. It returns "concave upward" if signs obey the regular expression $(+)^+(-)^+(+)^+$, "concave downward" if they obey $(-)^+(+)^+(-)^+$, and otherwise "neither." The default low setting on parameter s produces "less smooth" residuals and more frequent "neither" results.

DownUp(X, Y, s). The DownUp oracle checks whether successive differences in smoothed Y values obey the regular expression $(-)^+(+)^+$, returning `True` or `False`. The default low value of parameter s produces more frequent `False` results.

NextCoef(f, *direction*, *cstep*) and NextOrder(f, *direction*, *estep*). Some rules iterate over several guesses and require an oracle to supply the next guess. This implementation constructs functions $f(x) = ax^b$ for positive rationals a and b. *NextCoef* changes a according to ***direction*** (up or down) and the ***cstep*** size. If a decrement of size ***cstep*** would give a negative coefficient, then ***cstep*** is reset to ***cstep***/10 before decrementing. *NextOrder* changes the exponent b according to the ***estep*** size. Default ***estep*** is .001 and initial ***cstep*** is .01.

2.1 Guess Ratio

The first heuristic is called the Guess Ration (GR) rule. To justify GR, let F_{GR} contain $\bar{f}(x) = a_1x^{b_1} + a_2x^{b_2} + \cdots + a_tx^{b_t}$, with rationals a_i positive, and b_i such that $b_1 > 0$, $b_i \geq 0$, and $b_i > b_{i+1}$. Let the guess function be of the form $f(x) = x^b$. Then the ratio $\bar{f}(x)/f(x)$ has the following properties: (1) When $\bar{f}(x) \in O(f(x))$, the ratio decreases to a nonnegative constant as x increases; (2) When $\bar{f}(x) \notin O(f(x))$ the ratio eventually increases and has a unique minimum point at some location x_r. If $x_r > 0$, then the ratio shows an initial decrease followed by an eventual increase. These properties are established by an application of Descartes' Rule of Signs [17], which bounds the number of sign changes in the derivative of the ratio.

If a plot of a finite sample of the ratio (X vs $Y/f(X)$) shows an eventual increasing trend, then (2) must hold. If only a decrease is observed, then cases (1) and (2) cannot be distinguished. The Guess Ratio rule begins with a constant guess function and increments b until the ratios $Y/f(X)$ do not appear to eventually increase. The largest guess for which an eventual increase is observed is reported as a "greatest lower bound" found. When $\bar{f}(x) \in F_{GR}$ and $k \geq 2$, the correctness of GR can be guaranteed simply by defining "eventual increase" as $Y[k-1] < Y[k]$. However our implementation uses the Trend oracle for this test because of possible random noise in Y. For any data set (X, Y) and our Trend oracle, the rule must eventually terminate.

2.2 Guess Difference

The Guess Difference (GD) rule evaluates differences $f(X) - Y$ to produce an upper bound estimate. This rule is effective for the class F_{GD} which contains functions $f(x) = cx^d + e$ where c, d and e are positive rationals. Let the guess have the form $f(x) = ax^b$. Consider the *difference curve* $f(x) - \bar{f}(x)$. When $f(x) \notin O(\bar{f}(x))$ this curve eventually increases and has a unique minimum at some location x_d. Also, x_d is inversely related to the coefficient a: for large a the difference curve increases everywhere ($x_d = 0$), but for small a there might be an initial decrease. In the latter case we say the curve has the DownUp property.

The GD rule starts with an upper bound guess $f(x) = ax^b$ and searches for a difference curve with the DownUp property by adjusting the coefficient a. If a DownUp curve is found, the rule concludes that $f(x)$ overestimates the order of $\bar{f}(x)$, so it decrements b and tries adjusting a again. The lowest b for which the rule finds a DownUp curve is reported as a "least upper bound" found. Using an analysis similar to that for GR, we can show that when $\bar{f}(x) \in F_{GD}$ and $k \geq 4$, and X is fixed, there exists an a such that $f(X) - \bar{f}(Y)$ will have the DownUp property. If the rule is able to find a DownUp curve in its finite sample, then the upper bound it returns must be correct. We also show that the DownUp property cannot guarantee correctness for functions from F_{GR}. In our implementation, if the rule is unable to find an initial DownUp curve within preset limits, it stops and reports the original guess provided by the user.

2.3 Power Rule

The Power Rule (PW) modifies a standard method for curve-fitting (see [12]). Suppose that F_P contains functions $\bar{f}(x) = cx^d$ for positive c and d. Let $y = \bar{f}(x)$. Transforming $x' = \ln(x)$ and $y' = \ln(y)$, we obtain $y' = dx' + c$. The Power Rule applies this log-log transformation to X and Y and then reports d, the slope of a linear regression fit on the new scale. The Concavity oracle, applied to residuals from the regression, determines whether an upper or lower bound (or neither) is claimed. If $Y = \bar{f}(X)$ and $\bar{f}(X) \in F_P$ then the Power rule finds d exactly. If $Y = \bar{f}(X) + \epsilon$ and the random noise component ϵ obeys standard assumptions of independence and lognormality, then confidence intervals on the estimate of d can be derived.

High-End Power Rule (PW3). When $\bar{f}(x)$ has low-order terms (such as $ax^b + e$), the transformed points do not lie on a straight line, and regression using only the j highest design points might give a better asymptotic bound than one using all k design points. The PW3 variation tested in this paper applies the Power rule to the data points for $X[k-2]$, $X[k-1]$, $X[k]$.

Power Rule with Differences (PWD). The *differencing* variation on the power rule attempts to straighten out plots under log-log transformation by removing constant and logarithmic terms. This variation is applicable when the X are chosen such that $X[i] = \Delta \cdot X[i-1]$ for a positive constant Δ. The variation applies the Power rule to *successive differences* in adjacent Y values.

To justify this rule, suppose F_{PWD} contains $\bar{f}(x) = cx^d + e$ where c, d and e are positive constants, and let $Y = \bar{f}(X)$. Set $Y'[i] = Y[i+1] - Y[i]$ and $X'[1..k-1] = X[1..k-1]$. Then that $Y' = c'X'^d$ (with a new coefficient and with e gone), to which the basic power rule can be applied. When $\bar{f}(x) \in F_{PWD}$, $Y = \bar{f}(X)$ and $k > 2$, the PWD rule finds d exactly. Differencing affects other kinds of terms: for example, taking differences twice will remove logarithms.

2.4 The BoxCox rule.

A general approach to curve-fitting is to find transformations on Y or on X, or both, that produce a straight line in the transformed scale. For example, if $Y = X^2$, then a plot of X vs $\sqrt{Y}$ would produce a straight line, as would a plot of X^2 vs Y.

The Box-Cox ([1], [5]) transformation on Y is parameterized by λ. This transformation is applied together with a "straightness" statistic that permits comparisons across different parameter levels. The transformation is as follows:

$$Y^{(\lambda)} = \begin{cases} \frac{Y^\lambda - 1}{\lambda \bar{Y}^{\lambda-1}} & \text{if } \lambda \neq 0 \\ \bar{Y}\ln(Y) & \text{if } \lambda = 0 \end{cases}$$

where $\bar{Y}$ is the geometric mean of Y, equal to exp(mean(ln (Y))). The "best" transformation in this family minimizes the Residual Sum of Squares (RSS) statistic which is calculated from X and Y^λ.

Our BC rule iterates over a range of guesses $f(x) = x^b$, evaluating $Y^{(\lambda)}$ with $\lambda = 1/b$. The Concavity of residuals from the best transformation found determines the type of bound claimed. When $\bar{f}(x) = F_{PW}$, $Y = \bar{f}(X)$, $k > 2$, and the NextGuess oracle includes $\bar{f}(x)$, this rule finds the function exactly. With standard normality assumptions about an added random error term, it is possible to calculate confidence intervals for the estimate b; see [1] or [5] for details.

2.5 The Difference Rule.

The **Difference** heuristic extends Newton's divided difference method for polynomial interpolation (see [15] for an introduction) to be defined when Y contains random noise and nonpolynomial terms. The method iterates numerical differentiation on X and Y until the data appears non-increasing, according to the Trend oracle. The number of iterations d required to obtain this condition provides an upper bound guess x^d. When $\bar{f}(x)$ is a positive increasing polynomial of degree d, $k > d$, and $Y = \bar{f}(X)$ then this method is guaranteed correct. Much is known about numerical robustness, best choice of design points, and (non)convergence when $k \leq d$.

3 Experimental Results

The rules have been implemented in the S language [2], designed for statistical and graphical computations. The experiments were carried out on a Sun SPARCstation ELC, using functions running within the Splus statistical/graphics package; some supporting experiments were conducted using the CLASP statistical/graphics package, and the method labelled HY in Figure 1 was implemented in C. Timing statistics would be misleading in this context and are not reported in detail. Roughly, the Power rules required a few microseconds, and the iterative rules usually took no more than a few seconds per trial. The Guess Difference rule required a coarser *estep* value in the NextOrder oracle (.01 instead of .001) to produce comparable running times.

3.1 Parameterized Functions

The first experiment studies the sensitivity of the rules to second order terms, using functions $\bar{f}(x) = ax^b + cx^d + e$ (with no random term). Very roughly, the particular constants for this test were chosen after several months of exploration to highlight the boundary between functions that are "easy for all rules" and "hard for all rules." Vector X takes powers of two between 8 and 128. In Figure 1, the notations **l, u, c**, indicate the type of bound claimed. An underline marks an incorrect bound, and an $\underline{\text{X}}$ marks a case where the heuristic failed to return a meaningful result. We will defer discussing the results in column HY until section 4.

No.	Function	GR	GD	PW	PW3	PWD	BC	DF	HY
1	$3x^{.2} + 1$	.171l	(2.26).24u	.171l	.174l	.2u	.178l	1u	$.03x^{.60} + 4x^{0.15}$
2	$3x^{.2} + 10^2$	.011l	(2.26).24u	.011l	.012l	.2l	.012l	1u	$.12x^{.56} + 103x^{.01}$
3	$3x^{.2} + 10^4$	.0001l	(2.27).24u	.0001l	.0004l	.2l	$\underline{\text{X}}$	1u	$.12x^{.55} + 10003$
4	$3x^{.8} + 10^4$	.004l	(1.0)1u*	.004l	.006l	.8l	$\underline{\text{X}}$	1u	$1.55x^{.91} + 10003$
5	$3x^{.8} + x^{.2}$	.775l	(1.0)1u*	.774l	.784l	.793l	.792l	1u	$.22x^{1.1} + 4x^{.67}$
6	$3x^{.8} - x^{.2}$	$\underline{\text{.825l}}$	(1.0)1u*	.829u	.817u	.807u	.809u	1u	$-.34x^{1.26} + 2x^{1.03}$
7	$3x^{.8} + 10^4x^{.2}$	.201l	(1.0)1u*	.202l	.202l	.206l	.203l	1u	$x^{.97} + 10003x^{.2}$
8	$3x^{.8} + x^{.6}$	.771l	(1.0)1u*	.771l	.775l	.778l	.778l	1u	$.03x^{1.2} + 4x^{.75}$
9	$3x^{.8} - x^{.6}$	$\underline{\text{.838l}}$	(1.8).88u	.841u	.834u	.829u	$\underline{\text{.819l}}$	1u	$-.05x^{1.26} + 2x^{.89}$
10	$3x^{.8} + 10^4x^{.6}$	.600l	(1.0)1u*	.600l	.600l	.600l	.600l	1u	$.12x^{1.15} + 10003x^{.6}$
11	$3x^{.8} - 10^4x^{.6} + 10^6$	$\underline{\text{-.01l}}$	(1.0)1u*	$\underline{\text{-.059u}}$	$\underline{\text{-.086u}}$	$\underline{\text{X}}$	$\underline{\text{X}}$	$\underline{\text{0u}}$	$-3361x^{.77} + 990003x^{-.01}$
12	$3x^{1.2} + 10^4$	0.035l	(2.8)1.22u	.032l	.056l	1.2l	$\underline{\text{X}}$	2u	$2.4x^{1.25} + 10003$
13	$3x^{1.2} + x^{.2}$	1.187l	(2.8)1.22u	1.187l	1.194l	1.198l	1.2u	2u	$.48x^{1.4} + 4x^{1.01}$
14	$3x^{1.2} + 10^4x^{.2}$	0.213l	$\underline{\text{X}}$	0.212l	0.220l	0.263l	0.231l	$\underline{\text{1u}}$	$2.03x^{1.27} + 10003x^{.2}$
15	$3x^{1.2} + x^{.8}$	1.169l	(3.1)1.21u	1.168l	1.175l	1.178l	$\underline{\text{1.183u}}$	2u	$.11x^{1.54} + 4x^{1.11}$
16	$3x^{1.2} - x^{.8}$	$\underline{\text{1.235l}}$	(2.2)1.26u	1.238u	1.227u	1.223u	$\underline{\text{1.218l}}$	2u	$-.18x^{1.65} + 2x^{1.36}$
17	$3x^{1.2} + 10^4x^{.8}$	0.800l	(1.0)2u*	0.800l	0.801l	0.801l	$\underline{\text{0.801u}}$	$\underline{\text{1u}}$	$.48x^{1.45} + 10003x^{.8}$

Figure 1: Parameterized nonrandom functions

The functions tend to track large positive second terms. For example, for the function $\bar{f}(x) = 3x^{.8} + x^{.2}$, most of the methods estimate b to be in the range 0.77 to 0.79, which are correct and close lower bounds on the true value of .8. But for the function $\bar{f}(x) = 3x^{.8} + 10^4x^{.2}$, these same methods estimate b to be .2, tracking the exponent of the larger second term. Negated second terms can present problems, particularly for the GR method. GD does remarkably well at estimating the coefficient of the first term, although it is an iterative algorithm and its performance is sensitive to the choice of initial guess and step size. The starred entries mark cases where the rule failed to find a DownUp curve and returned the user-supplied guess which was either $1x^1$ (functions 1 through 11) or $1x^2$ (functions 12 through 17). Both PW3 and PWD give tighter bounds than PW; not only does PWD successfully eliminate constants, but it is slightly better than PW and PW3 when the second term is non–constant. The BC rule provides very competitive bounds when it works, but it goes into an infinite loop on functions with a large-magnitude constant as a second term; the failure of BC on these functions is an intrinsic property of the λ transformation. Like PWD, the differencing operation of DF makes it insensitive to large constant terms. Because DF returns an integer exponent its bound is never tight on this test set. Function 11 disastrous for all the rules because the negated second term causes Y to be decreasing within its range.

Larger Problem Size An obvious remedy to the problem of a dominant second-order term is to use larger problem sizes. A second experiment uses the same functions as above, except X takes values at powers of two in the range $8 \ldots 256$ rather than $8 \ldots 128$. That is, the largest problem size is doubled. This had very little effect on the bounds returned by Guess Ratio and the three Power

Rules. The ***change*** in estimate is generally only in the third decimal place, and incorrect bounds remain incorrect. We can argue that GR would probably be least affected by larger problem sizes, but one might expect greater responsiveness of PW3 because the new point should have greater leverage. The greatest improvement is found in the Guess Difference rule on functions 4 through 9 (excepting 7). In the previous experiment the rule failed to find an initial DownUp curve at all—now the rule finds upper bounds within .05 of the true exponent. BC also shows some very slight improvement; in two cases the rule produces ***close*** bound claims (which are hard to evaluate) where previously it had been incorrect.

Adding Random Noise. We added a random term to three easy functions (1, 5, and 13) to learn how rule performance degrades with increased variance. We let $Y = \bar{f}(X) + \epsilon_i$ with and $i = 1, 2, 3$. The ϵ_i are drawn independently from normal distributions with means 0 and standard deviations set to 1, 10 and the function means $\bar{f}(X[j])$, for $i = 1, 2, 3$ respectively. We ran two independent trials for each i.

The quality of results returned by all rules degrades as variance increases and the replication of tests in each category demonstrates that many correct bounds are spurious. Conversely, rule performance improves when variance decreases. We suspect that some of the decrement in performance is due to the oracles, some of which are not particularly robust. For example, a robust linear fit would probably make more sense than linear regression as an oracle for slope. Encouragingly, it is usually possible to reduce variance in data by increasing the number of trials or by applying variance reduction techniques [10].

With greater variance in Y the Power and the BoxCox rules more frequently return claims of ***close***, which are hard to evaluate. Large variance has less impact when the ***change*** in Y is large. Our implementations of the BC and PWD rules encounter difficulties with negative values and negative differences in case ϵ_3; the former can be remedied by adding a large positive constant to the data, but this introduces new inaccuracies.

3.2 Algorithmic Data Sets

This experiment applied the rules to eight data sets drawn from previous computational experiments by the first author. The data sets were not originally intended for this purpose and may give more realistic indications of performance. Data sets 1 and 2 are the expected costs of Quicksort and Insertion Sort, for which formulas are known exactly [9]. Sets 3 through 6 are from experiments on the FFD and FF rules for bin packing [3], [4]. Sets 7 and 8 are from experiments on distances in random graphs having uniform edge weights [11]. The X vectors have various ranges and intervals; except for the first two cases, the Ys represent means of several independent trials.

Results appear in Figure 2. The left column presents the best analytical bounds known for each. The entries NA for PWD mark cases where this rule was not applied because design points were not in required format. Results in column HY are discussed in the following section.

Contrary to experience with the constructed functions, GR obtains a correct and tight bound when a negated second term is present (case 2), but in four cases GR produces results violating known bounds. GD and the Power Rules rarely violate known bounds, although without tighter analyses it is impossible to tell whether the rules are always correct. BC nearly always returns a "close" bound which is difficult to evaluate. Interestingly, every incorrect bound produced by the rules is a lower bound.

The most interesting results are in cases 6,7 and 8, which have gaps in the known asymptotic bounds. In 6, the rules provide consensus support for a conjecture that $\bar{f}(x)$ is closer to linear than, say, to $\sqrt{x}$. In 7, there is some very slim support for super-linear growth in the data set, but the rules

	Known	GR	GD	PW	PW3	PWD	BC	DF	HY
1	$y = (x+1)(2H_{x+1} - 2)$	1.2l	1.24u	1.221u	1.181u	NA	1.181c	2u	$2.1x^{1.25} + 11.23x^{.64}$
2	$y = (x^2 - x)/4$	2.0l	2.03u	3.003u	3.001u	NA	2.0l	2u	$.08x^{2.71} - .01x^{3.11}$
3	$E(y) = x/2 + O(1/x^2)$	.99l	1u*	0.996l	.999u	1.0002c	1.203c	2u	$1.5x^{1.58} - .01x^{2.12}$
4	$E(y) \in \Theta(x^{.5})$	.52l	1u*	0.555c	.5716u	.7785c	0.999c	1u	$.2x^{.57} - .01x^{.8}$
5	$E(y) \in O(x^{2/3}(\log x)^{1/2})$	.68l	.72u	0.689c	.695u	.692c	.687c	1u	$.22x^{.71} - 0.00x^{1.07}$
	$E(y) \in \Omega(x^{2/3})$								
6	$E(y) \leq .68x$	.90l	1u	0.893l	.954l	1.269l	.976c	1u	$0.00x^{1.13} + .68x^{.47}$
7	$x - 1 \leq y \leq 13.5x \log_e x$	1.13l	1.18u	1.142u	1.125l	NA	1.109c	2u	$1x^{1.21} - 0.00x^{1.81}$
8	$x \log_e x < y < 1.2x^2$	1.30l	1.47u	1.318u	1.201l	NA	1.203c	2u	$.15x^{2.01} - 0.01x^{2.31}$

Figure 2: Tests on Algorithmic Data

are not really powerful enough to make such fine discriminations. In 8 the results give consensus support for a conjecture of sub-quadratic growth.

4 Iterative Refinement, Combining Methods

One known pathology of the heuristics presented so far is their sensitivity to low-order terms with large coefficients. For this type of function, the heuristics tend to track the low-order term.

This problem can be overcome by an iterative diagnosis and repair technique that combines the existing heuristics to produce improved models. The technique is designed to find upper bounds for functions of the form $ax^b + cx^d$ with rational exponents $b > d \geq 0$ and real coefficients $a << c$. This method represents a departure from our approach up to now: The earlier methods were intended to be general, but this one is specific to functions with relatively large coefficients on low order terms. This suggests a new role for the methods we have discussed so far: Instead of using them to guess at the order of a function, they can provide diagnostic information about the function (e.g., whether $a << c$), and then more specific, purpose-built methods, designed for particular kinds of functions, can estimate parameters.

To illustrate this new approach, we developed a three-step hybrid method for functions of the form $\bar{f}(x) = ax^b + cx^d$;

1. Apply a discrete derivative (the Difference rule) to the datasets, in order to find the integer interval of the exponent b.

2. Refine the guess for the exponent using the Guess Ratio rule. We start with the known upper and lower bound for the exponent, u and l. At each step we consider the model $x^{(u+l)/2}$ by plotting x against $y/x^{(u+l)/2}$. If the plotted points appear to be decreasing, then $(u+l)/2$ is overestimating the exponent, and we replace u by $(u+l)/2$. If the points are increasing, then l will be replaced. The estimates are refined until u and l get within a desired distance ϵ of each other. At this point, if the dataset $y/x^{(u+l)/2}$ has a DownUp feature, then we know that function $\bar{f}$ must have a relatively high coefficient c on a low order term. This diagnosis invokes the next step.

3. If, as we suspect, the current result is tracking a low-order term with a high coefficient, then this term will dominate $\bar{f}$ for small values of x. Thus we can approximate the upper bound for small x's to be cx^d. Let (x_1, y_1) and (x_2, y_2) be two points from the beginning part of the curve. If we consider that $y_1 \approx cx_1^d$ and $y_2 \approx cx_2^d$, then d can be approximated by $\frac{\log y_1 - \log y_2}{\log x_1 - \log x_2}$,

and c is $\frac{y_1}{x_1^d}$. Now we can correct the model using these estimates, in order to make the high-order term appear. For all points (x, y), we transform y into $\frac{y}{x^d} - c$. Now we can apply the same procedure as above to find the a and b parameters, assuming that $y \approx ax^b$. In this case, though, we use for our estimates two points that have high values of x, as the influence of the high-order term is stronger for these points.

This technique illustrates a way in which models can be improved by generating data and comparing it against the real values to obtain diagnostic information (step 2), which suggests a method specific to the diagnosis—in this case, a method specific to functions with large coefficients on low order terms. (We envision similar diagnostics and methods for functions with negative coefficients, but we haven't designed them, yet.)

The results of this method are found in the columns labelled HY in Figures 1 and 2. The results are tight upper bounds when $\bar{f}$ does in fact contain a low order term with a large coefficient (functions 7, 10, 11, 14, 17 in Fig. 1). In fact, these bounds are tighter than those returned by the other methods, and, remarkably, this hybrid method estimates coefficients and low order exponents very well. When the functions do not contain low order terms with large coefficients, the bounds returned by this method remain correct but they are looser than those given by other methods. Interestingly, this situation is often indicated by very low estimated coefficients on the high order terms; for example, in funtion 1 (Fig. 1), the coefficient of the first term is .03. The only cases when the technique fails are those in which negative coefficients appear in the low-order terms. The failure is probably due to the sensitivity of the Guess Ratio heuristic to such circumstances. This new method was also tested on noisy datasets but the noise had negligible effects. The new method used different oracles and different implementations of oracles from the previous methods, which might account for the relatively robust performance. Or, the small effects of noise might be due to a different method for sampling data from the given functions. Clearly, the effects of noise on these methods are still poorly understood.

5 Remarks

In our informal explorations and designed experiments with little or no random noise in the data, the rules generally get within a $\sqrt{x}$ factor of the exact bound. On data from algorithms, the rules can get within a factor of x and sometimes within $\sqrt{x}$. The rules are not reliable in discerning lower-order and logarithmic factors (this holds even when logarithms are added to the NextOrder oracle), and it doesn't seem likely that taking larger problem sizes would help.

Most rules do not much respond to larger problem sizes. However the quality of bound obtained is very responsive to variance in the data. This is good news for algorithm analyzers when Y is correlated with runtime, since variance can be reduced by taking more random trials, and trials are easier to get when Y grows slowly.

Can Humans Do Better? In one experiment, the third author was given the 25 data sets presented here, without any information about their provenance, and was allowed to use any data analysis tools to bound the function. He was more frequently incorrect than any of the implemented rules, and the human/machine interactions took considerably more time to accomplish. A second experiment involved strict application of the heuristics, but with a human oracle (the first co-author) who was familiar with the eight algorithmic data sets. Again, interactive trials require much more time to perform. Very preliminary results indicate that: GR produces worse (less close) bounds with a human Trend oracle; the human Concavity oracle tends to agree with the implemented one in the Power rules (no improvement); an interactive GD is more successful at finding DownUp curves (more frequent success, but not tighter bounds); an interactive BoxCox can be more successful by providing bounds that bracket the estimate rather than optimizing the transformation.

Removing Constant Terms. In many applications it may be possible to remove a constant from Y before analysis, either by testing with $x = 0$ or by subtracting an estimated constant. Our preliminary results suggest that subtraction of a known constant uniformly improves all the rules, but subtracting an estimated constant gives mixed results.

Some Negative Results. A basic requirement is that a heuristic be internally consistent. That is, should not be possible to reach the contradictory conclusions "Y is growing faster than X^2" and "Y is growing more slowly than X^2" on the same data set. Surprisingly, two plausible approaches turn out to have exactly this failure. The first, which is perhaps the most obvious approach to the bounding problem, is to use general regression to fit a function $f(x)$ and to read its leading term, using regression analysis to determine an upper/lower bound claim. In preliminary tests with this approach it quickly became clear that the results were primarily artifacts of the regression technique: contradictory bound claims, such as $\Omega(x^{2.2})$ and $O(x^{1.8})$ were easy to obtain by small changes in the regression method. This approach was abandoned early in this research. The second is based on Tukey's [16] "ladder of transformations." This approach also gives contradictory results depending on whether the transformation is applied to Y or X.

References

[1] A. C. Atkinson (1987) ***Plots, Transformations and Regression: an Introduction to Graphical Methods of Diagnostic Regression Analysis,*** Oxford Science.

[2] R. A. Becker, J. A. Chambers, and A. R. Wilks (1988) ***The New S Language: A Programming Enviornment for Data Analysis and Graphics***, Wadsworth & Brooks/Cole.

[3] J. L. Bentley, D. S. Johnson, F. T. Leighton, and C. C. McGeoch (1983) "An experimental study of bin packing," ***Proceedings of the 21st Allerton Conference on Communication, Control, and Computing***, University of Illinois, Urbana-Champaign. pp 51–60.

[4] J. L. Bentley, D. S. Johnson, C. C. McGeoch and L. A. McGeoch (1984). "Some unexpected expected behavior results for bin packing," ***Proceedings of the 16th Symposium on Theory of Computing***, ACM, NY. pp 279–298.

[5] G. P. Box, W. G. Hunter, and J. S. Hunter (1978) ***Statistics for Experimenters,*** Wiley & Sons.

[6] J. M. Chambers et al. (1983) ***Graphical Methods for Data Analysis***, Duxbury Press.

[7] P. R. Cohen (1995) ***Empirical Methods for Artificial Intelligence***, the MIT Press.

[8] T. Cormen, C. Leiserson and R. Rivest (1990) ***Introduction to Algorithms***, the MIT Press.

[9] D. E. Knuth (1981), ***The Art of Computer Programming: Vol. 3 Sorting and Searching***, Addison Wesley.

[10] C. C. McGeoch (1992), "Analyzing algorithms by simulation: Variance reduction techniques and simulation speedups," ***ACM Computing Surveys.*** (245)2, pp. 195–212.

[11] C. C. McGeoch (1995) "All pairs shortest paths and the essential subgraph," ***Algorithmica*** (13), pp. 426–441.

[12] J. O. Rawlings (1988) ***Applied Regression Analysis: A Research Tool***, Wadsworth & Brooks/Cole.

[13] C. Schaffer (1990) ***Domain-Independent Scientific Function Finding***, Ph.D. Thesis, Technical Report LCSR-TR-149, Department of Computer Science, Rutgers University.

[14] R. Sedgewick (1975), ***Quicksort.*** Ph. D. Thesis, Stanford University.

[15] J. Soer and R. Bulirsch (1993) ***Introduction to Numerical Analysis***, Springer-Verlag.

[16] J. W. Tukey (1977) ***Exploratory Data Analysis***, Addison-Wesley.

[17] L. Weisner (1938) ***Introduction to the Theory of Equations.***, Macmillan.