ON AVERAGE CASE ANALYSIS THROUGH STATISTICAL BOUNDS : LINKING THEORY TO PRACTICE

Niraj Kumar Singh[1], Soubhik Chakraborty[2*] and Dheeresh Kumar Mallick[3]

[1,3]Department of Computer Science & Engineering, BIT Mesra, Ranchi-835215, India
[2]Department of Applied Mathematics, BIT Mesra, Ranchi-835215, India
{niraj_2027, soubhikc}@yahoo.co.in,dkmallick@gmail.com

ABSTRACT

Theoretical analysis of algorithms involves counting of operations and a separate bound is provided for a specific operation type. Such a methodology is plagued with its inherent limitations. In this paper we argue as to why we should prefer weight based statistical bounds, which permit mixing of operations, instead as a robust approach. Empirical analysis is an important idea and should be used to supplement and compliment its existing theoretical counterpart as empirically we can work on weights (e.g. time of an operation can be taken as its weight). Not surprisingly, it should not only be taken as an opportunity so as to amend the mistakes already committed knowingly or unknowingly but also to tell a new story.

KEYWORDS

Theoretical analysis, empirical analysis, statistical bounds, empirical-O, average case analysis, computer experiment section.

1. INTRODUCTION

Arguably Computer Science is or at least has aspects of an experimental science – see for example the Turing Award Lecture of Juris Hartmanis and the various responses to this lecture [1]. Traditionally, algorithmic complexity is obtained through mathematical bounds which are operation specific and where each of the key operations is counted separately. Identifying correctly the dominant regions (so called key operations) in a complex code is a non trivial exercise [2]. We might come across a situation where the algorithms under comparison don't have all their operations in common (and such a case is not uncommon at all!). In such scenarios the comparative analysis becomes a tricky issue.

Further, the average case analysis of an arbitrary algorithm assumes some probability distribution from where the input is expected to come which is generally taken to be uniform. The relevance of a specific input distribution is context dependent and hence should not be pre-assumed to be uniform. Our argument is well supported by some recent breakthroughs [3], [4] and [5] in the field of algorithmic complexity analysis, which put a serious question mark on the robustness claim of theoretical methodology. We are of the view that though the theoretical analysis is a strong science in worst case (as the mathematical bounds here do give a guarantee), it is also saturated in the sense that if a guarantee giving bound becomes conservative, no certificate is given on the level of conservativeness. A scientific study of empirical analysis can be exploited to supplement and compliment the known concepts of theoretical analysis. In average case, especially where the code is complex or where the robustness can be challenged, empirical analysis assumes an acknowledged necessity. Statistical bounds were created to make average case analysis a meaningful science as well as provide the aforesaid certificate in worst case and guard against making a tall unrealistic theoretical claim in the best case.

2. WHY EMPIRICAL ANALYSIS IS NECESSARY?

In mathematical analysis, for the algorithms having more than one key operation, we need to correctly identify the various keys present in the pseudocode. Once the keys are identified, a separate bound is assigned to each of them. These bound are functions (in terms of size of input characteristics) specifying their frequencies. These functions are clearly operation specific. Different operations may have different (some time significantly different) bounding functions, and when mixed the resultant behavior might be a somewhat different function in the specified finite range of input sizes. So, it is not always justifiable to use such bounds to represent the complexity of the whole algorithm.

In the context of discussion made in the above section, theoretical analysis seems to be handicapped as it does not give much of an idea of how well a given algorithm will perform in a specific situation [6], [7] and the empirical analysis can be of great help here. According to Sedgewick [6], "it is unfortunately all too often the case that mathematical analysis can shed very little light on how well a given algorithm can be expected to perform in a given situation". Sedgewick and Flajolet [7] consider empirical analysis a crucial part of the analysis of algorithms. Their approach (see chapter 1 of their book) makes it clear that doing empirical analysis properly is a non-trivial exercise.

Empirical analysis is an important idea and should be used to supplement and compliment its existing theoretical counterpart. Not surprisingly, it should even be taken as an opportunity to amend the mistakes already committed knowingly or unknowingly.

Perhaps the biggest obstacle in the way towards the acceptance of empirical analysis lies in the lack of a uniform and homogeneous methodology. Brunskill and Turner [8] have identified some factors which have their contributions on the running time of an implementation. These factors include:-

- the CPU
- the compiler
- the programming language
- the way the program is constructed
- time for disk accesses and other IO devices
- whether the system is single or multitasking

If empirical approach is to be made a viable alternative to the theoretical one then it must have to be robust as well. It is a well known fact that the success of mathematical approach is largely attributed to its system independent nature. In empirical analysis, where we have to make our way through an entirely heterogeneous environment, we expect at least some equally powerful tool on our side. Empirical-O (see appendix for the definition) is such a tool which we are going to have with us throughout our journey while performing empirical analysis through statistical bounds. Chakraborty and Sourabh in their paper [5] strongly advocate as to why an algorithmic time Complexity measure can be system invariant rather than system independent. In the subsequent section we present strong arguments to prove our point.

3. AVERAGE CASE ANALYSIS THROUGH STATISTICAL BOUNDS

Although, it is not possible to define it precisely, an average case input typically corresponds to an unbiased set of data, where the term 'unbiased' obviously is context dependent. Drawing an exact boundary line between the biased and unbiased data sets, in our opinion, is both non trivial as well as non feasible exercise. Average case analysis is an important field of study in algorithmic analysis as it explains how certain algorithms with bad worst case complexity perform better on average [9]. A very common omission while making such a claim often lies in not verifying the robustness of the average complexity in question. Average behavior of an algorithmic performance is theoretically obtained by applying mathematical expectation to the dominant operation(s) present within the code. Identifying the key operations correctly is not always an easy task, particularly in case of complex codes. The empirical approach, where statistical bounds are used has much to offer positively in this direction. Instead of considering each operation separately, it permits collective consideration of all the operations, trivial or non-trivial, and thus the crucial task of identifying (and that too correctly) the key(s) is resolved. For definition of statistical bounds, empirical-O and more, see appendix and [10].

We are still not over here. Identifying the key(s) alone is not a guarantee of successful analysis. Yet another problem is of the probability distribution over which the expectation is taken. In most of the theoretical analyses these expectations are derived for uniform probability inputs. We should not pre-assume the distribution in this manner as such faulty assumptions might lead to misleading results, see for example rejection of Knuth's proof in [3] and Hoare's proof in [4]. In continuation to our arguments in favor of statistical bounds, it appears from [5] that the average complexity in Schoor's matrix multiplication algorithm is not the expected number of multiplications $O(d_1 d_2 n^3)$, d_1 and d_2 being the density (fraction of non zero elements) of pre and post factor matrices, but the exact number of comparisons which is n^2 provided there are sufficient zeroes and surprisingly we don't need a sparse matrix to get an empirical $O_{emp}(n^2)$ complexity. There are sufficient reasons to make us believe that it is possible to achieve the same complexity class empirically for still wider range of data and that too on a more robust set of inputs. Our work in this direction is in progress and we hope to come up with stunning results in the near future.

3.1 Statistical Bounds: An Extension to Parallel Computing

It seems that the idea of statistical bounds can be well extended even to the field of parallel computing. Theoretically it is impossible to get a speedup of more than p for any problem on a p-processor machine. Although the asymptotic speedup is ruled out, nevertheless, when the speedup is defined with respect to the actual run times on the parallel machines, it is possible to obtain super-linear speedup. Two of the possible reasons for such an anomaly are (1) p processors have more aggregate memory than one and (2) the cache hit frequency may be better for the parallel machine as the p-processors may have more aggregate cache than does one processor [11].

The possibility of getting a super-linear speedup might be an indication towards the presence of probable significant gaps between the mathematical and the statistical bounds (given that a statistical bound is empirically estimated for a feasibly finite range of inputs). In such a scenario it would be interesting to investigate such gaps for some practically important problems.

4. Conclusion And Future Work

The incapability of correctly identifying all the potential key operation(s) may lead to gaps (with varying extent) between the theoretical bounds and their corresponding empirical estimates. In certain situations dominant operation(s) might not be very obvious and may not even be present inside the algorithm. It may happen that some of such key(s) may be associated to run time access pattern or be related to some other less obvious factors which are concerned with the actual experimentation. So, our future work includes identifying (and quantifying as well) the factors leading to inconsistency in these two categories. Looking at some recent research papers even the possibility of rejection of already theoretically proved results cannot be ruled out. We hope to encounter a few more!

References

[1] Hartmanis, J.: Turing award lecture: On computational complexity and the nature of computer science. ACM Computing Surveys, 27(1):7-16, Responses – pages 17-61, (March 1995)

[2] Aho, A., Hopcroft, J., Ullman, J.: Data Structures and Algorithms. Pearson Education (2000)

[3] Chakraborty, S.,Sourabh, S.K.: How robust are Average Complexity Measures? A Statistical Case Study. Applied Mathematics and Computation, vol. 189(2), pp. 1787- 1797, (2007)

[4] Sourabh, S.K., Chakraborty, S.: How Robust is quicksort average case complexity? arXiv:0811.4376v1[cs.DS]

[5] Chakraborty, S., Sourabh, S.K.: On why algorithmic time complexity measure can be system invariant rather than system independent. Applied Mathematics and Computation. vol. 190(1), pp. 195-204, (2007)

[6] Sedgewick, R.: Algorithms. Addison-Wesley, Reading,MA, second edition, (1988)

[7] Sedgewick, R., Flajolet, P.: An Introduction to the Analysis of Algorithms. Addison-Wesley Reading,MA, (1996)

[8] D. Brunskill and J. Turner. Understanding Algorithms and Data Structures. Mac Graw-Hill, Maidenhead England, (1996) 1997 Reprint.

[9] Singh, N.K., Chakraborty, S., Pal, M.: Partition Sort Revisited: Reconfirming the Robustness in Average Case and Much More!. IJCSEA, Vol. 2, No. 1, (February 2012)

[10] Chakraborty, S., Sourabh, S.K.: A Computer Experiment Oriented Approach to Algorithmic Complexity. Lambert Academic Publishing, (2010)

[11] Horowitz, E., Sahni, S., Rajasekaran, S.: Fundamentals of Computer Algorithms. Galgotia Publications pvt. Ltd. (2003)

[12] Sacks, J., Weltch, W., Mitchel, T. and Wynn, H., Design and Analysis of Computer Experiments. Statistical Science Vol.4 (4), (1989)

Appendix:

Definitions 1 & 2 are taken from [10]
Definition 1: Statistical bound (non-probabilistic)

If w_{ij} is the weight of (a computing) operation of type i in the j-th repetition (generally time is taken as a weight) and y is a "stochastic realization" (which may not be stochastic) of the deterministic $T=\sum 1.\ w_{ij}$, where we count one for each operation repetition irrespective of the type, the statistical bound of the algorithm is the asymptotic least upper bound of y expressed as a function of 'n', where n is the input parameter characterizing the algorithm's input size.

Definition 2: Empirical –O (written as O with a subscript emp)

It is defined as "the O that corresponds to the simplest model fitted to (time) complexity data that is neither an under-fit nor an over-fit". More specifically, it corresponds to a model that does not invite the serious problem of ill-conditioning nor does it lead to a loss of predictive power.

It is an empirical estimate of the statistical bound over a finite range, obtained by supplying numerical values to the weights, which emerge from computer experiments. A computer experiment is a series of runs of a code for various inputs. A deterministic computer experiment is one which gives identical results if the code is re-run for the same inputs. Algorithmic complexity fortunately is expressed in terms of input size and not a specific input so that we have a ground for stochastic modeling as what is deterministic for a fixed input may be taken as stochastic for a fixed input size and randomly varying input elements at least for large input size. The other interest is that a stochastic model makes prediction cheap and efficient even for a deterministic response and here we are motivated by the works of Sacks and others [12].