Report: Experimental Algorithmics
by Catherine C. McGeoch

Methew Guda - A00381751
Nrimoni Chowdhury - A00371596

The two most important fundamental goals of research on algorithm is algorithm design and algorithm analysis. The main goal of algorithm analysis is to predict how well a given algorithm will perform in a given scenario under given conditions and assumptions. The goal of algorithm design is to build better algorithms that are faster and also return higher-quality solutions to a given problem. Algorithms that are both efficient and optimal are quite difficult to find in a large number of application areas. The quest for faster algorithms and the fundamental trade-off between “speed” and “quality” in applications important to industry, government, and science drives modern research in algorithm design. One of the important goal of algorithm analysis is to find an asymptotic upper bound (Big O) on algorithm performance using generic and abstract model, for example,Quicksort is $O(n^{2)}$ in the worst case. This kind of bound guarantees that no matter what programming language or platform is used, Quicksort will perform no more than cn^2 generic machine instructions (for some unspecified constant c) to sort a list of n numbers.

The abstract approach to design and analysis of an algorithm allows to discover universal truths about fundamental algorithmic properties. The goal of this approach what makes algorithms more and less efficient. Better algorithms become faster programs and produce higher-quality results when adopted by the software engineer. However, there are few difficulties found when engineers try to translate an asymptotic time bound into an accurate time prediction. One difficulty is the dependence on “worst case” assumptions. For example, Quicksort’s performance depends significantly on the input order of the n numbers to be sorted. On an input of 1,000 numbers, Quicksort might run hundreds of times faster than the worst-case prediction. In addition, the gap increases with n: on an input of one million numbers, Quicksort can run thousands of times faster than the worst-case prediction. The second difficulty arises when engineers try to translate “cn generic machine instructions” into microseconds. A real program encounters compiler optimization, pipelining, caching, and other platform-specific phenomena that dramatically reduce the predictive power of simple instruction counts.

Experimental methodology emphasizes highly controlled parameters, carefully placed datacollection probes, and sophisticated data analysis. The goal of the experiment is to generalize away from context specific measurements and build insight into fundamental structures and properties. This experimental approach can stimulate new forms of interplay between theory and practice, because experiments can be used to measure any aspect of algorithm performance. The term “experimental algorithmics” describes this new approach to algorithm design and analysis, combining theoretical questions and motivations with empirical research methods. The related term “algorithm engineering” is almost synonymous but emphasizes design over analysis.

Here is three example areas where experimental algorithmics has produced new results and significant progress in algorithm research. These examples also illustrate the variety of research questions that can be investigated through experimental methods.

Memory-efficient models of computation are a traditional theoretical approach to analyze counting basic operations performed on an abstract computer. These operations are general abstractions of CPU instructions. These instructions also have some unit costs. These instruction cost may vary in modern architectures. For example, a memory access may have several orders of magnitudes and more time consuming depending on where the operands are located. New abstract models are required to describe these new types of costs. Recently, the development of memory access models has been greatly stimulated and guided by experimental research.

One of the example of experimental research is, the development of new analytical models of computation to describe the cache performance of several sorting algorithms. There new analytical models have produced more accurate predictions of program running times.

One of the most popular sorting algorithm is quicksort that applies divide and conquer approach to place all the elements of an array in order. Quicksort sort all the elements of an array by recursively dividing the larger array into two smaller sub-arrays: one contains low elements and another contains high elements until the all the elements of the array is sorted. Quicksort’s cache performance depends on how the codes for partitioning is structured and how much time algorithm will take to execute to process small sublists. Even though the algorithm is well-designed, it will be less productive if the algorithm produce poor cache behaviour.

Researcher analyze memory access patterns by combining experiments and theory, develops efficient algorithms and programs for problem domains like search, dynamic tree and tire structures, matrix operations and permutations. These newly developed algorithms are “cache efficient”, which means these algorithms can be adjusted and exploited to its known cache properties. By exploiting these ideas researched have been able to develop sorting algorithm which are indeed very much faster.

Phase transitions in combinatorial problems, is a study of “Phase transition behaviors” in algorithms for NP-complete problems. An algorithm for NP-complete problem may run fast or slow, that depends on the properties of each inputs. But these properties are poorly comprehended, the gap between worst case and typical case for these NP-complete problems is immense and lastly, a reliable prediction of the performance of these algorithm is not possible to within years.

One of the wonderful example, the K-satisfiability problems serves to illustrate the larger question, where given a Boolean formula B in conjunctive normal form, containing n variables, m clauses, and k variables per clause. This type of problems is also know as Boolean Satisfiability Problems, which is to determine whether the variables of a given boolean formula B can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE. If it does evaluates to TRUE, then B is satisfiable, otherwise it is unsatisfiable.

Here is an example of boolean formula B with five variables, four clauses, and three variables per clause:

$$B = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_5) \wedge (x_3 \vee x_4 \vee x_5)$$

Since this boolean formula B has three variables per clause, it is a NP-complete problem. If the number of variables (k) per clause is equal or more than three, then it is a NP-complete problem and if k is less than three, then an efficient algorithm is known. In this case, B can be satisfied by setting x_1, x_3 to TRUE, and x_2, x_4, x_5 to FALSE.

K-satisfiability problems are extremely inefficient because till now, the best algorithm to determining K-satisfiability has to check all 2^n possible truth assignments. In that case, assuming one check per microsecond, checking a formula with only 50 variables would require about 435 years of computation.

Let’s consider generating the instances of the Boolean formula B_{nmk} uniformly at random, for the given parameters values n, m and k. Now Two closely related important questions are, what is the probability for B_{nmk} to be satisfiable? and What is the probability that a solution for B_{nmk} can be found efficiently? Let x be the ratio m/n. According to experiments performed by number of researchers, when each k greater than or equal to 2, there is a transition threshold value x_k such that the following property holds: when x is well below x_k, the probability that B_{nmk} is satisfiable is near 1;

when x is well above the threshold, the probability is near 0; and that probability makes a sharp change - a phase transition - when x is near x_k. This parameter x also can be used to predict that how difficult it will be to solve B_{nmk}.

Heuristic algorithms for satisfiability have more difficulty to solve instances that are near to the threshold value x_k then far from the threshold value. In fact, it is easy to show the satisfiability of the problem when x is low, because there are many more variables than clauses and therefore few conflicts among the variables; many satisfying truth assignments exist. Likewise, it is easy to show the unsatisfiability of the problem when x is large, because the clauses impose too many conflicts on the variables. It is astonishing, that the difficulty of solving a given random instance can be predicted so precisely based on just the ratio of clauses/variables.

Random satisfiability instances that are difficult to solve may be found in a localized region in the space of random instances near $m/n = x_k$. Locating the threshold values x_k for $k \geq 3$ and quantifying the difficulties of specific algorithms solving these parameterized instances are the most focused experimental efforts.

In 1991, through extensive experiments, it is found that this easy-hard-easy transitions can be possibly observed in a variety of NP-complete problems. A simple input parameter with threshold behavior can be found for each problem. This type of phase transition for random instances possibly be a property of all NP-complete problems.

The existence of a simple parameter like the ratio of m/n, a threshold value, and an easy-hard-easy transition in problem difficulty that depends on the parameter has been observed for many problems like Graph Coloring, Number Partitioning, and Traveling Salesman. This robust interaction between experimental and theoretical analysis has led to the development of new classification schemes for combinatorial problems, with deep implications for both theory and practice.

Algorithm engineering, which focuses the design aspect of algorithmic research. The aim algorithm engineering is to transform abstract algorithms into well-implemented programs with an emphasis on efficiency and generality. The DIMACS hosts an Implementation Challenge that stimulated much research in algorithm engineering by introducing a small set of problem areas and by inviting researchers to carry out design and analysis projects for algorithms for the given problems. Participants from the challenge adopt a common input format and common sets of input instances to test their implementations. Each Challenge involves cooperative research effort to find most efficient, robust and highly algorithms and implementations that takes a year-long of times. Participants also produce new analyses, data structures, and implementation strategies for these algorithms.

The Ninth Challenge, 2005–2006, participants presented practical and efficient algorithms for solving K-shortest paths, techniques for exploiting precomputation in memory-restricted applications, and efficient strategies for dynamic query-based problems. These types of algorithms are used in mapping and direction-finding systems for automobiles and other consumer applications. Comparing them revealed much new information about effective design strategies for the problems.

The Eighth Challenge, 2000–2001, participants are to evaluate a collection of algorithms for the Traveling Salesman problem. They have contributed their experimental results involving more than 150 implementations of state-of-the art algorithms. Direct comparisons of both time and tour quality was performed using a large collection of input instances through a companion Web site. The organizers presented a detailed discussion of how to select the best algorithms for solving real-world applications of the Traveling Salesman problem based on that experimental data. These applications are very useful for situations like delivery truck routing and fleet scheduling.

In conclusion, experimental analysis of algorithms is a progressive discipline. Together with a growing body of experimental results on algorithm performance, a collection of articles addressing methodological issues has also emerged. Since the experimental subject and the research questions are somewhat unusual compared to other problem domains much more work is needed to identify the statistical and data analysis tools most appropriate to these types of problems.