PROBABILISTIC COMPUTATIONS: TOWARD A UNIFIED MEASURE OF COMPLEXITY*

extended abstract

Andrew Chi-Chih Yao

Computer Science Department
Stanford University
Stanford, California 94305

1. Introduction

The study of expected running time of algorithms is an interesting subject from both a theoretical and a practical point of view. Basically there exist two approaches to this study. In the first approach (we shall call it the distributional approach), some "natural" distribution is assumed for the input of a problem, and one looks for fast algorithms under this assumption (see Knuth [8]). For example, in sorting n numbers, it is usually assumed that all n! initial orderings of the numbers are equally likely. A common criticism of this approach is that distributions vary a great deal in real life situations; furthermore, very often the true distribution of the input is simply not known. An alternative approach which attempts to overcome this shortcoming by allowing stochastic moves in the computation has recently been proposed. This is the randomized approach made popular by Rabin [10](also see Gill[3], Solovay and Strassen [13]), although the concept was familiar to statisticians (for example, see Luce and Raiffa [9]). Note that by allowing stochastic moves in an algorithm, the input is effectively being randomized. We shall refer to such an algorithm as a randomized algorithm.

These two approaches lead naturally to two different definitions of intrinsic complexity of a problem, which we term the distributional complexity and the randomized complexity, respectively. (Precise definitions and examples will be given in Sections 2 and 3.) To solidify the ideas, we look at familiar combinatorial problems that can be modeled by decision trees. In particular, we consider (a) the testing of an arbitrary graph property from an adjacency matrix (Section 2), and (b) partial order problems on n numbers, including sorting, selection, etc. (Section 3). We will show that for these two classes of problems, the two complexity measures always agree by virtue of a famous theorem, the Minimax Theorem of Von Neumann [14].

The connection between the two approaches lends itself to applications. With two different views (and in a sense complementary to each other) on the complexity of a problem, it is frequently easier to derive upper and lower bounds. For example, using adjacency matrix representation for a graph, it can be shown that no randomized algorithm can determine the existence of a perfect matching in less than $O(n^2)$ probes. Such lower bounds to the randomized approach were lacking previously. As another example of application, we can prove that for the partial order problems in (b), assuming uniform distribution (i.e., all n! permutations equally likely) always yields the greatest complexity.

We will also consider algorithms that allow errors (cf. Karp [6], Rabin[10]). Again useful connections (though not equality in this case) can be drawn between the distributional and the randomized complexities. For example, the $O(n^2)$ lower bound for perfect matching mentioned above still holds, even if we allow a randomized algorithm to make errors, say, 5 percent of the time.

Since discussions of low-order computational complexity have to be based upon precise models of algorithms, we shall treat (a) and (b) type problems seperately in Sections 2 and 3. Directions for further research are discussed in Section 4.

*This research was supported in part by NSF Grant No. MCS-7705313.

2. Testing Graph Properties

2.1 Notations and Definitions

Let $\mathcal{G}_n$ be the set of all undirected graphs on n vertices. Assume that P is a graph property on $\mathcal{G}_n$, that is, a mapping from $\mathcal{G}_n$ into {true, false}. Given any graph G in $\mathcal{G}_n$ represented by its adjacency matrix, we are interested in testing whether or not G has property P by successively probing entries of its adjacency matrix. In the standard model (see Rivest and Vuillemin [12]), an algorithm A for testing property P is represented as a binary decision tree, where branches correspond to whether a matrix entry tested is 0 or 1. To distinguish such conventional algorithms from the randomized algorithms to be described later, we call A a pure algorithm. For an input graph G of algorithm A, we use r(A,G) to denote the number of tests made by A. Let $\mathcal{A}$ be the family of pure algorithms that make no redundant tests. We are now ready to discuss the complexity concept in the distributional approach. Given a probability distribution d on $\mathcal{G}_n$, the average cost of a pure algorithm A is defined by

$$C(A,d) = \sum_{G\in\mathcal{G}_n} d(G)\cdot r(A,G) .$$

Under the input distribution d, the best that any pure algorithm can achieve is clearly $\min_{A\in\mathcal{A}} C(A,d)$. We now define the distributional complexity for testing P to be

$$F_1(P) = \sup_{d} \min_{A\in\mathcal{A}} C(A,d) .$$

This quantity $F_1(P)$ captures the following complexity notion:

> Provided that the distribution of inputs is known, $F_1(P)$ is the average cost we can always guarantee by finding a good algorithm..

We shall now explore the complexity concept from the randomized approach. A randomized algorithm R is specified by a probability distribution q on $\mathcal{A}$. We shall interpret R as an algorithm that has probability q(A) to proceed exactly as A. The expected cost of R for an input graph G is

$$E(R,G) = \sum_{A} q(A)\cdot r(A,G) .$$

The intrinsic cost of R is defined to be E(R,G) for the "worst" graph G, i.e., $\max_{G\in\mathcal{G}_n} E(R,G)$. A natural complexity measure of the problem from this viewpoint is thus the intrinsic cost of the best randomized algorithm. That is, we define the randomized complexity for testing P to be

$$F_2(P) = \inf_{R} \max_{G\in\mathcal{G}_n} E(R,G) .$$

The quantity $F_2(P)$ tells us how much time the best randomized algorithm takes for testing P.

A newly developed concept is to consider algorithms that allow a certain percentage of error in the final answer (see Karp [6], Rabin[10]). We shall now define the corresponding complexity measures for these classes of algorithms. Let us extend the family of pure algorithms $\mathcal{A}$ to a larger family $\mathcal{A}_0$, consisting of all decision trees that give a "yes" or "no" answer for any input graph $G\in\mathcal{G}_n$ (the answer need not be correct). For a decision tree $A\in\mathcal{A}_0$, and an input $G\in\mathcal{G}_n$, let $\epsilon(A,G)=0$ if A gives the correct answer for G, and $\epsilon(A,G)=1$ otherwise. As before, let r(A,G) be the number of tests made by A for input G.

Let λ be a number between 0 and 1. For any input distribution d, let $\mathcal{B}(\lambda)$ be the subset of $\mathcal{A}_0$ that consists of decision trees with error probability not exceeding λ under d. That is,

$$\mathcal{B}(\lambda) = \{A \mid A\in\mathcal{A}_0, \sum_{G\in\mathcal{G}_n} d(G)\epsilon(A,G) \leq \lambda\}.$$

The distributional complexity with error λ is defined as

$$F_{1,\lambda}(P) = \sup_{d} \min_{A\in\mathcal{B}(\lambda)} C(A,d)$$

where C(A,d) is as given previously.

We now consider the complexity with error from the randomized approach. A distribution q on the family $\mathcal{A}_0$ is said to be λ-tolerant if

$$\sup_{G\in\mathcal{G}_n} \sum_{A\in\mathcal{A}_0} q(A)\cdot\epsilon(A,G) \leq \lambda .$$

To insure that the error probability is bounded by λ for all input, we can only allow randomized algorithms whose characterizing distributions q are λ-tolerant. For a randomized algorithm R characterized by a λ-tolerant distribution q, the expected cost for input G is

$$E(R,G) = \sum_{A\in\mathcal{A}_0} q(A)\ r(A,G).$$

The randomized complexity with errorλ is then

$$F_{2,\lambda}(P) = \inf_{R} \max_{G\in\mathcal{G}_n} E(R,G)$$

where the infimum is taken over all randomized algorithms R with a λ-tolerant distribution. For $\lambda = 0$, $F_{1,\lambda}$ and $F_{2,\lambda}$ are reduced to F_1 and F_2, respectively.

2.2 Fundamental Theorems

The definitions of $F_1(P)$ and $F_2(P)$ naturally suggest methods for finding lower bounds to $F_1(P)$, and upper bounds to $F_2(P)$. In the first case, choose a specific distribution d, and a lower bound b for $F_1(P)$ can be obtained by showing that no pure algorithm can make less than b tests on the average under d. Similarly, upper bounds for $F_2(P)$ can be derived by analyzing the performance of specific randomized algorithms. These bounds may not be easy to obtain in practice, but at least we know what to attack. On the other hand, how can we prove lower bounds for $F_2(P)$ to know the limitation of the randomized approach? The following theorem thus is a useful bridge.

Theorem 1 $F_1(P) = F_2(P)$.

Theorem 1 can be proved by using the famous Minimax Theorem of Von Neumann [14]. It is aesthetically nice as the two complexity views are actually the same. It also provides practical means for proving lower bounds to $F_2(P)$ and upper bounds to $F_1(P)$. For example, limitations to the power of the randomized approach can now be established by considering "hard" input distributions. The following theorem suggests that distributions that have equal weights on isomorphic graphs in $\mathcal{G}_n$ are good candidates.

Theorem 2 Let d be any distribution on $\mathcal{G}_n$. Then there exists a distribution d_o such that

(i) $\min_{A\in\mathcal{A}} C(A,d) \le \min_{A\in\mathcal{A}} C(A,d_o)$, and

(ii) $d_o(G_1) = d_o(G_2)$ if G_1 and G_2 are isomorphic.

The exact evaluation of $F_i(P)$ can be shown to be a linear programming problem. For small n, $F_i(P)$ can be effectively computed, and Theorem 2 can be used to the advantage of reducing the number of variables considerably.

For algorithms with error tolerance, the following theorem links the complexities developed from the two approaches.

Theorem 3 $F_{2,\lambda}(P) \ge \frac{1}{2} F_{1,2\lambda}(P)$ for $0 \le \lambda \le \frac{1}{2}$.

2.3 Concrete Lower Bounds

In this subsection, we derive certain specific lower bounds for testing graph properties. By Theorems 1 and 3, it suffices to prove bounds only for $F_{1,\lambda}$. Occasionally, we find it possible to derive stronger results for $F_{2,\lambda}$ directly.

Definition We call P a normal graph property on $\mathcal{G}_n$ if P(empty graph) = false. For any graph S in $\mathcal{G}_n$, the size of S, denoted by $\|S\|$, is the number of edges in S. For a normal property P, a graph S is a minimal graph for P, if S is a smallest-sized graph satisfying P(S) = true. For any G in $\mathcal{G}_n$, let $\pi(G)$ denote the group of automorphisms for G. (An automorphism for G is a relabelling of the vertex set $\{1,2,\ldots,n\}$ that leaves G invariant.)

Theorem 4 Let P be a normal graph property on $\mathcal{G}_n$, S a minimal graph for P, and $0 \le \lambda \le 1/2$. Let $s = \|S\|$. Then

$$F_{i,\lambda}(P) \ge (\tfrac{1}{2} - \lambda)\, \frac{1}{s} \binom{n}{2} \qquad \text{for } i = 1, 2.$$

The spirit of Theorem 4 is similar to a result of Kirkpatrick [7, Theorem 2.3]. To give an application, let P be the property for a graph to be non-planar, then S is $K_{3,3}$, a 3 by 3 complete bipartite graph. Theorem 4 tells us that $\Omega(n^2)$ tests are required for any randomized algorithm, even allowing a 25% error. For more properties with a small minimal graph, see Kirkpatrick [7].

Theorem 4 can be generalized to bipartite graph properties. As in the worst-case complexity [7], this can then be used in conjunction with an embedding process to prove $\Omega(n^2)$ lower bounds for various "connectedness" properties. The following theorems can be proved by a non-trivial embedding process.

Theorem 5 Let T be any tree in $\mathcal{G}_n$, and P the property for a graph to contain T as a subtree. Then

$$F_{i,\lambda}(P) \ge \text{constant} \cdot n^2 \quad \text{for } 0 \le \lambda \le 0.1,\ i = 1, 2.$$

Theorem 6 Let $0 < k \le n$, and P the property of containing a clique of size k. Then

$$F_{i,\lambda}(P) \ge \text{constant} \cdot n^2 \text{ for } 0 \le \lambda \le 0.1,\ i = 1, 2.$$

Any graph property P has a linear lower bound $F_i(P) \geq \Omega(n)$. This can be seen as follows. Without loss of generality, we can assume P to be normal. Let S be a minimal graph for P of size s. It is easy to show that $F_i(P) \geq s$. When we combine this with Theorem 4, we have

$$F_i(P) \geq \frac{1}{4} \max \left\{s, \frac{n^2}{s}\right\},$$

which implies that $F_i(P) \geq \Omega(n)$. From the viewpoint of showing bounds stronger than $\Omega(n)$, properties with $s \approx \text{constant} \times n$ are the hardest to handle. The next theorem is often useful in such cases.

Theorem 7 Let S be a minimal graph for a normal graph property P on n vertices. Let $\|S\| = s$, then

$$F_i(P) \geq \frac{s}{s+1} \left(\frac{n!\ s!}{|\pi(S)|}\right)^{1/s}, \quad \text{for } i = 1,2.$$

As an illustration of its application, consider the property for a graph to be Hamiltonian. Clearly S is now a Hamiltonian circuit, and the group $\pi(S)$ is the cyclic group on $\{1,2,\ldots,n\}$. Thus, $s = n$ and $|\pi(S)| = n$. Therefore, for the Hamiltonian property P, we have

$$F_i(P) \geq \frac{n}{n+1} \left(\frac{n!\ n!}{n}\right)^{1/n} \approx \frac{n^2}{e^2} \quad \text{for } i = 1, 2.$$

Similar results can be derived for the property of containing a perfect matching. Although we have not succeeded in extending Theorem 7 to lower bounds for $F_{i,\lambda}$ (with $\lambda \neq 0$), for the special cases of being Hamiltonian and containing a perfect matching, the proofs can be modified to show the following theorem.

Theorem 8 Let P_1 and P_2 be the properties of being Hamiltonian, and containing a perfect matching, respectively, on $\mathcal{G}_n$. Then, for $\lambda \in [0, 0.1]$, we have

$$F_{2,\lambda}(P_i) \geq \text{constant} \times n^2 \quad \text{for } i = 1, 2.$$

3. Partial Order Problems

The complexity concept discussed in Section 2 can be extended to any problem with a finite set of possible inputs and algorithms. Theorem 1 and Theorem 3 remain valid, and Theorem 2 also has an analogue if there is a "symmetry" in the problem. (We shall see an example below.)

Perhaps the most extensively studied decision tree problems are those connected with partial orders (e.g. [8]). We shall concentrate on a special class that involve element selections. (See Fredman [4], for example, for other types of partial order problems.) Let the rank $\ell_V(x)$ of an element x in a linearly ordered set V be the number of elements less than or equal to x. For positive integers n and k with $n \geq k$, let $I_{n,k} = \{(i_1, i_2, \ldots, i_k) \mid \text{the } i_j\text{'s are distinct integers between 1 and } n\}$ be the set of all ordered k-tuples of distinct integers not exceeding n. A selection problem J on a set V of n elements is specified by a set $J \subseteq I_{n,k}$, and can be described as follows: Given a set V of n elements, we wish to find, by pairwise comparisons between the elements, k elements $(x_1, x_2, \ldots, x_k)$ such that $(\ell_V(x_1), \ell_V(x_2), \ldots, \ell_V(x_k))$ is in J. For example, the sorting problem corresponds to the choice $J = \{(1, 2, \ldots, n)\}$; and in the selection of the k smallest elements, we set J to be the set of all permutations of $(1, 2, \ldots, k)$.

For a selection problem J on a set $V = \{x_1, x_2, \ldots, x_n\}$, an algorithm A is a decision tree where each internal node performs a comparison of the form "$x_i:x_j$". At each leaf of the tree, an output $(x_{i1}, x_{i2}, \ldots, x_{ik})$ is specified. Let us use $F_i(J)$ and $F_{i,\lambda}(J)$ to denote the complexity of the selection problem J as was done in Section 2. Theorem 1 is still true, so we have $F_1(J) = F_2(J)$. There are two interesting results on the complexity of selection problems that we shall now discuss.

Firstly, there is a nice symmetry for any selection problem. Intuitively, if we take any algorithm A, and relabel the elements of V in any way, A will be transformed accordingly into another valid algorithm. This observation can be used to derive the following result parallel to Theorem 2. Let us still use $\mathcal{A}$ for the set of algorithms, and C(A,d) for the average number of comparisons made by algorithm A under input distribution d.

Theorem 9 Let J be a selection problem on a set V, and d_u be the uniform distribution on the input, i.e., every linear ordering on V is equally likely. Then

$$\min_{A \in \mathcal{A}} C(A,d_u) = F_i(J) \quad \text{for } i = 1, 2.$$

Informally, Theorem 9 tells us that the uniform distribution is the "hardest" distribution, and we can find bounds to $F_i(J)$ by studying optimal algorithms under d_u. There are many studies concerning the quantity $\min_A C(A,d_u)$ for various problems [2,8,11,15]. Some of these results involve rather difficult proofs. These results now acquire a new significance through Theorem 9. For example, it was shown by Floyd and Rivest [2] that for the problem of finding the median of n numbers, $1.375\, n \leq \min_{A \in \mathcal{A}} C(A,d_u) \leq 1.5\, n$. In the new interpretation, we can state that, (1) for an arbitrary distribution d, there always exists a (pure) algorithm that makes less than 1.5 n comparisons on the average; and (2) any randomized algorithm must make at least an expected 1.375 n comparisons for some input.

Now, for the second point of interest. So far in our models we have not seen any randomized algorithm that is order-of-magnitude better than the worst-case complexity of conventional (pure) algorithms solving the same problem. We shall now demonstrate the superiority (in the order-of-magnitude sense) of a randomized algorithm allowing a small error in a selection problem. Such an example comes from the selection of a mediocre element (see F. Yao[16]), which is defined, for our present purpose, to be an element with a rank in the middle third. That is, for any input set $V = \{x_1, x_2, \ldots, x_n\}$, we wish to find an element x_j such that $n/3 \leq \ell_V(x_j) \leq 2n/3$. For any fixed $0 < \lambda < 1$, it is not difficult to construct a randomized algorithm with error probability bounded by λ that uses only constant $\cdot \log(1/\lambda)$ comparisons. Since any pure algorithm has to make $2n/3$ comparisons in the worst case, this provides an example for our assertion. However, we do not have an analogous example where an errorless randomized algorithm performs much better than the worst-case complexity.

4. Concluding Remarks

We have developed a unified average complexity concept in some decision tree models, and demonstrated its usefulness. Many of these results have generalizations. For example, Theorem 4 can be extended to any string property whose invariant group is transitive. We list below some important unanswered questions.

(1) Despite the successful applications of randomization to several problems [1,11,13], there is little proof that the randomized approach really lowers the computation time by order of magnitude (compared to the worst-case complexity of pure algorithms). It seems that the only concrete demonstration that exists in the literature, apart from our example in Section 3, is a result of Frevald cited in Gill's paper [3, revised version]. It states that the language $\{ww \mid w \in \{0,1\}^*\}$ can be recognized by a 1-tape Turing machine with error $\lambda > 0$ in $O(n \log^2 n)$ steps (as compared with $c \cdot n^2$ for a conventional 1-tape Turing machine). There does not seem to be any similar example for an errorless randomized algorithm. Question: In our models, is there such an example?

(2) Is it true that $F_i(P) \geq c \cdot n^2$ for all monotone graph properties? The corresponding question for the worst-case complexity was answered by Rivest and Vuillemin [12]. It is in general desirable to develop more lower bound techniques for $F_i(P)$.

(3) With regard to Theorem 3, can one show a similar inequality in the opposite direction?

(4) Can one demonstrate non-linear lower bounds for 1-tape probabilistic Turing machines with error in concrete examples? A good candidate is the recognition of the language $L = \{wcw^R \mid w \in \{0,1\}^*\}$ (cf. [5] Theorem 10.7).

(5) The subject studied here shares certain common grounds with Game theory and statistical decision problems. It would be interesting to know of further connections.

References

[1] J. Carter and M. Wegman, "Universal Classes of Hash Functions", Proc. 9th Annual ACM Symposium on Theory of Computing (1977), 106 - 112.

[2] R. Floyd and R. Rivest, "Expected Time Bounds for Selection", CACM 18 (1975), 165-172.

[3] J. Gill, "Computational Complexity of Probabilistic Turing Machines", Proc. 6th Annual ACM Symposium on Theory of Computing (1974), 91-95, revised Feb. 1977.

[4] M. Fredman, "How Good Is the Information Theory Bound?", Theoretical Computer Science 1 (1976), 355-361.

[5] J. Hopcroft and J. Ullman, Formal Langrages and their Relation to Automata, Addison-Wesley(1969).

[6] R. Karp, "Probabilistic Analysis of Combinatorial Search", in Algorithms and Complexity, edited by J. Traub, Academic Press (1976), 1-20.

[7] Kirkpatrick, "Topics in the Complexity of Combinatorial Algorithms", University of Toronto Technical Report no. 74(1974).

[8] D. Knuth, The Art of Computer Programming vol 3, Addison-Wesley (1973).

[9] R. Luce and H. Raiffa, Games and Decisions, Wiley (1957).

[10] M. Rabin, "Probabilistic Algorithms", in Algorithms and Complexity, edited by J. Traub, Academic Press (1976), 21-40.

[11] I. Pohl, "Minimean Optimality in Sorting Algorithms" Proc. 16th Annual Symposium on Foundations of Computer Science (1975), 71-74.

[12] R. Rivest and J. Vuillemin, "On Recognizing Graph Properties From Adjacency Matrices", Theoretical Computer Science 3 (1976), 371-384.

[13] R. Solovay and V. Strassen, "Fast Monte-Carlo Test for Primality", SIAM J. on Computing 6(1977), 84-85.

[14] Von Neumann, "Zur Theorie der Gesellschaftsspiele", Math Annalen 100(1928), 295-320.

[15] A. Yao and F. Yao, "On the Average-Case Complexity of Selecting the t-th Best", to appear.

[16] F. Yao, "On Lower Bounds for Selection Problems", MIT Project Report TR-121 (1974).