Essay 9

On Probabilistic Machines, Bounded Rationality and Average-Case Complexity

Nimrod Megiddo[1]

ABSTRACT. The analogs of pure, mixed and behavior strategies in the context of algorithms are studied. It is shown that probabilistic machines are more powerful than probability distributions over deterministic ones, that best response may sometimes requires randomization, and that if nature's choices are computable then there exists a deterministic best response.

1. Introduction

In traditional game theory, deterministic strategies are called pure, and probability distributions over pure strategies are called mixed strategies. A strategy which makes probabilistic choices at every decision point is called a behavior strategy. It is well known that a mixed strategy is at least as powerful as a behavior strategy and the two are of equivalent power if the game has perfect recall [1]. Also, a best response to the opponent's mixed strategy can be played without randomization.

In recent years, there has been a growing literature on playing games through computing machines (see, for example, [2]), attempting to understand bounded rationality. On the other hand, complexity theory sometimes uses ideas from game theory (see, for example, [3]).

With some reservations, algorithms can be viewed as strategies, so concepts similar to mixed and behavior strategies can be defined with respect to algorithms. We show here that algorithms exhibit phenomena which are quite different from the ones we know from game theory. As usual in the theory of complexity, an algorithm is a procedure for solving a family of problem instances rather than a single one.

[1]IBM Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099, and School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel.

2. Models and results

Let $\mathcal{M}$ denote the set of all *deterministic* Turing machines M such that, given any natural number n as input, the machine M computes an n-bit number $M(n)$. Let $\mathcal{P}$ denote the analogous set of *probabilistic* Turing machines.[2]

The basic game

The definition of the basic game considered here is inspired by the concept of worst-case complexity of a computational problem. Player I (the player) designs a deterministic algorithm for the problem, and player II (the adversary) picks inputs for this algorithm for each input size. Thus, the player chooses a deterministic machine $M \in \mathcal{M}$, and the adversary chooses a function A which assigns to every natural n, an n-bit number $A(n)$. The adversary then pays the player an amount $C_n = C_n(M, A) = 2^n$, if $M(n) = A(n)$, and $C_n(M, A) = 0$ otherwise.

It is trivial to see that the player cannot secure for himself any positive payoff for any input size, since the adversary may choose $A(n) \neq M(n)$ for every n.

Mixed strategies

A mixed strategy is essentially a probability distribution over the deterministic strategies. If mixed strategies are allowed, then this game is played as follows. The player chooses any probability distribution $\pi : \mathcal{M} \to R$, i.e., $\pi(M) \geq 0$ and $\sum_{M \in \mathcal{M}} \pi(M) = 1$. Suppose the adversary now chooses his function $A(n)$, knowing this probability distribution. The expected payoff is evaluated with respect to π, i.e.,

$$E_n = E_n(\pi, A) = \mathcal{E}\, C_n(M, A) = \sum_{M \in \mathcal{M}} \pi(M) C_n(M, A) \ .$$

We note that since the adversary knows π, he cannot benefit from randomizing his choice of the mapping A, i.e., for every n the adversary has an optimal choice of a number $A(n)$, namely, one with minimum probability.

Ideally, the player would like to choose a probability distribution over $\mathcal{M}$ which would induce, for every n, a uniform distribution over the n-bit numbers. Given such a probability distribution, the adversary would be

[2] A probabilistic Turing machine is one that can also flip a coin and read its outcome.

indifferent, and the value of the game would be 1 for every n. However, such a probability distribution does not exist! The proof of the latter claim is as follows. Let π be any probability distribution over $\mathcal{M}$. Since $\mathcal{M}$ is countable, there exists an $M \in \mathcal{M}$ such that $\epsilon \equiv \pi(M) > 0$. It follows that for every n such that $2^{-n} < \epsilon$, the probability that the number $M(n)$ will be chosen is greater than 2^{-n}, hence there exists another n-bit number k_n which is chosen with probability less than 2^{-n}. Since the adversary knows the distribution π, he can find such a number k_n and choose $A(n) = k_n$. Thus, for every such n, the value of the game is less than 1.

In fact, we can prove a much stronger result. Let δ be any positive number. For any probability distribution π over $\mathcal{M}$, there exists a finite number K and machines $M_1, \ldots, M_K \in \mathcal{M}$ such that

$$\sum_{i=1}^{K} \pi(M_i) > 1 - \delta \ .$$

Let n be such that $2^n > K$. It follows that there exists an n-bit number k_n whose probability is less than $\delta/(2^n - K)$. If the adversary chooses $A(n) = k_n$ then the limsup of the expected payoff, as n tends to infinity is no more than δ. Since δ can be any positive number, the limit is indeed zero!

Note that the actual sequence of choices of numbers computed by the sampled deterministic machine is of course computable. The adversary can choose a noncomputable sequence $A(n)$. On the other hand, we have not assumed that the probability distribution itself is computable. If we assume the probability distribution is computable, and the adversary is given a program that computes $\pi(M)$ for every M, then we can also restrict the adversary to choose computable sequences $A(n)$; the expected payoff still tends to zero since in this case there exists a program which identifies an appropriate k_n for every n.

It is interesting to consider the rate of convergence of the value to zero. We now give an example. Given a natural number n, let $m = 2^n$. For simplicity (and with no loss of generality), let's change notation so that the object of the game is: given m, produce a number from the set $S_m = \{1, 2, \ldots, m\}$. We first construct a sequence of machines $M_1, M_2, \ldots$ as follows. The machine M_j computes a number $k \in S_m$ such that $j \equiv k \bmod m$. Let M_j be chosen with probability

$$p_j = \frac{j^{-c}}{\zeta(c)} \ ,$$

where $c > 1$ is a constant and $\zeta(c) = \sum_{i=1}^{\infty} i^{-c}$. It follows that, given m, a number $k \in S_m$ is chosen with probability

$$P_m(k) = \sum_{i=0}^{\infty} p_{k+im} \ .$$

The one with the smallest probability in S_m is m itself:

$$P_m(m) = \sum_{i=0}^{\infty} p_{m(i+1)} = m^{-c} \ .$$

The payoff is therefore, m^{1-c}, which tends to zero as m tends to infinity. By choosing c sufficiently small we can guarantee convergence at a polynomial rate with any degree. Of course, slower convergence can be achieved by choosing the $\pi(M_i)$ converging more slowly than any series of the form i^{-c}.

Behavior strategies

A behavior strategy is essentially a strategy that allows probabilistic decisions at any decision point. This concept corresponds to the notion of a probabilistic Turing machine in our game. Thus, if behavior strategies are allowed, then the game is played as follows. The player chooses a probabilistic machine $\boldsymbol{M} \in \mathcal{P}$ and then the adversary chooses a function A as above. The payoff is now a random variable $\boldsymbol{C}_n = \boldsymbol{C}_n(\boldsymbol{M}, A)$ whose value depends on the coin tosses of $\boldsymbol{M}$, so its expected payoff is evaluated with respect to this distribution.[3]

It is obvious that we have an optimal probabilistic Turing machine, namely, one which for every n samples an n-bit number simply by writing down n random bits. The value of the game is 1 for every n.

Distributions of inputs

We also consider games inspired by the concept of average-case complexity. First, suppose nature moves first and selects for every n, any probability distribution over the n-bit numbers. The player is then informed of these selections and then chooses either $M \in \mathcal{M}$ or $\boldsymbol{M} \in \mathcal{P}$, depending on the version being played. Next, for every n a number $A(n)$ is sampled from the probability distribution selected by nature, and the chosen machine produces a number, so payoffs can be determined as above.

If the player is allowed to choose a probabilistic machine, then the one discussed above guarantees the value of 1 for every n, regardless of nature's choice of distribution. However, if the player has to choose a deterministic machine then the value may be less than 1. For example, even if nature's choices are deterministic, but the sequence $A(n)$ is not computable, then the player will get 0 for infinitely many values of n.

[3] Mixtures of behavior strategies can also be defined.

Computable distributions of inputs

We now claim that if nature's choice of probability distribution is computable, then the player has an advantage. The variant we consider is as follows. Nature moves first and selects a Turing machine[4] which computes, for every n, a probability distribution over the n-bit numbers. The player is given the full description of the machine selected by nature and the game proceeds as above.

In this special case the player can design a deterministic machine M that analyzes nature's choices for any value of n and computes the best response to that probability distribution, namely, a number $M(n)$ of maximum probability. This yields an expected payoff of at least 1 for every n. Of course, nature can force the expected payoff to be not greater than 1.

3. Summary

To summarize, we have established the following facts, which are quite different from what traditional game theory suggests:

1. Probabilistic machines are more powerful than probability distributions over deterministic ones.
2. Randomization may be necessary in order to respond optimally to a known probability distribution chosen by nature.
3. If nature's choice of probability distribution is computable, then there exists a deterministic best response.

Acknowledgments: Thanks to Nathan Linial and other attendees of the IMSSS Summer Workshop on Bounded Rationality, for stimulating discussions on the material in the paper.

4. References

[1] H. W. Kuhn, "Extensive games and the problem of information," in: *Contributions to the Theory of Games I*, H. W. Kuhn and A. W. Tucker, eds., Annals of Mathematics Studies No. 28, Princeton University Press, Princeton, New Jersey, 1950.

[4] Here it does not matter whether this machine is deterministic or probabilistic.

[2] N. Megiddo and A. Wigderson, "On play by means of computing machines," in: *Theoretical Aspects of Reasoning about Knowledge*, J. Y. Halpern, ed., Morgan Kaufmann Publishers, Inc., Los Altos, California, 1986, pp. 259–274.

[3] A. C. Yao, "Probabilistic computations: toward a unified measure of complexity," in: Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science (1977), IEEE Computer Society Press, Long Beach, CA, 1977, pp. 222–227.