26. Derandomization

Given a randomized algorithm $\mathcal{A}$, a natural approach towards *derandomizing* it is to find a method for searching the associated sample space Ω for a good point ω with respect to a given input instance x; a point is good for x if $\mathcal{A}(x,\omega) = f(x)$. Given such a point ω, the algorithm $\mathcal{A}(x,\omega)$ becomes a deterministic algorithm and it is guaranteed to find the correct solution. The problem faced in searching the sample space is that it is usually exponential in size.

Two general methods for searching the sample space have emerged in recent years. One of them – the *method of conditional probabilities* – starts with a "trivial" (in most situations, uniform) sample space Ω whose size is usually exponential, and the idea is to make some non-trivial search procedure within it. The other method – the *method of small sample spaces* – first tries to design some "small" sample space Ω' (say, of polynomial size), and then performs an exhaustive search of it. In this chapter we will shortly discuss both these approaches.

26.1 The method of conditional probabilities

The aim of this method is to convert probabilistic proofs of *existence* of combinatorial structures into efficient deterministic algorithms for their actual *construction*. The idea is to perform a binary search of the sample space Ω for a good point. At each step, the current sample space is split into two equal halves and the *conditional probability* of obtaining a good point is computed for each half. The search is then restricted to the half where the conditional probability is higher. The search terminates when only one sample point (which must be good) remains. This method is applicable to large sample spaces Ω since it requires only $\log_2 |\Omega|$ steps. In situations, where the corresponding conditional probabilities can be effectively computed (or at least approximated) this approach works pretty well. To explain the idea, let us consider the following problem.

Given a 3-CNF formula $F(x_1, \ldots, x_n)$, we want to find an assignment of values 0 or 1 to $x_1, \ldots, x_n$ satisfying as many clauses as possible. If we assign to each variable the value 0 or 1 at random independently and with equal probability, then we may expect that at least $7/8$ fraction of clauses will be

satisfied, just because each clause is satisfied with probability $1 - 2^{-3} = 7/8$ (see Proposition 26.2).

But where is the assignment? The argument above guarantees only the *existence* of such an assignment and gives no idea about how to *find* it. An exhaustive search will always lead us to the desired assignment. But this dummy strategy will require exponential (in n) number of steps. Can we do better? It appears that we can "derandomize" the probabilistic *proof* of existence so that it leads to a *deterministic* algorithm which is only polynomial in the length of the input formula.

Before we turn to a formal description of the method, let us first try to solve our special problem with the help of a chimpanzee (this beautiful explanation is due to Maurice Cochand).

We build a binary tree whose 2^n leaves correspond to the 2^n possible assignments. Leaves are close to the sky, as they should be. Going up to the left branch at level i corresponds to choose the value 0 for x_i, going up to the right gives x_i the value 1.

In oder to motivate the chimpanzee for this fascinating problem, we attach at every leaf of the tree a black box containing a number of bananas equal to the number of clauses satisfied by the assignment corresponding to that leaf. We do then invite the chimpanzee to go up in oder to bring down one of the black boxes, making him most clear the potential benefit of the operation.

We repeat this experiment many times, with many different trees corresponding to as many formulas F, having different number of variables and clauses. The chimpanzee *newer* looked at the list of clauses (although he was allowed to do it), did not even care about the number of variables. He moved up quickly along the tree, and *always* brought back a box having a number of bananas at least equal to $7/8$ times the number of clauses!

We asked him for his secret (he definitely had one, this was more than luck!). For a number of bananas we do not dare to mentioned here, he gave the following answer:

"Embracingly simple," he said. "At every junction I do the same: because of the weight, the branch supporting the subtree having the biggest number of bananas is not as steep as the other one, there I go!"

26.1.1 A general frame

Suppose we have a sample space Ω, and assume, for simplicity, that it is symmetric (i.e., each point has probability $2^{-|\Omega|}$) and that $\Omega = \{0,1\}^n$. Let $A_1, \ldots, A_m$ be a collection of events, and consider the random variable $X = X_1 + \cdots + X_m$ where X_i is the indicator random variable for A_i. Hence, $\mathrm{E}[X] = \sum_{i=1}^{m} \mathrm{Prob}(A_i)$. Also suppose that we have a *proof* that $\mathrm{E}[X] \geqslant k$. So, there is a point $(\epsilon_1, \ldots, \epsilon_n)$ in the sample space in which at least k of the events hold. Our objective is to find such a point *deterministically*.

Introduce n random variables $Y_1, \ldots, Y_n$ where each Y_i takes value 0 or 1 independently with equal probability. We find the bits $\epsilon_1, \epsilon_2, \ldots$ sequentially