Chapter 1

Introduction

One of the fundamental questions in theoretical computer science is to fully understand why some functions are easy to compute, while others require considerable large computing resources. Unfortunately, the present state of the art falls far away from giving an answer to this question — there are very few functions for which we know exact complexity measures (i.e., matching upper and lower bounds).

The study of complexity of algorithms began far before the invention of the computer. For many centuries the notion of an algorithm has been used by mathematicians in the idea of a constructive proof. The first known reference to the study of an algorithm is Euclid’s work on the study of the greatest common divisor (GCD). This was done in approximately 300 B.C. Of course, Euclid’s motivation was to reduce the amount of pencil-and-paper work in order to compute the GCD of two numbers. Interestingly, this algorithm has survived to the present day: approximately 2300 years later Euclid’s algorithm and its variants still are the most efficient ones and in fact they are widely implemented in many computer programs. Knuth [29] calls Euclid’s GCD algorithm the “granddaddy of all algorithms”. The question whether this algorithm and its variants are the best possible stays as an important open problem. For a summary we refer the reader to [29].

The area of complexity theory is concerned with the fundamental problem: what makes some problems difficult to compute and what makes them easy. In pursuing this question, many important concepts have been introduced in theoretical computer science; we mention the concept of NP-complete problems, and circuit complexity. In general, many persons like to view the study of algorithms as providing upper bounds (in form of an efficient algorithm) and complexity theory as producing lower bounds (proof that certain functions actually are very difficult to compute). Of course, there is a lot of overlapping between these two areas such that distinction is not always clear.

This dissertation looks at problems in complexity theory; specifically, we
are interested in finding explicit Boolean functions which are provably hard
to compute in some reasonable model of computation. Although functions re-
quiring exponential resources are known to be plentiful by elementary counting
arguments [55], the problem of actually proving such lower bounds for explicit
functions, say functions computable in NP, is much more difficult. For a func-
tion in NP, the best known lower bound on circuit size are only linear [2], the
best known lower bounds on circuit depth are only logarithmic.

The two types of restrictions usually considered in complexity theory are
monotone circuits and bounded depth circuits. In the first part of this thesis
(Chapter 5) we concentrate on the depth restriction: we give (exponential) lower
bounds on threshold circuits for some concrete Boolean functions.

One natural motivation to study threshold circuits comes from the fact that
they are very closely related to neural nets (the basic element of a neural net
is very similar to a threshold gate). As the term “neural net” implies, it was
originally aimed towards modeling networks of real neurons in the human brain.
However, the models are extremely simplified when seen from a neurophysiolog-
ical point of view, though they are believed to be still of value to obtain insight
into the principles of “biological computing” (see [19]).

To be more precise, we give exponential lower bounds on threshold circuits
of depth 2 with bounded weights on the bottom layers. We look at very fun-
damental functions such as Boolean predicates of the discrete multiplication
function, the discrete division function, the discrete exponentiation function,
and the Diffie-Hellman function. The reasons for examining such functions are
manifold. First, the problems examined are of very fundamental structure, and sometimes
they are used repeatedly in more complex calculations. Some of the considered
functions are widely used in encryption schemes. In some cases, e.g. when
embedded in hardware, every single gate may be expensive to build and an
efficient implementation of such basic functions becomes very important. Our
lower bounds give an answer to the question whether the implementation may
be done with less hardware meaning lower cost and also more efficiency. Thus,
it becomes very important to give more insight in understanding the exact com-
plexity of such fundamental problems. Some of the examples for which we pro-
vide lower bounds are known to be computable with one more threshold layer
with polynomially many gates. So our given bounds are very close to the best
possible. Secondly, the analysis of easy functions adds important new tools to
the techniques available to provide lower bounds for more complex functions.
Third, some of the presented functions are actually used as the decryption func-
tion in various cryptographic protocols. Giving exponential lower bounds on the
number of threshold circuits can also be viewed as a weak impossibility result
to compute the function (to brake the cryptographic protocol) when the com-
putational model is restricted (in our case it is restricted to threshold gates of
depth 2). Thus presenting lower bounds on (very restricted) computational de-
vices may put a little more confidence in some widely believed assumptions that such functions are computationally hard. For example, one of such assumptions is the CDH assumption which states that the Diffie-Hellman function is hard to compute. Our results support this assumption in the sense that a Boolean predicate of the Diffie-Hellman functions is hard to compute when we (severely) restrict the model of computation. To put it in different words, we show that a certain kind of attack (namely an attack based on threshold circuits of depth 2) on the Diffie-Hellman function is doomed to fail since the number of necessary gates for this attack is too large. We want to stress that impossibility results of this kind should not be overestimated since the restricted model (based on threshold circuits of depth 2) seems to be far too weak to contain any practical cryptographic attack. However, since impossibility results (even in restricted models) are very rare, we think that they still are of some value.

In the second part of this thesis (Chapter 6) we look at interpolation results for (cryptographic) functions defined over a finite field. We give lower bounds on the degree and on the number of non-zero coefficients of such interpolation polynomials. Functions of interest will be functions related to the Diffie-Hellman function. Motivation of examining these function is given by the observation that whenever there is a simple (e.g. low degree) polynomial that coincides with such a functions on a sufficient large number of its inputs, then there exists an algorithm that computes the Diffie-Hellman function (and thus contradicting the DDH assumption).

Though the results seem to be similar in the two parts of this thesis, the methods are quite different. In the first part we pursue the strategy of giving lower bounds with algebraic and number theoretic methods. Specifically, we use the concept of matrices and matrix norms together with bounds on exponential sums. In the second part, the methods are more of combinatorial nature and results are obtained by counting arguments.

It is assumed that the reader is familiar with the basic notation and terminology used in complexity theory (such as the classes \( \text{NP} \)). Definitions of this terms can be looked up in widely used textbooks of theoretical computer science [29, 46].