BABEŞ-BOLYAI UNIVERSITY
FACULTY OF MATEMATICS AND COMPUTER SCIENCE

Solving NP complete problems using light based devices

Student: Oana Melinda Muntean
Advisor: Lect. dr. Mihai Oltean

June, 2007

Contents

List of Figures

Abstract

There are important problems such as the Hamiltonian path problem, the subset sum problem, the exact cover problem, solving diophantine equations problem that have been studied for decades and for which all known algorithms have a running time that is exponential in the length of the input. These four problems and many other problems belong to the set of NP-complete problems. Many NP-complete problems have been identified, and it seems that such problems are very difficult and probably exponential. If so, solutions are still needed, and in this paper we consider an approach to these problems based on a light based device.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Chapter 1

Introduction

It is widely accepted that there are certain problems that are easy to solve and there are certain problems that are hard to solve. These problems classes are loosely based on the ability of modern computers to provide solutions in a given time and space. We are also faced with limitations in modern computation on a physical level. Our current dominant paradigm, the universal Turing machine and the Von Neumann architecture, has allowed us to build faster and more powerful computers. It is becoming quickly obvious, however, that there are physical and theoretical limitations inherent in our current paradigm. Is our current view computing ideal for solving 'hard' problems? Are problem classes, as we understand them today, universal across other models of computation? In this paper, we will propose and investigate an alternative, unconventional model of computation to analyze these issues, namely optic computation.

Chapter 2

NP Completeness

The subject of computational complexity theory is dedicated to classifying problems by how hard they are. There are many different classifications; some of the most common and useful are the following:

1. **P** problems that can be solved in polynomial time,
2. **NP** problems - this stands for *nondeterministic polynomial time*. A problem is in NP if you can quickly (in polynomial time) test whether a solution is correct (without worrying about how hard it might be to find the solution).

The theory of NP-completeness is a solution to the practical problem of applying complexity theory to individual problems. NP-complete problems are defined in a precise sense as the hardest problems in P. Even though we don't know whether there is any problem in NP that is not in P, we can point to an NP-complete problem and say that if there are any hard problems in NP, that problems is one of the hard ones (conversely if everything in NP is easy, those problems are easy).

In this chapter the most important issues concerning the complexity theory, namely NP-completeness will be presented and analyzed. Before going into more details, some useful basic definitions are recall (Section 2.1). In Section 2.2 there are given the definitions of the main classes of problems with regard to NP-completeness theory and also the differences between these clases. Section 2.3 introduces some more definitions from complexity theory. In Section 2.4 it is discussed the most important question concerning NP-complete problems: is $P = NP$? In Section 2.5 it is brefly presented the theorem that represent the starting point of whole theory of NP-completness. In Section 2.6 some methods used to prove the NP-completness of a certain problem are given.

2.1 Problems, Algorithms and Complexity

A problem is a general question to be answered, usually possesing several *parameters*, or free variables, whose values are left unspecified. A problem is described by giving:

1. a general description of all its parameters,
2. a statement of what properties the answer, or textitsolution is required to satisfy.

An *instance* of a problem is obtained by specifying particular values for all the problem parameters.

Algorithms are general, step by step procedures for solving problems. An algorithm is said to solve a problem Π if that algorithm can be applied to any instance I of the problem.

In general we are intrested in finding the most efficient algorithm for solving a problem. In its broadest sense, the notion of efficiency involves all the various computing resources needed for executing an algorithm. However, by the most efficient algorithm one normally means the fastest. The time requirements of an algorithm are conveniently expressed in terms of a single variable, the *size* of a problem instance, which is intended to reflect the amount of input data needed to describe the instance. The *time complexity function* for an algorithm expresses its time requirements by giving, for each possible input length, the largest amount of time needed by the algorithm to solve a problem instance of that size. Different algorithms possess a wide variety of different time complexity functions, and the caracterization of which of these are efficiently enough and which are too inefficient will always depend on the situation. However there is a simple distinction that offers considerably insight into these matters. This is the distinction between *polynomial time algorithms* and *exponential time algorithms*. Let us say that a function $f(n)$ is $O(g(n))$ whenever there exist a constant c such that

$$|f(n)| \leq c \cdot |g(n)|$$

for all values of $n \geq 0$. A *polynomial time algorithm* is defined to be one whose time complexity function is $O(p(n))$ for some polynomial function p, where n is used to denote the input length.Any algorithm whose time complexity function cannot be so bounded is called an *exponential time algorithm* (although it should be noted that this definition includes certain non-polynomial time complexity functions like $n^{\log n}$, which are not normally regarded as exponential functions).

The distinction between these two types of algorithms has particular significance when considering the solution of large problem instances. Table 2.1 illustrates the difference in growth rates among several typical complexity functions of each type, where the functions express execution time in terms of microseconds. Notice the much more explosive growth rates for the two exponential complexity functions.

TCF/Size	20	30	40	50	60
n	.00002 sec	.00003 sec	.00004 sec	.00005 sec	.00006 sec
n^2	.00004 sec	.0009 sec	.0016 sec	.0025 sec	.0036 sec
n^3	.008 sec	.027 sec	.064 sec	.125 sec	.216 sec
n^5	3.2 sec	24.3 sec	1.7 min	5.2 min	13.0 min
2^n	1.0 sec	17.9 min	12.7 days	35.7 years	366 centuries
3^n	58 min	6.5 years	3855 centuries	$2*10^8$centuries	$1.3*10^{13}$centuries

Table 2.1: Comparison of several polynomial and exponential time complexity (TCF stands for *Time complexity function*)

Most exponential time algorithms are merely variations of exhaustive search, whereas polynomial tume algorithms generally are made possible only through the gain of some deeper insight into the structure of a problem. There is a wide agreement that a problem has not been well solved until a polynomial time algorithm is known for it. Hence we shall refer to a problem as *intractable* if it is so hard that no polynomial time algorithm can possibly solve it. This view is central to the theory of *NP-completness*.

2.2 P, NP and co-NP

Let define three classes of problems:

1. P is the set of yes/no problems that can be solved in polynomial time. Intuitively, P is the set of problems that can be solved quickly.

2. NP is the set of yes/no problems with the following property: If the answer is yes, then there is a proof of this fact that can be checked in polynomial time. Intuitively, NP is the set of problems where we can verify a YES answer quickly if we have the solution in front of us.

3. co-NP is the exact opposite of NP. If the answer to a problem in co-NP is no, then there is a proof of this fact that can be checked in polynomial time.

The relationship between the three classes can be seen in Figure 2.2.

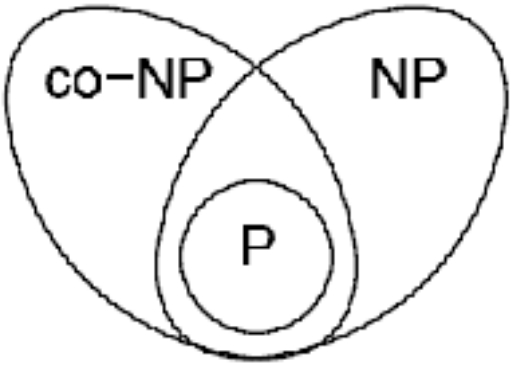

Figure 2.1: The relationship between P, NP and co-NP classes.

If a problem is in P, then it is also in NP to verify that the answer is yes in polynomial time, we can just throw away the proof and recompute the answer from scratch. Similarly, any problem in P is also in co-NP.

One of the most important open questions in theoretical computer science is whether or not $P = NP$. Nobody knows. Intuitively, it should be obvious that $P \neq NP$ (see [9],[6]); but nobody can prove it.

Notice that the definition of NP (and co-NP) is not symmetric. Just because we can verify every yes answer quickly, we may not be able to check no answers quickly, and vice versa. For example, as far as it is known, there is no short proof that a boolean circuit is not satisfiable. But again, there is no proof; everyone believes that $NP \neq co - NP$, but nobody really knows.

2.3 NP-Complete Problems

Before giving the definition of a NP-complete problem it is necessary to introduce the the notion of NP-hard problem.

A problem Π is NP-hard if a polynomial-time algorithm for Π would imply a polynomial-time algorithm for every problem in NP. More formally, a problem Π is NP-hard if and only if, for any problem Π_0 in NP, there is a polynomial-time Turing reduction from Π_0 to

Π a Turing reduction just means a reduction that can be executed on a Turing machine. Polynomial-time Turing reductions are also called Cook reductions.

In other words,

Π *is NP-hard* ⇔ *If* Π *can be solved in polynomial time, then P=NP*

A problem is NP-complete if it is both NP-hard and an element of NP (or NP-easy). NP-complete problems are between the hardest problems in NP. If anyone finds a polynomial-time algorithm for even one NP-complete problem, then that would imply a polynomial-time algorithm for every NP-complete problem. Literally thousands of problems have been shown to be NP-complete, so a polynomial-time algorithm for one (i.e., all) of them seems incredibly unlikely.

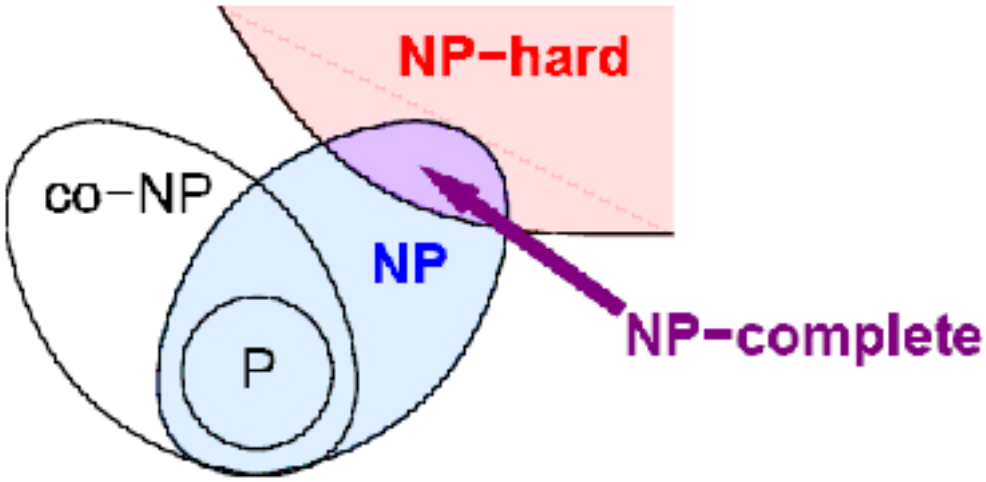

Figure 2.2: P, NP, NP-complete and NP-hard classes

Based on previous definitions, some differences and remarks concerning the two types of problems should be mentioned.

- It is easy to see that there are NP-hard problems that are not NP-complete.
- Although only decision problems can be NP-complete (according to the definition), it is used to refer also to the corresponding optimization problem as being NP-complete.
- The NP-hard refers to a problem that all problems in NP can be reduced to, but which itself is not in NP
- NP-hard problems are at least as hard as NP-complete problems.
- An optimization problem may be NP-hard. There also exist NP-hard decision problems that are not NP-complete.

2.4 NP vs. P

The relationship between the classes P and NP is fundamental for the theory of NP-completeness. The P versus NP problem is to determine whether every language accepted

by some nondeterministic algorithm in polynomial time is also accepted by some (deterministic) algorithm in polynomial time.

Problem Statement: Does P = NP?

The first observation, which is implicit is that

$$P \subseteq NP$$

Every decision problem solvable by a polynomial time deterministic algorithm is also solvable by a polynomial time nondeterministic algorithm. If a problem $\Pi \in P$ and A is a any polynomial time deterministic algorithm for Π, we can obtain a polynomial time nondeterministic algorithm for Π. Thus $\Pi \in P$ implies $\Pi \in NP$.

But are there search problems that cannot be solved in polynomial time? In other words, is $P \neq NP$? Most algorithms researchers think so. It is hard to believe that exponential search can always be avoided famously unsolved for decades and centuries. There are a variety of reasons why it is widely believed that $P \neq NP$. However, proving this has turned out to be extremely difficult, one of the deepest and most important unsolved puzzles of mathematics.

Computational complexity theory plays an important role in modern cryptography [Gol00]. The security of the internet, including most financial transactions, depend on complexity-theoretic assumptions such as the difficulty of integer factoring or of breaking DES (the Data Encryption Standard). If P= NP these assumptions are all false. Specifically, an algorithm solving 3-SAT in n^2 steps could be used to factor 200-digit numbers in a few minutes.

Although a practical algorithm for solving an NP-complete problem (showing P = NP) would have devastating consequences for cryptography, it would also have stunning practical consequences of a more positive nature, and not just because of the efficient solutions to the many NP-hard problems important to industry. For example, it would transform mathematics by allowing a computer to find a formal proof of any theorem which has a proof of reasonable length, since formal proofs can easily be recognized in polynomial time. Although the formal proofs may not be initially intelligible to humans, the problem of finding intelligible proofs would be reduced to that of finding a recognition algorithm for intelligible proofs. Similar remarks apply to diverse creative human endeavors, such as designing airplane wings, creating physical theories, or even composing music. The question in each case is to what extent an efficient algorithm for recognizing a good result can be found. This is a fundamental problem in artificial intelligence, and one whose solution itself would be aided by the NP-solver by allowing easy testing of recognition theories.

2.5 Satisfiability Problem. Cook's theorem

In complexity theory, satisfiability, or SAT, is a decision problem, whose instance is a Boolean expression written using only AND, OR, NOT, variables, and parentheses. The question is: given the expression, is there some assignment of TRUE and FALSE values to the variables that will make the entire expression true? A formula of propositional logic is said to be satisfiable if logical values can be assigned to its variables in a way that makes the formula true.

SAT is a problem of great practical importance, with applications ranging from chip testing and computer design to image analysis and software engineering. Given the im-

portance of the SAT search problem, researchers over the past 50 years have tried hard to find efficient ways to solve it, but without success. The fastest algorithms we have are still exponential on their worst-case inputs.

However, a very important theorem concerning this problem have been proved:

Theorem (Cook's Theorem): *SAT problem is NP-complete.*

We won't insist on the proof of this theorem. Still, it is important to mention the fact that this theorem had a great impact in the complexity theory since SAT is the first problem that has been shown to belong to the class of NP-complete problems. Moreover, no other problem has been proved to be NP-complete. However, a method for checking if a problem belong to the class of NP-complete problems have been found, namely showing that the given problem is 'harder' than SAT problem. This issue is discussed in Section 2.6.

2.6 Prove NP-Completeness

There are no truly standard approaches to prove a problem to be NP-Complete. However, there are several general types of proofs that occur frequently and that can provide a suggestive framework for deciding how to go about proving a new problem NP-complete. These techniques are:

1. **Restriction:** This is the easiest NP-completeness proof method, when it is applicable. This method consists of showing that a problem to be proved NP-complete contains a special case of a known NP-complete problem. For example, the Directed Hamiltonian Cycle problem is proved to be the special case of Undirected Hamiltonian Cycle, where each directed arc is complemented by its opposite direction counterpart, which is identical to the Undirected Hamiltonian Cycle problem.

2. **Local Replacement:** In this approach, we pick some aspect of the known NP-complete problem instance to make up a collection of basic units, and then we obtain the corresponding instance of the target problem by replacing each basic unit, in a uniform way, with a different structure. The transformation of SAT to 3-SAT follows this approach. The basic units of an instance of SAT were the clauses, and each clause was replaced by a collection of clauses of 3-SAT clauses according to the same general rule.

3. **Component design:** This is the most difficult of the three popular proof methods. The basic idea is to use the constituents (or components, parts) of the target problem instance to design certain components that can be combined to realize instances of the known NP-complete problem.

The process for proving a decision problem Π to be NP-complete is:

- Show that Π is in NP;
- Select a known NP-complete problem Π_0;
- Construct a transformation from Π to Π_0;
- Prove that the transformation has the complexity $O(p)$.

Selecting an NP-complete problem for transformation it is not a problem since at least one problem is provided by Cook's fundamental theorem(see Section 2.5). However, finding the relation between the given problem and the NP-complete problem is not very simple. The principal technique used for demonstrating that two problems are related is that of *reducing* one to the other by giving a constructive transformation that maps any instance of the first problem into an equivalent instance of the second. Such a transformation provides the means for converting any algorithm that solves the second problem into a corresponding algorithm for solving the first problem. It is necessary to emphasize here the significance of *polynomial time reducibility*, that is, reductions for which the required transformation can be executed by a polynomial time algorithm. If we have a polynomial time reduction from one problem to another, this ensures that any polynomial time algorithm for the second problem can be converted into a corresponding poynomial time algorithm for the first problem.

If we denote a reduction from A to B by

$$A \rightarrow B$$

then we can say that difficulty flows in the direction of the arrow, while efficient algorithms move in the opposite direction. It is through this propagation of difficulty that we know NP-complete problems are hard: all other search problems reduce to them, and thus each NP-complete problem contains the complexity of all search problems. If even one NP-complete problem is in P, then P = NP.

Reductions also have the convenient property that they compose. If

$A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$.

Starting with SAT as the common denominator, researchers have proven many decision problems to be NP-complete by this process. The most famous collection of these results are by Richard Karp (see [24]) where he proved 21 problems to be NP-complete.

In Figure 2.6 the reduction between some well-known NP-complete problems is shown. Note that for these problems algorithms for reduction have already been implemented.

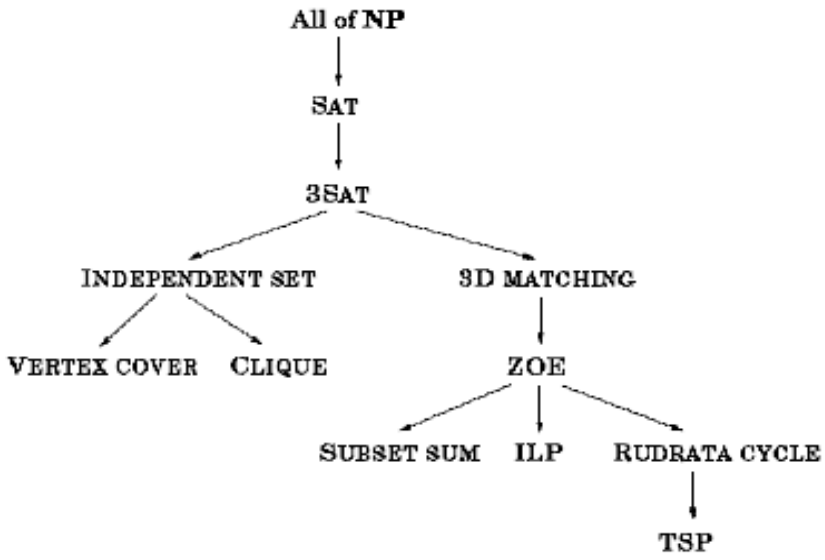

Figure 2.3: Reduction between NP-complete problems

Chapter 3

Unconventional computing

Unconventional computing is computing by a wide range of new or unusual methods. It is also known as alternative computing.

Unconventional computating is, according to [**?**], "'an interdisciplinary research area with the main goal to enrich or go beyond the standard models, such as the von Neumann computer architecture and the Turing machine, which have dominated computer science for more than half a century"'. These methods model their computational operations based on non-standard paradigms, and are currently mostly in the research and development stage.

The different methods of uncoventional computing can be seen in Figure 3.1.

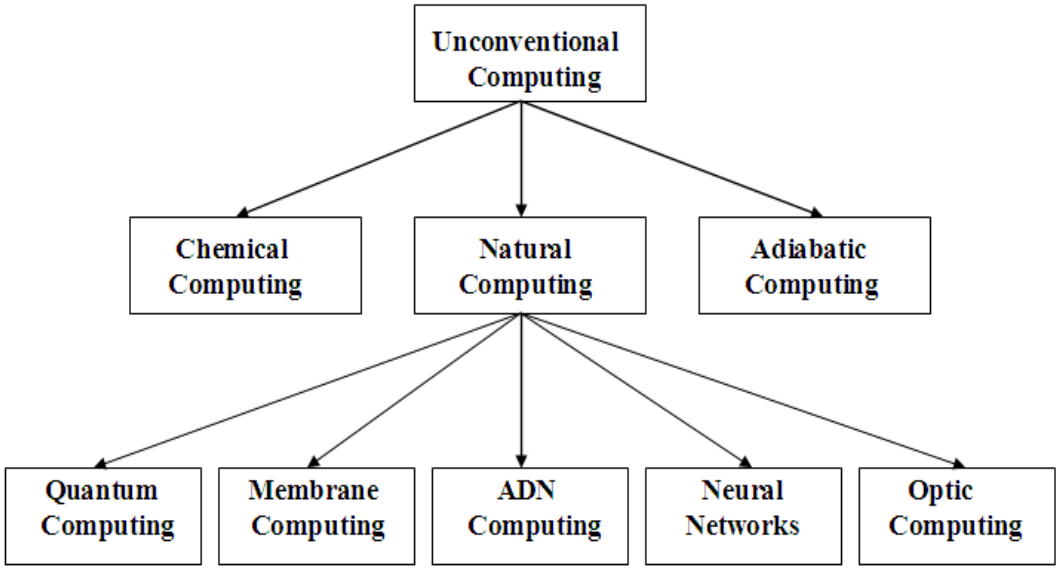

Figure 3.1: A schematic representation of the branches of unconventional computing

Natural Computing is one of the most important branches of unconventional computing and it refers to computing going on in nature and computing inspired by nature. When complex phenomena going on in nature are viewed as computational processes, our understanding of these phenomena and of the essence of computation is enhanced. In this way one gains valuable insights into both natural sciences and computer science. Characteristic for man-designed computing inspired by nature is the metaphorical use of concepts,

principles and mechanisms underlying natural systems. Thus, evolutionary algorithms use the concepts of mutation, recombination and natural selection from biology; neural networks are inspired by the highly interconnected neural structures in the brain and the nervous system; molecular computing is based on paradigms from molecular biology; and quantum computing based on quantum physics exploits quantum parallellism. There are also important methodological differences between various subareas of natural computing. Thus, e.g., evolutionary algorithms and algorithms based on neural networks are presently implemented on conventional computers. On the other hand, molecular computing also aims at alternatives for silicon hardware by implementing algorithms in biological hardware (bioware), e.g., using DNA molecules and enzymes. Also quantum computing aims at nontraditional hardware that would allow quantum effects to take place. Computer science undergoes now an important transformation by trying to combine the computing carried on in computer science with the computing observed in nature all around us. Natural computing is a very important catalyst of this transformation, and holds a lot of promise for the future.

In this chapter a brief description of some of the most important methods of unconventional computing are presented in Sections 3.2 - 3.6.

3.1 From Turing to present days

In some sense, the whole history of computer science is the history of a series of continuous attempts to discover, study, and, if possible, implement computing ideas, models, paradigms from the way nature the humans included computes. We do not enter here into the debate whether or not the processes taking place in nature are by themselves computations, or we interpret them as computations, but we just recall the fact that when defining the computing model which is known now as Turing machine and which provides the standard by now definition of what is computable, A. Turing (in 19351936) explicitly wanted to abstract and model what a clerk in a bank is doing when computing with numbers. One decade later, McCullock, Pitts, Kleene founded the finite automata theory starting from modelling the neuron and the neural networks. Later, this led to the area called now neural computing whose roots can be found in unpublished papers of the same A. Turing (see also Section 3.3 below).

Genetic algorithms and evolutionary computing/programming are now well established (and much applied practically) areas of computer science. One decade ago, the history making Adlemans experiment of computing with DNA molecules was reported, proving that one can not only get inspired from biology for designing computers and algorithms for electronic computers, but one can also use a biological support (a bio-ware) for computing. In the last years, the search of computing ideas/models/paradigms in biology, in general in nature, became explicit and systematic under the general name of natural computing.

This trend of computer science is not singular, many other areas of science and technology are scrutinizing biology in the hope (confirmed in many cases) that life has polished for billions of years numerous wonderful processes, tools, and machineries which can be imitated in domains apparently far from biology, such as materials and sensor technology, robotics, bionics, nanotechnology.

3.2 A typical case: Evolutionary computing

In order to see the (sometimes unexpected) benefits we can have in this framework, it is instructive to examine the case of genetic algorithms. Roughly speaking, they try

to imitate the bio-evolution in solving optimization problems: the space of candidate solutions for a problem are encoded as *chromosomes* (strings of abstract symbols), which are evolved by means of cross-overing and point mutation operations, and selected from a generation to the next one by means of a fittness mapping. The trials to improve the fitness mapping continue until either no essential improvement is done for a number of steps, or until a given number of iterations are performed.

The biological metaphors are numerous and obvious. What is not obvious (from a mathematical point of view) is why such a brute force approach searching randomly the space of candidate solutions, with the search guided by random cross-overings and point mutations is as successful as it happens to be (with a high probability, in many cases, the Genetic Algorithms provide a good enough solution in a large number of applications). The most convincing 'explanation' is probably because nature has used the same strategy in improving species. This kind of bio-mystical 'explanation' provides a rather optimistic motivation for related researches.

3.3 Neural computing

A special mentioning deserves another 'classic' area included nowadays in unconventional computing, namely neural computing. In short, the challenge is now to learn something useful from the brain organization, from the way the neurons are linked. The standard model consists of neuron-like computing agents (finite state machines, of very reduced capabilities), placed in the vertices of a network, with numerical weights on edges, aiming to compute a function.

In a first phase, the network is trained for the task to carry out, and the weights are adjusted, then the network is used for solving a real problem. Pattern recognition problems are typical to be addressed via neural networks. The successes (and the promises) are comparable with those of genetic algorithms, without having a similarly wide range of applications. However, the brain remains such a misterious and efficient machinery that nobody can underestimate the progresses in any area trying to imitate the brain.

It also deserves to mention the rather interesting detail that Alan Turing himself, some years after introducing Turing machines, had a paper where he proposed a computing device in the form of a network of very simple computing units, able to learn, and then to solve an optimization problem. Unfortunately, his paper remained unpublished and was only recently reevaluated(see http://www.AlanTuring.net and [41])

3.4 DNA Computing

Coming back to the history making Adlemans experiment mentioned above [2], it has the merit of opening (actually, confirming, because speculations about using DNA as a support for computations were made since several decades, while theoretical computing models inspired from the DNA structure and operations were already proposed in eighties, see [21]) a completely new research vista, not looking for better algorithms for existing computers, but for a new type of hardware, based on bio-molecules.

It is worth emphasizing the fundamental novelty of this event: the dream is to find an essentially new type of computers sometimes called *wet computer*. The great promise is to solve hard problems in a feasible time, by making use of the massive parallelism made possible by the very compact way of storing information on DNA molecules (bits at the molecular level, with some orders of efficiency over silicon supports). In this way, billions of 'computing chips' can be accommodated in a tiny test tube, much more than on silicon.

The possible (not yet very probable for the near future) DNA computer also has other attractive features: energetical efficiency, reversibility, evolvability.

For the practical computer science, DNA computing fuels several hopes, mainly related to the massive parallelism mentioned above. On this basis, we can simulate non-determinism (which is anyway present in biochemistry), so that one can address in this framework computationally hard problems, with the possibility to push with some steps the feasibility barriers at least for certain problems.

3.5 Membrane computing

Another component of this general intellectual enterprise is membrane computing, which starts from the observation that the cell is the smallest living thing, and at the same time it is a marvellous tiny machinery, with a complex structure, an intricate inner activity, and an exquisite relationship with its environment the neighboring cells included. Then, the challenge is to find in the structure and the functioning of the cell those elements useful for computing. Distribution, parallelism, non-determinism, decentralization, (non)synchronization, coordination, communication, robustness, scalability, are only a few keywords related to this challenge. For instance, a problem which cannot be easily solved in terms of silicon engineering, but which was misteriously and very efficiently solved by nature at the level of the cell is related to the coordination of processes, the control pathways which keep the cell alive, without a high cost of coordination (in parallel computing the communication complexity is sometimes higher than the time and space complexity). Then, interesting questions appear in connection with the organization of cells into tissues, and this is also related to the way the neurons cooperate in the brain.

3.6 Quantum computing

There are aspects of physics, such as reversibility, that are relevant to computation and can be brought under better control by incorporating them directly in the computation scheme. Quantum computation (QC) represents an important further step in this direction. *Quantum mechanics* is of course used extensively in the design of semiconductor devices and communication systems. However, until recently the most peculiar, nonclassical aspect of quantum mechanics were hidden within the devices and didn't affect the logic variables that are the object of a computation.

In computer science, however, this theory was restricted to being referenced in terms of physical hardware contexts such as that of nano-technology and materials science. Only recently, however,have scientists begun to understand that quantum phenomenon can be used inthe simulation and representation of classical computation [23].

The basis of building a quantum computer is the *qubit*. This term refers to a physical system that can be arranged in one of the two states commonly labeled as 0 and 1 or in a superposition of these two states. It is because of this ability to be in multiple states at one time, that a QC can store many inputs. So, the *qubit* is a mechanism which is analogous to the state-based classical computers of today. Our dominant paradigm is based on two states: on and off (or high and low current). The qubit is a powerful extension of this. Not only can we represent two separate states, we can represent any linear combination of these two states in one qubit. This inturn leads to massive parallelism on a scale that can not be imagined in classical computers (see [11],[23]).

Chapter 4

Using light in computations

Using light, instead of electric power, for performing computations is an exciting idea whose applications can be already seen on the market. This choice is motivated by the increasing number of real-world problems where the light-based devices could perform better than electric-based counterparts. They are immune to electromagnetic interference, and free from electrical short circuits. They have low-loss transmission and provide large bandwidth; i.e. multiplexing capability, capable of communicating several channels in parallel without interference. They are capable of propagating signals within the same or adjacent fibers with essentially no interference or cross-talk. They are compact, lightweight, and inexpensive to manufacture, and more facile with stored information than magnetic materials[1].

Optical interconnections and optical integrated circuits will provide a way out of these limitations to computational speed and complexity inherent in conventional electronics. Optical computers will use photons traveling on optical fibers or thin films instead of electrons to perform the appropriate functions. In the optical computer of the future, electronic circuits and wires will be replaced by a few optical fibers and films, making the systems more efficient with no interference, more cost effective, lighter and more compact. Optical components would not need to have insulators as those needed between electronic components because they don't experience cross talk. Indeed, multiple frequencies (or different colors) of light can travel through optical components without interfacing with each others, allowing photonic devices to process multiple streams of data simultaneously.

It is hoped that optical computing could advance computer architecture and can improve the speed of data input and output by several orders of magnitude [13].

Good examples in this direction can be found in the field of Optical Character Recognition [44]. Another example is the n-point discrete Fourier transform computation which can be performed, optically, in only unit time [18, 36]. Based on that, a solution to the subset sum problem can be obtained by discrete convolution. The idea is that the convolution of 2 functions is the same as the product of their frequencies representation.

In the following sections some interesting examples in the field of optical computation are presented.

4.1 Optical Character Reecognition

Optical character recognition, usually abbreviated to OCR, is a type of computer software designed to translate images of handwritten or typewritten text (usually captured by a scanner) into machine-editable text, or to translate pictures of characters into a standard encoding scheme representing them (e.g. ASCII or Unicode). OCR began as a field of

research in pattern recognition, artificial intelligence and machine vision.

The quest for the light-based computers was started in 1929 by G. Tauschek who has obtained a patent on Optical Character Recognition (OCR) in Germany, Next step was made by Handel who obtained a patent on OCR in USA in 1933 (U.S. Patent 1,915,993). Those devices were mechanical and used templates for matching the characters. A photodetector was placed so that when the template and the character to be recognized were lined up for an exact match, and a light was directed towards it, no light would reach the photodetector. [44]. Since then, the field of OCR has grown steadily and recently has become an umbrella for multiple pattern recognition techniques (including Digital Character Recognition).

The accurate recognition of Latin-script, typewritten text is now considered largely a solved problem. Typical accuracy rates exceed 99%, although certain applications demanding even higher accuracy require human review for errors. Recognition of hand printing, cursive handwriting, and even the printed typewritten versions of some other scripts (especially those with a very large number of characters), is still the subject of active research.

4.2 Wave all-silicon laser

An important practical step was made by Intel researchers (see Figure 4.2) who have developed the first continuous wave all-silicon laser using a physical property called the Raman Effect [12, 34, 37, 38] (the Raman effect is a process by which light interacts with vibrating atoms in a material resulting in the transfer of energy).

Figure 4.1: Intel concept: a continuous wave laser embedded into a silicon chip.

In this case, the Raman effect, in which light is amplified when it passes through a transparent material, is used to extend the range of lasers used in communication systems based on glass fiber. The new Intel process miniaturizes the Raman effect so it can be produced within a silicon chip, rather than requiring strands of glass fiber that stretch for miles. The Intel researchers demonstrated that the effect was 10, 000 times as strong in silicon as it was in glass fiber.

The device could lead to such practical applications as optical amplifiers, lasers, wavelength converters, and new kind of lossless optical devices.

4.3 Rainbow Sort

Rainbow Sort was first described by Schultes [39]. It utilises the phenomenon of dispersion, where light beams of longer wavelengths are refracted to a lesser degree than beams of a

shorter wavelength(see Figure 4.2).

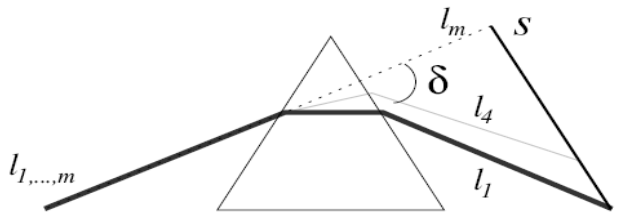

Figure 4.2: Schematic view of rainbow sort

A set of elements is to be sorted. Each element is encoded as a beam of light of a distinct wavelength. The separate beams are then combined into one beam of light and this is passed through a prism. The component beams are refracted at different angles and so emerge from the prism separately and in an order dictated by their wavelength. A light measurement device can be positioned to sequentially read the ordered component beams.

There is a relationship between the angle of deviation δ (the angle between the input beam and the output beam) and the refractive index of the prism medium for each wavelength of light [39]. We will restrict the possible wavelengths of the input beam so that the distance from where the uninterrupted beam would have reached the sensor to where the diffracted beam reaches it is linear in $\tan\delta$ and the distance between the sensor and the prism. This can be expressed as

$$s = p \tan \delta$$

where s is the distance along the sensor and p is the distance between the sensor and the prism surface.

For implementing the Rainbow Sort one need to perform the following steps:

- encode multiple wavelengths (representing the numbers to be sorted) into a light ray,
- send the ray through a prism which will split the ray into n monochromatic rays that are sorted by wavelength,
- read the output by using a special detector that receives the incoming rays.

However, Rainbow Sort cannot guarantee to return a stable sorting as it returns a sorted list of wavelengths not a list of indices. Also Rainbow Sort cannot be directly used to sort lists with repeated values. Rainbow Sort is not an instance of the Model.

Generalization of the Rainbow Sort

In [29] Wood (et al.) outline a natural generalisation of Rainbow Sort. Generalised Rainbow Sort is very similar to Rainbow Sort except that it utilises the full geometry of the prism. It keeps each input beam at a different depth in the prism, as shown in Figure 4.3. The output is then detected by a two dimensional sensor and is read off sequentially. Generalised Rainbow Sort sorts the input in a manner that is similar to Rainbow Sort but instead returns a list of indices, and naturally deals with repeated elements in the input.

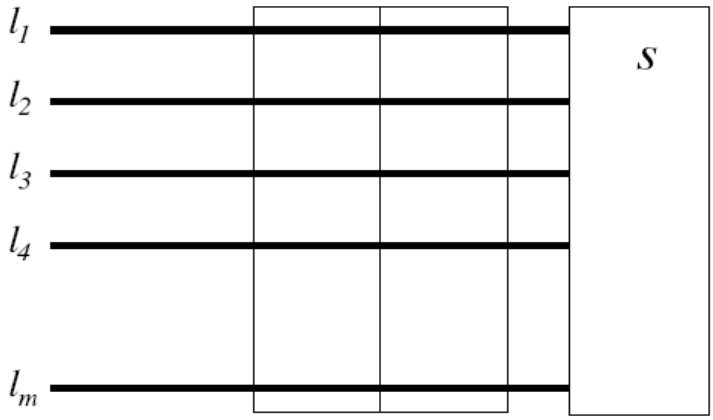

Figure 4.3: Generalized rainbow search

The resulting Generalised Rainbow Sort is a stable sort. It is interesting to note that the introduction of stability to Rainbow Sort does not decrease its time complexity.

Another stable version of the Rainbow Sort is proposed in [30].

4.4 Lenslet's matrix multiplication

Another solution comes from Lenslet [25] which has created a very fast processor for vector-matrix multiplications (see Figure 4.4). This processor can perform up to 8000 Giga Multiple-Accumulate instructions per second. The device is based on a 3-core engine: An optical Vector-Matrix Multiplier (VMM), a Vector Digital Processing Unit (VPU), and an off-the-shelf scalar DSP. The Optical core is the prime contributor to the outstanding processing power.

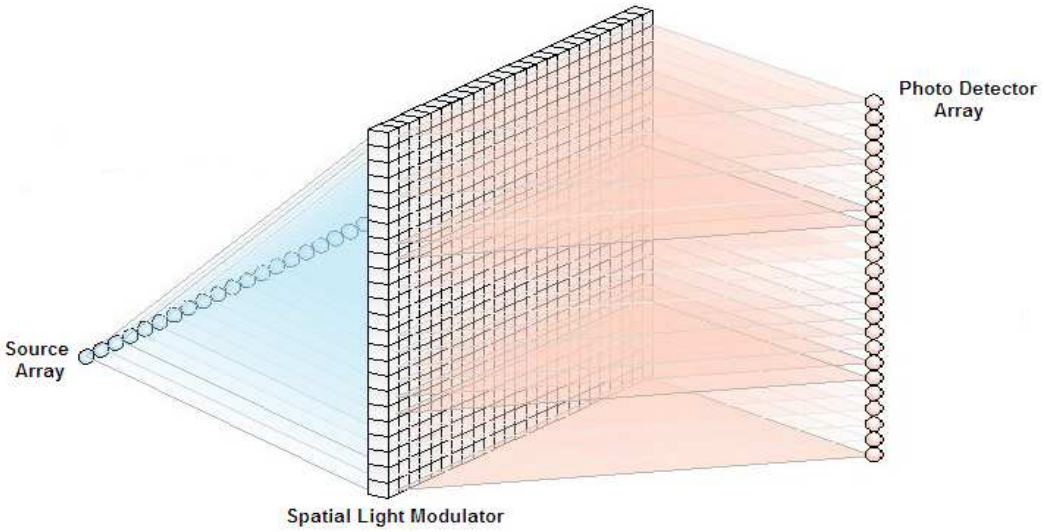

Figure 4.4: A sketch of the Lenslet device used for performing vector-matrix multiplications

The representation of the solution starts from the fact that any linear transform can be represented by means of optical elements (see Figure 4.4).

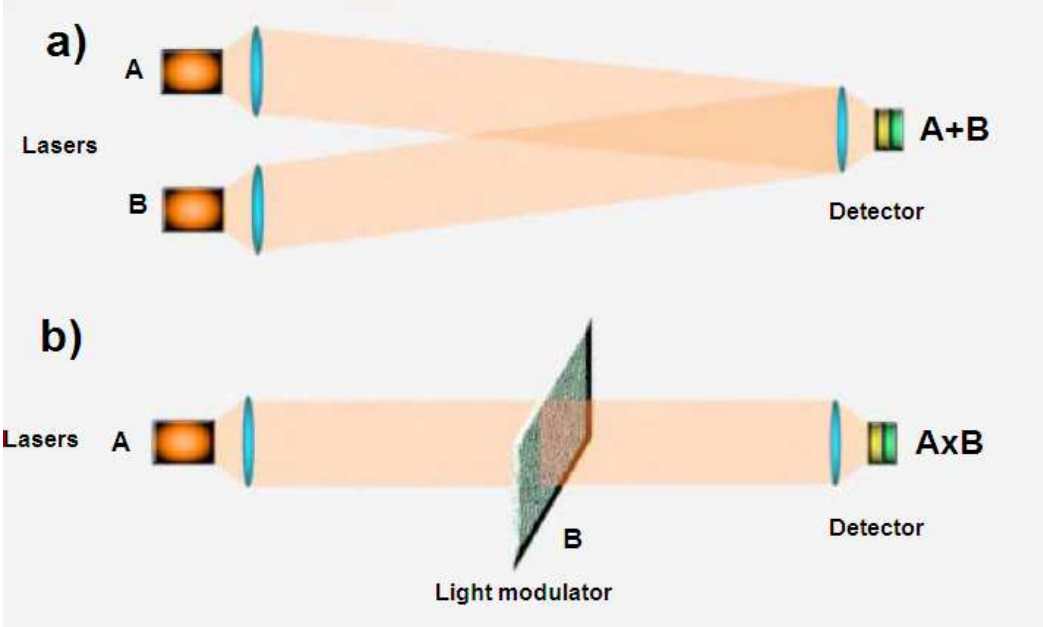

Figure 4.5: (a). Addition of two numbers using an optic device; (b). Multiplication of two numbers using an optic device.

Lenslet technology has already been applied for solving different problems. Some of the most important are listed below.

- *Wireless Communications:* robust communications, e.g. multi-Spread Spectrum techniques, also including spatial processing through array antennas;
- *Real time video compression:* H.264 video compression for full quality standard and HDTV transmissions single and multi channel;
- *Radars and Electronic Warfare:* Being able to run long Fourier Transform tasks at a tremendously high pace makes it a natural candidate for advanced RADAR applications;
- Data Mining: Analysis of large networks requires matrix multiplication and manipulation. Other data mining applications require pattern searches. All these are accelerated by 3-4 orders of magnitude using this device as compared to standard processors;
- *Software definable radio:* A single radio will be able to simultaneously transmit and receive several communication channels of a variety of communication modes. For example this may solve communication problems between different security forces that may be involved in same events.

4.5 The Continuous Space Machine

Naughton (et al.) proposed and investigated [31, 43] a model called the Continuous Space Machine (CSM) inspired by the theory of Fourier optics. The CSM operates in discrete

timesteps over a finite number of two-dimensional (2D) complex-valued images of finite size and infinite spatial resolution.A finite control is used to traverse, copy, and perform other optical operations on the images.

Each instance of the CSM consists of a memory containing a program (an ordered list of operations) and an input. Informally, the memory structure is in the form of a 2D grid of rectangular elements. The grid has finite size and a scheme to address each element uniquely. Each grid element holds a 2D image. There is a program start address and two well-known addresses. The model has a number of operations that effect optical image processing tasks. For example, two operations available to the programmer, st and ld (parameterised by two column addresses and two row addresses), copy rectangular subsets of the grid out of and into the first image address, respectively. Upon such loading and storing the image contents are rescaled to the full extent of the target location. The other operations are image Fourier transform (FT), complex conjugation, multiplication, addition, amplitude thresholding, and some control flow operations (all these operations are performed in constant time).

It was proved that this model can simulate analog recurrent neural networks, thus establishing a lower bound on its computational power. The authors also defined an $O(\log_2 n)$ unordered search algorithm with CSM.

Chapter 5

Solving problems

In this chapter are presented some well known NP complete problems solved using the light-based device presented in Chapter 4.

Our idea is based on two properties of light:

- The speed of light has a limit. The value of the limit is not very important at this stage of explanation. The speed will become important when we will compute the size of the graphs that can be solved by our device. What is important now is the fact that we can delay the ray by forcing it to pass through an optical fiber cable of a certain length.
- The ray can be easily divided into multiple rays of smaller intensity/power by using some beam-splitters.

Initially a light ray is sent to the start node. Generally speaking two operations must be performed when a ray passes through a node:

- The light ray is marked uniquely so that we know that it has passed through that node.
- The ray is divided into a number of rays equal to the external degree of that node. Each obtained ray is directed toward one of the nodes connected to the current node.

At the destination node we will search only for particular rays that have passed only once through each node.

5.1 Hamiltonian path problem

5.1.1 Definition

The description of the Hamiltonian Path problem (HPP) for a directed graph is the following:

Given a directed graph $G = (V, E)$ with $|V| = n$ nodes and a start node (v_{start}) and a stop node (v_{stop}), the problem asks to compute is there is a simple path, beginning with node v_{start} and ending with node v_{stop}, containing all nodes exactly once. The output for this decision problem is either YES or NO depending on whether the Hamiltonian path does exist or not.

The Hamiltonian path problem arises in many real-word applications [4, 10].

5.1.2 The proposed device

This section deeply describes the proposed system. First step is to find a way to mark the signals which passes through nodes such that the interesting signals can be easily identified at the destination node. The mathematical background required for this operation is described in what follows and also the physical implementation of the labeling system is given.

Labeling system

At the destination node we will wait for a particular ray which has passed through all nodes of the graph exactly once. This is why we need to find a way to label that particular ray so that it could be easily identified.

Actually we are interested in marking all rays which pass through a particular node with a unique label, such that Hamiltonian path is uniquely identified at the destination node (v_{stop}).

In the solution proposed in this paper, the rays passing through a node are marked by delaying them with a certain amount of time. This delay can be easily obtained by forcing the rays to pass through an optical fiber of a certain length. Roughly speaking, we will know if a certain ray has traversed a Hamiltonian path only if its delay (at the destination node) is equal to the sum of delays of all nodes in that graph. We will also know the particular moment when the expected ray (the one which has completed a Hamiltonian path) will arrive. In this case the only thing that we have to do is to "listen" if there is a fluctuation in the intensity of the signal at that particular moment. Due to the special properties of the proposed system we know that no other ray will arrive, at the destination node, at the moment when the Hamiltonian path ray has arrived.

The delays, which are introduced by each node, cannot take any values. If we would put random values for delays we might have different rays (which are not Hamiltonian paths) arriving, at the destination node, in the same time with a ray representing a Hamiltonian path.

We need only the ray, which has traversed a Hamiltonian path, to arrive in the destination node at the moment equal to the sum of delays of each node (the moment when the ray has entered in the start node is considered moment 0). Thus, the delaying system must have the following property:

Property of the delaying system

Let us denote by d_1, d_2, ..., d_n the delays introduced by each node of the graph. A correct set of values for this system must satisfy the condition:

$d_1 + d_2 + ... + d_n \neq a_1 \cdot d_1 + a_2 \cdot d_2 + ... + a_n \cdot d_n$,

where a_i $(1 \leq i \leq n)$ are natural numbers $(a_i \geq 0)$ and cannot be all 1 in the same time.

Basicaly speaking a_k tell us how many times the ray has passed through node k. Thus, if value a_k is strictly greater than 1 it means that the ray has passed at least twice through node k.

Finding the appropriate labeling system was a three-step process. First of all we have written a computer program which generates this numbers by using a backtracking procedure [9]. Then we have extracted a general formula and we have proved its correctness.

Backtracking procedure

We also wanted to generate numbers such that the highest number in a system is the smallest possible. This will ensure that the network is constructed in an efficient way. The labeling systems generated by our computer programs are given in Table 5.2.

Table 5.1: The labeling system generated by our backtracking procedure. First column contains the number of nodes of the graph. The second column represents the labels applied to nodes.

n	Labels (delays)
1	1
2	2, 3
3	4, 6, 7
4	8, 12, 14, 15
5	16, 24, 28, 30, 31
6	32, 48, 56, 60, 62, 63

Extracting the general formula

From Table 5.2 it can be easily seen that these numbers follow a general rule. For a graph with n nodes one of the possible labeling systems is:

$2^n - 2^{n-1}$,
$2^n - 2^{n-2}$,
$2^n - 2^{n-3}$,
... ,
$2^n - 2^0$.

Remarks

- The numbers in this set are also called Nialpdromes (sequence A023758 from The On-line Encyplopedia of Integer Numbers [40]). They are numbers whose digits in base 2 are in nonincreasing order.
- These numbers have been used in [22] for solving NP-complete problems in the context of DNA computers [2],
- The delaying system described above has the advantage of being a general one, but it also has a weakness: it is exponential.

5.1.3 Proving the correctness of the system

We have to prove that the property of delaying system holds for this sequence of numbers.

Actually we have to prove that the equality:

$$\begin{gathered} 2^n - 2^{n-1} + 2^n - 2^{n-2} + 2^n - 2^{n-3} + ... + 2^n - 2^0 = \\ a_1 \cdot (2^n - 2^{n-1}) + a_2 \cdot (2^n - 2^{n-2}) + a_3 \cdot (2^n - 2^{n-3}) + ... + a_n \cdot (2^n - 2^0) \end{gathered} \tag{5.1}$$

(where $a_i \geq 0$) is not possible unless all a_i are equal to 1.

The left part of the equality is:

$2^n - 2^{n-1} + 2^n - 2^{n-2} + 2^n - 2^{n-3} + ... + 2^n - 2^0 =$
$n \cdot 2^n - (2^{n-1} + 2^{n-2} + 2^{n-3} + ... + 2^0) =$
$n \cdot 2^n - 2^n + 1 =$
$(n-1) \cdot 2^n + 1.$

First of all we have to see that the equality does not hold if all a_i numbers are at least 1 and at least one number is strictly greater than 1. If this happens the 2^n term will be represented at least n times. But, it must be represented only $n-1$ times (see above).

Thus, if at least one number a_i is strictly greater than 1, it means that other numbers a_j must be 0. We will prove that the equality (1) does not hold in this case too.

As discussed above, if one of the coefficients is 0, at least one of the other coefficients must be strictly greater than 1. For instance, if a_1 is set to 0, it means that a_2 must be set to 3 in order to compensate the missing 2^{n-1} term. This is a direct consequence of the fact that $2^{n-1} = 2 \cdot 2^{n-2}$. Of course, we also have to take into account that 2^{n-2} must be represented once. This is why we have to set a_2 to value 3.

If a_2 is also 0 we have to set a_3 to value 7 (we need $4 \cdot 2^{n-3}$ in order to compensate the missing term 2^{n-1}, we also need $2 \cdot 2^{n-3}$ to compensate the missing term 2^{n-2} and, of course, the term 2^{n-3} must be represented 1 time).

As a general idea: if a particular term (2^{n-j}) is missing (the corresponding coefficient a_j is set to 0), it can be compensate by setting one of the next coefficients to a value of at least 3. But, a coefficient set to 0 means that the term 2^n is missing once, and by setting another coefficient to at least 3 we will get at least 2 extra representations for 2^n. This will mean that right part of the equation (1) will be at least $(n+1) \cdot 2^n + 1$. But, the left part of the equation is only $(n-1) \cdot 2^n + 1$.

With this we have shown that all $a_i (1 \leq i \leq n)$ must be 1 in order to have equality in equation (1). For any other values of a_i the equality (1) does not hold.

5.1.4 Proving the minimality

An important question is whether this system is the minimal possible (the greatest number is the minimal possible). We prove this by showing a counter-example.

Let us suppose that there is another delaying system which is smaller than the one described above. This means that the largest delay is smaller than $2^n - 2^0$. We are not interested in the other $(n-1)$ delays because the largest one imposes the largest length for the cables.

Consider a graph with 2 nodes. In this case the delaying system is {2, 3}. We will show that the largest delay cannot be smaller than 3.

If it is smaller it should be 2 or 1 (actually it cannot be 1 because, in this case, the other delay should be 0 and we do not allow this delay in our system). If the delay is 2 it means that the delay for the other node is 1. The sum of the delays is 3 which can be simply obtained by visiting 3 times the node with delay 1. This is in conflict with the property of the delaying system (see section 5.4.5).

With this we have proved that the largest delay cannot be smaller than $2^n - 2^0$.

5.1.5 How the system works

An schematic example of a graph-like system is given in Figure 5.1.

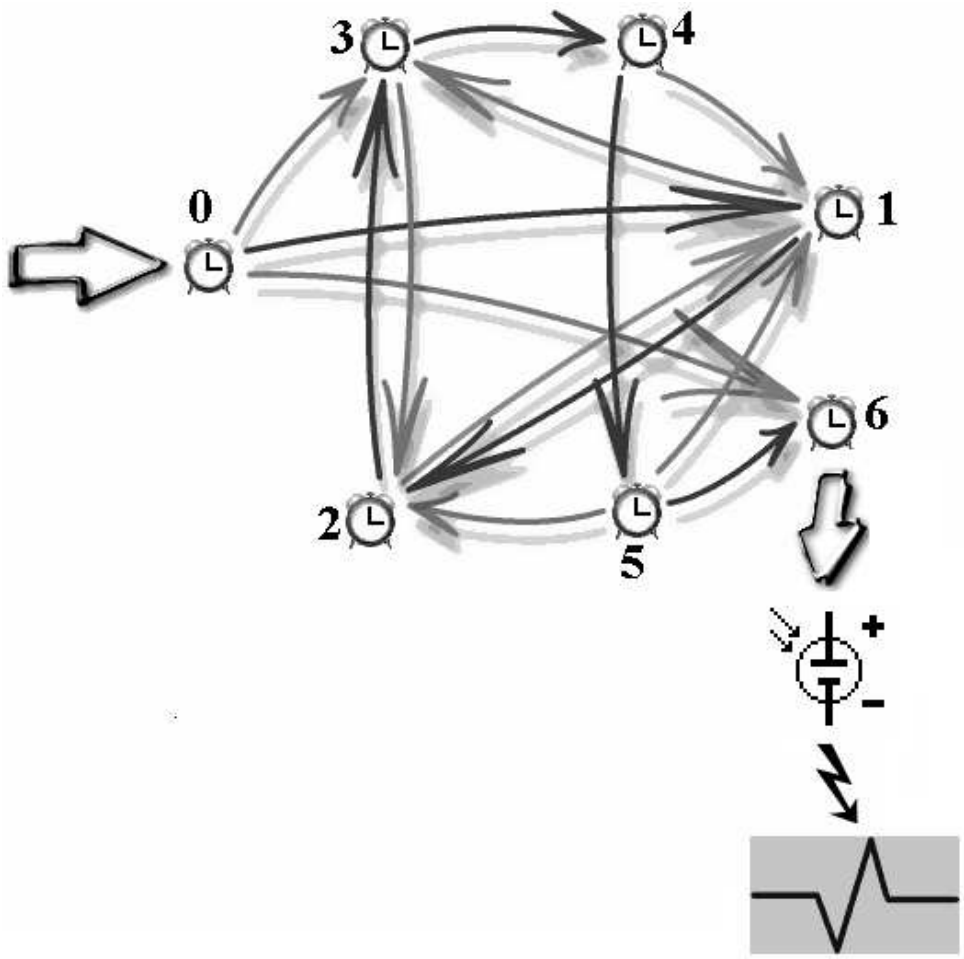

Figure 5.1: A schematic representation of the proposed device. Consider a graph with 7 nodes. Start node is 0 and the destination node is 6. The list of arcs is: (0, 1), (0, 3), (0, 6), (1, 2), (1, 3), (2, 1), (2, 3), (3, 2), (3, 4), (4, 1), (4, 5), (5, 2), (5, 6). In each node the rays are delayed with a certain amount of time. At the destination node we will have a photodiode and an oscilloscope. When a ray will arrive in the destination node there will be a fluctuation, in the light intensity, which could be measured by the oscilloscope.

In the graph depicted in Figure 5.1 the light will enter in node 0. It will be delayed with a certain amount of time and then it will be divided into 3 rays which will be sent to nodes 1, 3 and 6. In node 1 the ray will be delayed (with the amount of time corresponding to node 1) and then it will be sent to nodes 2 and 3. One of the rays will be able to generate and Hamiltonian path (0, 1, 2, 3, 4, 5, 6).

Note that there are also several cycles in the graph: 1, 3, 4 or 1, 2 or 2, 3. The cycles will make that some particular rays to be trapped within the system. The rays which have passed once through the previously described cycle are not considered Hamiltonian paths because the moments when they arrive, at the destination node, are different from the sum of delays introduced by each node.

Because we are working with continuous signal we cannot expect to have discrete output at the destination node. This means that rays arrival is notified by fluctuations in the intensity of the light. These fluctuations will be transformed, by the photodiode, in fluctuations of the electric power which will be easily read by the oscilloscope.

5.1.6 Complexity

This section answers a very important question: *Why the proposed approach is not a polynomial-time solution for the HPP?*

At the first sight one may be tempted to say that the proposed approach provides a solution in polynomial time to any instance of the HPP. The reason behind such claim is given by the ability of the proposed device to provide output to any instance by traversing only once all nodes ($O(n)$ complexity). This could mean that we have found a polynomial-time algorithm for the HPP. A direct consequence is obtaining solutions, in polynomial time, for all other NP-Complete problems - since there is a polynomial reduction between them [15].

However, this is not our case. As can be seen from Table 5.2 the delay time increases exponentially with the number of nodes. Even if the ray has to traverse only n nodes (resulting a complexity of $O(n)$), the total time required by the ray to reach the destination node increases exponentially with the number of nodes.

There are two direct consequences which are derived from here:

- The length of the optical fibers, used for delaying the signals, increases exponentially with the number of nodes,
- The intensity of the signal decreases exponentially with the number of nodes that are traversed.

These two issues are discussed in sections 5.1.7 and 5.6.2.

5.1.7 Problem size

We are interested in computing the size of the cables required to solve a certain instance of the problem in a small amount of time. This will give as a rough indication on the size of the graphs that can be solved using our system in reasonable time.

This size heavily depends on the accuracy of the measurement tools (response time of the photodiode and the rise time of the oscilloscope).

The rise-time of the best oscilloscope available on the market is in the range of picoseconds (10^{-12} seconds). This means that we should not have two signals that arrive at 2 consecutive moments at a difference smaller than 10^{-12} seconds.

Knowing that the speed of light is $3 \cdot 10^{11} m/s$ we can easily compute the minimal cable length that should be traversed by the ray in order to be delayed with 10^{-12} seconds. This is obviously 0.3 meters.

This value is the minimal delay that should be introduced by a node in order to ensure that the difference between the moments when two consecutive signals arrive at the destination node is greater or equal to the measurable unit of 10^{-12} seconds. This will also ensure that we will be able to correctly identify whether the signal has arrived in the destination node at a moment equal to the sum of delays introduced by each node. No other signals will arrive within a range of 10^{-12} seconds around that particular moment.

Once we have the length for that minimal delay is quite easy to compute the length of the other cables that are used in order to induce a certain delay.

Recall from section 5.4.5, Table 5.2 that a graph with 5 nodes has the following delaying system:

16, 24, 28, 30, 31.

From the previous reasoning line we have deduced that the smaller indivisible unit is 0.3. So, we have to multiply these numbers by 0.3. We obtain:

4.8, 7.2, 8.4, 9.0, 9.3.

These numbers represent the length of the cables that must be used in graph's nodes in order to induce a certain delay.

Note that the delay introduced by the cables connecting the nodes was not taken into account in this example. This is not a limitation of our system. The cables connecting nodes can be set to have some length which must obey the property of delaying system (see section 5.4.5). Note that all cables must have the same length. In this case if we have a graph with 4 nodes the length of every cable connecting the nodes must be set to 16 units (the shortest possible - in order to reduce the costs). The length of cables within the nodes should be 24, 28, 30 and 31 units.

The largest length in this sequence is 9.3 meters. This length is not very big, but for larger graphs the length of the cables within nodes can be a problem.

Once we have the length for that minimal delay is quite easy to compute the maximal number of nodes that a graph can have in order to find the Hamiltonian path in one second. We know the facts:

- the largest delay has the form $2^n - 1$ (see equation 5.1),
- the distance traversed by light in 1 second is $3 \cdot 10^8$ meters,
- the shortest delay possible is 0.3 meters.

We simply have to solve the equation:

$$2^n \cdot 0.3 = 3 \cdot 10^8 \tag{5.2}$$

This number is about 33 nodes. However, the length of the optic fibers used for inducing the largest delay for this graph is huge: about $8 \cdot 10^{11}$ meters. We cannot expect to have such long cables for our experiments.

However, shorter cables (of several hundreds of kilometers) are already available in the internet networks. They can be easily used for our purpose. Assuming that the longest

cable that we have is about 300 kilometers we may solve instances with about 17 nodes. The amount of time required to obtain a solution is about 10^{-6} seconds.

Note that the maximal number of nodes can be increased when the precision of our measurement instruments (oscilloscope and photodiode) is increased.

Also note that this difficulty is not specific to our system only. Other major unconventional computation paradigms, trying to solve NP-complete problems share the same fate. For instance, a quantity of DNA equal to the mass of Earth is required to solve HPP instances of 200 cities using DNA computers [19].

5.1.8 Improving the performance of the device for particular graphs

The labeling system proposed in section 5.4.5 is a general one. It can be used for any kind of graph (with any number of nodes and arcs). We have shown that this system has a big problem: the value of the involved numbers increase exponentially with the number of nodes in the graph being solved.

But, for particular graphs we can find other labeling systems which are not exponential. For example, the linear graph (see Figure 5.2 can be solved by our device by using virtually no delays. In this case the moment when the signal arrives in the destination node is equal to sum of delays introduced by the cables connecting the nodes.

Figure 5.2: A linear graph with 7 nodes. No delays are required for the nodes of this graph in order to find a Hamiltonian path

Finding the optimal labeling system for a particular graph is an interesting problem which will be investigated in the near future.

5.2 Bounded subset sum problem

5.2.1 Definition

The description of the subset-sum problem [15] is the following:

Given a set of positive numbers $A = \{a_1, a_2, ..., a_n\}$ and another positive number B. Is there a subset of A whose sum equals B?

We focus our attention on the YES / NO decision problem. We are not interested in finding the subset generating the solution. Actually we are interested to find only if such subset does exist.

The subset-sum problem arises in many real-word applications [17].

5.2.2 The proposed device

The first idea for our device was that numbers from the given set A represent the delays induced to the signals (light) that passes through our device. For instance, if numbers a_1, a_3 and a_7 generate the expected subset, then the total delay of the signal should be $a_1 + a_3 + a_7$. If using light we can easily induce some delays by forcing the ray to pass through an optical cable of given length.

This is why we have designed our device as a direct graph. Arcs, which are implemented by using optical cables, are labeled with numbers from the given set A. Each number is assigned to exactly one arc and there are no two arcs having assigned the same number. There are $n + 1$ nodes connected by n arcs. At this moment of explanation we have a linear graph as the one shown in Figure 5.8.

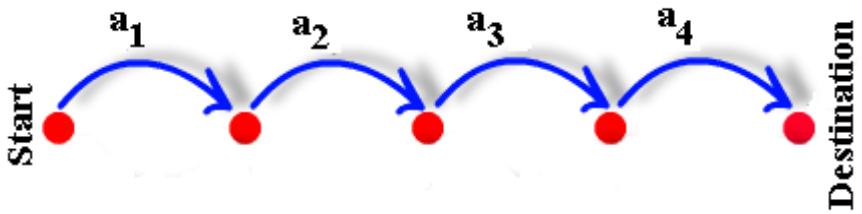

Figure 5.3: First version of our device. Each arc delays the ray by the amount of time written on it. Note that this device is not complete because it cannot generate all possible subsets of A.

However, this is not enough because we also need an mechanism for skipping an arc. Only in this way we may generate all possible subsets of A.

A possible way for achieving this is to add an extra arc (of length 0) between any pair of consecutive nodes. Such device is depicted in Figure 5.9. A light ray sent to start node will have the possibility to either traverse a given arc (from the upper part of figure) or to skip it (by traversing the arc of length 0 from the bottom of figure).

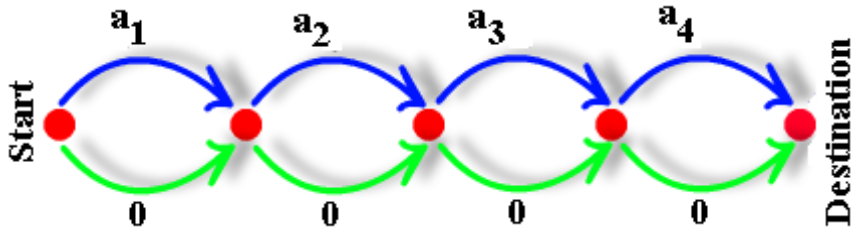

Figure 5.4: Second version of our device. Each subset of A is generated, but this device cannot be implemented in practice because we cannot have cables of length 0.

In each node (but the last one) we place a beam-splitter which will split a ray into 2 subrays of smaller intensity.

The device will generate all possible subsets of A. Each subset will delay one of the ray by an amount of time equal to the sum of the lengths of the arcs in that path.

There is a problem here: even if theoretically we could have arcs of length 0, we cannot have cables of length 0 in practice. For avoiding this problem we have multiple solutions. The first one was to use very short cables (let's say of length ϵ) for arcs which are supposed to have length 0. However, there is another problem here: we could obtain for instance the sum B written as $B = a_1 + 3 * \epsilon$. Even if there is no subset of sum B, still there will be possible to have a signal at moment B due to the situation presented above.

For avoiding this situation we have added a constant k to the length of each cable. The schematic view of this device is depicted in Figure 5.11.

We can see that each path from *Start* to *Destination* contains exactly n time value k. Thus, at the destination we will not wait anymore at moment B. Instead we will wait for a solution at moment $B + n * k$ since all subsets will have the constant $n * k$ added.

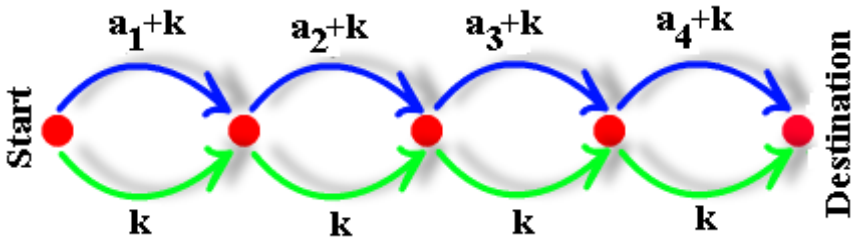

Figure 5.5: A schematic representation of the device used for solving an instance with 4 numbers. On each arc we have depicted its length. There are n cables of length k and n cables of length $a_i + k$ $(1 \leq i \leq n)$. This device does generate all possible subsets of A and it can be implemented in practice

The device will generate all possible subsets of A. The good part is that we do not have to check all 2^n possible solutions. We will only have to check if there is a ray arriving at moment $B + n * k$ in the destination node. The signals generated by all other subsets are ignored and not recorded in any way.

5.2.3 How the system works

In the graph depicted in Figure 5.11 the light will enter in *Start* node. It will be divided into 2 subrays of smaller intensity. These 2 rays will arrive into the second node at moments $a_1 + k$ and k. Each of them will be divided into 2 subrays which will arrive in the 3^{rd} node at moments $2 * k, a_1 + 2 * k, a_2 + 2 * k, a_1 + a_2 + 2 * k$. These rays will arrive at no more than 4 different moments.

In the destination node we will have 2^n rays arriving at no more than 2^n different moments. The ray arriving at moment $n * k$ means the empty set. The ray arriving at moment $a_1 + a_2 + ... + a_n + n * k$ represents the full set. If there is a ray arriving at moment $B + n * k$ means that there a subset of A of sum B.

If there are 2 rays arriving at the same moment in the *Destination* it simply means that there are multiple subsets which have the same sum. This is not a problem for us because we want to answer the YES/NO decision problem (see section 5.2.1). We are not interested at this moment which is the subset generating the solution.

Because we are working with continuous signal we cannot expect to have discrete output at the destination node. This means that rays arrival is notified by fluctuations in the intensity of the light. These fluctuations will be transformed, by a photodiode, in fluctuations of the electric power which will be easily read by an oscilloscope.

5.3 Unbouded subset sum problem

5.3.1 Definition

The description of the unbounded subset-sum problem [15, 17, 35] is the following:

Given a set of real positive numbers $A = \{a_1, a_2, ..., a_n\}$ and a positive number B. Are there some integer values $x_i \geq 0$ $(1 \leq i \leq n)$ such that $\sum_{i=1}^{n}(x_i * a_i) = B$?

We are interested in the YES/NO decision problem. That is, we are interested to find whether a solution does exist. We are not interested to find the actual values of x_i.

5.3.2 Device 1

The idea for our device is that numbers from the given set A represent the delays induced to the signals (light) that passes through our device. For instance, if numbers a_1, appearing twice, and a_4 generate the expected multiset, then the total delay of the signal should be $2 * a_1 + a_4$. By using light we can easily induce some delays by forcing the ray to pass through an optical cable of given length.

This is why we have designed our device as a directed graph. Arcs, which are implemented by using optical cables, are labeled with numbers from the given set A. There is a total of 2 nodes. One of them has $card(A)$ loops and is also connected by one arc to the 2^{nd} one, representing the destination node. A possible representation of this device is shown in Figure 5.7.

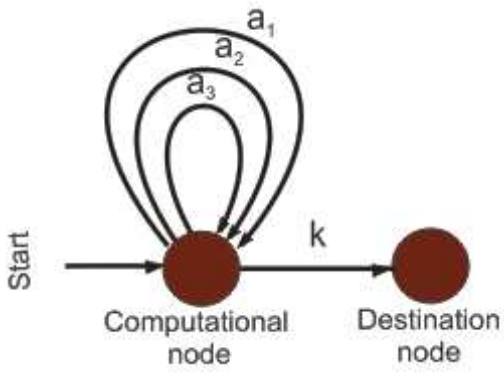

Figure 5.6: A schematic representation of the device used for solving an instance with 3 numbers. On each arc we have depicted its length. The loops have the lengths a_i $(1 \leq i \leq 3)$ and the other arc has length k. The light is initially sent to node computational node.

In the computational node we place a beam-splitter which will split a ray into $card(A)+1$ subrays of smaller intensity.

The device will generate all possible multisets of elements from the given set A. Each combination will delay one of the ray by an amount of time equal to the sum of the lengths of the arcs in that path. It can be easily seen that each element of the set can have any number of occurences.

To the arc that connect the computational node to destination we have assigned a positive constant value k. This is why instead of waiting for a solution at moment B, we will have to wait for it at moment $B+k$, since all multisets will have the k constant added.

How the system works

In the graph depicted in Figure 5.7 the light will enter in the computational node. It will be divided into $card(A)+1$ subrays of smaller intensity. One of these rays will arrive into the destination node and the rest of them will enter the loops, arriving again in the computational node at moments a_i $(1 \leq i \leq card(A))$. Each of these subrays will be divided into $card(A)+1$ subrays, one that goes to destination node and the rest go back to the 1^{st} node, arriving at moments $a_i + a_j$ $(1 \leq i \leq card(A), 1 \leq j \leq card(A))$.

In the destination node we will have more rays arriving at different moments. If there is a ray arriving at moment $B + k$ means that there is a subset of A of sum B.

If there is more than one ray arriving at the moment $B+k$ in the destination it simply means that there are multiple multisets whose elements have the same sum. This is not

a problem for us because we want to answer the YES/NO decision problem (see section 5.3.1). We are not interested at this moment which is the subset generating the solution.

Because we are working with continuous signal we cannot expect to have discrete output at the destination node. This means that rays arrival is notified by fluctuations in the intensity of the light. These fluctuations will be transformed, by the photodiode, in fluctuations of the electric power which will be easily read by the oscilloscope.

Complexity

This section answers a very important question: *Why the proposed approach is not a polynomial-time solution for the subset-sum problem?*

At the first sight one may be tempted to say that the proposed approach provides a solution in polynomial time to any instance of the subset-sum problem. The reason behind such claim is given by the ability of the proposed device to provide output to any instance in O(B) complexity. This could mean that we have found a polynomial-time algorithm for the subset-sum problem. A direct consequence is obtaining solutions, in polynomial time, for all other NP-Complete problems - since there is a polynomial reduction between them [15].

However, this is not our case.

The main reason is that the intensity of the signal decreases exponentially with the number of nodes that are traversed. When the ray is passing through one of the 2 computational nodes it is divided into $n+1$ subrays, where $n = card(A)$. If divided uniformly we could have a $n+1$ times decrease in the intensity for each subray. If a ray is passing 5 times through each of the 2 nodes we can have a decrease of $(n+1)^{10}$ times of power. A possible way for handling this issue is discussed in section 5.6.2.

Precision Another problem is that we cannot measure the moment $B+k$ exactly. We can do this measurement only with a given precision which depends on the tools involved in the experiments. Actually it will depend on the response time of the photodiode and the rise time of the oscilloscope.

The rise-time of the best oscilloscope available on the market is in the range of picoseconds (10^{-12} seconds). This means that we should not have two signals that arrive at 2 consecutive moments at a difference smaller than 10^{-12} seconds. We cannot distinguish them if they arrive in a smaller than 10^{-12}s interval. In our case it simply means that if a signal arrives in the destination in the interval $[B+k-10^{12}, B+k+10^{12}]$ we cannot be perfectly sure that we have a correct subset or another one which does not have the wanted property.

Knowing that the speed of light is $3 \cdot 10^{11} m/s$ we can easily compute the minimal cable length that should be traversed by the ray in order to be delayed with 10^{-12} seconds. This is obviously 0.3 meters.

This value is the minimal delay that should be introduced by an arc in order to ensure that the difference between the moments when two consecutive signals arrive at the destination node is greater or equal to the measurable unit of 10^{-12} seconds. This will also ensure that we will be able to correctly identify whether the signal has arrived in the destination node at a moment equal to the sum of delays introduced by each arc.

Once we have the length for that minimal delay is quite easy to compute the length of the other cables that are used in order to induce a certain delay.

Note that the maximal number of nodes can be increased when the precision of our measurement instruments (oscilloscope and photodiode) is increased.

Power decrease

Beam splitters are used in our approach for dividing a ray into more subrays. Because of that, the intensity of the signal is decreasing. In the worst case we have an exponential

decrease of the intensity. For instance, having a set A with n elements, each signal is divided (within the two computational nodes) into $n+1$ signals. Roughly speaking, the intensity of the signal will decrease $(n+1)^{[B/min(A)]+1}$ times, where $min(A)$ represents the smallest value in the set A.

This means that, at the destination node, we have to be able to detect very small fluctuations in the intensity of the signal. For this purpose we will use a photomultiplier [14] which is an extremely sensitive detector of light in the ultraviolet, visible and near infrared range. This detector multiplies the signal produced by incident light by as much as 10^8, from which even single photons can be detected.

Also note that this difficulty is not specific to our system only. Other major unconventional computation paradigms, trying to solve NP-complete problems share the same fate. For instance, a quantity of DNA equal to the mass of Earth is required to solve Hamiltonian Path Problem instances of 200 cities using DNA computers [19].

5.3.3 Device 2

The idea for our device is that numbers from the given set A represent the delays induced to the signals (light) that passes through our device. For instance, if numbers a_1, appearing twice, and a_4 generate the expected multiset, then the total delay of the signal should be $2 * a_1 + a_4$. By using light we can easily induce some delays by forcing the ray to pass through an optical cable of given length.

This is why we have designed our device as a directed graph. Arcs, which are implemented by using optical cables, are labeled with numbers from the given set A. Each number is assigned to exactly 2 arcs, connecting two nodes. There is a total of 3 nodes. Two of them are connected by $card(A)$ arcs and each of these 2 nodes is connected by one arc to the 3^{rd} one, representing the destination node. A possible representation of this device is shown in Figure 5.7.

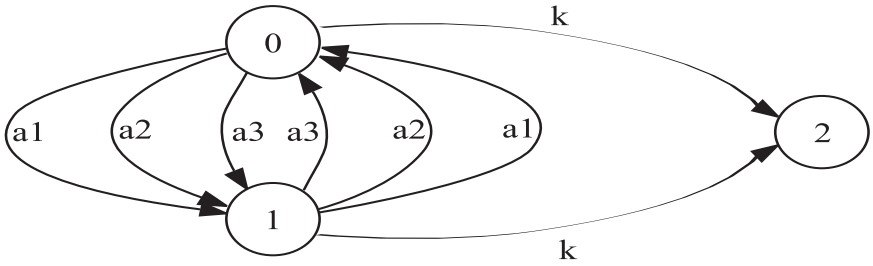

Figure 5.7: A schematic representation of the device used for solving an instance with 3 numbers. On each arc we have depicted its length. There are 8 arcs in this device. Six of them have the length given by numbers a_i $(1 \leq i \leq 3)$ and 2 of them have length k. The light is initially sent to node 0. Node 2 is the destination node.

In each node (but the destination one) we place a beam-splitter which will split a ray into $card(A)+1$ subrays of smaller intensity.

The device will generate all possible multisets of elements from the given set A. Each combination will delay one of the ray by an amount of time equal to the sum of the lengths of the arcs in that path. It can be easily seen that each element of the set can have any number of occurances.

To the two nodes that connect the computational nodes to destination we have assigned a possitive constant value k. This is why instead of waiting for a solution at moment B,

we will have to wait for it at moment $B + k$, since all multisets will have the k constant added.

How the system works

In the graph depicted in Figure 5.7 the light will enter in first node used for performing the computations. It will be divided into $card(A) + 1$ subrays of smaller intensity. One of these rays will arrive into the destination node and the rest of them will arrive into the second computational node at moments a_i $(1 \leq i \leq card(A))$. These rays will arrive at $card(A)$ different moments. Each of these subrays will be divided into $card(A) + 1$ subrays, one that goes to destination node and the rest go back to the 1^{st} node, arriving at moments $a_i + a_j$ $(1 \leq i \leq card(A), 1 \leq j \leq card(A))$.

In the destination node we will have more rays arriving at different moments. If there is a ray arriving at moment $B + k$ means that there a subset of A of sum B.

If there is more than one ray arriving at the moment $B+k$ in the destination it simply means that there are multiple multisets whose elements have the same sum. This is not a problem for us because we want to answer the YES/NO decision problem (see section 5.3.1). We are not interested at this moment which is the subset generating the solution.

Because we are working with continuous signal we cannot expect to have discrete output at the destination node. This means that rays arrival is notified by fluctuations in the intensity of the light. These fluctuations will be transformed, by the photodiode, in fluctuations of the electric power which will be easily read by the oscilloscope.

Complexity This section answers a very important question: *Why the proposed approach is not a polynomial-time solution for the subset-sum problem?*

At the first sight one may be tempted to say that the proposed approach provides a solution in polynomial time to any instance of the subset-sum problem. The reason behind such claim is given by the ability of the proposed device to provide output to any instance in $O(B)$ complexity. This could mean that we have found a polynomial-time algorithm for the subset-sum problem. A direct consequence is obtaining solutions, in polynomial time, for all other NP-Complete problems - since there is a polynomial reduction between them [15].

However, this is not our case.

The main reason is that the intensity of the signal decreases exponentially with the number of nodes that are traversed. When the ray is passing through one of the 2 computational nodes it is divided into $n + 1$ subrays, where $n = card(A)$. If divided uniformly we could have a $n + 1$ times decrease in the intensity for each subray. If a ray is passing 5 times through each of the 2 nodes we can have a decrease of $(n + 1)^{10}$ times of power. A possible way for handling this issue is discussed in section 5.6.2.

Precission

Another problem is that we cannot measure the moment $B + k$ exactly. We can do this measurement only with a given precision which depends on the tools involved in the experiments. Actually it will depend on the response time of the photodiode and the rise time of the oscilloscope.

The rise-time of the best oscilloscope available on the market is in the range of picoseconds (10^{-12} seconds). This means that we should not have two signals that arrive at 2 consecutive moments at a difference smaller than 10^{-12} seconds. We cannot distinguish them if they arrive in a smaller than 10^{-12}s interval. In our case it simply means that if a signal arrives in the destination in the interval $[B + k - 10^{12}, B + k + 10^{12}]$ we cannot be perfectly sure that we have a correct subset or another one which does not have the wanted property.

Knowing that the speed of light is $3 \cdot 10^{11} m/s$ we can easily compute the minimal cable

length that should be traversed by the ray in order to be delayed with 10^{-12} seconds. This is obviously 0.3 meters.

This value is the minimal delay that should be introduced by an arc in order to ensure that the difference between the moments when two consecutive signals arrive at the destination node is greater or equal to the measurable unit of 10^{-12} seconds. This will also ensure that we will be able to correctly identify whether the signal has arrived in the destination node at a moment equal to the sum of delays introduced by each arc.

Once we have the length for that minimal delay is quite easy to compute the length of the other cables that are used in order to induce a certain delay.

Note that the maximal number of nodes can be increased when the precision of our measurement instruments (oscilloscope and photodiode) is increased.

Power decrease

Beam splitters are used in our approach for dividing a ray into more subrays. Because of that, the intensity of the signal is decreasing. In the worst case we have an exponential decrease of the intensity. For instance, having a set A with n elements, each signal is divided (within the two computational nodes) into $n+1$ signals. Roughly speaking, the intensity of the signal will decrease $(n+1)^{[B/min(A)]+1}$ times, where $min(A)$ represents the smallest value in the set A.

This means that, at the destination node, we have to be able to detect very small fluctuations in the intensity of the signal. For this purpose we will use a photomultiplier [14] which is an extremely sensitive detector of light in the ultraviolet, visible and near infrared range. This detector multiplies the signal produced by incident light by as much as 10^8, from which even single photons can be detected.

Also note that this difficulty is not specific to our system only. Other major unconventional computation paradigms, trying to solve NP-complete problems share the same fate. For instance, a quantity of DNA equal to the mass of Earth is required to solve Hamiltonian Path Problem instances of 200 cities using DNA computers [19].

5.4 Exact cover

5.4.1 Definition

The description of the Exact Cover problem [15] is the following:

Given a set $U = \{u_1, u_2, ..., u_n\}$ of elements and a set C of subsets of U, an exact cover is a subset S of C such that every element in U is contained in exactly one set in S.

Our purpose is to solve the YES/NO decision problem. This means that we are not interested to compute the actual subset S. Rather, we are interested in finding whether a solution does exist or not.

In what follows we denote by m the cardinal of C.

5.4.2 Example

$U = \{u_1, u_2, ..., u_5\}$
$C = \{C_1, C_2, C_3, C_4, C_5, C_6\}$
$C_1 = \{u_1, u_3\}$
$C_2 = \{u_1, u_2, u_5\}$
$C_3 = \{u_2\}$
$C_4 = \{u_1, u_4, u_5\}$
$C_5 = \{u_2, u_3, u_4\}$

$C_6 = \{u_4, u_5\}$

An exact cover for U is :

$$S = \{C_1, C_3, C_6\}.$$

We are not thinking to U as a set of numbers. Rather we adopt a more general definition: U is a set of items. We do this in order to avoid confusion which might appear because, later in this paper (see section 5.4.5), we will attach positive integer numbers to each item in U. Those numbers have some special properties and they should not be confounded with the actual value of the items u_i (in the case that these values are numerical).

5.4.3 The proposed device

We have divided our problem into 2 small subproblems:

- to generate all possible subsets of C. We show how our device is capable of performing this operation in section 5.4.4.
- to find if there is a subset of C which contains each element of U exactly once. This is possible due to the special properties of the numbers assigned to each item from U (see section 5.4.5).

5.4.4 Generating all subsets of C

First of all we assign to each item u_i from U a positive number (d_i). The way in which these numbers are constructed is described in section 5.4.5. We will use these numbers when we will compute the moment when a particular ray has arrived at the output of our device. The sum of all these numbers is denoted by B.

In this section we want to design a device that will generate all possible subsets of C.

The first idea for our device was that elements from the given set C represent the delays induced to the signals (light) that passes through our device. For instance, if elements C_1, C_3 and C_6 generate the expected exact cover, then the total delay of the signal should be $delay(C_1) + delay(C_3) + delay(C_6)$. By $delay(C_i)$ we denote the delay introduced by C_i (which is a subset of U). The $delay(C_i)$ is computed as sum of delays induced by each item (from U) belonging to C_i.

If using light we can easily induce some delays by forcing the ray to pass through an optical cable of given length.

This is why we have designed our device as a direct graph. Arcs, which are implemented by using optical cables, are labeled with numbers representing delays of subsets $C_i (1 \leq i \leq m)$. Each subset is assigned to exactly one arc and there are no two arcs having assigned the same subset. There are $m + 1$ nodes connected by m arcs. At this moment of explanation we have a linear graph as the one shown in Figure 5.8.

However, this is not enough because we also need an mechanism for skipping an arc encoding an element of C. Only in this way we may generate all possible subsets of C.

A possible way for achieving this is to add an extra arc (of length 0) between any pair of consecutive nodes. Such device is depicted in Figure 5.9. A light ray sent to start node will have the possibility to either traverse a given arc (from the upper part of figure) or to skip it (by traversing the arc of length 0 from the bottom of figure).

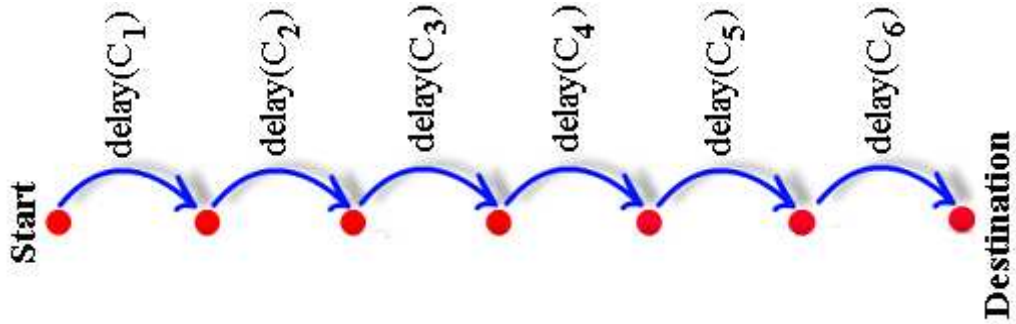

Figure 5.8: First version of our device. Each arc delays the ray by the amount of time written on it. Note that this device is not complete because it cannot generate all possible subsets of C.

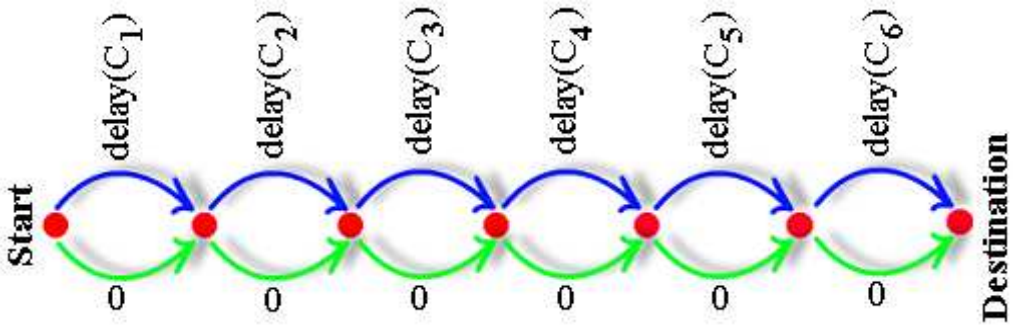

Figure 5.9: Second version of our device. Each subset of C is generated, but this device cannot be implemented in practice because we cannot have cables of length 0.

In each node (but the last one) we place a beam-splitter which will split a ray into 2 subrays of smaller intensity.

The device will generate all possible subsets of C. Each subset will delay one of the ray by an amount of time equal to the sum of the lengths of the arcs in that path.

There is a problem here: even if theoretically we could have arcs of length 0, we cannot have cables of length 0 in practice. For avoiding this problem we have multiple solutions. The first one was to use very short cables (let's say of length ϵ) for arcs which are supposed to have length 0. However, there is another problem here: we could obtain for instance the sum B written as $B = a_1 + 3 * \epsilon$. Even if there is no subset of sum B, still there will be possible to have a signal at moment B due to the situation presented above.

For avoiding this situation we have added a constant k to the length of each cable. The schematic view of this device is depicted in Figure 5.11.

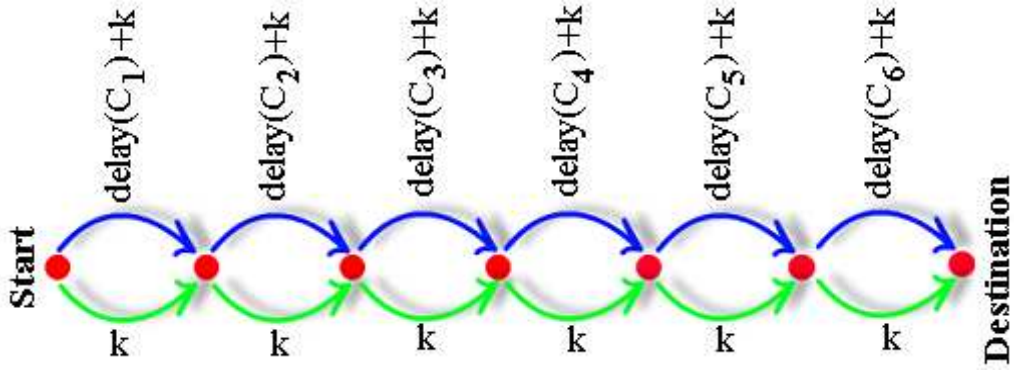

Figure 5.10: A schematic representation of the device used for solving an instance with 6 elements in C. On each arc we have depicted its length. There are n cables of length k and m cables of length $delay(C_i) + k$ $(1 \leq i \leq m)$. This device does generate all possible subsets of C and it can be implemented in practice

We can see that each path from *Start* to *Destination* contains exactly m time value k. Thus, at the destination we will not wait anymore at moment B. Instead we will wait for a solution at moment $B + m * k$ since all subsets will have the constant $m * k$ added.

In Figures 5.8, 5.9 and 5.11 we have not depicted the items u_i from each C_k $(1 \leq k \leq m)$. The delays introduced by each u_i (see section 5.4.5) are actually in the delays of all C_k containing u_i.

This trick has solved one of the problems related to our device. It shows how to generate all possible subsets of C. However, this is not enough. Another requirement of the problem is that each element of U to be represented once in the exact cover solution. For solving this problem we assign to each element in U a special integer value. These values will be chosen in such way will be possible to easily identify the exact cover solution at the destination node.

In the next section we show how to decide if is there a subset of C which contains each element of U exactly once.

5.4.5 Labeling system

We start by a simple definition about what we mean by a ray passing through an item of U.

We say that a ray has visited (passed through) a certain element u_i of U if that ray has passed through an arc representing one of the subsets of C_k which contains element

u_i. Please note that C_k can contain more elements from U and if one of them is visited by a ray it means that all of them are visited by that ray since the light has no chance of avoiding parts of an arc once it has started to traverse it.

Take for instance set C from our example (section 5.4.2). If the arc encoding C_4 is visited by a light ray it means that all items from that set (namely u_1, u_4 and u_5) are visited.

At the destination node we will wait for a particular ray which has visited each element of U exactly once. This is why we need to find a way to label that particular ray so that it could be easily identified.

Recall from section 5.4.4 that the rays passing through an arc are marked by delaying them with a certain amount of time. This delay can be easily obtained by forcing the rays to pass through an optical fiber of a certain length. Roughly speaking, we will know if a certain ray has generated an exact cover only if its delay (at the destination node) is equal to the sum of delays induced by all items from U.

We know exactly the particular moment when the expected ray (the one which has generated an exact cover) will arrive. This moment is B (the sum of numbers attached to items from U). In this case the only thing that we have to do is to "listen" if there is a fluctuation in the intensity of the signal at that particular moment. Due to the special properties of the proposed system we know that no other ray will arrive, at the destination node, at the moment when the ray generating the exact cover has arrived.

The delays, which are introduced by each arc (which is actually the sum of delays of particular items from U), cannot take any values. If we would put random values for delays we might have different rays (which are not covering U exactly) arriving, at the destination node, in the same time with a ray representing an exact cover. This is why we assign, to each element u_i of U, a special number (denoted by d_i).

We need only the ray, which has generated the exact cover, to arrive in the destination node at the moment equal to the sum of delays of each item u_i (the moment when the ray has entered in the start node is considered moment 0). Thus, the delaying system must have the following property:

Property of the delaying system

Let us denote by d_1, d_2, ..., d_n the delays that we plan to attached to u_1, u_2, ..., u_n. A correct set of values for this system must satisfy the condition:

$d_1 + d_2 + ... + d_n \neq a_1 \cdot d_1 + a_2 \cdot d_2 + ... + a_n \cdot d_n$,

where a_i $(1 \leq i \leq n)$ are natural numbers $(a_i \geq 0)$ and cannot be all 1 in the same time.

Basically speaking a_k tell us how many times the ray has passed through an element u_k. Thus, if value a_k is strictly greater than 1 it means that the ray has passed at least twice through element u_k. Note that an element u_i can belong to multiple subsets from C. For instance, in the example from section 5.4.2 we can see that element u_1 belongs to 3 subsets (C_1, C_2 and C_4).

Actually each item u_i can appear no more than m times on any path between Start and Destination (see Figure 5.11). This is because the cardinal of C is m and each C_k $(1 \leq k \leq m)$ is a subset of U (that is each element from U can appear no more than once). In our case we were not interested to depend on particular values for m. This is why when we have designed the system of numbers $d_i (1 \leq i \leq n)$ we have assumed that each item u_i can appear any times in the final solution (not limited to m times). This is the most general case. Other cases, where has a fixed value can lead to smaller values for numbers d_i.

Some examples of numbers complying with our property are given in Table 5.2.

Table 5.2: Some examples. First column contains the cardinal of U. The second column represents the numbers assigned to each element of U.

n	Labels (delays)
1	1
2	2, 3
3	4, 6, 7
4	8, 12, 14, 15
5	16, 24, 28, 30, 31
6	32, 48, 56, 60, 62, 63

From Table 5.2 it can be easily seen that these numbers follow a general rule. For a problem with n elements in U one of the possible labeling systems is:

$2^n - 2^{n-1}$,
$2^n - 2^{n-2}$,
$2^n - 2^{n-3}$,
... ,
$2^n - 2^0$.

Remarks

- In Section 5.1 these numbers have been used for solving the Hamiltonian path problem with a light based-device. It was proved that these numbers satisfy the previously exposed property.
- The numbers in this set are also called Nialpdromes (sequence A023758 from The On-line Encyplopedia of Integer Numbers [40]). They are numbers whose digits in base 2 are in nonincreasing order.
- These numbers have been used in [22] for solving NP-complete problems in the context of DNA computers [2],
- The delaying system described above has the advantage of being a general one, but it also has a weakness: it is exponential.

5.4.6 How the system works

In the graph depicted in Figure 5.11 the light will enter in *Start* node. It will be divided into 2 subrays of smaller intensity. These 2 rays will arrive into the second node at moments $delay(C_1) + k$ and k. Each of them will be divided into 2 subrays which will arrive in the 3^{rd} node at moments $2*k, delay(C_1)+2*k, delay(C_2)+2*k, delay(C_1)+delay(C_2)+2*k$. These rays will arrive at no more than 4 different moments.

In the destination node we will have 2^m rays arriving at no more than 2^m different moments. The ray arriving at moment $m * k$ means the empty set. The ray arriving at moment $delay(C_1) + delay(C_2) + ... + delay(C_n) + m * k$ represents the full set (C). If there is a ray arriving at moment $B + m * k$ means that there a subset of C of sum B (this means an exact cover for U).

If there are 2 rays arriving at the same moment in the *Destination* it simply means that there are multiple subsets which have the same sum (generating the same cover - exact or not). This is not a problem for us because we want to answer the YES/NO decision problem (see section 5.4.1). We are not interested at this moment which is the subset generating the solution.

Because we are working with continuous signal we cannot expect to have discrete output at the destination node. This means that rays arrival is notified by fluctuations in the intensity of the light. These fluctuations will be transformed, by a photodiode, in fluctuations of the electric power which will be easily read by an oscilloscope.

5.4.7 Complexity

This section answers a very important question: *Why the proposed approach is not a polynomial-time solution for the Exact Cover problem?*

At the first sight one may be tempted to say that the proposed approach provides a solution in polynomial time to any instance of the exact cover problem. The reason behind such claim is given by the ability of the proposed device to provide output to any instance in $O(B)$ complexity. This could mean that we have found a polynomial-time algorithm for the Exact Cover problem. A direct consequence is obtaining solutions, in polynomial time, for all other NP-Complete problems - since there is a polynomial reduction between them [15].

However, this is not our case. There are two main reasons for this:

- the intensity of the signal decreases exponentially with the number of nodes that are traversed. When the ray is passing through a node it is divided into 2 subrays. If divided uniformly we could have a 2 times decrease in the intensity for each subray. If a ray is passing through 10 nodes we can have a decrease of 2^{10} times from the initial power. A possible way for handling this issue is discussed in section 5.6.2.
- The numbers assigned to each item in U increase exponentially with the cardinal of U.

Precission

Another problem is that we cannot measure the moment $B + m * k$ exactly. We can do this measurement only with a given precision which depends on the tools involved in the experiments. Actually it will depend on the response time of the photodiode and the rise time of the oscilloscope.

The rise-time of the best oscilloscope available on the market is in the range of picoseconds (10^{-12} seconds). This means that if a signal arrives at the destination in the interval $[B + m * k - 10^{-12}, B + m * k + 10^{-12}]$ we cannot be perfectly sure that we have an exact cover or another one which does not have the wanted property. This problem can be avoided if all cables are long enough. In what follows we will try to compute the length of the cables.

We know that the speed of light is $3 \cdot 10^{11} m/s$. Based on that we can easily compute the minimal cable length that should be traversed by the ray in order to be delayed with 10^{-12} seconds. This is obviously 0.3 meters.

This value is the minimal delay that should be introduced by an arc. More than that, all lengths must be integer multiples of 0.3. We cannot allow to have cables whose lengths can be written as $p * 0.3 + q$, where p is an integer and q is a positive real number less than 0.3 because by combining this kind of numbers we can have a signal in the above

mentioned interval and that signal does not encode a subset whose sum is the expected one.

Once we have the length for that minimal delay is quite easy to compute the length of the other cables that are used in order to induce a certain delay. First of all we have to divide all delays introduced by $C_k(1 \leq k \leq m)$ with such factor that the less significant digit (greater than 0) to be on the first position before the decimal place. For instance if we have $delay(C_1) = 100$ and $delay(C_2) = 2000$ we have to divide both numbers by 100.

After this operation we will multiply the obtained numbers by 0.3 factor.

This will ensure that if a signal will arrive in the interval $[B + m * k - 10^{-12}, B + m * k + 10^{-12}]$ we can be sure that it encodes the sum $B + m * k$.

5.5 Diophantine equations

5.5.1 Definition

Diophantine equations are indeterminate polynomial equations in which only integer solutions are allowed. So, if we are given a polynomial equation $f(x_1, ..., x_n) = 0$ with integer coefficients, we are asked to find any integer solution.

Solving diophantine equations is not an easy task. Even the decision whether an equation has solutions or not can be a real challenge. In [27] it was proved that the problem of deciding if there are positive integer solutions for the equation $a * x^2 + b * y = c$ where a, b and c are positive integers, is NP-complete [15]. Some specific cases of Diophantine equations and their computational complexities were studied[42].

A linear Diophantine equation is an equation of the general form $\sum_{i=1}^{n} a_i * x_i = c$, where a_i $(1 \leq i \leq n)$ and c are integer values.

For the case of linear diophantine equation in two variables $a * x + b * y = c$ with integer solutions the existence of solutions is a very simple problem concerning divisibility of numbers. If c is a multiple of the least common divisor of a and b then the equation has solution. In all other situations the equation has no solution.

However, in this paper we are interested in finding if a diophantine equation has non-negative solutions. This case is more difficult than the previous one since more restriction are imposed here[7]. Moreover, we work with equations that have more than 2 variables, so the problem gets harder.

5.5.2 The proposed device

The device is represented similar to an oriented graph having a number of nodes equal to the number of variables of the diophantine equation plus the destination node. The coefficients of the equation are assign to the arcs of the graph and represent the delays induced to the signals (light) that passes through this device. Delays are induced by forcing a signal to pass through a cable of a given length.

An example of representation for an equation with 4 variables is depicted in Figure 5.11. In each node (but the destination one) we place a beam-splitter which will split a ray into 2 subrays of smaller intensity, one subray going back to the same node and the other one to the next node.

We can see that each path from *Start* to *Destination* contains exactly n time value k. Thus, at the destination we will not wait anymore at moment c. Instead we will wait for a solution at moment $c + n * k$ since all results will have the constant $n * k$ added.

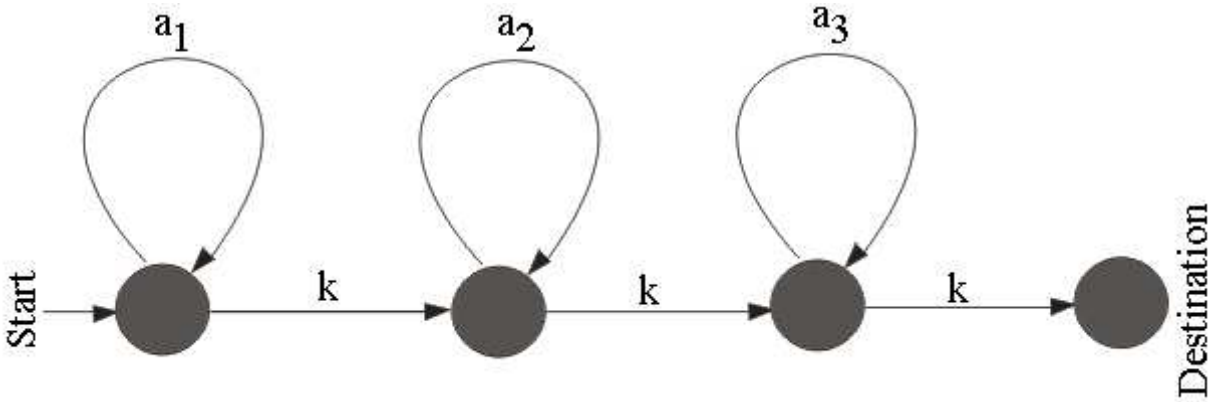

Figure 5.11: A schematic representation of the device used for solving an instance with 4 numbers. On each arc we have depicted its length. There are n cables of length k and n cables of length $a_i + k$ $(1 \leq i \leq n)$.

5.5.3 How the system works

In the graph depicted in Figure 5.11 the light will enter in *Start* node. It will be divided into 2 subrays of smaller intensity.

These 2 rays will arrive into the second node at moments $a_1 + k$ and k. Each of them will be divided into 2 subrays which will arrive in the 3^{rd} node at moments $2 * k, a_1 + 2 * k, a_2 + 2 * k, a_1 + a_2 + 2 * k$. These rays will arrive at no more than 4 different moments.

In the destination node we will have light rays arriving at different moments. The ray arriving at moment $n * k$ means that the equation has the trivial solution. If there is a ray arriving at moment $c + n * k$ means that the equation has solution.

If there are 2 rays arriving at the same moment in the *Destination* it simply means that there are multiple solutions of the equation. This is not a problem for us because we want to answer the YES/NO decision problem (see section 5.5.1). We are not interested at this moment which are the values of the variables representing the solution.

Because we are working with continuous signal we cannot expect to have discrete output at the destination node. This means that rays arrival is notified by fluctuations in the intensity of the light. These fluctuations will be transformed, by a photodiode, in fluctuations of the electric power which will be easily read by an oscilloscope.

5.5.4 Precission

One important problem is that we cannot measure the moment $c + n * k$ exactly. We can do this measurement only with a given precision which depends on the tools involved in the experiments. Actually it will depend on the response time of the photodiode and the rise time of the oscilloscope.

The rise-time of the best oscilloscope available on the market is in the range of picoseconds (10^{-12} seconds). This means that we should not have two signals that arrive at 2 consecutive moments at a difference smaller than 10^{-12} seconds. We cannot distinguish them if they arrive in a smaller than 10^{-12}s interval. In our case it simply means that if a signal arrives in the destination in the interval $[c + n * k - 10^{12}, c + n * k + 10^{12}]$ we cannot be perfectly sure that we have a correct subset or another one which does not have the wanted property.

Knowing that the speed of light is $3 \cdot 10^{11} m/s$ we can easily compute the minimal cable length that should be traversed by the ray in order to be delayed with 10^{-12} seconds. This

is obviously 0.3 meters.

This value is the minimal delay that should be introduced by an arc in order to ensure that the difference between the moments when two consecutive signals arrive at the destination node is greater or equal to the measurable unit of 10^{-12} seconds. This will also ensure that we will be able to correctly identify whether the signal has arrived in the destination node at a moment equal to the sum of delays introduced by each arc.

Once we have the length for that minimal delay is quite easy to compute the length of the other cables that are used in order to induce a certain delay.

Note that the maximal number of nodes can be increased when the precision of our measurement instruments (oscilloscope and photodiode) is increased.

5.6 The end

5.6.1 Physical implementation of the labeling system

For implementing the proposed device we need the following components:

- a source of light (laser),
- Several beam-splitters for dividing light rays into multiple subrays. A standard beam-splitter is designed using a half-silvered mirror. For dividing a ray into k subrays we need $k-1$ beam-splitters. An example on how to split a ray in more than 2 subrays is given in Figure 5.12,
- A high speed photodiode for converting light rays into electrical power. The photodiode is placed in the destination node,
- A tool for detecting fluctuations in the intensity of electric power generated by the photodiode (oscilloscope),
- A set of optical fiber cables having certain lengths. These cables are used for connecting nodes and for delaying the signals within nodes. The length of the cables must obey the rules described in section 5.4.5.

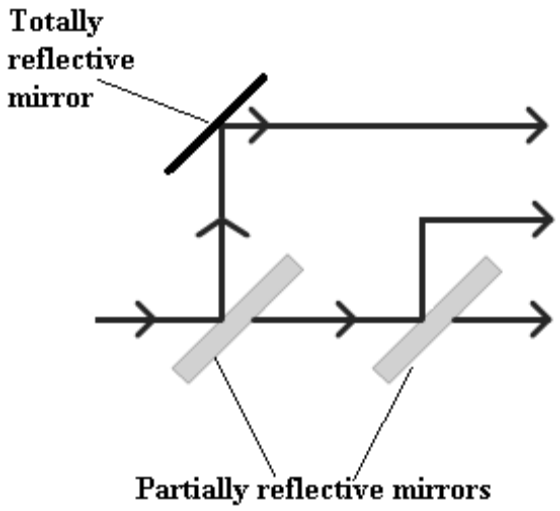

Figure 5.12: The way in which a ray can be split into 3 sub-rays by using 2 beam-splitters.

5.6.2 Amplifying the signal

Beam splitters are used in our approach for dividing a ray in two or more subrays. Because of that, the intensity of the signal is decreasing. In the worst case we have an exponential decrease of the intensity. For instance, in a graph with n nodes, each signal is divided (within each node) into $n-1$ signals. Roughly speaking, the intensity of the signal will decrease n^n times.

This means that, at the destination node, we have to be able to detect very small fluctuations in the intensity of the signal. For this purpose we will use a photomultiplier [14] which is an extremely sensitive detector of light in the ultraviolet, visible and near infrared range. This detector multiplies the signal produced by incident light by as much as 10^8, from which even single photons can be detected.

5.6.3 Improving the device by reducing the speed of the signal

The speed of the light in optic fibers is an important parameter in our device. The problem is that the light is too fast for our measurement tools. We have either to increase the precision of our measurement tools or to decrease the speed of light.

It is known that the speed of light traversing a cable is significantly smaller than the speed of light in the void space. Commercially available cables have limit the speed of the ray wave up to 60% from the original speed of light. This means that we can obtain the same delay by using a shorter cable.

However, this method for reducing the speed of light is not enough for our purpose. The order of magnitude is still the same. This is why we have the search for other methods for reducing that speed. A very interesting solution was proposed in [20] which is able to reduce the speed of light by 7 orders of magnitude and even to stop it [5, 26]. In [5] they succeeded in completely halting light by directing it into a mass of hot rubidium gas, the atoms of which, behaved like tiny mirrors, due to an interference pattern in two control beams.

This could help our mechanism significantly. However, how to use this idea for our device is still an open question because of the complex equipment involved in those experiments [20, 26].

By reducing the speed of light by 7 orders of magnitude we can reduce the size of the involved cables by a similar order (assuming that the precision of the measurement tools is still the same). This will help us to solve larger instances of the problem.

5.6.4 Technical challenges

There are many technical challenges that must be solved when implementing the proposed device. Some of them are:

- Cutting the optic fibers to an exact length with high precision. Failing to accomplish this task can lead to errors in detecting a ray which has passed through each node once.
- Finding a high precision oscilloscope. This is an essential step for solving larger instances of the problem,
- Finding cables long enough so that larger instances of the problem could be solve. This problem might have a simple solution: the internet networks connecting the world cities. It is easy to find cables of hundreds of kilometers connecting distant

cities. This will help us to solve instance of more than 10 nodes. However, this solution introduces a difficulty too: cables of certain lengths must be found or the system must be rescaled in order to fit the existing lengths.

Chapter 6

Conclusions and further work

The thesis has presented several optical devices for solving NP-complete problems. The proposed models are based on the massive parallelism of the light rays.

It has been shown the way in which a light-based devices can be used for solving the following problems:

- Hamiltonian path problem. First device for this problem have been presented in [32, 33],
- Subset-sum problem,
- Unbounded subset-sum problem,
- Exact cover,
- Diophantine equations.

Most of the proposed models are able to solve the considered problems in polynomial time. However, we cannot claim that we have completely answered the P=NP question because the proposed devices require an exponential amount of energy.

Note that each problem requires its own dedicated optical device. The same design cannot be used for 2 different problems. This is why a lot of effort has been spent on designing each model of computation. For some problems multiple devices have been presented. Each model has been analysed in detail. An example on how the device works was given for each problem.

Several details for the physical implementation of each device have been given. The required components are not sophisticated: a continuous source of light, several optical cables, several beam splitters, a photodiode and an osciloscope.

Further work directions will be focused on:

- Implementing the proposed device,
- Cutting new cables each time when a new instance has to be solved is extremely inefficient. This is why finding a simple way to reuse the previously utilized cables is a priority for the proposed system,
- Automate the entire process. This will be the first step for creating an general model of computation based on the ideas proposed in this thesis,

- The proposed devices cannot find the solution. They can only say if there is a solution or not. Moreover, if there are multiple solutions we cannot distinguish them. However, the YES/NO decision problems are still NP-complete problems [15]. Currently several ways for computing the actual solution are investigated,
- Finding other non-trivial problems which can be solved by using the proposed device,
- Finding other ways to introduce delays in the system. The current solution requires cables that are too long and too expensive,
- Using other type of signals instead of light. Possible candidates are electric power and sound.

Bibliography

[1] Abdeldayem H., Frazier D.O., Paley M.S., Witherow W.K., Recent Advances in Photonic Devices for Optical Computing. NASA Marshall Space Flight Center, Space Sciences Laboratory Report, 2000

[2] Adleman L.M., Molecular Computation of Solutions to Combinatorial Problems. Science, 226, 10211024, 1994

[3] Agrawal G.P., Fiber-optic communication systems. Wiley-Interscience; 3^{rd} edition, 2002

[4] Ascheuer N., Hamiltonian path problems in the on-line optimization of flexible manufacturing systems. PhD thesis, TU Berlin, 1995

[5] Bajcsy M., Zibrov A.S., Lukin M.D., Stationary pulses of light in an atomic medium. Nature 426:638-641, 2003

[6] Balcazar J., Daz J., Gabarro J., Structural Complexity, I.Springer, Berlin, DE, 2^{nd} edition, 1995

[7] Borosh I., A Sharp Bound for Positive Solutions of Homogeneous Linear Diophantine Equations, Proceedings of the American Mathematical Society, Vol. 60, No. 1, pp. 19-21, 1976

[8] Černý V, Quantum computers and intractable (NP-Complete) computing problems. Phys. Rev. A, 48:116-119, 1993

[9] Cormen T.H., Leiserson C.E., Rivest R.R., Introduction to algorithms. MIT Press, 1990

[10] Doniach S., Garel H., Orland H., Phase diagram of a semiflexible polymer chain in a θ solvent: Application to protein folding. Journal Of Chemical Physics, 105:1601-1608, 1996

[11] Ekert A., Macchiavello C., An Overview of Quantum Computing. Unconventional Models of Computation, pages 19-44. Springer,1998.

[12] Faist J., Optoelectronics: silicon shines on. Nature, 433:691-692, 2005

[13] Feitelson D.G., Optical computing: A survey for computer scientists, MIT Press, 1998

[14] Flyckt S.O., Marmonier C., Photomultiplier tubes: Principles and applications. Photonis, Brive, France, 2002

[15] Garey M.R., Johnson D.S., Computers and intractability: A guide to NP-Completeness. Freeman & Co, San Francisco, CA, 1979

[16] Greenwood G.W., Finding solutions to NP problems: Philosophical differences between quantum and evolutionary search algorithms. in Proceedings CEC'2001, IEEE Press, 815-822, 2001

[17] Gilmore P.C., Gomory R.E., Multistage Cutting Stock Problems of Two and More Dimensions, Operations Research, Vol. 13, No. 1, pp. 94-120, 1965

[18] Goodman J.W., Architectural development of optical data processing systems. Aust. J. Electr. Electron. Eng. 2:139149, 1982

[19] Hartmanis J., On the weight of computations. Bulletin of the EATCS 55:136-138, 1995

[20] Hau L.V., Harris S.E., Dutton Z., Behroozi C.H., Light speed reduction to 17 meters per second in an ultracold atomic gas. Nature 397:594-598, 1999

[21] Head T., Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors. Bulletin of Mathematical Biology, 49, 737759, 1987

[22] Henkel C., Frisco P., Tengely Sz. An algorithm for SAT without an extraction phase. DNA Computing, Eleventh International Meeting on DNA Based Computers, LNCS 3892, 2005

[23] Hughes R.J., Quantum Computation. Feynman and Computation:Exploring the Limits of Computers, 191-224. Perseus Books, 1998.

[24] Karp R., Reducibility among combinatorial problems, Complexity of Computer Computations, 1972

[25] Lenslet website, www.lenslet.com

[26] Liu C., Dutton Z., Behroozi C.H., Hau L.V. Observation of coherent optical information storage in an atomic medium using halted light pulses. Nature 409:490-493, 2001

[27] Manders K., Adleman L., NP-complete decision problems for binary quadratics, J. Comput. Syst. Sci., vol. 16, pp. 168184, 1978.

[28] MacQueen J. Some methods for classification and analysis of multivariate observations. In LeCam LM, Neyman J, (eds.) Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. University of California press, Berkeley, 281-297, 1967

[29] Murphy N., Woods D., Naughton T.J., Stable Sorting Using Special-Purpose Physical Devices. BCRI Preprint 06/2006, Boole Centre for Research in Informatics, University College Cork, Ireland, 2006

[30] Murphy N., Naughton TJ, Woods D, Henley B, McDermott K, Duffy E, van der Burgt PJM, Woods N., Implementations of a model of physical sorting. From Utopian to Genuine Unconventional Computers workshop, Adamatzky A, Teuscher C, (editors), Luniver Press 79-100, 2006

[31] Naughton T.J., A model of computation for Fourier optical processors. In Lessard RA, Galstian T (editors), Optics in Computing, Proc. SPIE 4089:24-34, 2000

[32] Oltean M., A light-based device for solving the Hamiltonian path problem. Unconventional Computing, Calude C. (et al.) (Eds), LNCS 4135, Springer-Verlag, 217-227, 2006

[33] Oltean M., Solving the Hamiltonian path problem with a light-based device, Natural Computing, Vol 7, Issue 3, 2007 (in press)

[34] Paniccia M., Koehl S., The silicon solution. IEEE Spectrum, IEEE Press, October, 2005

[35] Pisinger D., Dynamic Programming on the word RAM. Algorithmica, 35:128-145, 2003

[36] Reif J.H., Tyagi A., Efficient parallel algorithms for optical computing with the discrete Fourier transform primitive. Applied optics 36(29):7327-7340, 1997

[37] Rong H., Jones R., Liu A., Cohen O., Hak D., Fang A., Paniccia M., A continuous-wave Raman silicon laser. Nature 433:725-728, 2005

[38] Rong H., Liu A., Jones R., Cohen O., Hak D., Nicolaescu R., Fang A., Paniccia M., An all-silicon Raman laser. Nature 433:292-294, 2005

[39] Schultes D., Rainbow Sort: Sorting at the speed of light. Natural Computing. Springer-Verlag, 5(1):67-82, 2005

[40] Sloane N., The on-line encyclopedia of integer sequences. *http* : *//www.research.att.com/* $\sim$ *njas/sequences/A*023758

[41] Teuscher C., Alan Turing. Life and Legacy of a Great Thinker. Springer-Verlag, Berlin, 2003

[42] Tung S.P., Computational complexities of diophantine equations with parameters, J. Algorithms,vol. 8, pp. 324336, 1987

[43] Woods D., Naughton T.J., An optical model of computation. Theoretical Computer Science, 334 (1-3):227-258, 2005

[44] Optical Character Recognition @ Wikipedia,
http : *//en.wikipedia.org/wiki/Optical_character_recognition*