SOLVING NARROW-INTERVAL LINEAR EQUATION SYSTEMS IS NP-HARD

PATRICK THOR KAHL

Department of Computer Science

APPROVED:

Vladik Kreinovich, Chair, Ph.D.

Luc Longpré, Ph.D.

Mohamed Amine Khamsi, Ph.D.

Pablo Arenaz, Ph.D.
Dean of the Graduate School

to my

MOTHER and FATHER

with love

SOLVING NARROW-INTERVAL LINEAR EQUATION SYSTEMS IS NP-HARD

by

PATRICK THOR KAHL

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

July 1996

Acknowledgements

I would like to express my deep-felt gratitude to my advisor, Dr. Vladik Kreinovich of the Computer Science Department at The University of Texas at El Paso, for his advice, encouragement, enduring patience and constant support. He was never ceasing in his belief in me (though I was often doubting in my own abilities), always providing clear explanations when I was (hopelessly) lost, constantly driving me with energy (*Where does he get it?!*) when I was tired, and always, *always* giving me his time, in spite of anything else that was going on. His response to my verbal thanks one day was a very modest, "It's my job." I wish all students the honor and opportunity to experience his ability to perform at that job.

I also wish to thank the other members of my committee, Dr. Luc Longpré of the Computer Science Department and Dr. Mohamed Amine Khamsi of the Mathematics Department, both at The University of Texas at El Paso. Their suggestions, comments and additional guidance were invaluable to the completion of this work. As a special note, Dr. Longpré graciously volunteered to act as my advisor while Dr. Kreinovich was working abroad in Europe. He was extremely helpful in providing the additional guidance and expertise I needed in order to complete this work, especially with regard to the chapter on NP-hard problems and the theory of NP-completeness.

Additionally, I want to thank The University of Texas at El Paso Computer Science Department professors and staff for all their hard work and dedication, providing me the means to complete my degree and prepare for a career as a computer scientist. This includes (but certainly is not limited to) the following individuals:

Dr. Andrew Bernat

> He made it possible for me to have many wonderful experiences I enjoyed while a student, including the opportunity to teach beginning computer science students the basics of UNIX and OpenWindows (something I wish I had been taught when I first started), and the ability to present some of my work at the University of Puerto Rico, Mayagüez Campus.

Dr. Michael Gelfond

> His influence, though unbeknownst to him, was one of the main reasons for my return to UTEP and computer science after my extended leave from school while island hopping in the navy. He taught me many things about computer science—and life. Among the many things he showed me was that there really is science in computer science.

And finally, I must thank my dear wife for putting up with me during the development of this work with continuing, loving support and no complaint. I do not have the words to express all my feelings here, only that I love you, Yulia!

NOTE: This thesis was submitted to my Supervising Committee on the May 31, 1996.

Abstract

Solving systems of linear equations is a common computational problem well known to mathematicians, scientists and engineers. Several algorithms exist for solving this problem. However, when the equations contain *interval coefficients* (i.e., intervals in which the desired coefficient values are known to lie), the problem may not be solvable in any reasonable sense. In fact, it has been shown that the general problem of solving systems of linear equations with interval coefficients is NP-*hard*, i.e., extremely difficult and (it is believed) unsolvable; thus, no feasible algorithm can ever be developed that will solve all particular cases of this problem.

It turns out, though, that the widths of the interval coefficients are quite small in a large number of the linear systems having interval coefficients. This becomes readily apparent when we learn that the intervals typically come from measurements.

Any measurement of a physical quantity is limited by the precision and accuracy of the measuring device. To be of practical use, the measuring devices used in science and industry must be reasonably accurate. This implies that, for the most part, the actual values associated with measurements lie within relatively narrow intervals. Indeed, manufacturers often guarantee the error of their instruments to be very small.

Thus, we desire to look only at *narrow-interval* coefficients when considering the development of an algorithm for solving linear systems with interval coefficients. As there already exists an algorithm that solves most such systems, developing such an algorithm seems indeed promising. Therefore, the goal of this thesis is to answer the following question:

> *Can a feasible algorithm be developed for the general problem of solving systems of linear equations with narrow-interval coefficients?*

We show here that this problem, that of solving systems of linear equations with narrow-

interval coefficients, is NP-hard; thus, we do not consider it possible to develop a feasible algorithm that will solve all particular cases of this problem.

Table of Contents

Chapter 1

Introduction

1.1 Data Processing

Processing engineering and scientific data is one of the main functions of computing. This processing is needed if we are interested in the value of some physical quantity y, and it is difficult or impossible to measure this quantity directly. To find an estimate $\tilde{y}$ of such a quantity y, we measure other quantities $x_1, \ldots, x_n$ that are related to y in a known way, and then compute $\tilde{y}$ from the results $\tilde{x}_1, \ldots, \tilde{x}_n$ of measuring $x_1, \ldots, x_n$. As an example, it is impossible to directly measure the amount of oil in a well; thus, geophysicists measure ultrasound conductivity and other measurable parameters, and then estimate the amount of oil based on the values of these measurements.

In such situations, we have an algorithm $\mathcal{U}_f$ specifying a function f that transforms the values of directly measured quantities $x_1, \ldots, x_n$ into the value of the desired quantity $y = f(x_1, \ldots, x_n)$. To estimate the value of the desired quantity, we apply this algorithm $\mathcal{U}_f$ to the measurement results $\tilde{x}_1, \ldots, \tilde{x}_n$ and compute the estimate $\tilde{y} = f(\tilde{x}_1, \ldots, \tilde{x}_n)$.

1.2 Estimating the Accuracy of the Results of Data Processing

In many cases, the algorithm specifying a function f used in data processing is *precise* in the sense that in the ideal situation where we use the exact (actual) values of x_i as input, the algorithm returns the exact value of the desired quantity y.

In reality, however, measurements are almost never absolutely exact; i.e., the results $\tilde{x}_i$ of direct measurements are (in general) somewhat different from the actual values x_i. Because of these *measurement errors* $\Delta_{x_i} = \tilde{x}_i - x_i$, the result $f(\tilde{x}_1, \ldots, \tilde{x}_n)$ of data processing is, in general, different from the actual value $y = f(x_1, \ldots, x_n)$ of the desired quantity.

It is clear that we need to know how accurate the results of data processing are in order to make a qualitative decision about these results. In other words, we need to know what the possible values of the error $\Delta_y = \tilde{y} - y$ are as a result of data processing. For example, whether we decide to drill for oil depends on the estimate of the amount of oil in the area *and* on the accuracy of this estimate. Suppose we require a minimum of 50 million tons of oil to make drilling profitable. If we compute an estimate of 100 million tons of oil in an area by processing data obtained from preliminary measurements, should we decide to start drilling? If this estimate is very inaccurate (e.g., an absolute accuracy of 100 million tons would mean that the actual value could be any value from 0 to 200 million tons), then further measurements are warranted if they will lead to a more accurate estimate.

The desire to solve a particular case of the general problem of determining the accuracy of such indirect measurements is the motivation for the work in this thesis.

1.3 Traditionally, Statistical Methods Are Used

Since the error in the computed value of y comes from measurement errors, we must use some information about the possible values of the measurement errors in order to determine the maximum possible error (or the likelihood of different possible values of the error) in the computed value of y.

In some cases, for each measuring device we know not only the values of the measurement errors that are possible, but also the *probabilities* of the different possible values. In such cases, we can use statistical methods to estimate the statistical characteristics of the error in the computed value of y. These are the methods traditionally used by engineers and scientists (see, e.g., [14]).

1.4 Statistical Methods Are Not Always Applicable

In spite of the success in the use of statistical methods for some special cases, in many real-life situations the probabilities of the different error measurement values are not known. These are common situations in manufacturing and robotics, for example, where various sensors are used. In such situations, the only information we may have is the set of possible measurement error values; thus, the traditional statistical methods are not applicable. For a typical example, many measuring instruments used in industry have a manufacturer-provided *guaranteed* absolute accuracy, i.e., a number $\Delta^{max} > 0$ that the absolute value of the measurement error cannot exceed.

Consequently, if a measured value is $\tilde{x}$ and the guaranteed absolute accuracy of the measuring device is Δ_x^{max}, then the actual value x is guaranteed to lie in the *interval* $[\tilde{x} - \Delta_x^{max}, \tilde{x} + \Delta_x^{max}]$. In the following text, we will denote this interval as $[x^-, x^+]$ or simply as $\mathbf{x}$. As an example, if a thermometer has an accuracy of $\Delta_T^{max} = 2^\circ F$, and it reads $\tilde{T} = 40^\circ F$, the actual value T of the temperature lies between $\tilde{T} - \Delta_T^{max} = 38^\circ F$ and $\tilde{T} + \Delta_T^{max} = 42^\circ F$, i.e., $T \in [38^\circ F, 42^\circ F]$.

1.5 Interval Computations

In such situations, since we only know the *intervals* $[x_i^-, x_i^+]$ that contain the input data x_i, we cannot compute the exact value of $y = f(x_1, \ldots, x_n)$; we can only compute the *interval* $[y^-, y^+] = \{f(x_1, \ldots, x_n) | x_1 \in [x_1^-, x_1^+], \ldots, x_n \in [x_n^-, x_n^+]\}$ of possible values of the desired quantity y. This interval is usually denoted by $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

Computing the bounds y^- and y^+ is an example of *interval computations*. The history of interval computations is usually said to have started with the pioneer work of R. E. Moore. In 1959, while working for Lockheed Space and Missile Corporation, Moore wrote a technical report containing a description of what we now call interval computations (see [11]). For a survey of the modern state of interval computations, see, e.g., [4].

1.6 A Typical Data Processing Problem: Solving Linear Systems

Occasionally, there are known explicit formulas for describing a desired value y directly in terms of measurable parameters $x_1, \ldots, x_n$. For example, if we use Ohm's law $V = I \cdot R$, we can compute the voltage ($y = V$) by simply measuring the current ($x_1 = I$), measuring the resistance ($x_2 = R$), and applying the simple formula $y = x_1 \cdot x_2$.

In most cases, however, the known dependency between x_i and y is *implicit*: we have some equations that connect x_i and y, but we do not have exact formulas that describe y in terms of x_i. These equations typically include, in addition to x_i and y, some other quantities that we may not be directly interested in, but that we have to compute if we want to find y. For example, in the geophysical problem, we may have to determine certain geophysical parameters like the density of the rocks at different depths before we are able to estimate the amount of oil.

Thus, we may have several unknown quantities $y_1, \ldots, y_q$ that are related to the directly measured quantities $x_1, \ldots, x_n$. Usually, each measurement x_i leads to a new restriction on the values on the unknown quantities, i.e., to a new equation $F_i(y_1, \ldots, y_q, x_i) = 0$ that relates the unknown values y_j and the measurement result x_i. Therefore, we have (in general) a system of non-linear equations from which we must determine the values y_j.

Often, we know the approximate values $y_j^{(0)}$ of the unknown quantities y_j. In such situations, we need only compute differences $\Delta_{y_j} = y_j - y_j^{(0)}$. In terms of these differences, the equations take the form

$$F_i(y_1^{(0)} + \Delta_{y_1}, \ldots, y_q^{(0)} + \Delta_{y_q}, \tilde{x}_i) = 0.$$

Since the differences are small, if each function F_i is smooth and continuous then we can expand every function F_i into Taylor series, neglect terms that are quadratic or of higher order in Δ_{y_j}, and get a system of equations that are *linear* in terms of the unknowns. In such cases, we must solve a system of linear equations in order to find (close approximations

for) the unknown values.

Thus, we see that solving systems of linear equations is a typical example of data processing. In this thesis, we will be analyzing the problem of error estimation for such data processing algorithms.

In order to avoid possible confusion, please note that in the previous text, we followed the standard denotation of measurement analysis and denoted by x_i the values of the directly measured quantities. For linear equations, x usually denotes an unknown, i.e., a desired result of data processing. In the following text, we will only be considering systems of linear equations, and thus, x will denote an unknown in a system of linear equations hereafter. In these denotations, a general system of linear equations will take the following form:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i. \tag{1.1}$$

1.7 Interval Linear Systems

The coefficients of linear equations (i.e., a_{ij} and b_i) often come as results of measurements. Let us assume that we only know the interval of possible values associated with each measurement result; i.e., traditional statistical methods are not applicable. From this we conclude that

- we only know the intervals $\mathbf{b}_i$ that contain the actual values of b_i, and
- we only know the intervals $\mathbf{a}_{ij}$ that contain the actual values of a_{ij}.

Thus, we have a system of linear equations with interval coefficients. We call such a system a *system of interval linear equations.*

Since the only thing we know about the actual values of the measured quantities is that they lie within some intervals $\mathbf{a}_{ij}$ and $\mathbf{b}_i$, we can rewrite system (1.1) as

$$\sum_{j=1}^{n} \mathbf{a}_{ij} x_j = \mathbf{b}_i. \tag{1.2}$$

We say that a tuple $(x_1, \ldots, x_n)$ is a *solution* to this system (1.2) of interval linear equations if for some $a_{ij} \in \mathbf{a}_{ij}$ and $b_i \in \mathbf{b}_i$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i$$

for all $i = 1, \ldots, m$. The set of all solutions to system (1.2) is usually denoted by X.

Note that for different solutions $(x_1, \ldots, x_n) \in X$ we may get different values of x_j; therefore, since we cannot find a unique value for x_j, we need to find the *interval of possible values of* x_j for all $j = 1, \ldots, n$. In other words, we need to find the interval $\mathbf{x}_j = [x_j^-, x_j^+]$ where

$$x_j^- = \min\{x_j \,|\, (x_1, \ldots, x_n) \in X\}$$

and

$$x_j^+ = \max\{x_j \,|\, (x_1, \ldots, x_n) \in X\}$$

for all $j = 1, \ldots, n$. By *solving* a system of interval linear equations, we mean computing the bounds x_j^- and x_j^+ of the interval $\mathbf{x}_j$ for all $j = 1, \ldots, n$.

1.8 Example

Let us take a simple system of interval linear equations

$$\mathbf{a}_{11} x_1 = \mathbf{b}_1$$

with one unknown ($n = 1$) and one measurement ($m = 1$), for which:

- as a result of measuring b_1, we get $\tilde{b}_1 = 2.5$;
- as a result of measuring a_{11}, we get $\tilde{a}_{11} = 1.5$;
- the absolute accuracy of both measurements is $\Delta_a^{max} = \Delta_b^{max} = 0.5$.

This means that $\mathbf{b}_1 = [2.5 - 0.5, 2.5 + 0.5] = [2, 3]$ and $\mathbf{a}_{11} = [1.5 - 0.5, 1.5 + 0.5] = [1, 2]$. Therefore, the system (1.2) reduces to the single interval linear equation

$$[1, 2] x_1 = [2, 3].$$

To solve this system, we need to find the solution set X to this problem; i.e., we need to find the endpoints x_1^- and x_1^+ of the interval $\mathbf{x}_1$ such that $x_1 \in \mathbf{x}_1$. Here $x_1 = b_1/a_{11}$ where $a_{11} \in \mathbf{a}_{11}$ and $b_1 \in \mathbf{b}_1$. The value of x_1 is smallest when b_1 takes the smallest possible value in $\mathbf{b}_1$ and a_{11} takes the largest possible value in $\mathbf{a}_{11}$. Similarly, the value of x_1 is largest when b_1 takes the largest possible value in $\mathbf{b}_1$ and a_{11} takes the smallest possible value in $\mathbf{a}_{11}$. Hence,

$$x_1^- = \frac{\min(\mathbf{b}_1)}{\max(\mathbf{a}_{11})} = \frac{\min([2,3])}{\max([1,2])} = \frac{2}{2} = 1$$

and

$$x_1^+ = \frac{\max(\mathbf{b}_1)}{\min(\mathbf{a}_{11})} = \frac{\max([2,3])}{\min([1,2])} = \frac{3}{1} = 3.$$

Thus, $\mathbf{x}_1 = [1,3]$ and $X = \{(x_1)|x_1 \in [1,3]\}$.

1.9 Solving Interval Linear Systems Is NP-Hard

There exist many useful algorithms for solving interval linear systems (see, e.g., [13]); however, all these algorithms face the following problem: there are systems of linear interval equations of size $n \times m$, for which the running time grows exponentially with n. As a result, even for reasonably small n, the running time can be longer than the estimated lifetime of the universe!

It was later shown that this problem is not caused by the imperfection of the algorithms, but by the computational complexity of the problem itself; i.e, it was shown (see [5]) that this problem is *computationally intractable* (i.e., NP-*hard*) (see [2]). Crudely speaking, this means that unless *all* possible problems can be solved in reasonable time[1], any algorithm for solving this problem must go exponential in some case. We will look at this in more detail in later chapters.

[1]This highly improbable hypothesis is usually denoted as "P=NP."

1.10 What If Intervals Are Narrow?

Most measuring devices used in industry are reasonably accurate, and thus, the intervals associated with measurements are almost always narrow. It would be nice if we could write a computer program that could solve interval linear systems with narrow intervals since this appears to be the sort of problem typically seen in data processing. If we then restrict ourselves to narrow intervals only, will the problem of solving interval linear systems still be computationally intractable?

Of particular relevance to this question is the fact that for narrow intervals there exists an algorithm that solves *almost all* interval linear systems in reasonable time (see [6, 9]). We will look at this in more detail in Chapter 4.

1.11 Our Result

In spite of the above optimistic result, we will show that *the problem of solving interval linear systems with narrow intervals is computationally intractable (i.e.,* NP*-hard).* In other words, (unless P=NP) there exists no algorithm that can solve every narrow-interval linear system of reasonable size in a reasonable amount of time. Thus, it is believed that no one will ever be able to write a computer program that can solve narrow-interval linear systems in general, no matter how fast computers of the future become. We will look at this problem in more detail in later chapters. The proof of this result can be found in Chapter 5.

Chapter 2

NP-Hard Problems

In this chapter we provide a brief introduction to the theory of NP-completeness and provide the notation and definitions necessary to accurately describe what we mean when we say that a problem is NP-hard. The theory of NP-completeness was first developed by Cook (see [1]) and Karp (see [3]), and later, independently by Levin (see [10]). We also provide here the theorems that form the basis for the proof of the main result of this thesis. This material is inspired by the well-known book *Computers and Intractability: A Guide to the Theory of* NP-*Completeness* by M. R. Garey and D .S. Johnson (see [2]). Material on NP-completeness can also be found in basic textbook on theory of computation.

2.1 What Is a Problem?

Informally, a *problem* is a general *proposition* (i.e., question proposed) for which we desire a satisfactory *solution*. Usually, a problem possesses one or more *parameters* (i.e., free variables) whose values are left unspecified. We describe a problem by defining:

- the problem parameters, and
- the properties of a desired solution.

For example, finding the prime factors of a natural number $n > 1$ is a problem called *prime factorization*. The problem is defined as follows:

1. There is one parameter, a natural number $n > 1$.

2. A solution is the set $\{p_1, \ldots, p_n\}$ of all prime numbers p_i such that

$$n = p_1^{m_1} \cdot \ldots \cdot p_n^{m_n}$$

where $n > 1$, $m_i > 0$, and n and m_i are natural numbers.

In order to specify a particular *instance* of the problem, we need to specify the values of all problem parameters. For example, we can specify an instance of the prime factorization problem by providing the parameter $n = 60$. Since $60 = 2^2 \cdot 3 \cdot 5$, the solution to this instance of the problem is $\{2, 3, 5\}$.

Informally, a *decision problem* is any general problem with only two possible solutions: "yes" or "no." For any decision problem Π, we denote by D_Π the *set of all instances of* Π, and by $Y_\Pi \subseteq D_\Pi$ we denote the *set of "yes" instances of* Π (i.e., instances of Π where the solution is "yes"). For example, the problem of determining whether a number is prime is a decision problem whereas prime factorization is not.

Definition 1 A *search problem* Π consists of a set D_Π of finite objects called *instances* and, for each instance $I \in D_\Pi$, a set $S_\Pi[I]$ of finite objects called *solutions* for I.

Definition 2 An algorithm is said to *solve* a search problem Π if, given as input any instance $I \in D_\Pi$, it returns "no" as a solution whenever $S_\Pi[I] = \emptyset$, and some solution $s \in S_\Pi[I]$ otherwise.

Definition 3 A *decision problem* Π is a search problem where

$$S_\Pi[I] = \begin{cases} \{\text{"yes"}\} & \text{if } I \in Y_\Pi \\ \emptyset & \text{otherwise.} \end{cases}$$

Definition 4 For any alphabet Σ, a subset of strings $L \subseteq \Sigma^*$ is called a *language over the alphabet* Σ.

Definition 5 An *encoding scheme* e for a problem Π is a fixed method for mapping each instance I of Π to some appropriate string $x \in \Sigma^*$ for some fixed alphabet Σ.

Example. Consider the decision problem of determining whether a given number n is prime. Specifying the value of n provides a particular instance of this problem. Let us now look at two different possible encoding schemes for the problem:

1. An encoding scheme e_{asc} that maps any instance of the problem to an ASCII character file (i.e., one long string of ASCII characters that is, e.g., the source code for a computer program representing this instance).

2. An encoding scheme e_{bin} that maps the same instance to a binary file (i.e., one long string of 0's and 1's that is, e.g., the object code for a the computer program representing this instance).

Notice that both encoding schemes provide a method of representing the same instances of the same problem, even though they use different alphabets.

Since it is easy to translate any reasonable encoding into any other reasonable encoding, we will not concern ourselves with any particular encoding scheme in the contents of this thesis. We require only that an encoding scheme be reasonable (e.g., is easily decoded, does not excessively pad, does not provide problem solution as part of the encoding, etc.) (For a more detailed discussion of what is meant by "reasonable" see [2]).

Definition 6 For any decision problem Π and for any encoding scheme e for Π that (for some fixed alphabet Σ) maps to Σ^*, the *language L associated with Π and e* is the set of "yes" instances of Π encoded under e, i.e.,

$$L[\Pi, e] = \{x \in \Sigma^* | x \text{ is the encoding under } e \text{ of an instance } I \in Y_\Pi\}.$$

2.2 Turing Machines

We now need to discuss the models of computation that we will be using, namely Turing machines. For a formal definition, the reader is referred to any basic textbook on theory of computation. Informally, a Turing machine is a computing device that has one read/write

tape (memory) and a finite control (the program). The machine accesses the tape with a read/write head positioned on the tape. In one step, the program can read the symbol at the current position, optionally write a symbol, and optionally move the head backward or forward one position. The result of any computation is the contents of the tape when the Turing machine reaches a halt state. If the Turing machine only gives "yes" or "no" answers, then the *set accepted by the Turing machine* is the set of inputs that lead the machine to a "yes" answer. The *time* taken by a Turing machine on given input x is defined to be the number of steps the Turing machine takes to reach a halt state, starting with x on the tape.

A nondeterministic Turing machine is one that (possibly) has more than one next configuration from the current configuration. This leads to many possible computations from the same initial configuration, usually modelled as a *tree of possible computations*. The language L accepted by a nondeterministic Turing machine is the set of all strings x such that the Turing machine has at least one possible computation that leads to a "yes" answer with x as its input. The *time* taken by a nondeterministic Turing machine with given input x is defined to be the number of steps it takes in the longest possible computation for x before reaching a halt state. (There are actually many ways to define time for a nondeterministic Turing machine, but for our purposes, all are equivalent.)

An oracle Turing machine is a Turing machine with the added capability that it can query an *oracle* in order to determine set membership. To be more precise, let us start by choosing a set A to be used as the oracle. The oracle Turing machine has a special query tape where it can write strings that are queries to the oracle. After writing a string x on the query tape, the oracle Turing machine enters a special state $q_?$ called the *query state* from which it goes automatically to a new state, either q_{no} or q_{yes} depending on whether $x \in A$. This sequence of state transformations is defined to be only one step in a computational process. A common way of looking at an oracle Turing machine is to see an oracle query as if the program calls a hypothetical procedure that determines membership in A, but only gets charged one step for the call. An oracle Turing machine can be deterministic or

nondeterministic and its time is defined in like fashion.

2.3 The Classes P and NP

The complexity classes P and NP are based on polynomial-time bounded deterministic and nondeterministic Turing machines. By *polynomial time* we mean that the number of steps is bounded by some polynomial of the input size; therefore, a polynomial-time bounded Turing machine is one in which the number of steps required before reaching a halt state is bounded by a polynomial of the input size.

Definition 7 The *class* P is the set of all languages L such that there exists a polynomial-time deterministic Turing machine accepting L.

Definition 8 The *class* NP is the set of all languages L such that there exists a polynomial-time nondeterministic Turing machine accepting L.

The class NP was first defined by Cook in the early 1970's (see [1]). With its introduction came one of the most important open problems in theoretical computer science: whether P=NP. Most computer scientists believe that P≠NP. One reason for this is that simulating a nondeterministic Turing machine with a deterministic Turing machine appears to require exploring the whole tree of possible computations, which would require an exponential amount of time in (at least) some cases. However, no proof that P≠NP is known, nor does it seem likely that one will be found in the near future.

2.4 NP-Complete, NP-Hard, and Reductions

One important notion about the class NP is the existence of *complete* problems. Crudely speaking, complete problems for a class are the hardest problems in the class. Thus, NP-complete problems are the hardest problems in the class NP. Also of importance is the notion of NP-hardness. Informally, NP-hard problems are problems that are at least as

hard as NP-complete problems. Cook (see [1]) proved that the problem of satisfiability of Boolean formulas (known as SAT) is an NP-complete problem, and Karp (see [3]) showed that many other important problems are also NP-complete by use of *reductions* (generically denoted as $\leq_r$). Completeness is formally defined in terms of reductions. We will now define two types of reductions, both dealing with the notion of polynomial time. The *time* that an algorithm (e.g., a reduction) requires in order to solve a problem is defined to be the number of steps as a function of the size of the input. The size here, of course, depends on the encoding scheme chosen. Recall that polynomial time means that the number of steps is bounded by some polynomial of the input size. As an example, if we say that an n^3 algorithm $\mathcal{U}$ exists for solving a problem Π of size n, we mean that $\mathcal{U}$ will take no more than n^3 steps in order to solve Π. Therefore, we say that $\mathcal{U}$ is a polynomial-time algorithm. Though not explicitly stated as being algorithms, the following two definitions refer to polynomial-time algorithms.

Definition 9 A *polynomial-time many-one reduction* $\leq_m^p$ (or *polynomial transformation*) from a language $L_1 \subseteq \Sigma_1^*$ to a language $L_2 \subseteq \Sigma_2^*$ (denoted $L_1 \leq_m^p L_2$) is a polynomial-time computable function $f : \Sigma_1^* \to \Sigma_2^*$ such that

$$\forall x \in \Sigma_1^* \ (x \in L_1 \iff f(x) \in L_2).$$

Definition 10 A *polynomial-time Turing reduction* $\leq_T^p$ from a language $L_1 \subseteq \Sigma_1^*$ to a language $L_2 \subseteq \Sigma_2^*$ (denoted $L_1 \leq_T^p L_2$) is a polynomial-time oracle Turing machine that accepts L_1 using L_2 as its oracle.

We now are ready to formally define NP-hardness and NP-completeness. In the following, $\leq_r^p$ stands for either many-one or Turing polynomial-time reductions.

Definition 11 A language L is NP-*hard under* $\leq_r^p$ *reductions* (alternatively, $\leq_r^p$-*hard for* NP) if

$$\forall L' \in \text{NP} \ (L' \leq_r^p L).$$

Definition 12 A language L is NP-*complete under* $\leq_r^p$ *reductions* (alternatively, $\leq_r^p$-*complete for* NP) if $L \in$ NP and L is NP-hard under $\leq_r^p$ reductions.

The following theorem says that unless P=NP, no one will ever be able to find a polynomial-time algorithm to solve any NP-hard problem.

Theorem 1 *If a language* L *is* NP*-hard under Turing reductions and* $L \in$ P*, then* P=NP*.*

Corollary *If a language* L *is* NP*-hard under many-one reductions and* $L \in$ P*, then* P=NP*.*

Thus, if someone some day finds a polynomial-time algorithm for any NP-hard problem, then this algorithm could be used to solve any NP-complete problem in polynomial time, which would solve many important open problems (and would also be very useful). However, the current wisdom is that this will never be possible. This means that proving that a problem is NP-hard is strong evidence that no polynomial-time algorithm exists for that problem. For this reason, NP-hard problems are said to be *computationally intractable*.

Once we have a collection of problems that are proven to be NP-hard, these problems can be used to prove other problems are NP-hard as well via reductions. This is the method most used to prove NP-hardness, and it is precisely how we prove our main result. This method is formalized by the following theorem:

Theorem 2 *If* L *is* NP*-hard under* $\leq_r^p$ *reductions and* $L \leq_r^p L'$ *then* L' *is* NP*-hard under* $\leq_r^p$ *reductions.*

It should be noted that NP-completeness is usually defined using many-one reductions, but philosophically, it may make more sense to define NP-completeness using Turing reductions. This distinction is important because definitions based on different reductions are believed not to be identical; however, since many-one reductions are a special case of Turing reductions, proving that a language is many-one-hard for NP implies that it is Turing-hard for NP.

Chapter 3

Solving Interval Linear Systems Is NP-Hard: Known Result

In this chapter we will define formally what it means to solve a system of interval linear equations and present a negative result related to such systems that was first discovered by Kreinovich, Lakeyev and Noskov in the early 1990's.

3.1 Definitions

Definition 13 An expression of the form

$$\sum_{j=1}^{n}[a_j^-, a_j^+]x_j = [b^-, b^+] \tag{3.1}$$

is called an *interval linear equation* with n unknowns, $x_1, \ldots, x_n$.

Definition 14 Let

$$\sum_{j=1}^{n}[a_{ij}^-, a_{ij}^+]x_j = [b_i^-, b_i^+] \tag{3.2}$$

for $i = 1, \ldots, m$ be a *system of interval linear equations*. The tuple $(x_1, \ldots, x_n)$ is called a *solution* of the system (3.2) if there exist values $a_{ij} \in [a_{ij}^-, a_{ij}^+]$ and $b_i \in [b_i^-, b_i^+]$ such that

$$\sum_{j=1}^{n} a_{ij}x_j = b_i$$

for all $i = 1, \ldots, m$.

Definition 15 By *solving* a system (3.2) of interval linear equations, we mean computing the bounds x_j^- and x_j^+ where

$$x_j^- = \min\{x_j | (x_1, \ldots, x_n) \text{ is a solution to system (3.2)}\}$$

and

$$x_j^+ = \max\{x_j | (x_1, \ldots, x_n) \text{ is a solution to system (3.2)}\}$$

for all $j = 1, \ldots, n$.

To clarify the problem, consider a (hypothetical) feasible algorithm that can be used to solve any system of interval linear equations. Such an algorithm would require the following input to be given:

- the number of equations m,
- the number of unknowns n,
- the endpoints a_{ij}^- and a_{ij}^+ for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$, and
- the endpoints b_i^- and b_i^+ for all $i = 1, \ldots, m$.

Such input would define a particular system of the form (3.2). For the algorithm to be feasible it must be able, given any particular system defined by the input, to compute in polynomial time the following output:

- the endpoints x_j^- and x_j^+ such that

  $$x_j^- = \min\{x_j | (x_1, \ldots, x_n) \text{ is a solution to the given system}\}$$

  and

  $$x_j^+ = \max\{x_j | (x_1, \ldots, x_n) \text{ is a solution to the given system}\}$$

  for all $j = 1, \ldots, n$.

3.2 Theorem

Theorem 3 *The problem of solving systems of interval linear equations is computationally intractable (*NP*-hard).*

This theorem was proven (see [5]) in 1993 by Kreinovich, Lakeyev and Noskov by a reduction from SAT[1].

3.3 What If Intervals Are Narrow?

The proof of theorem 3 uses intervals that are of type $[0, 1]$ that correspond, in measurement terms, to measuring a value of magnitude 0.5 with an absolute accuracy of 0.5, or a relative accuracy of 100%. Such measurements, though possible, are not very accurate. Most measuring devices of practical use in science and industry are far more accurate, and hence, intervals are much narrower. The natural question is:

> *If we restrict ourselves to systems with narrow intervals only, will the problem of solving systems of interval linear equations still be* NP*-hard?*

This question was first analyzed by Lakeyev and Kreinovich in 1995 (see [9]). In the next chapter we will present the result of that analysis.

[1]Recall from chapter 2 that SAT is the problem of satisfiability of Boolean formulas.

Chapter 4

Solving Most Narrow-Interval Linear Systems Is Easy: Known Result

In this chapter we will define what we mean by *narrow intervals* and present a positive result concerning systems of interval linear equations having such narrow intervals. This result was first discovered by Lakeyev and Kreinovich in the mid 1990's.

4.1 Definitions

Definition 16 We say that the number $\tilde{x}$ represents a number x with *absolute accuracy* $\Delta > 0$ if $|x - \tilde{x}| \leq \Delta$.

Definition 17 By *absolute half-width* of the interval $\mathbf{x} = [x^-, x^+]$, we mean the smallest number $\Delta > 0$ for which every number in $\mathbf{x}$ is represented with an absolute accuracy Δ by $\tilde{x} = \dfrac{x^- + x^+}{2}$.

Proposition It can be seen that the absolute half-width of $\mathbf{x}$ is

$$\max_{x \in [x^-, x^+]} |x - \tilde{x}| = \frac{x^+ - x^-}{2}.$$

Definition 18 By *absolute width* W of an interval $\mathbf{x} = [x^-, x^+]$, we mean twice the absolute half-width of $\mathbf{x}$, i.e.,

$$W([x^-, x^+]) = x^+ - x^-.$$

Definition 19 We say that an interval $\mathbf{x}$ is Δ-*narrow in the sense of absolute accuracy* if $W(\mathbf{x}) \leq \Delta$.

Definition 20 Let $\varepsilon > 0$ be a real number, let $D \subseteq R^N$ be a closed and bounded set of positive N-dimension volume $V(D) > 0$, and let $P(x)$ be a property that is true for some points $x \in D$. We say that $P(x)$ is true for (D, ε)-*almost all* x if

$$\frac{V(\{x \in D | \neg P(x)\})}{V(D)} \leq \varepsilon.$$

Definition 21 Let $\eta > 0$ be a real number. We say that intervals

$$[r_1 - d_1, r_1 + d_1], \ldots, [r_n - d_n, r_n + d_n]$$

are η-close to intervals

$$[\tilde{x}_1 - \Delta_1, \tilde{x}_1 + \Delta_1], \ldots, [\tilde{x}_n - \Delta_n, \tilde{x}_n + \Delta_n]$$

if $|r_i - \tilde{x}_i| \leq \eta$ and $|d_i - \Delta_i| \leq \eta$ for all i.

Definition 22 Let $\mathcal{U}$ be an algorithm that solves some systems of interval linear equations, and let $\eta > 0$ be a real number. We say that an algorithm $\mathcal{U}$ is η-*exact* for the interval matrices $\mathbf{a}_{ij}$ and $\mathbf{b}_i$ if for every interval matrix $\mathbf{a}'_{ij}$ and $\mathbf{b}'_i$ that are η-close to $\mathbf{a}_{ij}$ and $\mathbf{b}_i$, the algorithm $\mathcal{U}$ returns the exact solution to the system

$$\sum_{j=1}^{n} \mathbf{a}'_{ij} x_j = \mathbf{b}'_i.$$

Definition 23 We say that an algorithm $\mathcal{U}$ is *almost always exact for narrow input intervals* if for every closed and bounded set $D \subseteq R^N (N = n \cdot m + n)$ there exist $\varepsilon > 0$, $\Delta > 0$ and $\eta > 0$ such that, for (D, ε)-almost all $\tilde{a}_{ij}$ and $\tilde{b}_i$, if all input intervals $\mathbf{a}_{ij}$ and $\mathbf{b}_i$ (containing $\tilde{a}_{ij}$ and $\tilde{b}_i$ respectively) are Δ-narrow (in the sense of absolute accuracy), then the algorithm $\mathcal{U}$ is η-exact for $\mathbf{a}_{ij}$ and $\mathbf{b}_i$.

4.2 Theorem

Theorem 4 *There exists a feasible (polynomial-time) algorithm $\mathcal{U}$ that is almost always exact for narrow input intervals.*

This theorem was proven in 1995 by Lakeyev and Kreinovich (see [9]).

4.3 Open Problem

Theorem 4 says that we can have a feasible algorithm that solves *almost all* narrow-interval linear equation systems, but it does not say whether we can solve *all* of them in reasonable time. Thus, there still remains an open question:

> *Can a feasible algorithm be developed for the general problem of solving systems of linear equations with narrow-interval coefficients?*

The answer to this open question is the main concern of this thesis.

We will show that the problem of solving all narrow-interval linear equation systems is NP-hard; moreover, we will show that the problem is NP-hard not only for intervals that are narrow in the sense of absolute accuracy, but also in the sense of relative accuracy.

Chapter 5

Solving Narrow-Interval Linear Systems Is NP-Hard: New Result

In this chapter we present the main result upon which this thesis is centered—a new theorem and a proof for it based upon the reduction[1] of a system of (arbitrary) interval linear equations to a system of narrow-interval linear equations.

5.1 Definitions

Definition 24 We say that the (non-zero) number $\tilde{x}$ represents a number x with a *relative accuracy* $\delta > 0$ if $\dfrac{|x - \tilde{x}|}{|\tilde{x}|} \leq \delta$.

Definition 25 By *relative half-width* of the interval $\mathbf{x} = [x^-, x^+]$ where $0 \notin [x^-, x^+]$, we mean the smallest number $\delta > 0$ for which every number in $\mathbf{x}$ is represented with a relative accuracy δ by $\tilde{x} = \dfrac{x^- + x^+}{2}$.

Proposition It can be seen that the relative half-width of $\mathbf{x}$ where $0 \notin [x^-, x^+]$ is

$$\max_{x \in [x^-, x^+]} \frac{|x - \tilde{x}|}{|\tilde{x}|} = \frac{x^+ - x^-}{2|\tilde{x}|}.$$

Definition 26 By *relative width* W^{rel} of an interval $\mathbf{x} = [x^-, x^+]$ where $0 \notin [x^-, x^+]$, we mean twice the relative half-width of $\mathbf{x}$, i.e.,

$$W^{rel}([x^-, x^+]) = \frac{x^+ - x^-}{|\tilde{x}|}.$$

[1] The reduction used is called a *polynomial-time one-one reduction*, a special case of polynomial-time many-one reductions.

Definition 27 We say that an interval $\mathbf{x}$ is *δ-narrow in the sense of relative accuracy* if $W^{rel}(\mathbf{x}) \leq \delta$.

Definition 28 We say that an interval $\mathbf{x}$ is *δ-narrow* if it is both δ-narrow in the sense of absolute accuracy and δ-narrow in the sense of relative accuracy.

5.2 Theorem

Theorem 5 *For every $\delta > 0$, the problem of solving systems of interval linear equations with δ-narrow intervals is computationally intractable (*NP*-hard).*

Corollary 1 *For every $\Delta > 0$, the problem of solving systems of interval linear equations with intervals Δ-narrow in the sense of absolute accuracy is computationally intractable (*NP*-hard).*

Corollary 2 *For every $\delta > 0$, the problem of solving systems of interval linear equations with intervals δ-narrow in the sense of relative accuracy is computationally intractable (*NP*-hard).*

5.3 Proof

5.3.1 Part I: Reduction of Interval Linear System to Narrow-Interval Linear System

We have seen that the general problem of solving interval linear equation systems (3.2) is NP-hard. We now intend to show that this general problem can be reduced to the problem of solving a system of interval linear equations with δ-narrow intervals.

Let us start with an arbitrary interval linear equation system (3.2). To reduce it to a δ-narrow interval linear equation system, we introduce new variables w_i, y_{ij} and z_{ij} for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$. For the enlarged list of $m + n + 2mn$ variables (i.e., x_j, w_i, y_{ij} and z_{ij}) we introduce the following new system of $m + n + 2mn$ δ-narrow interval linear equations[2]:

$$\sum_{j=1}^{n} y_{ij} + \sum_{j=1}^{n} [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}] z_{ij} + [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}] w_i = c_i, \tag{5.1}$$

$$w_i = \gamma_i, \tag{5.2}$$

$$y_{ij} - \mu_{ij} x_j = 0, \tag{5.3}$$

$$z_{ij} - \nu_{ij} x_j = 0, \tag{5.4}$$

where $\delta > 0$ and the numerical (non-interval) coefficients c_i, $\gamma_i \geq 0$, μ_{ij} and $\nu_{ij} \geq 0$ will be chosen in such a way that for every solution of this δ-narrow interval linear equation system (5.1)–(5.4) we have

$$c_i - [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}] w_i = [b_i^-, b_i^+] \tag{5.5}$$

and

$$y_{ij} + [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}] z_{ij} = [a_{ij}^-, a_{ij}^+] x_j \tag{5.6}$$

for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

[2]For clarity, non-interval coefficients can be thought of as intervals of zero width; e.g., the coefficient for each variable y_{ij} is $[1, 1]$ and each coefficient c_i is equivalent to the interval $[c_i, c_i]$.

Let us first find the values for c_i and γ_i. From equation (5.2) we know that $w_i = \gamma_i$. By substituting this expression into equation (5.5) we get

$$c_i - [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]\gamma_i = [b_i^-, b_i^+].$$

Intervals are equal *if and only if* their endpoints coincide. Since we are looking for a solution with $\gamma_i \geq 0$, the equations for the endpoints take the following forms:

$$c_i - (1 - \frac{\delta}{2})\gamma_i = b_i^+ \tag{5.7}$$

and

$$c_i - (1 + \frac{\delta}{2})\gamma_i = b_i^-. \tag{5.8}$$

Subtracting the second equation from the first we get

$$\delta \cdot \gamma_i = b_i^+ - b_i^-,$$

and hence,

$$\boxed{\gamma_i = \frac{1}{\delta}(b_i^+ - b_i^-).} \tag{5.9}$$

Substituting this value into equation (5.7), we get

$$c_i - (1 - \frac{\delta}{2})\frac{1}{\delta}(b_i^+ - b_i^-) = b_i^+$$

from which we get

$$c_i - (\frac{1}{\delta} - \frac{1}{2})(b_i^+ - b_i^-) = b_i^+,$$

and hence,

$$\boxed{c_i = \frac{1}{2}(b_i^+ + b_i^-) + \frac{1}{\delta}(b_i^+ - b_i^-).} \tag{5.10}$$

Now let us find the values for μ_{ij} and ν_{ij}. From equations (5.3) and (5.4) we conclude that $y_{ij} = \mu_{ij}x_j$ and $z_{ij} = \nu_{ij}x_j$. Substituting these expressions into equation (5.6) we get

$$\mu_{ij}x_j + [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]\nu_{ij}x_j = [a_{ij}^-, a_{ij}^+]x_j.$$

For this equality to be true for all x_j, coefficients for x_j on both sides of the equation must coincide. Thus, we conclude that

$$\mu_{ij} + [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]\nu_{ij} = [a_{ij}^-, a_{ij}^+].$$

As stated before, for intervals to be equal their endpoints must coincide, and since we are looking for a solution for $\nu_{ij} \geq 0$, the equations for the endpoints take the following forms:

$$\mu_{ij} + (1 + \frac{\delta}{2})\nu_{ij} = a_{ij}^+ \tag{5.11}$$

and

$$\mu_{ij} + (1 - \frac{\delta}{2})\nu_{ij} = a_{ij}^-. \tag{5.12}$$

Subtracting the second equation from the first we get

$$\delta \cdot \nu_{ij} = a_{ij}^+ - a_{ij}^-,$$

and hence,

$$\boxed{\nu_{ij} = \frac{1}{\delta}(a_{ij}^+ - a_{ij}^-).} \tag{5.13}$$

Substituting this value back into equation (5.11) we get

$$\mu_{ij} + (1 + \frac{\delta}{2})\frac{1}{\delta}(a_{ij}^+ - a_{ij}^-) = a_{ij}^+$$

which leads to

$$\mu_{ij} + (\frac{1}{\delta} + \frac{1}{2})(a_{ij}^+ - a_{ij}^-) = a_{ij}^+,$$

and hence,

$$\boxed{\mu_{ij} = \frac{1}{2}(a_{ij}^+ + a_{ij}^-) - \frac{1}{\delta}(a_{ij}^+ - a_{ij}^-).} \tag{5.14}$$

5.3.2 Part II: The Two Systems Are "Equivalent"

Let us show that every system (3.2) of interval linear equations is "equivalent" to the δ-narrow interval linear equation system (5.1)–(5.4) in the following sense:

Claim 1 If $(x_1, \ldots, x_n)$ is a solution of the original system (3.2), then for the same values $x_1, \ldots, x_n$, and for some values $w_1, \ldots, w_m$, $y_{11}, \ldots, y_{mn}$ and $z_{11}, \ldots, z_{mn}$, the extended tuple

$$(x_1, \ldots, x_n, w_1, \ldots, w_m, y_{11}, \ldots, y_{mn}, z_{11}, \ldots, z_{mn}) \tag{5.15}$$

is a solution of the δ-narrow interval system (5.1)–(5.4).

Claim 2 Conversely, if tuple (5.15) is a solution of the δ-narrow interval linear equation system (5.1)–(5.4), then $(x_1, \ldots, x_n)$ is a solution of the original system (3.2) for the same values $x_1, \ldots, x_n$.

We will now prove that the two systems are indeed "equivalent" in the above sense.

Proof of Claim 1

Let us assume that $(x_1, \ldots, x_n)$ is a solution of the original system (3.2). By definition, if $(x_1, \ldots, x_n)$ is a solution to the original system (3.2), then there must exist values $a_{ij} \in [a_{ij}^-, a_{ij}^+]$ and $b_i \in [b_i^-, b_i^+]$ such that

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \tag{5.16}$$

for all $i = 1, \ldots, m$. With this in mind, let us introduce new parameters $\alpha_{ij} = a_{ij} - a_{ij}^-$ and $\beta_i = b_i - b_i^-$; thus, $a_{ij} = a_{ij}^- + \alpha_{ij}$ and $b_i = b_i^- + \beta_i$. Substituting these expressions into (5.16) we get

$$\sum_{j=1}^{n} (a_{ij}^- + \alpha_{ij}) x_j = b_i^- + \beta_i. \tag{5.17}$$

Since $a_{ij} \in [a_{ij}^-, a_{ij}^+]$, we conclude that $\alpha_{ij} \in [0, a_{ij}^+ - a_{ij}^-]$. Similarly, since $b_i \in [b_i^-, b_i^+]$, we conclude that $\beta_i \in [0, b_i^+ - b_i^-]$.

Now let us show that there exists an extended tuple (5.15) that is a solution to system (5.1)–(5.4) for the same values $x_1, \ldots, x_n$ and for appropriately chosen values of $w_1, \ldots, w_m$, $y_{11}, \ldots, y_{mn}$ and $z_{11}, \ldots, z_{mn}$. In order to have equations (5.2)–(5.4) automatically satisfied, we will take $w_i = \gamma_i$, $y_{ij} = \mu_{ij} x_j$ and $z_{ij} = \nu_{ij} x_j$. It is thus sufficient to show that equation (5.1) is satisfied for all $i = 1, \ldots, m$; in other words, we need to show that there exist values $\kappa_i \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$ and $\lambda_{ij} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$ such that

$$\sum_{j=1}^{n} y_{ij} + \sum_{j=1}^{n} \lambda_{ij} z_{ij} + \kappa_i w_i = c_i \tag{5.18}$$

for all $i = 1, \ldots, m$.

Note that for any given i, if $\gamma_i = 0$ (and thus $w_i = 0$), then $\kappa_i w_i = 0$, implying that κ_i can be *any value* as long as $\kappa_i \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$ (e.g., $\kappa_i = 1$). Likewise, for any given i and j, if $\nu_{ij} = 0$ (and thus $z_{ij} = 0$), then $\lambda_{ij} z_{ij} = 0$, implying that λ_{ij} can be *any value* as long as $\lambda_{ij} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$ (e.g., $\lambda_{ij} = 1$). Therefore, for the remainder of the proof we will be choosing κ_i only for those i for which $\gamma_i > 0$ (i.e., $b_i^- < b_i^+$), and we will be choosing λ_{ij} only for those i and j for which $\nu_{ij} > 0$ (i.e., $a_{ij}^- < a_{ij}^+$).

We have chosen w_i and c_i for which the equation (5.5) is true, i.e., for which

$$c_i - [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}] w_i = [b_i^-, b_i^+].$$

Crudely speaking, this equality means that when we take different values for κ_i such that $\kappa_i \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$, the set

$$c_i - [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}] w_i$$

of possible values of $c_i - \kappa_i w_i$ coincides with the interval $[b_i^-, b_i^+]$. In particular, since $b_i \in [b_i^-, b_i^+]$, there must exist a value $\kappa_i \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$ for which

$$c_i - \kappa_i w_i = b_i = b_i^- + \beta_i.$$

From this equation we can find the exact value of κ_i.

We know from equation (5.2) that $w_i = \gamma_i$, so by substitution we get

$$c_i - \kappa_i \gamma_i = b_i^- + \beta_i,$$

and hence,

$$\kappa_i = \frac{1}{\gamma_i}(c_i - b_i^- - \beta_i).$$

Substituting the values of γ_i and c_i from equations (5.9) and (5.10) we get

$$\kappa_i = \frac{\delta}{b_i^+ - b_i^-}(\frac{1}{2}(b_i^+ + b_i^-) + \frac{1}{\delta}(b_i^+ - b_i^-) - b_i^- - \beta_i).$$

Simplifying, we get

$$\kappa_i = 1 + \frac{\delta}{b_i^+ - b_i^-}(\frac{1}{2}(b_i^+ + b_i^-) - b_i^- - \beta_i)$$

from which we get

$$\kappa_i = 1 + \frac{\delta}{b_i^+ - b_i^-}(\frac{1}{2}(b_i^+ - b_i^-) - \beta_i),$$

and hence,

$$\boxed{\kappa_i = 1 + \frac{\delta}{2} - \frac{\delta \cdot \beta_i}{b_i^+ - b_i^-}.} \tag{5.19}$$

We have also chosen y_{ij} and z_{ij} for which the equation (5.6) is true, i.e., for which

$$y_{ij} + [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]z_{ij} = [a_{ij}^-, a_{ij}^+]x_j$$

Crudely speaking, this equality means that when we take different values for λ_{ij} such that $\lambda_{ij} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$, the set

$$y_{ij} + [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]z_{ij}$$

of possible values of $y_{ij} + \lambda_{ij} z_{ij}$ coincides with the interval $[a_{ij}^-, a_{ij}^+]x_j$. In particular, since $a_{ij} \in [a_{ij}^-, a_{ij}^+]$, there must exist a value $\lambda_{ij} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$ for which

$$y_{ij} + \lambda_{ij} z_{ij} = (a_{ij}^- + \alpha_{ij})x_j.$$

From this equation we can find the exact value of λ_{ij}.

We conclude from equations (5.3) and (5.4) that $y_{ij} = \mu_{ij} x_j$ and $z_{ij} = \nu_{ij} x_j$, so by substitution we get

$$\mu_{ij} x_j + \lambda_{ij} \nu_{ij} x_j = (a_{ij}^- + \alpha_{ij})x_j,$$

and hence,

$$\lambda_{ij} = \frac{1}{\nu_{ij}}(a_{ij}^- + \alpha_{ij} - \mu_{ij}).$$

Substituting the values of the coefficients ν_{ij} and μ_{ij} from equations (5.13) and (5.14) we get

$$\lambda_{ij} = \frac{\delta}{a_{ij}^+ - a_{ij}^-}(a_{ij}^- + \alpha_{ij} - \frac{1}{2}(a_{ij}^+ + a_{ij}^-) + \frac{1}{\delta}(a_{ij}^+ - a_{ij}^-)).$$

Simplifying, we get

$$\lambda_{ij} = 1 + \frac{\delta}{a_{ij}^+ - a_{ij}^-}(a_{ij}^- + \alpha_{ij} - \frac{1}{2}(a_{ij}^+ + a_{ij}^-))$$

from which we get

$$\lambda_{ij} = 1 - \frac{\delta}{a_{ij}^+ - a_{ij}^-}(\frac{1}{2}(a_{ij}^+ - a_{ij}^-) - \alpha_{ij}),$$

and hence,

$$\boxed{\lambda_{ij} = 1 - \frac{\delta}{2} + \frac{\delta \cdot \alpha_{ij}}{a_{ij}^+ - a_{ij}^-}.} \tag{5.20}$$

To show that these values for κ_i and λ_{ij} are indeed the ones desired, we first note that, since $\beta_i \in [0, b_i^+ - b_i^-]$ and $\alpha_{ij} \in [0, a_{ij}^+ - a_{ij}^-]$, we can conclude that

$$\frac{\beta_i}{b_i^+ - b_i^-} \in [0, 1]$$

and

$$\frac{\alpha_{ij}}{a_{ij}^+ - a_{ij}^-} \in [0, 1].$$

Thus,

$$\frac{\delta \cdot \beta_i}{b_i^+ - b_i^-} \in [0, \delta]$$

and

$$\frac{\delta \cdot \alpha_{ij}}{a_{ij}^+ - a_{ij}^-} \in [0, \delta]$$

from which it follows that

$$1 + \frac{\delta}{2} - \frac{\delta \cdot \beta_i}{b_i^+ - b_i^-} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$$

and

$$1 - \frac{\delta}{2} + \frac{\delta \cdot \alpha_{ij}}{a_{ij}^{+} - a_{ij}^{-}} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}],$$

and hence,

$$\kappa_i \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$$

and

$$\lambda_{ij} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}].$$

Our goal is to prove that equation (5.1) is satisfied. By showing that we have found appropriate values for κ_i and λ_{ij} such that equation (5.18) is satisfied, we can do just that. We have already chosen the values for y_{ij} and z_{ij}. Substituting these values into equation (5.18) we get an equivalent equation

$$\sum_{j=1}^{n} \mu_{ij} x_j + \sum_{j=1}^{n} \lambda_{ij} \nu_{ij} x_j + \kappa_i w_i = c_i.$$

This equation, in its turn, is equivalent to the equation

$$\sum_{j=1}^{n} (\mu_{ij} + \lambda_{ij} \nu_{ij}) x_j = c_i - \kappa_i w_i. \tag{5.21}$$

Let us now show that this equation is satisfied and thus prove that the equivalent equation (5.18) is satisfied as well.

- According to our choice of κ_i, the right-hand side of equation (5.21) is equal to b_i.
- According to our choice of λ_{ij}, we have

  $$\mu_{ij} + \lambda_{ij} \nu_{ij} = a_{ij},$$

  and hence, the left-hand side of equation (5.21) is equal to

  $$\sum_{j=1}^{n} a_{ij} x_j.$$

We started with $x_1, \ldots, x_n$ for which $\sum_{j=1}^{n} a_{ij} x_j = b_i$; hence, the left-hand side and the right-hand side of equation (5.21) coincide. Thus, we have proven that equation (5.18) is satisfied.

Since we are able to show that there do indeed exist values $\kappa_i \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$ and $\lambda_{ij} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$ such that equation (5.18) is satisfied for all $i = 1, \ldots, m$, it follows that if $(x_1, \ldots, x_n)$ is a solution of system (3.2), then for the same values $x_1, \ldots, x_n$ and for some values $w_1, \ldots, w_m$, $y_{11}, \ldots, y_{mn}$ and $z_{11}, \ldots, z_{mn}$, there exists an extended tuple (5.15) that is a solution of system (5.1)–(5.4). This proves Claim 1.

Proof of Claim 2

Now we must show that the converse is true, namely, that if tuple (5.15) is a solution of the δ-narrow interval linear equation system (5.1)–(5.4), then $(x_1, \ldots, x_n)$ is a solution of the interval linear equation system (3.2) for the same values $x_1, \ldots, x_n$.

Let us assume tuple (5.15) is a solution of system (5.1)–(5.4). For this to be true, equations (5.2)–(5.4) must be satisfied, and thus we can conclude that $w_i = \gamma_i$, $y_{ij} = \mu_{ij} x_j$ and $z_{ij} = \nu_{ij} x_j$. The fact that equation (5.1) is satisfied by a given tuple means that

$$\sum_{j=1}^{n} y_{ij} + \sum_{j=1}^{n} \lambda_{ij} z_{ij} + \kappa_i w_i = c_i \tag{5.22}$$

for some $\lambda_{ij} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$ and $\kappa_i \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$. Let us show that

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \tag{5.23}$$

for some values $a_{ij} \in [a_{ij}^-, a_{ij}^+]$ and $b_i \in [b_i^-, b_i^+]$.

From equation (5.22) we conclude that

$$\sum_{j=1}^{n} y_{ij} + \sum_{j=1}^{n} \lambda_{ij} z_{ij} = c_i - \kappa_i w_i.$$

Since $w_i = \gamma_i$, $y_{ij} = \mu_{ij} x_j$ and $z_{ij} = \nu_{ij} x_j$, by substitution we conclude that

$$\sum_{j=1}^{n} (\mu_{ij} + \lambda_{ij} \nu_{ij}) x_j = c_i - \kappa_i \gamma_i.$$

Note that this equation has the desired form (5.23) where

$$a_{ij} = \mu_{ij} + \lambda_{ij} \nu_{ij} \tag{5.24}$$

and

$$b_i = c_i - \kappa_i \gamma_i. \tag{5.25}$$

Thus, to complete the proof, we need only show that $a_{ij} \in [a_{ij}^-, a_{ij}^+]$ and $b_i \in [b_i^-, b_i^+]$.

Let us first show that $b_i \in [b_i^-, b_i^+]$. Since $\kappa_i \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$, we have that

$$1 + \frac{\delta}{2} \geq \kappa_i \geq 1 - \frac{\delta}{2}.$$

By multiplying all sides of the inequality by $\gamma_i \geq 0$, we conclude that

$$(1 + \frac{\delta}{2})\gamma_i \geq \kappa_i \gamma_i \geq (1 - \frac{\delta}{2})\gamma_i.$$

By subtracting all parts of this inequality from c_i, we conclude that

$$c_i - (1 + \frac{\delta}{2})\gamma_i \leq c_i - \kappa_i \gamma_i \leq c_i - (1 - \frac{\delta}{2})\gamma_i.$$

According to our choice of c_i and γ_i (which led to equations (5.7) and (5.8)), this is equivalent to

$$b_i^- \leq c_i - \kappa_i \gamma_i \leq b_i^+.$$

Thus, the value $b_i = c_i - \kappa_i \gamma_i$ we have chosen indeed belongs to the interval $[b_i^-, b_i^+]$.

Let us now show, in like fashion, that $a_{ij} \in [a_{ij}^-, a_{ij}^+]$. Since $\lambda_{ij} \in [1 - \frac{\delta}{2}, 1 + \frac{\delta}{2}]$, we have that

$$1 - \frac{\delta}{2} \leq \lambda_{ij} \leq 1 + \frac{\delta}{2}.$$

By multiplying all sides of the inequality by $\nu_{ij} \geq 0$, we conclude that

$$(1 - \frac{\delta}{2})\nu_{ij} \leq \lambda_{ij}\nu_{ij} \leq (1 + \frac{\delta}{2})\nu_{ij}.$$

By adding all parts of this inequality to μ_{ij}, we conclude that

$$\mu_{ij} + (1 - \frac{\delta}{2})\nu_{ij} \leq \mu_{ij} + \lambda_{ij}\nu_{ij} \leq \mu_{ij} + (1 + \frac{\delta}{2})\nu_{ij}.$$

According to our choice of μ_{ij} and ν_{ij} (which led to equations (5.11) and (5.12)), this is equivalent to

$$a_{ij}^- \leq \mu_{ij} - \lambda_{ij}\nu_{ij} \leq a_{ij}^+,$$

i.e., the value $a_{ij} = \mu_{ij} - \lambda_{ij}\nu_{ij}$ that we have chosen indeed belongs to the interval $[a_{ij}^-, a_{ij}^+]$.

Thus, we have shown that if tuple (5.15) is a solution of system (5.1)–(5.4), then $(x_1, \ldots, x_n)$ is a solution of system (3.2) for the same values $x_1, \ldots, x_n$. This proves Claim 2.

5.3.3 Conclusion

We saw in part II that the two systems, namely, the interval linear equation system (3.2) and the δ-narrow interval linear equation system (5.1)–(5.4), are "equivalent" in the sense described therein. In part I we saw that the number of equations and unknowns of system (5.1)–(5.4) is bounded by a polynomial of the number of equations and unknowns of system (3.2). Further, the number of steps required for reducing an interval linear equation system (3.2) to a δ-narrow interval linear equation system (5.1)–(5.4) is bounded by a polynomial of the number of equations and unknowns of system (3.2). Thus, it follows from parts I and II that if we have an algorithm $\mathcal{U}$ that can solve δ-narrow interval linear equation systems in polynomial time, then we can solve an arbitrary system (3.2) of interval linear equations in polynomial time by applying this hypothetical algorithm $\mathcal{U}$ to the "equivalent" δ-narrow interval linear equation system (5.1)–(5.4). But, as stated in part I, solving interval linear equation systems is an NP-hard problem.

So, if we can (in general) solve interval linear equation systems in polynomial time, we can solve any problem from the class NP in polynomial time. Therefore, if we can (in general) solve δ-narrow interval linear equation systems in polynomial time, we can solve any problem from the class NP in polynomial time. Thus, we conclude that *the problem of solving δ-narrow interval linear equation systems is* NP*-hard*. Q.E.D.

Chapter 6

Concluding Remarks

6.1 Significance of the Result

Now that we have shown that we cannot solve narrow-interval linear equation systems in general, what does this mean to the computing and mathematical communities? Unless P=NP, attempts at developing a feasible algorithm to solve this problem in general will assuredly fail; however, the very fact that there exists an algorithm for solving a large number of these systems (see [9]) shows that work on solving a subclass of the class of all narrow-interval linear equation systems is certainly a worthwhile endeavor.

6.2 Future Work

The problem now shifts to identifying new subclasses of the class of all narrow-interval linear equation systems for which the problem of solving them is possible with the development of new algorithms. Also, if the general problem (or any problem in the class NP) shows up often enough in industry, science, research, etc., work on improving existing and/or creating new approximation methods (including heuristic and/or statistical methods, where applicable) is certainly warranted. Since we cannot compute the exact bounds for the general case, good approximation methodologies are the most we can hope for or expect.

References

[1] S. Cook, "The complexity of theorem-proving procedures," *Proceedings of the 3rd ACM Symposium on Theory of Computing*, Shaker Heights, Ohio, 1971, pp. 151–158.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of* NP-*Completeness*, W. H. Freeman, San Francisco, 1979.

[3] R. Karp, "Reducibility among combinatorial problems," in: R. Miller and J. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York, 1972, pp. 85–103.

[4] R. B. Kearfott and V. Kreinovich (eds.), *Applications of Interval Computations*, Kluwer Academic Publishers, Norwell, MA, 1996.

[5] V. Kreinovich, A. V. Lakeyev and S. I. Noskov, "Optimal solution of interval linear systems is intractable (NP-hard)," *Interval Computations*, 1993, No. 1, pp. 6–14.

[6] V. Kreinovich, A. V. Lakeyev and J. Rohn , "Computational complexity of interval algebraic problems: some are feasible and some are computationally intractable: a survey," in: G. Alefeld and A. Frommer (eds.), *Scientific Computing and Validated Numerics*, Akademie-Verlag, Berlin, 1996, pp. 293–306.

[7] V. Kreinovich, A. V. Lakeyev, J. Rohn and P. Kahl, *Feasible? Intractable? On Computational Complexity of Data Processing and Interval Computations*, Kluwer Academic Publishers, Norwell, MA, 1996 (to appear).

[8] U. Kulisch and W. L. Miranker, *Computer Arithmetic in Theory and Practice*, Academic Press, NY, 1981.

[9] A. V. Lakeyev and V. Kreinovich, "If input intervals are small enough, then interval computations are almost always easy," *Reliable Computing*, Supplement (Extended

Abstracts of APIC'95: International Workshop on Applications of Interval Computations), 1995, pp. 134–139.

[10] L. Levin, "Universal sequential search problems," *Problems of Information Transmission*, 1973, Vol. 9, No. 3, pp. 265–266.

[11] R. E. Moore, "Automatic error analysis in digital computation," *Technical Report LMSD-48421*, Lockheed Missiles and Space Co., Palo Alto, CA, January 1959.

[12] R. E. Moore, *Interval Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1966.

[13] A. Neumaier, *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, 1990.

[14] S. G. Rabinovich, *Measurement Errors: Theory and Practice*, American Institute of Physics, NY, 1995.

Curriculum Vitae

Patrick Thor Kahl was born on July 12, 1961. The first son of Ulf Thor Gustav Kahl and Carolyn Kahl, he graduated from Coronado High School, El Paso, Texas, in the spring of 1979. He entered Auburn University in the fall of 1979, and, in the spring of 1982, The University of Texas at El Paso. In 1985 he joined the United States Navy where he served for eight years, most of it aboard the submarine USS Narwhal (SSN671). In the fall of 1993, after being honorably discharged from the navy, Patrick resumed his studies at The University of Texas at El Paso. While pursuing his bachelor's degree in Computer Science he worked as a Teaching Assistant, and as a programmer at the National Solar Observatory at Sunspot, New Mexico. He received his bachelor's degree in Computer Science in the summer of 1994.

In the fall of 1994, he entered the Graduate School of The University of Texas at El Paso. While pursuing a master's degree in Computer Science he worked as a Teaching and Research Assistant, and as the Laboratory Instructor for the 1995 Real-Time Programming Seminar at the University of Puerto Rico, Mayagüez Campus. He was a member of the Knowledge Representation Group and the Rio Grande Chapter of the Association for Computing Machinery.

Permanent address: 6216 Sylvania Way
El Paso, Texas 79912-4927