REFERENCES

1. Waltz D.L. , Applications of the Connection Machine, Computer,pp.85-97,January 1987.

2. POGUE C. A.,Willet P.,Use of Text Signature for Document Retrieval in a Highly Parallel Enviroment , Parallel Computing,4 ,pp.259-268,1987.

3. Meltzer A.C. and Kowalski G., Text Searching Using an Invertion Database Consisting of Trigrams , IEEE Proc. of Sec.Int.Conf.on Computers and Applications,pp 65-69,1987

4.Berkovich S.Y. and Abd El Fatah A.Hegazy, Matching String Patterns in Large Textual Files ,IEEE Computer Society Reprint,pp.122-127, 1985.

5.HarrisonM.C.,Implementation of the Substring Test by Hashing,Comm.ACM,V.14,pp.777-779,1971.

(NASA-CR-192943) INTELLIGENT PARALLEL SIMULATION: A KEY TO INTRACTABLE PROBLEMS IN INFORMATION PROCESSING (Texas Univ.) 6 p

N93-72059

Unclas

Z9/82 0159191

NASA-CR-192943

NAG 9-482 N1S

INTELLIGENT PARALLEL SIMULATION – A KEY TO INTRACTABLE PROBLEMS IN INFORMATION PROCESSING

7N-82-CR

159191

P.6

Andrew Bernat[1] Luis Cortes[1] Vladik Kreinovich[1]
Karen Villaverde[1]

[1] University of Texas at El Paso, El Paso, TX

ABSTRACT

We describe simulation-type methods that solve (in reasonable time) the following problems of intelligent information processing: estimate the result's precision; estimate the result's reliability; and, decision-making in case of a conflict.

We also propose a special architecture that enables to parallelize these algorithms efficiently (without wasting time on communication protocols.)

INTRODUCTION

The main objective of science, and why we need information processing. The whole purpose of science is to cognize the world, to be able to predict what will happen if we do not interfere, what will be the result of this or that action, and how to choose the best action ("best" in some reasonable sense). In our predictions, we rely on our knowledge and on the results of experiments. Our knowledge usually consists of physical laws that enable us to predict how the current state of the world will evolve. But in order to apply them, we must know this current state of the world, i.e., we must know the values of all the physical variables that characterize the world.

For example, if we are interested in the trajectory of a satellite, then Newton's equation enable us to predict it, on the condition that we know precisely the masses of all the bodies in the Solar system and all the distances between them.

Some of these variables can be measured more or less directly: e.g., in the satellite example, we can measure the distance to the Moon by sending a laser signal from Earth to a Lunar mirror, and measuring the time that it takes for a signal to make this round trip. However, for many important variables, we do not know how to measure them directly. We know of no measuring instrument that would directly measure the mass of a planet or of an elementary particle, or the amount of oil in an oil well, etc. So, in these cases, we measure whatever we can, and then we use our knowledge to reconstruct the value y of the desired variable Y from the values $x_1, \ldots, x_n$ of the measured variables.

For example, in order to measure the mass y of a planet, we observe the trajectories of its satellites, and then we use Newton's equations to extract the value of its mass.

This is the main reason why we need "number-crunching" supercomputers that process gigabytes of data from radiotelescopes, satellites, supercolliders, from geophysical observatories: the final result of this processing consists of the few values that we are interested in. For example, in weather forecasting we process tons of data to produce a few temperature and density estimates.

To be able to process these data, we must know the relationship between the measurable data x_i and the desired value y. This relationship must be somehow extracted from the existing knowledge.

Information processing: the resulting picture. So, if we are interested in the value of some physical quantity y that we cannot directly measure (e.g., tomorrow's temperature, or the amount of oil in a well), we must do the following:

1) we must find out what we know about this quantity y; this knowledge can take the form of differential equations (like Newton's laws or typical equations of physics), semi-empirical laws, rules, or simply experts' intuitive guesses;

2) from this knowledge we must extract a relationship between y and some other quantities x_i (whose values we can measure directly);

3) we must express this relationship in an algorithmic form, and write a program to compute y from x_i;

4) we must measure the values x_i, and substitute them into our program. As a result, we have an estimate for y. Meanwhile, we may reveal some additional laws or rules that can be used to solve further problems.

All these steps are often extremely complicated, and no wonder that they attract the main attention of the media. The first person to measure the size of a galactic cluster, or the age of a species is guaranteed this attention. This part of information processing is a mainstream, and it is so complicated that it sounds reasonable to consider it the main challenge of science.

Our results. We start on a pessimistic note: yes, discovering a relationship between y and x_i is a challenge, but there are other problems in information processing. These problems are sometimes considered to be minor, but they are even more challenging and hardly tractable.

And here parallel simulation comes into action: it enables us to solve these new problems. For that, we need a more detailed view of information processing.

Information processing: a more detailed view. Of course, if we are interested in the value of some physical quantity y, then we must compute an estimate $\tilde{y}$. But this is not all.

First, all measurements are approximate. Every measurement device has some range of possible errors, and these errors lead to inevitable imprecision of the resulting estimate. If we measured X_i with, say, a 1% precision, then the resulting estimate is also not precise. And it is often extremely important to know the precision of y. For example, if we estimate that an oil well has, say, 50 million tons of oil, then, if it is 50 ± 5, then it is really good estimate, but if it is 50 ± 100, then this estimate is senseless.

Second, we want to know the reliability of this estimate. The majority of real-life estimates are based not only on the well-established physical laws, but on some semi-empirical rules and expert statements that are not always 100% reliable. Therefore, the relationship between y and x_i that we extracted from all that knowledge, can also turn out to be wrong. What is the reliability of our conclusion? This aspects is most evident for weather predictions: even if the forecast predicts 60 F, and we know the precision (say ± 5), we also know that these predictions are not always valid, so we would like to estimate in what percentage of cases they are true: is it a forecast with a 95% reliability, or its reliability is close to 50% which makes it senseless.

These two aspects dealt with the estimate. But in the majority of cases, we do not need

estimates per se. We want them in order to choose an appropriate action: to drill or not to drill the well, to take an umbrella or not, etc. In many cases, it is a problem of individual action. A person can make a decision about taking his umbrella, and no matter what he decides, no one else will be hurt. In more complicated cases, e.g., when we discuss a location of a plant, there are usually several interests involved, and to find a solution that is satisfactory for all the parties is often not so easy.

The frequent viewpoint (maybe slightly exaggerated). These three aspects are usually treated as second-class citizens in the land of Information Processing: media rarely pays any attention to a person who estimated a precision or reliability. The main objective is to get an estimate. People who get them are praised as geniuses. Of course, not everyone is an Einstein, so for those who cannot, there is always a dull but necessary work: to estimate precision and reliability, and to apply this knowledge to real-life situations with a conflict of interests. This is not that difficult, but someone needs to do it.

Alas, life is more complicated. Live would have been much easier if this viewpoint was true, but, alas, it turns out that to estimate precision and reliability is *much more difficult* than to solve the problem itself. So, from the philosophical viewpoint, *the main bottleneck is not in discovering the methods* to estimate the desired value y from observations, *but in estimating precision and reliability of the resulting estimates, and in resolving the resulting conflict situations.*

What we are planning to do. We want to:
1) show that these three problems are really extremely complicated;
2) propose simulation-type methods of solving them, and
3) propose a special parallel computer architecture that is maximally suitable for solving these problems.

In other words, we are planning to show that *intelligent parallel simulation is a key to the intractable bottleneck problems of information processing.*

The algorithms that we come up with are rather simple. The non-trivial point that we want to make is that in all the three cases *without simulation these problems are intractable.*

Some results of this paper appeared first in [1-4].

FIRST PROBLEM: ESTIMATING PRECISION

Informal introduction to the problem. Suppose that we are interested in some quantity y (e.g., the amount of oil in a given region), and it is difficult (or even impossible) to measure y directly. So we measure some physical quantities $x_1, \ldots, x_n$ that are indirectly connected with y (e.g., the conductivity of different layers, the results of ultrasound screening, etc.) and then compute y by applying some known algorithm to these values x_i. This algorithm is actually the numerical method for solving the equations that connect y with x_i (these equations can be algebraic, integral, differential, stochastic etc).

What is the possible error of the resulting estimate? The traditional approach to answering this question is to analyze the corresponding mathematical model. Engineers who have gone through numerical mathematics know very well that this is often an extremely difficult problem. Even for the simplest case, when y and x_i are related by a system of a linear equations, error estimates are difficult to compute, and for some nonlinear systems error estimates are not even known. So these methods are hard to use, and for an engineer, who is not a specialist in numerical analysis, it is a very tough (and sometimes even impossible) job. So, what to do?

Illustrative example. In order to understand the models and the solutions better we'll

give an illustrative example. This example is so simple that it does not constitute any problem from the computational viewpoint and is given here only for illustration purposes.

Suppose we cannot measure the voltage V directly, so we decided to measure it indirectly: i.e., measure the current I, the resistance R and then multiply the resulting values (i.e., apply Ohm's law). Suppose the result i of measuring current is 2.5 (lower case letters represent the actual measured quantities), the result r of measuring resistance is 4.0; then the resulting voltage value is $v = ir = 10.0$. The real value of voltage can be different from v, e.g., if we know that the precision of both measurements is 0.1 (the real values of I and R can differ from the measured ones by no more than 0.1) then these real values can be, e.g., 2.59 and 4.08, in which case the real value V of voltage is 10.56..., i.e., the error in this case is 0.56.... In the other cases the error can be smaller or greater. It is impossible to enumerate all possible values of I and R (there are infinitely many), therefore the problem of estimating all possible values of error is sometimes non-trivial.

We want to formulate this problem in mathematical terms. By n we denote the number of all measurement results. The result of ith measurement will be denoted by x_i; the real value of the measured quantity will be denoted by X_i. The difference between x_i and X_i will be denoted by d_i and called the error of the ith measurement.

By $f(x_1, \ldots, x_n)$ we denote a value of the quantity y that corresponds to the values x_i of the measured quantities. The analytical expression for the function f may not be known, but there must exist a program that computes the value of f for any input data. So the real value Y of the desired quantity equals to $f(X_1, \ldots, X_n)$, the value y that is produced by the given program equals to $f(x_1, \ldots, x_n)$. The difference between them $d = y - Y$ is called an *error* of the resulting indirect measurements of y.

In the above example $n = 2, x_1 = i, x_2 = r, f(x, y) = xy$. X_1 is the real value I of current, $R = X_2$ is the real value of resistance, $v = ir$ is the result of our indirect measurement procedure and $V = IR$ is the real voltage.

We are also supposed to know the estimates D_i for d_i, and from them we want to compute an estimate D for d. Following the traditional engineering approach [5–6], we'll consider two kinds of errors: systematic and random errors.

Systematic errors. By saying that an error is *systematic* we mean (from the mathematical viewpoint) that the only thing we know about the error d_i is that it belongs to a certain interval $[-D_i, D_i]$. In this case the possible values of X_i form an interval $[x_i - d_i, x_i + d_i]$. The resulting possible values of Y also fill the interval, and the estimate we are interested in is, in this case, the maximum possible value D of $|d| = |y - Y|$.

Random Errors. Another typical situation in engineering applications is when the error is *random*, which means that d_i are random variables, and we know their probabilistic distributions. These distributions are usually considered to be independent; the average value of an error is 0; we know the mean square deviation D_i and assume that the probabilistic distribution is Gaussian. In this case the resulting error d is also a random variable, and we are interested in the mean square deviation D of this variable, i.e., $D^2 = E(d^2) = E((y - Y)^2) = E((f(x_1, \ldots, x_n) - f(X_1, \ldots, X_n))^2)$, where E means mathematical expectation.

Mathematical formulation. We know the following: an integer n (the number of measurement results), a program f that, given n real numbers $x_1, \ldots, x_n$, computes the estimate $f(x_1, \ldots, x_n)$ for y; n real numbers $x_i, i = 1, 2, \ldots, n$ (the measurement results), and n positive real numbers $D_i, i = 1, 2, \ldots, n$ (the estimates of the measurement errors).

The Shapley value describes a reasonable compromise between the interests of all the participants, and allocates to every player i the sum $x_i = \sum_{S: i \notin S} p(S)(v(S \cup \{i\}) - v(S))$, where the sum is taken over all the sets S that do not contain i, $p(S) = (1/n)/C(n-1, |S|)$, where $|S|$ denotes the number of elements in the set S, and $C(m, r) = m!/(r!(m-r)!)$ is the number of combinations.

This explicit formula is not computationally feasible for one simple reason: it uses the sum over all possible subsets S such that $i \notin S$, and there are 2^{n-1} of them. For example, for $n = 100$ (a reasonable amount of interest groups), we need 2^{99} computational steps, which is impossible.

Simulation helps to make Shapley value feasible. The possibility to apply Monte-Carlo methods to Shapley value stems from the original paper of Shapley [15], who proved that the i-th component x_i of the Shapley value is equal to the average value of the difference $d = v(S \cup \{i\}) - v(S)$ with respect to the following probabilistic distribution: consider all the orderings of $\{1, 2, \ldots, n\}$ equally probable (with probability $1/n!$), then S is the set of all the players who precede i in this random ordering.

So, in order to estimate x_i, we can simulate random permutations several (N) times, for each of these N permutation estimate the corresponding value $d^{(k)}$ of the difference d, and take an average $(d^{(1)} + \ldots + d^{(N)})/N$ as a desired estimate X_i for x_i.

To get a random permutation $\pi_1, \ldots, \pi_n$ of the sequence $1, 2, \ldots, n$, we first take a random element π_1 from the set $1 \ldots n$; if $\pi_1, \ldots \pi_k$ are already computed, we pick random elements from this same set $1, \ldots, n$ until we find an element that is different from all the previously chosen values $\pi_1, \ldots, \pi_k$; this element will be chosen as π_{k+1}.

How many iterations do we need? Suppose that we want to estimate x_i with the precision cV/n (where $V = V(\{1, 2, \ldots, n\})$ is the total amount of money that all the participants get, so V/n is per capita outcome, and c is a relative precision) and with reliability p. In other words, we want the probability $P(|X_i - x_i| \le cV/n)$ to be not smaller than p.

Let us apply Tchebychev inequality to estimate the difference between our result X_i and the desired value x_i: this inequality guarantees that $P(|X_i - x_i| < t\sigma) > 1 - 1/t^2$, where σ denotes a mean square value of the difference $X_i - x_i$. In order to get the desired inequality, we must take such t and N that $1 - 1/t^2 \ge p$ and $t\sigma \le cV/n$. The first inequality can be obtained, if we take $1 - 1/t^2 \ge p$, i.e. $t = (1 - p)^{-1/2}$.

Let us now estimate σ. Since $X_i = (d^{(1)} + \ldots + d^{(N)})/N$, the difference $X_i - x_i$ is an arithmetic mean of N equally distributed differences $d^{(k)} - x_i$. Therefore, $\sigma = \sigma'/\sqrt{N}$, where σ' denotes a mean square value of the difference $d^{(k)} - x_i$. The quantity $d = v(S \cup \{i\}) - v(S)$ takes values from 0 to $V = v(\{1, 2, \ldots, n\})$; x_i is also limited by V. Therefore, the absolute value of their difference is $\le V$. Therefore, the mean square value σ' of this difference is also bounded by V. So, $\sigma = \sigma'/\sqrt{N} \le V/\sqrt{N}$, hence $t\sigma \le cV/n$ if $N \ge n^2/(c^2(1-p))$. Note that this number of iterations grows slowly with n.

This is a crude upper estimate; in reality smaller N will be sufficient.

This algorithm is parallelizable. All simulations can be done in parallel. If we use the architecture that was outlined in the description of the first problem, then we do not waste time on protocols. As a result, the computation time will be equal to the time that is necessary to compute one value $v(S)$, i.e., to the running time of a linear programming problem (which is known to be quite feasible).

according to the Gaussian law with 0 average and standard deviation D_i. Then we substitute $x_i + s_i$ into f, and compute the difference between the resulting value and the value $y = f(x_1, \ldots, x_n)$, that was obtained from the initial data. This difference $d = f(x_1 + s_1, \ldots, x_n + s_n) - f(x_1, \ldots, x_n) = f_{,1}s_1 + \ldots + f_{,n}s_n$ is a linear combination of the normally distributed independent random variables, and therefore its probabilistic distribution is also Gaussian, with standard deviation $D = ((f_{,1})^2 D_1^2 + \ldots + (f_{,n})^2 D_n^2)^{1/2}$ equal to the desired value. So if we repeat this experiment several (N) times, then the resulting values $d_1, \ldots, d_N$ are distributed according to this Gaussian law and therefore we can use the standard statistical formula to determine D: $D = \sqrt{\sum_i (d_i)^2}$.

The precision of this formula is determined by the size N of the sample (and thus does not depend on n), e.g., if we want to estimate D with relative precision 20% (that is quite sufficient from an engineering viewpoint, since 1.2% precision is actually the same as 1%) we need $N = 25$. Therefore we need only 25 calls of f; for large n this is smaller than the n calls of the first algorithm.

Simulation method for systematic errors. That simulation methods work for random errors is no wonder. But they can be applied for systematic errors as well (for details see [2], [9]).

This method is based on the special properties of the Cauchy distribution and is as follows: For $k = 1, 2, \ldots, N = 50$ repeat the following:
1) use the random number generator to compute n numbers $r_i, i = 1, 2, \ldots, n$, that are uniformly distributed on the interval $[0, 1]$;
2) compute $s_i = D_i \tan(\pi(r_i - 0.5))$
3) substitute $x_i + s_i$ into the function f and compute $d_k = f(x_1 + s_1, \ldots, x_n + s_n) - f(x_1, \ldots, x_n)$
4) compute D by applying the bisection method to solve the equation $(1 + (d^1/D)^2) - 1 + \ldots + (1 + (d^N/D)^2)^{-1} = N/2$.

Parallelization is possible and helpful here. The main time-consuming part of these simulation algorithms is the part where we must call the program f. But all these computations of $f(\{x_i + s_i\})$ can be done in parallel if we have sufficiently many processors (25 or 50). Since all the processors work simultaneously, the resulting time is equal to the time spent by one of them, i.e., to the time that is necessary to compute just one value of f. Therefore, in addition to the processed data, we get its error estimate practically in no additional time (at the expense of using additional processors).

Not all parallel computers lead to this estimate: the necessity of a new architecture. In real parallel systems a lot of time is consumed on communication protocols, information exchange, waiting in the queues, etc. But in our case during the main ("calling f") stage no communication between the processors is necessary, and the only communication we need is: sending the initial values x_i to the processors and sending the resulting values d_k from the processors to one processor which will compute D from them. The fact that we need to send the same values to all the processors means that we can use small shared memory. The second communication task can be also solved by a small shared memory if we allocate 1 word for every processor (at the expense of maximally 50 additional words). In this case there can be no attempts to write contradictory data into this common memory, so we can implement it in the easiest possible way, e.g., we can choose one of the processors (that actually reads x_i) to be the Coordinator. In the memory of each processor we select two small pieces: for drop and for pick. In the beginning the signal is picked from the Coordinator and dropped consequently into the "drop" part of all other processors. At the end the same consequent process can be used to "pick" the results from the processors and drop them into the Coordinator's memory. So what we need is to connect all the processors by a loop and organize a continuous pick-drop process: a signal is picked from the pick locations and dropped into the Coordinator's memory.

Such a simple loop system has been produced by Septor Electronics for use in machinery control applications [10]. We have implemented the above-described algorithms on a system based upon the Septor Electronics boards and indeed achieved the desired speedup.

SECOND PROBLEM: RELIABILITY

Due to lack of space, we will only briefly discuss this problem. Suppose that our knowledge consists of statements $E_1, \ldots, E_n$, and that we are not absolutely sure that all of them are correct. We can express our degree of belief in each of these statements by a number $p(E_i)$ that is called its *subjective probability*, or simply *probability*. From the set of all these statements E_i, we made a conclusion Q. We are interested in computing the probability $P(Q)$ that Q is true. This problem was also proved to be NP-hard ([1], [12]) and thus in general intractable.

But we are talking about real-life problems, and in real life we do not need (and never get) 100% reliability; in all engineering problems some reliability is enough (like 99.9%). And so, although the negative theorem forbids the algorithms that always solve the problem, it does not explicitly forbids algorithms that work in 99.9% of cases.

And indeed, simulation-type algorithms were proposed that solve this problem in reasonable time [1], [13]. These algorithms are based on the following idea. Since we are not sure whether all statements E_i are true, we can generate a "random" knowledge base by adding each statement E_i with probability p_i. We can repeat this procedure N times, and for each of the resulting knowledge bases ask whether Q is deducible from it or not. Then $p(Q)$ can be estimated as the ratio M/N, where M is the total number of cases in which Q was deducible.

If we have several processors, then we can generate each of N knowledge bases in parallel. The only operation that cannot be parallelized are: sending the results (*yes*, Q is deducible, or *no*, not deducible) to one of the processors, and computing M/N, where M is the total number of *yes* answers.

Therefore, we can parallelize this algorithm, and utilize the above-described architecture.

THIRD PROBLEM: RESOLVING CONFLICT SITUATIONS

Among all possible methods of conflict resolving, we will choose Shapley value. There exist many different methods of conflict resolving [14]. We choose Shapley value because, first, it is uniquely determined by some natural axioms [14]; second, it always exists, and, third, it chooses a single payoff vector (and therefore, unlike the other notions like the core and the von Neumann-Morgenstern solution, it does not demand any further choice).

Definition of a Shapley value. Suppose that we have a conflict situation that involves n participants. In game theory the participants are called *players*. Let us enumerate the players in a sequence 1,2,3,..., n. If S is a subset of the set $\{1, 2, \ldots, n\}$ of all players, then we can define $v(S)$ as a sum of money that players from S are guaranteed to win in case they decide to form a coalition against all the others. The value of $v(S)$ corresponds to the so-called *zero-sum game*, for which the algorithms (reducing it to an easily solvable linear programming problem) are well known [14]. So we can assume, that there is an algorithm that allows us for a given S to compute $v(S)$.

Without simulation, the problem is intractable. "Without simulation" means that we restrict ourselves to the algorithms that do not use random number generators at all. In other words, we consider only deterministic algorithms, in which the result is uniquely determined by the input data. For such algorithms, reliability makes no sense (either we have the result, or not), so we arrive at the following problem: we must get an estimate X_i for x_i such that $|X_i - x_i| \le cV/n$.

An algorithm that solves this problem must include two kind of computational steps: standard operations with numbers (addition, multiplication, etc) and calls for the procedure that computes $v(S)$. So we arrive at the following definition.

Definitions and the main result. By a *game* we mean a pair of a positive integer n and a program v that for every given subset S of $\{1, 2, ..., n\}$ computes a number $v(S)$ so that: $v(\phi) = v(\{a\}) = 0$ and $v(S \cup T) \ge v(S) + v(T)$ if $S \cap T = \phi$ [14]. Suppose that $c > 0$. By an *c−precise algorithm* we mean an algorithm (e.g., a Turing machine) that uses a procedure v as an oracle (see, e.g., [8]) and computes the value X_i for which $|X_i - x_i| \le cV/n$. Suppose that t_e and t_0 are positive real numbers. By a *running time* of an algorithm for a game v we mean $t_e N_e + t_0 N_0$, where N_e is the total number of calls of v, N_0 the total number of other computational steps. By the *time complexity* (or *worst-case time complexity*) of an algorithm we mean a function, that assigns to every integer n the biggest running time of this algorithm on all games with $\le n$ players.

THEOREM. *If $c < 1/4$, then the time complexity of every c−precise algorithm is $\ge t_e 2^n/(4\sqrt{n})$.*

Comment. So if we do not use simulations, exponentially increasing running time is inevitable, and for big n the problem is intractable.

Proof. We will prove this result using reduction to a contradiction. Suppose that there exists a c−precise algorithm with time complexity $< t_e 2^n/(4\sqrt{n})$. This means that for every game v, its running time is always less than this number. So, for every game, it asks for the values of $v(S)$ for $< 2^n/(4\sqrt{n})$ different value of S.

Let us consider a game v in which $v(S) = 1$ if $|S| > n/2$, else $v(S) = 0$. For this game, $x_1 = x_2 = ... = x_n = 1/n$. In this game, the difference $v(S \cup \{i\})) - v(S)$ is nonzero only if S includes precisely half of the players. Therefore, we are interested only in the sets S with $|S| = n/2$ (and in the corresponding sets $S \cup \{i\}$). The total number of such sets is equal to $C(n, n/2) = n!/((n/2)!(n/2)!)$. Applying Stirling formula $n! \approx (n/e)^n (C_1^n)^{1/2}$, where $C_1 = 2\pi$, we conclude that the total number of such sets is less than $2^n/\sqrt{n}$. Therefore, our hypothetic algorithm asks for the value of $v(S)$ for less than a quarter of such sets S.

Let us now construct another game v' with the same values of $v(S)$ for all the sets S for which $v(S)$ is called, and essentially different values for other S. For such v' the result X_i of applying the algorithm to v and v' will be the same: for all S for which it asks for $v(S)$, the algorithm gets the same values. Take $v'(S) = 0$ if $|S| \le n/2$, $v'(S) = 1$ if $|S| > n/2 + 1$; if $|S| = n/2 + 1$ and S does not include 1, then $v'(S) = 1$; if $|S| = n/2 + 1$ and $1 \in S$, then $v(S) = 1$ iff this S is one of those for which $v(S)$ was asked, else $v'(S) = 0$.

Informally, the voice of the player 1 is ignored in all the cases, except those cases when the algorithm checks it.

It's easy to check that v' satisfies the inequalities and is therefore, a game. Let's denote by x' the Shapley value of this game. The difference $v(S \cup \{1\}) - v(S)$ is different from 0 (and equal to 1) only if $|S| = n/2$ and S is one of the "checked" coalitions. So, all

In case d_i correspond to systematic errors, we want to know the maximum possible value D of the expression $|d|$, where $d = f(x_1, \ldots, x_n) - f(X_1, \ldots, X_n)$ and each X_i takes all possible values from the interval $[x_i - D_i, x_i + D_i]$.

In case d_i correspond to random errors, we want to know the value $D = (E(d^2))^{1/2}$, where d is defined as above, $X_i = x_i + d_i$ and d_i are n independent normally distributed random variables with average value 0 and standard deviation D_i.

The resulting problem is intractable. In [7] it was proved that in the general case, this problem is intractable, or, using the precise mathematical term, NP-hard [8]. In other words, if we could solve this problem in reasonable computation time (at least not exceeding some polynomial of the length of the input), then we would be able to solve all discrete problems in polynomial time, which is generally believed to be impossible.

It is intractable even for the simplest possible functions f (e.g., for polynomials).

How this problem is usually solved. The usual engineering approach takes into consideration the fact that the value D that we are looking for is just the error estimate, so there is no need to compute it with great precision. A statement that "the precision of this measuring device is 10.3%" sounds crazy to an engineer; if we measure with precision about 10% (just 1 decimal digit), there's no sense to argue about the third decimal point in the error.

Engineers and physicists normally express this idea by saying that we can "neglect" terms quadratic in errors d_i. In computational terms this means that in order to compute any function F of d_i we expand the function in a Taylor series and neglect all quadratic (or higher order) terms, so that only linear terms remain.

In our case we are interested in the function d, that is described by above-given equation. The values x_i are given, so the only thing that depends on d_i is $X_i = x_i - d_i$. Substituting this expression into that equation, we get $d = f(x_1, \ldots, x_n) - f(x_1 - d_1, \ldots, x_n - d_n)$. Expanding this expression into a power (Taylor) series with respect to d_i and retaining only terms that either do not contain d_i at all or are linear in d_i, we conclude that $d = f_{,1}d_1 + \ldots + f_{,n}d_n$, where $f_{,i}$ means partial derivative of the function f with respect to the variable x_i and evaluated at the point $(x_1, \ldots, x_n)$.

How does this representation help to solve the computational problem of finding the value D?

In this case, both the maximization problem (in case of the systematic errors) and the problem of finding the mean square (in case of random errors) become easier to solve. In case of systematic errors $D = \sum_i |f_{,i}| D_i$. In case of random errors, $D = \sqrt{\sum_i (f_{,i})^2 D_i^2}$. So, if we known the derivatives, then it's easy to estimate D. How to compute the derivatives of f? If f is an analytical expression (like in case of Ohm's law), then we can differentiate this expression analytically. In other cases, we can apply numerical differentiation: $f_{,i} = (f(x_1, ..., x_{i-1}, x_i + s, x_{i+1}, ..., x_n) - f(x_1, x_2, ..., x_i, ...x_n))/s$ for some small s.

This is still a very time-consuming method. To compute n values of $f_{,i}$, we must apply f n times. Therefore, the time that is necessary to estimate the precision of y is n times bigger than the time that is necessary to compute the value y itself. For complicated information processing, where we process 10^4 or more values of x_i, the resulting procedure becomes practically non-feasible. Let us show that simulation can make this problem feasible.

Simulation method for random errors. We can generate s_i which are distributed

the terms in Shapley's formula with $|S| \neq n/2$ are 0, and according to Shapley's formula, $x'_1 = (1/n)(N' - N)$, where N is a total number of all the sets S with $|S| = n/2$, that do not contain 1, and N' is a number of those sets, for which $v(S \cup \{1\}) = 1$. The number N is equal to the half of the total number of sets with $|S| = n/2$, and according to our argument N' is less than 1/4 of this total number, therefore, $x'_1 < (1/n)(1/4)/(1/2) = 1/(2n)$.

Now let's use the fact that the hypothetic algorithm is c-precise for $c < 1/4$. For the game v with equal outputs, it means that $|x_i - X_i| < cV/n < 1/(4n)$ and $x_i = 1/n$. Therefore, $3/(4n) < X_i < 5/(4n)$. In particular, $X_1 > 3/(4n)$.

For v', we also have $|X_1 - x'_1| \leq cV/n < 1/(4n)$, hence $x'_1 > X_1 - 1/(4n) \geq 3/(4n) - 1/(4n) = 1/(2n)$, and $x'_1 > 1/(2n)$, but we have proved that $x'_1 < 1/(2n)$. This contradiction proves that our assumption was false, and computational complexity cannot be smaller than $t_* 2^n/(4\sqrt{n})$. Q.E.D.

CONCLUSION

We prove that although the problem of finding an estimate for a desired physical quantity y from the measured data x_i is often very complicated, but the problems of computing precision and reliability of these estimates and the problem of choosing an appropriate decision in case of conflicting interests, are in general even more complicated.

We develop simulation-based algorithms that help to avoid exponentially increasing computation time for these three problems (precision, reliability, and decision making). These algorithms are parallelizable, and we describe a special architecture that makes this parallelization most efficient.

ACKNOWLEDGEMENTS

This work was supported by a NSF Grant No. CDA-9015006, NASA Research Grant No. 9-482 and the Institute for Manufacturing and Materials Management grant. The authors are greatly thankful to S. Berkovich (George Washington University) for valuable discussions.

REFERENCES

1. Dantsin, E. and Kreinovich, V. "Probabilistic inference in prediction systems," *Academy of Sciences Doklady*, 1989, Vol. 307, No. 1, pp. 17-21 (in Russian); English translation: *Soviet Mathematics Doklady*, 1990, Vol. 40, No. 1, pp. 8-12.

2. Kreinovich, V., Bernat, A., Villa, E. and Mariscal, Y. "Parallel computers estimate errors caused by imprecise data," *Proceedings of the Fourth ISMM (International Society on Mini and Micro Computers) International Conference on Parallel and Distributed Computing and Systems*, Washington, 1991, Vol. 1, pp. 386-390. Reprinted in *Interval Computations*, 1991, Vol. 1, No. 2, pp. 31-46.

3. Cortes, A. L. "Probabilistic methods that are parallelizable", *Abstracts of the First El Paso ACM Computer Science Students Conferences*, El Paso, TX, 1991.

4. Kosheleva, O., Kreinovich, V., and Villaverde, K. "A polynomial-time algorithm for calculating the Shapley vector (Monte-Carlo method)", University of Texas at El Paso, Computer Science Department, Technical Report UTEP-CS-90-18, 1990.

5. Clifford, A. A. "Multivariate Error Analysis", J. Wiley & Sons, New York, 1973.

6. Fuller, W. A. "Measurement Error Models", J. Wiley & Sons, New York, 1987.

7. Gaganov, A. A. "Interval Estimates are NP-Hard", M.S. thesis, Leningrad University, 1981 (in Russian). English translation in *Cybernetics*, 1985.

8. Garey, M., and Johnson, D. "Computers and Intractability: a Guide to the Theory of NP-Completeness", Freeman, New York, 1979.

9. Kreinovich, V. and Pavlovich, M. I. "Error Estimate of the Result of Indirect Measurements by Using a Calculational Experiment", *Measurement Techniques*, Vol. 28, pp. 201-205, 1985.

10. Hardin, J. and Taylor, J. "A Distributed Parallel Processing System Can Solve Today's Complex Automated Machine Control Problems", *Proc. 19th Annual International Programmable Controllers Conference*, 1990.

11. Shafer, G. and Pearl, J. (eds.) "Readings in Uncertain Reasoning", Morgan Kaufmann, San Mateo, CA, 1990.

12. Orponen, P. "Dempster's rule of combination is #-P complete, *Artificial Intelligence*, Vol. 44, pp. 245-253, 1990.

13. Wilson, N. "A Monte-Carlo algorithm for Dempster-Shafer belief", *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, Los Angeles, CA, 1991.

14. Owen, G. "Game theory". Academic Press, N. Y., 1982.

15. Shapley, L. S. "A value for n-person games", In: *Contributions to the Theory of Games, II*, ed. by H. W. Kuhn and A. W. Tucker. Princeton, Princeton University Press, 1953, pp. 307-317.

KEYWORDS

Parallel computations, Monte-Carlo methods, error estimation, probabilistic reasoning, game theory, Shapley value.