URC97076

How the Theory of Computing Can Help in Space Exploration

Vladik Kreinovich and Luc Longpré

Center for Theoretical Research and its
Applications in Computer Science (TRACS)
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968
email {vladik,longpre}@cs.utep.edu

Abstract

The opening of the NASA Pan American Center for Environmental and Earth Sciences (PACES) at the University of Texas at El Paso made it possible to organize the student Center for Theoretical Research and its Applications in Computer Science (TRACS).

In this abstract, we briefly describe the main NASA-related research directions of the TRACS center, and give an overview of the preliminary results of student research.

1 Preamble

The opening of the NASA Pan American Center for Environmental and Earth Sciences (PACES) at the University of Texas at El Paso made it possible to organize the student Center for Theoretical Research and its Applications in Computer Science (TRACS).

This center was one of the sponsors of the student regional conference in Computational Sciences SC-COSMIC (October 1996, El Paso, TX) [7], and it was instrumental in bringing the 1997 Annual ACM Symposium on Theory of Computing to El Paso.

The main emphasis of this center is not only on today's engineering problems, but also on the fundamental problems that can eventually be of use in different areas of practice, in particular, in the area of space exploration. In this abstract, we briefly describe the main NASA-related research direction of the TRACS center, and give an overview of its results.

2 Space-related computations: specific features, specific problems

How are computations and data processing related to space research different from computations in other application areas? There are many differences, regarding the *input,* the *computational resources* available for these computations, and the desired *result.*

Input. Many on-Earth computations deal with the reasonably well known areas, for which a lot of information already exists. Most space missions are, to the large extent, missions into the unknown. As a result, in computations related to space mission, there is usually a much smaller amount of a *priori* information.

The *amount of information* coming from the space missions is, usually, also *much smaller* than usual. There are several reasons for that; the three main reasons are:

- first, each additional sensor on-board costs a lot to launch and to maintain, and therefore, the designers try to keep the total number of sensors reasonably low;
- second, the more accurate and more reliable sensors are usually more complicated, weigh more, require more energy and other resources, and are usually less robust in the sense that may require too many

additional resources to protect them against the often hostile space environment; therefore, the sensors that are actually launched are often not the most accurate and the most reliable;

- finally, each additional bit of information has to be collected and transferred to the Earth, creating an additional burden on the spaceship communication system.

Computational resources. Processors launched into space (or placed on board of a planetary rover) must take as small amount of space, energy, etc., as possible. As a result, we must be able to perform all necessary computations on the available *limited* computational resources.

Desired result. Since space missions are usually very expensive, every potential error is very costly, and in manned space missions, errors are simply inadmissible. So, space-related computations must be 100% *reliable.*

3 Due to these specific features, problems of space-related computations are, in general, computationally intractable

Interval computations. First of all, in space-related computations, we need the *guaranteed* results. Together with the fact that the input comes from not 100% accurate sensors, this means that we have to estimate the accuracy of the results of data processing, while in traditional computing, accuracy is usually not an issue.

This bring up the importance of computations that take this inaccuracy into consideration and lead to guaranteed results. Due to the fact that we want not simply a *numerical estimate* of the desired quantity, but an *interval* within which the actual values of this quantity is *guaranteed* to lie, the corresponding area of theory of computing is called *interval computations.*

The problems become computationally intractable. In interval computations, there are important theoretical results, but it turns out that if we take the inaccuracies of the input data into consideration and require the results to be 100% reliable, then even the simplest tasks such as solving a liner system become, in general, computationally intractable (N P-hard) [1 O, 1].

What can we do?

4 How can we solve computationally intractable problems?

First approach: finding classes of solvable problems. The fact that the problem is computationally intractable means that there is no general algorithm for solving these problems. Therefore, a natural approach is to find *classes* of problems for which there are feasible algorithms. In this, theory of computing, with its large experience of designing new algorithms, can be of great help.

In particular,

- for general interval data processing, theory-motivated algorithms are presented in [11];
- for a specific problem of chemical i dentificat ion, a problem that is very important for space exploration, algorithms based **on** theoretical ideas are presented in [4, 3].

Second approach: using human intelligence. In many real-life situations, the problem that we must solve does not belong to any of the classes for which feasible algorithms are known. In some of these cases, we know that human experts can often solve these problems really well. How can we describe the experience of these experts in such a way that the computer will be able to understand this knowledge and use it in data processing'?

Here, there are three types of theoretical problems:

- is a certain type of formalism *sufficient* for a certain class of problems?
- what is a computational *complexity* of using this formalism?
- how can we make the resulting computations faster?

In TRACS, all three types of problems are analyzed:

- *Sufficient.* In [12], it is shown that one of the logic programming formalisms is sufficient to represent reasonably general type of knowledge. In [13], a similar problem of sufficiently is analyzed for *robots.*

- *Complexity.* In [5], it is shown that even if *we* manage to describe all the knowledge in terms of an expert system, the computational problems related to *using* this expert system are, in general, computationally intractable.

- *Faster.* In [9], a fast algorithm is developed for an important *particular* class of *fuzzy data processing* algorithms.

Third approach: trying to make the existing computers faster. If the existing algorithms are too slow, and no expert knowledge is available, then we can try to speed up the computers.

The design of a faster computer usually involves lots of engineering problems and heuristic methods, but, as it is shown in [8], theory of computing, with its experience of evaluating the computational abilities and computational times of different hypothetic computational devices, can definitely help in selecting the most promising design of a real-life computer.

Fourth approach: looking for radically new ways of computing. If the existing computers cannot solve our problems, then maybe some radically new ways of computing will be more helpful. In particular, one way to speed up computations is to miniaturize the computers.

At TRACS, we have looked into the potential abilities of *chemical computers [6]* that operation on the level of molecules and *quantum computers [2]* that operate on the level of elementary particles. Baaed on the results of this analysis, both approaches seem to be very promising.

Acknowledgments. This work was partially supported by NSF Grants No. CCR-9211174 and EEC-9322370, and by the NASA Pan American Center for Environmental and Earth Studies (PACES). The authors are thankful to Ann Gates and Scott St arks for their help and encouragement.

References

[1] G. Alefeld, M. Koshelev, and G. Mayer, "Fixed Future and Uncertain Past: Theorems Explain Why It Is Often More Difficult To Reconstruct the Past Than to Predict the Future", *These Proceedings.*

[2] A. Beltran, V. Kreinovich, and L. Longpré, *QFT + NP = P: Quantum Field Theory (QFT): A Possible Way of Solving NP- Complete Problems* in *Polynomial Time,* University of Texas at El Paso, Department of Computer Science, Technical Report UTEP-CS-96-45, November 1996; LaTeX file available as **ftp:** //es. **utep.edu/pub/report s/tr96-45.** t **ex**.

[3] A. Beltran and J. Salvador, "The Ulam Index: Methods of Theoretical Computer Science Help in Identifying Chemical Substances", *These Proceedings.*

[4] A. Beltran and J. M. Salvador, "The Ulam index", *Abstracts of the Second SC-COSMIC Conference in Computational Sciences, October 25-.27, El Paso, TX,* Rice University Center for Research on Parallel Computations and University of Texas at El Paso, 1996, p. 6.

[5] L. Chee, *Computing the Value of a Boolean expression with intervals is NP-hard,* Master Thesis, Department of Computer Science, University of Texas at El Paso, 1996.

[6] B. Cloteaux, On *the Computational Power of Using Chemical Reactions,* Master Thesis, Department of Computer Science, University of Texas at El Paso, 1996.

[7] *Abstracts of the Second SC-COSMIC Conference in Computational Sciences, October 25–27, El Paso, TX,* Rice University Center for Research on Parallel Computations and University of Texas at El Paso, 1996.

[8] M. Hampton, "How difficult is it to add 1? A pedagogical example of how theory of computing maybe useful", *These Proceedings.*

[9] M. Hampton and O. Kosheleva, "Fast Fuzzy Arithmetic Operations", *These Proceedings.*

[10] P. Kahl, Solving *Narrow-Interval Linear Equation Systems Is NP-Hard,* Master Thesis, Department of Computer Science, University of Texas at El Paso, 1996.

[11] M. Koshelev and P. Taillibert, "Optimal Approximation of Quadratic Interval Functions", *These Proceedings.*

[12] O. Kosheleva, "An Arbitrary First Order Theory Can Be Represented by a Logic Program: a Theorem", *These Proceedings.*

[13] M. Nogueira, "What Can Robots Do? Towards Theoretical Analysis", *These Proceedings,*