Hypothetic Paraparticles and How They Can Potentially Speed Up Computations

Sebastian Balderrama, Daniel Marin, and Vladik Kreinovich

Abstract While modern computers are fast, their speed is not sufficient for many practical problems. Thus, we need faster computers. A significant part of computation time is spent of moving data from one location to the other – e.g., from a memory cell to the processor. According to relativity theory, all communications are limited by the speed of light – and computer communications are already close to that limit speed. Thus, the only way to further speed up computations is to decrease the size of a memory cell and of other components of a computer, The ultimate decrease is when we use the smallest possible object as a memory cell – an elementary particle. For this purpose, we cannot use any of the known elementary particles, since they have a single stable state – while to store a bit, we need an object with two stable states that would represent 0 and 1. Recently, researchers came us with a suggestion that there are elementary particles that have two stable states; such particles are known as *paraparticles*. Thus, paraparticles can serve as bit-storing elements – and so, their use can speed computations. In this paper, we show that paraparticles can speed computations even further – namely, they may be able to help us solve known NP-hard problems in feasible time.

1 Formulation of the problem

Modern computers are fast, but we need even faster computers. Modern computers are fast, but there are many practical problems for which modern computers are not fast enough. For example, modern computers can reasonably reliably predict tomorrow's weather. Such a prediction is based on solving the Navier-Stokes equa-

Sebastian Balderrama, Daniel Marin, and Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: sbalderrama3@miners.utep.edu, djmarin1@miners.utep.edu, vladik@utep.edu

tions describing all atmospheric phenomena. The corresponding computations take several hours on a high-performance computer.

The same equations describe not only regular weather situations, they also describe extreme atmospheric effects like tornadoes. In principle, in case of tornadoes, we can use the same algorithms to solve these equations, and predict the direction in which a tornado will move in the next 15 minutes. The problem is that tornadoes are much faster than the usual atmospheric phenomena – this is, by the way, why tornadoes are so dangerous and why they cause so much destruction. As a result, the same relative change that takes a day or so for the usual weather, for a tornado, happens in 15 minutes. Because of this, predicting where a tornado is going in 15 minutes takes as much computation time as predicting tomorrow's weather – which is several hours on a high-performance computer. For predicting tomorrow's weather, several hours of computation make sense. However, for predicting where a tornado will go in the next 15 minutes, the hours-long prediction makes no sense: we will "predict" way after the tornado has already moved. To predict the dynamics of a tornado, we therefore need faster computers.

How can we make computers faster? One of the main factors that limits computer speed is computer size. For example, the size of a usual laptop is about 30 cm. If we divide this size by the speed of the fastest possible process – speed of light $c = 300\ 000$ km/sec, we conclude that we need at least 1 nanosecond for a signal to travel from one location on a computer to another. During this time, the usual 4 GHz laptop already performs 4 computational steps. So, to make computers faster, we need to make them smaller.

Next step – quantum computing. To make computers smaller means, in particular, to make every bit-storing element in a computer memory smaller. Already such elements contain a relatively small number of atoms – hundreds or thousands. Reducing the size even further will lead to elements consisting of a few atoms – or even of a single atom. This will bring us into the microworld, where all the physical processes follow quantum physics (and not the usual Newtonian physics). Thus, we need computing under quantum conditions, i.e., quantum computing; see, e.g., [3].

What next? Suppose that we have managed to reduce the size of a bit-storing element to a single atom. What next? Atoms consist of elementary particles, so a natural idea seems to be to use elementary particles as bit-storing elements. (And, by the way, there is nothing beyond elementary particles, so this will be the final size reduction.)

But how can we use elementary particles as bit-storing elements? A usual way to store a bit is to have an element that has two (or more) stable states, i.e., states with the smallest possible energy. One of these states represents 0, another stable state represent one. However, all known elementary particles have a single stable state, i.e., the state with the smallest possible energy; see, e.g., [1, 5]. So what can we do?

Paraparticles and how they can help. Recently, researchers raised the possibility of having particle that have two different minimum-energy states; these particles

are called *paraparticles* [6, 7]. The two-stable-states feature makes paraparticles a perfect tools for storing one bit of information. If we can use only one elementary particle to store a bit (instead of several molecules), that will make computers drastically smaller and thus, faster.

What else can do with paraparticles computation-wise. Paraparticles are a next step after quantum computing. Quantum computing started with the goal to decrease the size of computational devices. However, later on, it turned out that with quantum elements, we can achieve additional speed up (see, e.g., [3]); for example:

- in a quantum-based computer, we can search in an unsorted n-element array in time proportional to $\sqrt{n}$ – while without quantum effects, we cannot guarantee fewer than n steps – otherwise we do not check at least one of the elements and this unchecked element may be the desired one;
- spectacularly, quantum computer are, potentially, able to efficiently factor large integers and thus, decode all currently encrypted messages – whose security is based on the fact that without quantum computers, no efficient factoring method is known.

These successes prompt a natural question: how else can we speed up computing if we use paraparticles?

What we do in this paper. In this paper, we show that, similarly to quantum computing, paraparticles have the potential to further speed up computations: namely, they may be able to solve NP-hard problems in feasible time.

The structure of the paper. We start, in Section 2, with a brief reminder of what are NP-hard problems and why they are important. After that, in Section 3, we describe how paraparticles may be able to solve NP-hard problems in reasonable time.

2 NP-hard problems: a brief reminder

In this section, we will briefly follow the usual description of NP-hardness; see [2, 4] for details.

What is feasible and what is not feasible. The usual description of NP-hard problems starts with describing which algorithms are feasible and which are not feasible. Some algorithms are practically useful – we can them *feasible*. Other algorithms require, for reasonable-size inputs, time which is larger than the lifetime of the Universe.

For example, one way to find a length-n sequence of 0s and 1s that has the desired property is to try all possible binary sequences of length n. However, there are 2^n such sequences, so already for $n = 500$, this algorithm requires $2^{500} \approx 10^{150}$ computational steps. This is not realistic. Even if each of the 10^{90} particles in the Universe is used as a computational devices, and each of them performs each computational step as fast as possible – during the time when light passes through the

smallest elementary particle – the overall computation time will be several orders of magnitude longer than the lifetime of our Universe – which is about 10-20 billion years,

In practice, most algorithms whose running time is limited by some polynomial $P(n)$ of the input length n are feasible – and vice versa, the running time of most known feasible algorithms is bounded by some polynomial. Thus, usually, feasibility is defined as having such a polynomial upper bound on computation time.

This is not a perfect definition: some polynomial-time algorithms are not practically feasible and, vice versa, the running time of some practically feasible algorithms are not bounded by a polynomial. However, such cases are rare and, in any case, this is the best definition of feasible that we have so far.

What problems are we solving? In science and engineering, we usually solve well-formulated problems, i.e., problems for which, once someone comes up with a candidate for a solution, we can check, in feasible (polynomial) time, whether it is indeed a solution. Finding a solution may be difficult. For example, in mathematics, it sometimes takes hundreds of year to find a proof of a hypothesis – but once a detailed proof is given, it is relatively straightforward to check its correctness. In engineering, once we have a design of, e.g., a bridge, it is relatively easy to simulate different loads and thus confirm that the bridge satisfies all the specification – but coming up with such a design can be difficult.

In other words, if we guess a solution, then we can feasibly check that our guess is a solution. In computer science, "algorithms" that include guessing are called *non-deterministic*. Because of this, such problems – for which checking is feasible – are known as Non-deterministic Polynomial, NP for short.

What is NP-hard? Can we solve all NP problems in feasible time? The class of all problems that can be solved in feasible (polynomial) time is usually denoted by P. In these terms, the above question takes the form: is P equal to NP? This is a known open problem. Most computer scientists believe that classes P and NP are different – i.e., that there are problems from the class NP that cannot be solved in feasible time – but no one has proved it so far.

What is known, however, is that in the class NP, there are problems which are as hard as possible – in the sense that all other problems from the class NP can be feasibly reduced to such "hard" problems. These problems are called *NP-hard.* Such problems are most probably not feasible – otherwise, if there existed a feasible algorithm for solving such a problem, then, by using reduction, we could solve all the problems from the class NP in polynomial time and we would thus get P = NP – while, as we have mentioned, most computer scientists believe that P $\neq$ NP.

An example of an NP-hard problem. There are many examples of problems whose NP-hardness have been proven. In this paper, we will use one such example: the *subset sum* problem.

In this problem, we are given a list of natural numbers $s_1, \ldots, s_n$ and a natural number s, and we need to find a subset $S \subseteq \{1, \ldots, n\}$ over which the sum of s_i's is equal to s. This is equivalent to finding $x_i \in \{0, 1\}$ for which

$$\sum_i x_i \cdot s_i = s.$$

How do we usually prove that a new problem is NP-hard. A usual way to prove that a new problem is NP-hard is to prove that one of the known NP-hard problems can be feasibly reduced to this new problem. Indeed, by definition of NP-hardness, each problem form the class NP can be feasibly reduced to the known problem, and – since the known problem can reduced to the known one, we can conclude that every problem from the class NP can be feasibly reduced to the new problem as well. This means exactly that the new problem is NP-hard.

Now we are ready to argue that paraparticles can help us solve NP-hard problems in feasible time.

3 Back to paraparticle

How can we describe paraparticles in precise terms? We know that parapartciles have two different states with the smallest possible value of energy.

We do not know the exact equations describing the paraparticle's energy, it can be any analytical function. In such situations, to get a first approximation to the particle's description, it is reasonable to restrict the general Taylor expansion of the energy function to the first few terms – as long as that allow to explain the basic behavior of the particle. This is a usual strategy in physics; see, e.g., [1, 5].

For many physical systems – such as a pendulum – the first approximation comes from consider quadratic terms. To find the minimum-energy state, we can differentiate the energy function with respect to all unknowns and equate all the resulting derivatives to 0. Differentiating a quadratic function leads to a linear expression, so we get a system of linear equations. Such systems either have a unique solution, or the whole linear space of solutions – but they cannot have just two solutions. So, to describe paraparticles, we need to go beyond quadratic terms.

The next after quadratic are cubic terms. However, a cubic polynomial cannot have a global minimum: in some directions, it reaches infinity and thus, in opposite directions, it tends to $-\infty$. Thus, to describe paraparticles, we need to also take into account the next order terms – i.e., we need to consider 4th order polynomials.

How this can help to solve NP-hard problems. An interesting feature of 4th order polynomials is that for them, finding a global minimum is NP-hard (see proof in the next subsection). So, in general, finding the minimum-energy state of a paraparticle is probably an NP-hard problem. A particle, left to itself, reaches its minimum-energy state – and usually does it fast. So by observing hypothetic paraparticles, we may be able to find, in short time, solutions to an NP-hard problem. By definition, NP-hardness means that we can reduce any problem from the class NP to this problem in feasible time. Thus, paraparticles may lead to feasible algorithms may lead

to a feasible way of solving all the problems from the class NP – i.e., in effect, to solution of all the problems in mathematics, physics, and engineering!

How to prove that minimizing 4th order polynomials is NP-hard. We can prove this by reducing, to this problem, the known NP-hard subset sum problem: we have a list of natural numbers $s_1, \ldots, s_n$ and a natural number s, and we need to find a subset $S \subseteq \{1, \ldots, n\}$ over which the sum of s_i's is equal to s. This is equivalent to finding $x_i \in \{0, 1\}$ for which

$$\sum_i x_i \cdot s_i = s.$$

For each instance of this problem, let us form the following 4th order polynomial:

$$\sum_i (x_i \cdot (1 - x_i))^2 + \left(\sum_i x_i \cdot s_i - s \right)^2.$$

This polynomial is always non-negative, and its minimum is equal to 0 if and only if all the terms in the sum are 0s. In particular, this means that $x_i \cdot (1 - x_i) = 0$ (so either $x_i = 0$ or $1 - x_i = 0$, i.e., $x_i = 1$) and

$$\sum_i x_i \cdot s_i = s.$$

So, if we could minimize this polynomial, we would be able to solve the subset sum problem. This reduction proves that the above minimization problem is also NP-hard.

Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-2225395 (Center for Collective Impact in Earthquake Science C-CIES), and by the AT&T Fellowship in Information Technology.

It was also supported by a grant from the Hungarian National Research, Development and Innovation Office (NRDI), by the Institute for Risk and Reliability, Leibniz Universitaet Hannover, Germany, and by the European Union under the project ROBOPROX (No. CZ.02.01.01/00/22 008/0004590).

References

1. R. Feynman, R. Leighton, and M. Sands, *The Feynman Lectures on Physics*, Addison Wesley, Boston, Massachusetts, 2005.

2. V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, 1998.
3. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, U.K., 2011.
4. C. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994.
5. K. S. Thorne and R. D. Blandford, *Modern Classical Physics: Optics, Fluids, Plasmas, Elasticity, Relativity, and Statistical Physics*, Princeton University Press, Princeton, New Jersey, 2021.
6. Z. Wang and K. R. A. Hazzard, "Particle exchange statistics beyond fermions and bosins", *Nature*, 2025, Vol. 639, pp. 314–318.
7. K. Wright, "Strange swapping behavior defines new particle candidate", *Physics*, 2025, Vol. 18, Paper 11.