4-1-2024

Number Representation With Varying Number of Bits

Anuradha Choudhury
The University of Texas at El Paso, achoudhury@miners.utep.edu

Md Ahsanul Haque
The University of Texas at El Paso, mhaque3@miners.utep.edu

Saeefa Rubaiyet Nowmi
The University of Texas at El Paso, srnowmi@miners.utep.edu

Ahmed Ann Noor Ryen
The University of Texas at El Paso, aryen@miners.utep.edu

Sabrina Saika
The University of Texas at El Paso, ssaika@miners.utep.edu

See next page for additional authors

Comments:
Technical Report: UTEP-CS-24-17

Authors
Anuradha Choudhury, Md Ahsanul Haque, Saeefa Rubaiyet Nowmi, Ahmed Ann Noor Ryen, Sabrina Saika, and Vladik Kreinovich

Number Representation With Varying Number of Bits

Anuradha Choudhury, Md Ahsanul Haque, Saeefa Rubaiyet Nowmi,
Ahmed Ann Noor Ryen, Sabrina Saika, and Vladik Kreinovich

Abstract In a computer, usually, all real numbers are stored by using the same number of bits: usually, 8 bytes, i.e., 64 bits. This amount of bits enables us to represent numbers with high accuracy – up to 19 decimal digits. However, in most cases – whether we process measurement results or whether we process expert-generated membership degrees – we do not need that accuracy, so most bits are wasted. To save space, it is therefore reasonable to consider representations with varying number of bits. This would save space used for representing numbers themselves, but we would also need to store information about the length of each number. In view of this, the first natural question is whether a varying-length representation can lead to a drastic decrease in needed computer space. Another natural question is related to the fact that while potentially, allowing number of bits which is not proportional to 8 bits per byte will save even more space, this would require a drastic change in computer architecture, since the current architecture is based on bytes. So will going from bytes to bits be worth it – will it save much space? In this paper, we provide answers to both questions.

1 Formulation of the problem

How real numbers are represented in a computer now. Computers usually use the same number of bits to store all real numbers: 64 bits, which is 8 bytes.

This length potentially enables us to represent real numbers with relative precision up to 2^{-64}, which is approximately 10^{-19}.

Anuradha Choudhury, Md Ahsanul Haque, Saeefa Rubaiyet Nowmi, Ahmed Ann Noor Ryen, Sabrina Saika, and Vladik Kreinovich
University of Texas at El Paso, 500 W. University, El Paso, Texas 79968, USA
e-mail: achoudhury@miners.utep.edu, mhaque3@miners.utep.edu, srnowmi@miners.utep.edu, aryen@miners.utep.edu, ssaika@miners.utep.edu, vladik@utep.edu

But do we need that many bits? In some cases, we *do* need this precision – and sometimes, we even need double precision corresponding to 128 bits. However:

- In many practical situations, we process measurement results, that are measured with precision 1% or even less; see, e.g., [9]; in such cases, we do not need that many bits, so additional bits are simply wasted.
- In other practical situations, we process degrees of confidence provided by experts; see, e.g., [1, 5, 6, 7, 8, 10]. These degrees are known even with less accuracy, usually the accuracy about 10%, so even more bits are wasted when we use the usual computer representation of real numbers.

It is therefore reasonable to consider number representations with *varying* number of bits. Such representations have indeed been proposed; see, e.g., [2, 3].

Resulting questions. The above idea – of using number representations with varying number of bits – leads to several questions.

First question: how much space will we save? The first question is: how much space do we save?

- On the one hand, we need fewer bits to store each number.
- On the other hand, we will need to store, for each number, information about its length – which also takes a few bits.

Second question: shall we abandon byte structure? The second question is related to the fact that in a computer, bits are usually organized into bytes. From this viewpoint, it is easier to design a computer in which real numbers can use 1, 2, etc. bytes than to allow also any number of bits – including the number of bits that does not divide by 8 and thus, does not constitute several bytes.

The bit-representation complication may be worth it, if it allows us to save a significant portion of memory. So, the second question is: how much memory do we save if we use bits and not bytes?

What we do in this paper. In this paper, we provide answers to both questions.

2 How do we answer these questions: our methodology

Challenge. Strictly speaking, to answer these questions, we need to know how frequently we encounter numbers of different length. At present, this information is not available. So what can we do?

Let us use Laplace Indeterminacy Principle. In such situations, when we do not know probabilities of different situations, it makes sense to use what is called *Laplace Indeterminacy Principle* (see, e.g., [4]):

- since we have no reason to believe that some situations (in our case, some lengths) lengths are more frequent than others,

- it makes sense to assume that all possible lengths are equally frequent.

Let us use this assumption to answer both questions.

3 Case of byte representation

What are the options? In the byte representation, a number can occupy 1, 2, . . . , 8 bytes.

What are the probabilities of these options? In line with the Laplace Indeterminacy Principle – as described in the previous section – we assume that each of these eight cases has the same probability 1/8.

In this case, what us the average number of bits taken by a real number. Based on the lengths and probabilities of different options, we can conclude that the average length of the real number is equal to

$$\frac{1+2+\ldots+8}{8} = 4.5 \text{ bytes} = 36 \text{ bits.}$$

How many additional bits do we need to store information about each number's length. In general, with k bits, we can describe 2^k different options. In our case, we have 8 possible options, and $8 = 2^3$. Thus, to store information about the length, we need 3 bits.

How many bits per number do we need overall. For each real number:

- we need, on average, 36 bits to store the number itself, and
- we need 3 bits to store the information about the number's length.

So overall, we need $36+3 = 39$ bits.

How much space do we save this way. The resulting amount of 39 bits is much smaller than the current 64 bits – about 40% smaller.

So, is this worth doing? Since this representation can potentially save a lot of space, the answer to the first question is: yes, number representation with varying number of bytes is worth pursuing.

4 Case of bit representation

What are the options? In the bit representation, a number can occupy 1, 2, . . . , 64 bits.

What are the probabilities of these options? In line with the Laplace Indeterminacy Principle – as described earlier – we assume that each of these 64 cases has the same probability 1/64.

In this case, what us the average number of bits taken by a real number. Based on the lengths and probabilities of different options, we can conclude that the average length of the real number is equal to

$$\frac{1+2+\ldots+64}{64} = 32.5 \text{ bits.}$$

How many additional bits do we need to store information about each number's length. In general, with k bits, we can describe 2^k different options. In our case, we have 64 possible options, and $64 = 2^6$. Thus, to store information about the length, we need 6 bits.

How many bits per number do we need overall. For each real number:

- we need, on average, 32.5 bits to store the number itself, and
- we need 6 bits to store the information about the number's length.

So overall, we need $32.5 + 6 = 38.5$ bits.

How much space do we save this way. The difference between the resulting average number of 38.5 bits and 39 bits corresponding to byte representation is very small – about 1%.

So, is this worth doing? The bit representation requires a drastic redesign of computer architecture but saves only a tiny amount of space. So, the answer to the second question is: no, it is probably not worth doing.

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).

References

1. R. Belohlavek, J. W. Dauben, and G. J. Klir, *Fuzzy Logic and Mathematics: A Historical Perspective*, Oxford University Press, New York, 2017.

2. M. Ceberio, C. Lauter, and V. Kreinovich, “Just-in-accuracy: mobile approach to uncertainty”, *Journal of Mobile Multimedia*, 2024, Vol. 20, No. 3, to appear.
3. J. F. Gustafson, *The End of Error: Unum Computing*, Chapman & Hall/CRC, Boca Raton, Florida, 2015.
4. E. T. Jaynes and G. L. Bretthorst, *Probability Theory: The Logic of Science*, Cambridge University Press, Cambridge, UK, 2003.
5. G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, New Jersey, 1995.
6. J. M. Mendel, *Explainable Uncertain Rule-Based Fuzzy Systems*, Springer, Cham, Switzerland, 2024.
7. H. T. Nguyen, C. L. Walker, and E. A. Walker, *A First Course in Fuzzy Logic*, Chapman and Hall/CRC, Boca Raton, Florida, 2019.
8. V. Novák, I. Perfilieva, and J. Močkoř, *Mathematical Principles of Fuzzy Logic*, Kluwer, Boston, Dordrecht, 1999.
9. S. G. Rabinovich, *Measurement Errors and Uncertainty: Theory and Practice*, Springer Verlag, New York, 2005.
10. L. A. Zadeh, “Fuzzy sets”, *Information and Control*, 1965, Vol. 8, pp. 338–353.