The Challenge of Hyper-Spectral Satellite Imaging and Integer-Valued Fuzzy Sets

Maria Beltran, Vladik Kreinovich, and Scott A. Starks

Abstract— **Satellite images already produce huge amounts of data, which makes their processing a serious computational challenge. This problem will become even more complicated with the launch of multi-spectral Earth-imaging satellites that will increase the amount of information by at least two orders of magnitude. With such a huge amount of information, it is necessary to come up with data processing methods that are as fast as possible. In particular, we show that for fuzzy processing techniques, this leads to the necessity to use integer-valued fuzzy sets.**

Keywords— **Satellite imaging, multi-spectral, computational complexity, integer-valued fuzzy sets**

I. Hyper-Spectral Satellite Imaging: Challenges

A. Satellite Imaging

Nowadays, satellite imaging is one of the most important sources of geographical, geophysical, and environmental information. Satellite images can determine the amount and type of vegetation, the geological type of the underlying soils (and often, of the minerals below), etc.

However, with the current satellite images, it is sometimes difficult to decide what exactly we observe, because the existing Earth-sensing satellites, such as Landsat, only take the images at a few (≤ 7) frequencies.

B. An Example of a Problem in Which a Satellite Image is Currently not Sufficient: Kaolinite vs. Dickite

Based on the (inevitably imprecise) measurements on the few frequencies, it is difficult, e.g., to distinguish between kaolinite and its rare amorphous but chemically similar forms such as dickite.

Kaolinite and dickite are the principle ingredients of *kaolin*, a soft white-clay mineral that is an essential ingredient in the manufacture of china and porcelain and is also widely used in the making of paper, rubber, paint, and many other products (see, e.g., [32]). It is also used in medicine: e.g., in the treatment of diarrhea, kaolin powder is the most widely used absorbent powder. Due to kaolin's importance, it is desirable to determine not only its *presence*, but its *type* as well.

Since crystal-based kaolinite and amorphous dickite are chemically similar, their spectra are very similar. Therefore, currently, in order to distinguish between these two minerals, we have to complement satellite images with geophysical and radar data (see, e.g., [26] and [27]).

C. Hyper-Spectral Satellite Imaging

To produce more data, NASA is planning to launch the imaging satellites of the new generation, satellites that will have the ability to map the Earth on up to 500 optical frequencies. These coming satellites are nicknamed *Lewis* after the famous 19 century US geographer.

From the resulting multi-spectral images, it is, in principle, possible to determine many characteristics of soil and vegetation without using additional data; see [29] and [28]. For example, it is, in principle, possible to distinguish between kaolinite and dickite because from hyper-spectral images, we can extract spectra in each point, i.e., the dependence of its brightness $I(f)$ on the frequency f; the corresponding spectra, although similar, have different number of local maxima.

D. The Challenge

This two orders of magnitude increase in the amount of processed data makes processing this data an extremely difficult task.

The complexity of data processing is a difficult problem for many different application areas, but is especially difficult for areas in which it is relatively easy to get new data. *Environmental* and earth studies, the area for which satellite images are mainly used, is one of such areas: lots of relatively *easily accessible data* come from satellites, and processing this data is already extremely difficult.

This problem is going to become even more acute with the launch of the new satellites. Therefore, when designing algorithms for processing this data, we must use computational methods that are as fast as possible.

Among important data processing methods are fuzzy methods, that enable us to process expert information together with measurement results. In this paper, we will discuss how we can make fuzzy data processing methods as fast as possible.

M. Beltran and V. Kreinovich are with the Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968, USA. E-mails: {mbeltran,vladik}@cs.utep.edu.

S. A. Starks is with the NASA Pan-American Center for Environmental and Earth Studies, University of Texas at El Paso, El Paso, TX 79968, USA. E-mail: sstarks@utep.edu.

This work was supported in part by NASA under cooperative agreement NCCW-0089, by NSF under grants No. DUE-9750858 and EEC-9322370, and by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518.

II. Speeding Up Fuzzy Data Processing

A. Main Idea: Let Us Use Directly Hardware Supported Operations

Moore's law: computers are fast. Modern computers are very fast, and they are getting faster and faster. A known empirical law called Moore's law says that the computer speed doubles every 18 months. Nowadays, a simple PC can now perform approximately 200 million operations per second.

Directly hardware supported operations are very fast; mathematical operations are much slower. The impressive numbers in computer advertisements may create an erroneously over-optimistic picture.

As anyone who is doing data processing knows, these impressive numbers refer only to computer cycles, i.e., to computer operations that are directly hardware supported; the speed with which we can process real data (e.g., arrays, or floating point numbers) is much slower.

Conclusion: we must stay as close to directly hardware supported operations as possible. From the above remark, we can make a following natural conclusion: if we want to speed up computations on the existing computer, then we must try to stay as close to the (real fast) directly hardware supported operations as possible.

In order to apply this idea to fuzzy data processing, let us briefly recall what data types and what operations are usually hardware supported, and which of these operations are faster.

Historical comment. The problem of choosing the computationally simplest fuzzy operations is not completely new: e.g., in the paper [22], the simplest operations are described under the assumption that we are dealing with numbers from the interval $[0,1]$. However, the computational challenge of the multi-spectral satellite imaging is such that simply choosing the simplest operations with real numbers is not enough: we need to go *beyond* that and simplify the data type as well.

B. Which Computer Operations are Directly Hardware Supported: Brief Reminder

The main hardware supported computer operations are *logical* operations, i.e., bitwise "and", "or", "not", and *arithmetic* operations (a computer processor is even sometimes called ALU, short for arithmetic-logical unit).

Logical operations, with binary ("true"-"false") data, are definitely the fastest. However, since we are talking about using *fuzzy* logic, in which we have additional "truth values", we cannot use these operations. So, we have to use operations with *strings* of 0's and 1's, operations that are usually called *arithmetic* operations.

The simplest and fastest hardware supported operation with strings of 0's and 1's is not even, strictly speaking, an arithmetic operation: because it does not perform any true arithmetic operations, it simply compares the two given strings and returns the largest (or the smallest) in lexicographic order. If we interpret these strings as natural numbers, then this comparison simply turns into a comparison of two given integers.

The next fastest arithmetic operation (and the first truly arithmetic one) is *unary minus* that transform a number x into $-x$, followed by *addition of integers*. All other arithmetic operations are reduced to additions and unary minuses:

- integer *subtraction* $a-b$ is usually implemented as $a+(-b)$;
- integer *multiplication* is based on several shifts and additions (the computer actually multiplies the numbers digit-by-digit, just like we would have done it by hand);
- integer *division* is implemented as a sequence of subtractions (just as we divide numbers by hand);
- addition and multiplication of *fixed point real numbers* is usually done, crudely speaking, as follows:
  - first, we represent each fixed-point real number as a fraction,i.e., as a ratio of a generic integer and of an integer of the type 2^n;
  - second, we add and multiply these fractions by using the standard rules of fraction arithmetic (i.e., performing several necessary arithmetic operations with integers);
  - finally, we transform the resulting rational number back into a fixed-point real number format;
- operations with *floating-point* real numbers are implemented in pretty much the same manner.

All these operations are usually hardware supported, and therefore fast; however, of course, if an operation A consists of several operations B, A is slower than B.

C. The Resulting Choice

We need integer-valued fuzzy logic. From this description, it follows that the data type for which the operations is the integer type. So, ideally, we should implement truth values as *integers*, i.e., we should consider *integer-valued fuzzy logic.*

Can we simply use all integers? The first natural idea is to simply use *all* integers.

According to the main ideas behind fuzzy logic, the truth values, that we will be representing, must be ordered from the smallest (that corresponds to "false") to the largest (that corresponds to "true"). There is a natural order on the set of all integers, so ordering itself is not a problem. In view of this ordering, it is natural:

- to identify the largest computer-representable integer (which is usually denoted by `MaxInt`, and which we will denote as $+\infty$) with "true", and
- to identify the smallest computer integer, which we will denote by $-\infty$, with "false" (this integer is often equal to $-$`MaxInt`).

We need then to pick up operations that correspond to "not", "and", and "or" in such a way, that these operations be appropriately monotonic, and, when restricted to $+\infty$ and $-\infty$, these operations would become standard logical operations of classical logic: For example:

- $\neg(-\infty) = +\infty$, because the negation of "false" is "true";
- $(-\infty) \vee (+\infty) = +\infty$, because "false" or "true" makes "true", etc.

It is easy to see that from all hardware supported operations min, max, $+$, $-$, $\cdot$ (multiplication), and / (division), the only operation that correctly represents "or" is max, and the only operation that correctly represents "and" is min. (For example, $+$ works for neither of them, because $(-\infty) + (+\infty) = 0$ and is thus different from both $-\infty$ (needed for "and") and from $+\infty$ (needed for "or").)

Thus defined fuzzy logic is, thus, similar to the usual fuzzy logic with min and max operations. The only problem with this choice is that in many applications of fuzzy logic, e.g., in fuzzy control, min and max are not the best pair of operations (see, e.g., [21]). It is therefore desirable to have alternative easy-to-compute "and" and "or" operations, and the choice of all integers does not allow us this possibility.

It is, therefore, desirable to restrict the class of all integers. The natural restriction is to use all *non-negative* integers, i.e., all natural numbers.

Let us use natural numbers. For natural numbers, a similar analysis of all possible operations reveals the following possible choices:

- For &, the only choice is min.
- For $\vee$, we have two choices: max and $+$.

Thus, if we are not satisfied with the fastest-to-compute pair (min, max), we can use a still fast-to-compute alternative pair of operation (min, $+$).

This is the integer-valued fuzzy logic that we plan to use.

III. A Similar Integer-Valued Fuzzy Logic Has Already Been Successfully Used

Our hope that using the integer-valued fuzzy logic can bring definite computational advantages is also fed by the fact that a similar approach has already been successfully used in inference engines for expert systems.

What is an expert system. An *expert system*, for a certain area of expertize, is a computer system that tries to simulate experts' answers to different questions (*queries*) about this area; e.g., a medical expert system must, given symptoms of a patient, return the possible diagnoses and a reasonable treatment. To be able to do this, an expert system must contain the expert's knowledge; this computer-stored knowledge is called a *knowledge base*.

In addition to the knowledge base, the system must contain a program for answering queries; such a program is called an *inference engine*.

Designing an inference engine is a very difficult problem: Even when we have *crisp* knowledge, and the knowledge base contains only *propositional* statements F_i — i.e., statements obtained from the elementary statements $S_1, \ldots, S_n$ (like "a patient has a flu") by using "and" (&), "or" ($\vee$), and "not" ($\neg$) — the question of whether a given query follows from the knowledge $F_1, \ldots, F_m$ is, in general, computationally intractable (NP-hard) [14].

NP-hard means that this problem is *universal* in the following sense: any other problem from a very reasonable class (called *NP*) can be reduced to a particular case of this query-answering problem. This universality means that this problem is indeed very difficult to solve: if we could have an algorithm that solves all particular cases of this problem in reasonable time, then we would be thus be able to solve not only this problem, but also *all* reasonable problems. So, this query-answering problem is as computationally difficult as the most complicated of these (realistic) problems.

Heuristic methods are needed. NP-hard means, crudely speaking, that no algorithm can solve *all* particular cases of this problem in reasonable time; thus, *heuristic methods* are needed. In other words, we not only need expert's knowledge about the *domain* to which this expert system is applied, but we also need expert knowledge about the *way* experts answer queries.

It is natural to use fuzzy values (between 0 and 1) to describe heuristic methods. If we ask an expert about a certain query, this expert will often come up with a crisp answer ("yes" or "no"). Producing this answer takes some time. If we ask for an expert's opinion before this time, we will get his *preliminary* opinion; in this preliminary opinion, she is not yet sure whether the answer will be "yes" or "no", but she will probably have some *degree of belief* either in a "yes" answer, or in a "no" answer, or maybe in both. If we ask an expert for the reasons for this degree of belief, she will probably describe some beliefs in the elementary statements $S_1, \ldots, S_n$ and/or their negations.

Therefore, it is natural to simulate this expert reasoning as a step-by-step procedure, in which we start with no beliefs at all (except for the knowledge contained in the knowledge base) and then *modify* our degrees of belief $d(S_i)$ and $d(\neg S_i)$ in the basic statements S_i and their negations $\neg S_i$.

One can use of a fuzzy control-type technique. In order to apply fuzzy control methodology, we must describe this change in degrees of belief by if-then rules.

A natural way to do this comes from the fact that our knowledge consists of propositional formulas, and it is known that every propositional formula can be reformulated in Conjunctive Normal Form (CNF), i.e., in the form $D_1 \& \ldots \& D_k$, where each of the formulas D_j (called *disjunctions*) is of the type $a \vee b \vee c$, and a, b, and c are *literals* (i.e., elementary statements S_i or their negations $\neg S_i$). Each disjunction D_j, in its turn, can be reformulated as three implications: "if $\neg a$ and $\neg b$, then c"; "if $\neg b$ and $\neg c$, then a"; and "if $\neg a$ and $\neg c$, then b". Thus, the entire knowledge can be represented as a set of such if-then rules.

These rules describing *knowledge* naturally lead to the rules describing *change in degrees of belief*: e.g., the knowledge rule "if $\neg a$ and $\neg b$, then c" leads to the update rule "if $\neg a$ and $\neg b$, then increase the degree of belief in c". Thus, we can apply the standard fuzzy control methodology to these rules: at any given moment of time, we know the degree of belief $d(\neg a)$ in $\neg a$ and the degree of belief $d(\neg b)$ in $\neg b$. Therefore, we can compute the degree of belief p_j that this particular rule is applicable as $f_\&(d(\neg b), d(\neg c))$, and we can compute the degree of belief $i(c)$ that we should increase $d(c)$ as $f_\vee(p_j, \ldots, p_K)$, i.e., as an aggregation for all the rules whose conclusion is this particular increase. Then, for every literal c, we have two conclusions: "update" with degree of belief $i(c)$, and "do not update" with the remaining degree of belief $1 - i(c)$.

If we interpret "update" as adding a constant α to the previous degree of belief, then the standard defuzzification leads to the change from $d(c)$ to the updated value $d(c) + \alpha \cdot i(c)$.

By applying this update procedure again and again, we will get an answer to a query.

If we start with all 1's, the the resulting degrees are all non-negative integers!

This method is successful and related to neural networks. The resulting algorithm coincides with a successful heuristic method proposed by S. Maslov in the 1980s [23], [24], [25] as a way of simulating biological neurons (see also [17], [18], [19], [20], [35]).

This method was originally proposed based on the idea of simulating biological *neurons*, but it later turned out that exactly these same formulas follow from *fuzzy logic* heuristics, from the ideas of *chemical computing* (i.e., simulating chemical reactions), from heuristics of *numerical optimization* approach, from the ideas of *freedom of choice*, etc. (for a survey and latest results, see [18], [20]).

IV. Final Comment: Relation to Bags

Our suggestion reformulated. Traditionally, a fuzzy set is defined by its *membership function*, i.e., by a function that assigns, to every element of a universal set, a number from the interval $[0, 1]$. This notion is a generalization of the notion of a *characteristic function* of a crisp set, i.e., a function that is equal to 1 or 0 depending on whether a given element belongs to this set or not.

We are proposing to use, instead of the more traditional membership functions, functions that assign to every element x, a number $\mu(x)$ from the set of all natural numbers. Such functions can also be viewed as generalizations of characteristics functions, generalization in which an element can have 0 occurrences in the "set", 1 occurrence, 2 occurrences, etc.

Bags. Such a generalization of a set is well-known and widely used in computer science under the name of a *bag* (*multiset*).

Namely, a *bag* is a collection of elements over some domain. Unlike sets, bags allow *multiple* occurrences of elements. For example $\{a, a, b\}$ is a bag but not a set.

Bags have been successfully used in computing. Bags are actively used in computing:

- The main sorting algorithms (see, e.g., [16]) actually sort bags, not sets of data.
- Bags are used to describe Petri nets ([4], [33], [34]): namely, a state of a Petri net at any given moment of time is described by specifying how many tokens are there in each location. So, a state is a bag of locations.
- Bags are not only a good way to describe algorithms, but also a good way to describe specifications (see, e.g., [13]), because unsorted collection of elements with possible repetitions is a frequent example of input. Moreover, other, more complicated data structures can be naturally expressed in bag terms, to an extent that bags have been proposed as a basic data type for a new generation of high-level programming languages [1], [2], [3], [5], [6], [7], [8], [9], [11], [12], [36]

Bags have been successfully used in processing fuzzy data. In particular, in [10], bags have been successfully used for processing fuzzy data.

Our general idea explains that this success was not accidental.

Acknowledgments

The authors are thankful to Ann Q. Gates and Daniel E. Cooke for valuable discussions.

References

[1] J.-P. Banâtre, A. Courant, D. Le Métayer, "A parallel machine for multiset transformation and its programming style, *Future Generation Computer Systems*, vol. 4, pp. 133–144, 1988.

[2] J.-P. Banâtre, D. Le Métayer, "The GAMMA model and its discipline of programming", *Sci. Comput. Program.*, vol. 15, pp. 55–77, 1990.

[3] J.-P. Banâtre, D. Le Métayer, "Programming by multiset transformation", *Communications of the ACM*, vol. 36, No. 1, pp. 98–111, 1993.

[4] V. Cerf, E. Fernandez, K. Gostelow, and S. Volansky. *Formal Control Flow Properties as a Model of Computation*, Report ENG-7178, Computer Science Department, University of California at Los Angeles, December 1971.

[5] D. Cooke, "Arithmetic over multisets leading to a high level language", *Proceedings of the Computers in Engineering Symposium*, Houston, TX, 1993, pp. 31–36.

[6] D. Cooke, "Possible effects of the next generation programming language on the software process model", *International Journal of Software Engineering and Knowledge Engineering*, 1993.

[7] D. Cooke, "A high level computer language based upon ordered multisets", *IEEE Fifth International Conference on Software Engineering and Knowledge Engineering*, San Francisco, 1993.

[8] D. E. Cooke, "A Comparison of Languages SEQUENCEL and FP," in *Proceedings of the First World Conference on Integrated Design and Process Technology*, Society of Design and Process Science, Austin, TX, 1995, pp. 43–48.

[9] D. Cooke, "An Introduction to SEQUENCEL: A Language to Experiment with Nonscalar Constructs," *Software Practice and Experience*, Vol. 26, No. 11, pp. 1205–1246.

[10] D. E. Cooke, R. Duran, and A. Gates, "Bag language speeds up fuzzy interval computations", In: L. Hall, H. Ying, R. Langari, and J. Yen (eds.), *NAFIPS/IFIS/NASA'94, Proceedings of the First International Joint Conference of The North American Fuzzy Information Processing Society Biannual Conference, The Industrial Fuzzy Control and Intelligent Systems Conference, and The NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic, San Antonio, December 18–21, 1994*, IEEE, Piscataway, NJ, pp. 398–399.

[11] D. E. Cooke, R. Duran, A. Gates, and V. Kreinovich, "Bag Languages, Concurrency, Horn Logic, and Linear Logic", in *Proceedings of IEEE Sixth International Conference on Software Engineering and Knowledge Engineering, Riga, Latvia*, IEEE Press, 1994, pp. 289-297.

[12] D. E. Cooke, A. Gutierrez, "An introduction to BagL", *IEEE Fourth International Conference on Software Engineering and Knowledge Engineering*, Capri, Italy, 1992, pp. 479-486.

[13] G. Dromey, *Program Derivation. The Development of Programs from Specifications*, Addison-Wesley, Sydney, 1989.

[14] M. Garey and D. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[15] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: theory and applications*, Prentice Hall, Upper Saddle River, NJ, 1995.

[16] D. Knuth, *The Art of Computer Programming. Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 1969.

[17] V. Kreinovich, "Semantics of S. Yu. Maslov's iterative method," In: *Problems of Cybernetics*, Moscow, 1987, Vol. 131, pp. 30–62 (in Russian); English translation in: V. Kreinovich and G. Mints (eds.), *Problems of reducing the exhaustive search*, American Mathematical Society, Providence, RI, 1997, pp. 23–52.

[18] V. Kreinovich, "S. Maslov's Iterative Method: 15 Years Later (Freedom of Choice, Neural Networks, Numerical Optimization, Uncertainty Reasoning, and Chemical Computing)", In: V. Kreinovich and G. Mints, eds. *Problems of reducing the exhaustive search*, American Mathematical Society, Providence, RI, 1997, pp. 175–189.

[19] V. Kreinovich and L. O. Fuentes, "Simulation of chemical kinetics - a promising approach to inference engines," in: J. Liebowitz (ed.), *Proceedings of the World Congress on Expert Systems, Orlando, Florida, 1991*, Pergamon Press, N.Y., Vol. 3, pp. 1510–1517.

[20] V. Kreinovich and G. Mints, eds. *Problems of reducing the exhaustive search*, American Mathematical Society, Providence, RI, 1997.

[21] V. Kreinovich and H. T. Nguyen, "Methodology of fuzzy control: an introduction", In: H. T. Nguyen and M. Sugeno (eds.), *Handbooks of Fuzzy Systems. Vol. VI: Fuzzy Modeling and Control*, Kluwer Academic, 1997 (to appear).

[22] V. Kreinovich and D. Tolbert, "Minimizing computational complexity as a criterion for choosing fuzzy rules and neural activation functions in intelligent control", In: M. Jamshidi, C. Nguyen, R. Lumia, and J. Yuh (Editors), *Intelligent Automation and Soft Computing. Trends in Research, Development, and Applications. Proceedings of the First World Automation Congress (WAC'94), August 14–17, 1994, Maui, Hawaii*, TSI Press, Albuquerque, NM, 1994, Vol. 1, pp. 545–550.

[23] S. Yu. Maslov, "Iterative methods in intractable problems as a model of intuitive methods", *Abstracts of the 9th All-Union Symposium on Cybernetics*, 1981, pp. 52–56 (in Russian).

[24] S. Yu. Maslov, "Asymmetry of cognitive mechanisms and its implications", *Semiotika i Informatika*, 1983, Vol. 20, pp. 3–31 (in Russian).

[25] S. Yu. Maslov, *Theory of deductive systems and its applications*, MIT Press, Cambridge, MA, 1987.

[26] E. Merényi, B. Csathó, M. Bodrogi, M., and Á. Gulyás, "Utilization of Landsat images for mapping natural resources and for enviromnetal protection in Hungary", In *Proc. Tenth Thematic Conference on Geologic Remote Sensing, San Antonio, TX, May 9–12*, 1994, Vol. II, pp. 491–502.

[27] E. Merényi, B. Csathó, M. Bodrogi, M., and Á. Gulyás, "Integration of Landsat images, geophysical and radar data for mapping soil composition in temperate climate environment, Hungary", Submitted to *Remote Sens. Environ.*

[28] E. Merényi, J. V. Taranik, T. B. Minor, and W. H. Farrand, "Quantitative comparison of neural networks and conventional classifiers for hyperspectral imagery" In: R. O. Green, ed., *Summaries of the Sixth Annual JPL Airborne Earth Science Workshop*, Pasadena, CA, March 4–8, 1996, Vol. 1.

[29] T. Moon and E. Merényi, "Classification of hyperspectral images using wavelet transforms and neural networks", In: *Proceedings of the Annual SPIE Conference*, 1996, Vol. 2569.

[30] H. T. Nguyen and M. Sugeno (eds.), *Handbooks of Fuzzy Systems. Vol. VI: Fuzzy Modeling and Control*, Kluwer Academic, 1997 (to appear).

[31] H. T. Nguyen and E. A. Walker, *A first course in fuzzy logic*, CRC Press, Boca Raton, Florida, 1997.

[32] S. H. Patterson, *Kaolin, refractory clay, ball clay, and halloysite in North America, Hawaii, and the Caribbean region*, U.S. Department of the Interior, Geological Survey, Alexandria, VA, 1984.

[33] J. L. Petersen, "Computation sequence sets", *Journal of Computer and System Sciences*, vol. 13, No. 1, pp. 1–24, 1976.

[34] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Inc., 1981.

[35] O. Sirisaengtaksin, L. O. Fuentes, and V. Kreinovich, "Nontraditional neural networks that solve one more intractable problem: propositional satisfiability", *Proceedings of the First International Conference on Neural, Parallel, and Scientific Computations*, Atlanta, GA, May 28–31, 1995, Vol. 1, pp. 427–430.

[36] S. A. Starks and D. E. Cooke, "Navigating Large Databases Using a New High-Level Language," *Proceedings of IEEE Technical Applications Conference and Workshops*, NORTHCON/95, Seattle, WA, 1995, pp. 49–54.

[37] L. Zadeh, "Fuzzy sets", *Information and control*, 1965, Vol. 8, pp. 338–353.