Intelligent Technologies

(Preface to the Special Issue)

1 What are intelligent technologies

Many practical problems, e.g., many engineering design problems, can be solved by known (routine) techniques. For many of these problems, these techniques have been incorporated into software packages, so the solution to these problems can be (largely) automated.

However, there are still many other problems for which no routine techniques are known. For example, it is usually rather routine to design a new chip based on the known technologies, but when technologies change, the problems becomes non-trivial. In order to solve such hard-to-solve problems, we must use new ideas, we must use our intelligence. Any technology that helps us in solving such problems is called *intelligent technology.*

2 Traditional intelligent technologies: search for algorithms, especially for algorithms that solve dynamical optimization problems

For each class of hard-to-solve practical problems, ideally, we would like to have a feasible algorithm that would solve all problems from this class in reasonable time.

What do we mean by "solving a problem"? For example, when we design a bridge, we must create a design that satisfies certain constraints: that the bridge should be able to withstand certain total weight, a certain wind-power, should be buildable within the budget constraints, etc. However, if we simply satisfy all the constraints, it is good, but not perfect. Ideally, we are not just looking for *a* solution, we want to find *the best possible* solution – the best with respect to some optimality criterion. Therefore, most practical problems can be naturally formalized as optimization problems.

Since in most practical problems, we deal with a *dynamic* situation, a situation that changes with time, the corresponding optimization problems involve quantities that change with time – in other words, the corresponding optimization problems are *dynamic* optimization problems.

Techniques that help in solving such problems in hard-so-solve situations can be classified as intelligent technologies. Such technologies are very useful in areas related to Artificial Intelligence, e.g., in robot control, airplane control, etc. For example, an autopilot that automatically flies a plane is a seemingly intelligent device – and it is mainly based on traditional intelligent technologies.

3 Limitations of traditional intelligent technologies

As we have mentioned, for some classes of hard-to-solve practical problems, traditional intelligent technologies are very successful: they lead to an algorithm that solves all the problems from this class (or at least from a large subclass of this class) in reasonable time; once we implemented this algorithm, we do not have to worry about solving these problems anymore.

Not all classes of problems have been thus solved. In some case, it may still be possible to find a general algorithm, but we have not found it yet. There are other case when one can prove that (crudely speaking) such a general algorithm is not possible,. To be more precise, many such classes of problem turn out to be *NP-hard.* Informally, NP-hardness means that, unless there is an efficient algorithm for solving all such

practical problems (and most researchers strongly believe that no such algorithm is possible), no general algorithm is possible that solves all the problems from this class in reasonable time.

For such classes, it is still possible and reasonable to looks for algorithms, but (in contrast to success stories of traditional intelligent technologies) there will always be cases when the resulting algorithm will not lead to solution – and thus, that a human expert will be needed. In other words, for such classes, we need to go beyond traditional intelligent technologies.

4 Beyond traditional intelligent technologies: main idea

In real life, for each class of practical problems, we usually have experts who have successfully solved some problems from this class. It is therefore natural, in designing new techniques for solving these problems, to use the experience of these experts. We can also use the experience of experts in solving similar problems outside the desired class. In other words, we want to emulate (simulate) the way human experts solve these problems.

What do we need to do to emulate? Depending on what exactly we simulate, what is the level of our simulation, we end up with different intelligent technologies. There are several different aspects of expert reasoning:

- Sometimes, experts have a clear logical understanding of how they solve the problem, what steps they take and why these steps, with certainty, lead to the desired results. In many other real-life cases, the situation is made more difficult by the existence of *uncertainty*. This may be an "*objective*" uncertainty – when the exact same expert's action leads to different results because the real world is sometimes unpredictable. This may be (and often is) a more "*subjective*" type of uncertainty – when an expert uses some terms like "small", "large" in the rules that lead to a decision, but the expert does not have a crisp understanding of these terms. We must be able to simulate both the experts' "crisp" reasoning and their reasoning under both types of uncertainty.

- In some situations, an expert cannot explain his or her reasoning, the expert simply follows an *intuition*. We must therefore be able to simulate such situations in which the only thing that we can base our simulation on is the resulting behavior of the experts – and in which we do not know (and the expert him/herself does not know) how exactly we came up with this behavior.

- Sometimes, a single expert succeeds in solving a problem. However, when a problem becomes more complex, an individual expert cannot solve it by itself, he or she needs help. In view of this fact, we must be able to simulate both *individual* and *group* expert problem solving.

- Sometimes, the problem is difficult to solve, but at least easy to grasp and to explain. In many practical situations, however, the problem itself is difficult to formulate. For example, when we design a complex system such as an airplane, it is very difficult to grasp the complexity of the design because in reality what we are designing is a hierarchical complex system – with the shape of the wings to provide aerodynamic stability and efficiency, with the engine to provide power, etc. Usually, different teams of experts design – in collaboration with each other – different subsystems. The interaction between these subsystems is what make this design problem very difficult. We must therefore simulate both the design of simpler systems and the design of complex *hierarchical* systems.

Let us describe how this classification of aspects of expert problem solving translates into a classification of intelligent technologies themselves (see, e.g., [7]).

5 Resulting classification of new intelligent technologies

According to the above discussion, the simplest case is when we have an individual expert who applies crisp reasoning to an easy-to-describe problem. (It is worth mentioning here that we are dealing with hard-to-solve problems, so "simplest" does not mean "simple", it rather means "hard to solve but relatively simpler than other hard-to-solve problems".) In this case, we have describe the expert's knowledge by facts, and the

expert's reasoning by crisp rules like "if A then B". Such a system of crisp facts and rules constitutes an important particular case of what is called an *expert system*; hence, the corresponding intelligent technologies are usually called *expert system technologies*.

Situations with "objective" uncertainty are usually well covered by *probabilistic* (*statistical*) intelligent techniques.

To help in situations with "subjective" uncertainty – in which probabilistic methods are not always successful – *fuzzy* techniques have been designed.

To simulate experts' intuition, we need to use *intelligent learning* techniques, techniques that enables us to use the known examples of expert decisions to train a computer system that would make the same decisions and thus, emulate the expert. The most well-known of such techniques is the technique of *neural networks*.

What if we must go from individual to group decisions? There are several intelligent technologies that emphasize and simulate agent interaction. These technologies differ by the level of detail with which an individual agent is described:

- The simplest technology corresponds to the case when we consider a crude model of an agent, in which an we only distinguish between a small finite number of different states of an agent. The resulting model of an agent is called a *finite automation*, and the corresponding interacting agents are called a *cellular automata*. (These technologies have been recently boosted by a nice popular exposition in [15].)
- A somewhat more realistic model is a model in which we take into consideration that each agent can be in infinitely many different states – but we restrict this infinity by allowing only finitely many parameters to describe each state. Social insects such as ants and bees provide a natural example of such agents. As a result, this approach is usually called *ant intelligence*, or *swarm intelligence*.
- Finally, there are more sophisticated *multi-agent* techniques that consider even more complex models of interacting agents.

To describe how we can solve hierarchical problems, we must consider technologies that take into consideration that the simulated objects are hierarchical. The corresponding intelligent technologies are usually called *object-oriented*.

Finally, instead of simulating how we solve problems, an alternative idea is to simulate how we come up with such solutions, how different ideas are generated, how they compete with each other – and how the best ideas win. Even more ambitiously, we can simulate how, in nature, simple organisms evolved into us human beings capable of solving complex problems. The corresponding algorithms are called *evolutionary computations*. Since the biological evolutions is performed by evolving genes, this is also called *genetic algorithms*.

Each of these classes of intelligent technologies captures only one aspect of expert decision making. In real life, many of these aspects are present: we usually have some objective uncertainty, some subjective uncertainty, some intuition, some group interaction, etc. To capture this fact, we must *combine* the existing the existing intelligent technologies.

6 In what application areas can we benefit most from using intelligent technologies?

As we have mentioned, intelligent technologies are needed when no routine algorithms exist. For most practical problems that have been around for a while, reasonable algorithms have been developed. It is still possible to (somewhat) improve the quality of these algorithms by using intelligent technologies; however, the need for such technologies is much more acute in new problems, problems for which we did not have

time to develop efficient algorithms. The newer the problem, the more rapid the progress in an area, the more important it is to use new intelligent technologies.

There is a constant progress in many areas of human endeavor, but none is so rapid as the progress in computers and – an even faster one – the progress in computer connections. For several decades already, computers follow Moore's law – according to which the computer speed doubles every 18 months, and the World Wide Web grows even faster. Hence, computers in general – and especially the web – are the areas which can benefit the most from the new intelligent technologies.

What are the most important problems related to the web?

- The most frequent use of the web is to *search* for information. In this sense, the web acts as a large knowledge base.
- Sometimes, we know what we are looking for, so it is literally a search, sometimes, we just look for some information without knowing what exactly we look for – in other words, we need *data mining*.
- Instead of simply learning *facts*, we may can use the web to learn concepts, skills, etc. – in other words, we need *web-based learning*.
- Even faster than the web itself grows, grows the amount of *images* on the web. It is therefore extremely important to search for information in images – and in the process, to classify the stored image.
- Most people simply use the web, by searching through the material that someone else has placed there. But for this information to be there, someone needs to design and store it. We therefore need intelligent methods for *simulating* objects – as before, simulating images is an especially important task.

Once we come up with a new algorithm, we must:

- check how good this algorithm is, and
- if necessary, produce a better one.

So, we must be able to predict the running time of the existing algorithms (i.e., to *analyze* these algorithms), and to provide improvements. Nowadays, a natural way to speed up computations is to perform them in parallel on several processors. The efficiency of a multi-processor system depends on the efficiency of an algorithm that schedules tasks on different processes. Thus, we need to develop and use intelligent techniques for *scheduling* tasks.

Finally, an important task is to make sure that everything is working right. *Testing* and checking techniques for large knowledge bases is also, therefore, one of the important areas of application of intelligent technologies.

7 About this issue

This Special Issue collects selected papers on Intelligent Technologies presented at the First International Conference on Intelligent Technologies InTech'2000, Assumption University, Bangkok, Thailand. These papers cover all the aspects and all major applications of intelligent technologies.

Specifically, R. Keinprasit and P. Chongstitvatana [1] combine a new (improved) version of ant computations with a more traditional dynamic optimization technique and successfully apply the resulting techniques to scheduling problems. They also compare their results with genetic algorithms.

V. Kreinovich and C.-W. Tao [2] describe the existing algorithms for checking combinatorial identities – a problem useful in analyzing complexity of different algorithms – and show that the drawbacks of the existing algorithms are not accidental: the corresponding problem is NP-hard.

P. Mahatthanapiwat and W. Rivepiboon [3] combine cellular-automata type bit techniques with object oriented technologies and design an efficient algorithm for search in object-oriented databases and knowledge bases.

W. Nakapan, G. Halin, J.-C. Bignon, and M. Wagner [4] use expert system-type rule base to extract and index images stored on the World Wide Web.

E. Nantajeewarawat and R. Sombatsirisrisomboon [5] provide an exact semantics of a Unified Modeling Language (UML) – a standard (albeit somewhat informal) way of handling object-oriented systems.

E. Nantajeewarawat, V. Wuwongse, C. Anutariya, K. Akama, and S. Thiemjarus [6] provide expert system-type crisp rules for reasoning with UML objects.

H. T. Nguyen, T. Wang, and B. Wu [8] show how to combine probabilistic and fuzzy techniques. Specifically, they show how probabilistic methods can be used to analyze and improve fuzzy techniques, and how the resulting combination is helpful in data fusion.

Y. Rodkaew, S. Chuai-aree, S. Siripant, C. Lursinsap, and P. Chongstitvatana [9] combine formal grammars (analogue of cellular automata), sigmoidal functions (from neural networks), and genetic algorithms to produce an efficient simulation of a complex and difficult-to-simulate process of leaf growth in plants.

C. Sanrach and M. Grandbastien [10] design new techniques for web-based learning.

M. Siddaiah, M. A. Lieberman, S. E. Hughs, and N. S. Prasad [11] combine fuzzy and neural techniques, and use this combination in image analysis, to detect and classify trash in cotton.

N. Soonthornphisaj and B. Kijsirikul [12] use probabilistic and other learning techniques to classify unlabeled web pages. Such a classification is very important for improving the quality and efficiency of searching the web.

T. Theeramunkong [13] uses hierarchical techniques to improve the quality of data mining on the web.

W. Vatanawood, V. Sriratanalai, and W. Rivepiboon [14] show how to translate traditional relational databases into a formal specification language that enables the user to employ rule-based type reasoning for testing correctness of database design.

X. Yang and M. Zhang [16] analyze multi-agent systems; specifically, they analyze different methods of fusing the results of several web search engines into a single "meta-engine".

Finally, Y.-J. Zhang and Z.-Q. Liu [17] use known clustering techniques to improve the efficiency of web search.

These papers cover all intelligent technologies:

- traditional intelligent technologies are developed and analyzed in [1, 2];
- expert system methods are developed and used in [4, 6, 14];
- probabilistic methods are developed and used in [8, 12, 17];
- fuzzy techniques are developed and used in [8, 11];
- learning techniques, in particular, neural techniques are used in [9, 10, 11, 12];
- cellular automata-type methods are developed and used in [3, 9];
- ant intelligence methods are developed and used in [1];
- multi-agent techniques are developed and used in [16];
- hierarchical and object-oriented methods are developed and used in [3, 5, 6, 13, 17];
- finally, genetic algorithms are used in [1, 9].

These technologies are applied to major practical problems:

- applications to search in the web and, more generally, in large knowledge bases, are described in [3, 8, 12, 16, 17];

- applications to data mining are described in [13];
- applications to web-based learning are considered in [10];
- applications to search for images on the web and to image processing in general are described in [4, 11];
- applications to efficient simulation, in particular, simulation of images, are described in [5, 6, 9];
- applications to algorithm analysis are described in [2];
- applications to scheduling are described in [1];
- applications to testing are described in [14].

Acknowledgments

We are grateful to Dr. Ronald R. Yager, Editor-in-Chief of the *International Journal of Intelligent Systems*, for encouraging us to edit this Special Issue. We also thank all contributors to this Issue.

References

[1] R. Keinprasit and P. Chongstitvatana, "High-Level Synthesis by Dynamic Ant", this issue.

[2] V. Kreinovich and C.-W. Tao, "Checking identities is computationally intractable (NP-hard), so human provers will always be needed", this issue.

[3] P. Mahatthanapiwat and W. Rivepiboon, "Virtual Path Signature: An Approach for Flexible Searching in OODB", this issue.

[4] W. Nakapan, G. Halin, J.-C. Bignon, M. Wagner, "Building Product Image Extraction from the Web", this issue.

[5] E. Nantajeewarawat and R. Sombatsirisrisomboon, "On the Semantics of UML Diagrams Using Z Notation", this issue.

[6] E. Nantajeewarawat, V. Wuwongse, C. Anutariya, K. Akama, and S. Thiemjarus, "Towards Reasoning with UML Diagrams Based on XML Declarative Description Theory", this issue.

[7] H. T. Nguyen and V. Kreinovich, *Applications of continuous mathematics in computer science*, Kluwer, Dordrecht, 1997.

[8] H. T. Nguyen, T. Wang, and B. Wu, "On probabilistic methods in fuzzy theory", this issue.

[9] Y. Rodkaew, S. Chuai-aree, S. Siripant, C. Lursinsap, and P. Chongstitvatana, "Animating Plant Growth in L-System by Parametric Functional Symbols", this issue.

[10] C. Sanrach and M. Grandbastien, "Adaptive and intelligent web-based environments", this issue.

[11] M. Siddaiah, M. A. Lieberman, S. E. Hughs, and N. S. Prasad, "Automation in Cotton Ginning", this issue.

[12] N. Soonthornphisaj and B. Kijsirikul, "Iterative cross-training: an algorithm for learning from unlabeled web pages", this issue.

[13] T. Theeramunkong, "Applying Passage in Web Text Mining", this issue.

[14] W. Vatanawood, V. Sriratanalai, and W. Rivepiboon, "Formal Specification Synthesis for Database Applications", this issue.

[15] S. Wolfram, *A new kind of science*, Wolfram Media, Champaign, IL, 2002.

[16] X. Yang and M. Zhang, "Rational Constraints for Fusion Methods in Meta Search Engineering Systems", this issue.

[17] Y.-J. Zhang and Z.-Q. Liu, "Refine web search engine results using incremental clustering", this issue.

Guest Editors:

Vladik Kreinovich (University of Texas at El Paso, USA)
Hung T. Nguyen and Nadipuram S. Prasad (New Mexico State University, USA)
Pratit Santiprabhob (Assumption University, Thailand)

November 2002