UDC:004
Original scientific paper

THE RICH WORLD OF COMPUTER MODELLING AND PROBLEM SOLVING

Vlatko Čerić
Graduate School of Economics and Business, University of Zagreb
vceric@efzg.hr

Abstract: *This paper presents some interesting aspects of the rich world of computer modelling and problem solving. After discussing basic model building principles attention is devoted to several modelling methods aimed for solving different types of problems. These are methods for modelling of optimisation problems, dynamic systems with random characteristics, knowledge, uncertain phenomena, systems with graph structure, learning, as well as cooperation and competition. Basic characteristics of these methods and examples of their application are given.*

Keywords: *computer modelling, problem solving, modelling methods.*

1. INTRODUCTION

For the last five decades computer modelling was used as a powerful approach for representing and solving complex problems. Since a variety of problems were waiting to be solved, diverse modelling and problem solving approaches were developed. Some of the typical problems that were modelled have been: finding the optimal production plan, solving a travelling salesman problem, designing airports and assessing their performances, management of workforce education in organizations, and finding best strategies for playing business or social games.

Efforts directed at solution of these problems resulted in creation of a rich and interesting computer modelling world. Computer modelling methods originated from various disciplines such as operations research, artificial intelligence and mathematics. Some of these methods are (Winston, 1991; Michalewicz and Fogel, 2002) optimisation methods (like linear programming, genetic algorithms and multiple criteria optimisation), simulation methods (including discrete simulation and system dynamics), transportation models, project management, expert systems, neural networks, genetic algorithms, fuzzy logic, theory of graphs and theory of games.

We don't intend to present taxonomy of computer modelling methods here, or to describe computer modelling methods in detail. Instead, we will demonstrate some of the methods aimed for solving different type of problems. For each of these methods we will see for what kind of problems the method is appropriate, outline how the method functions, and describe some interesting problems that are being solved by the method.

In section 2 we describe basic facts about models and principles of model building. In section 3 we present optimisation modelling, and in section 4 simulation modelling of dynamic systems with random characteristics is presented. Section 5 describes modelling of knowledge while section 6 is devoted to modelling of uncertainties. Section 7 discusses systems with graph structure. Section 8 describes modelling of learning, while section 9 presents modelling of competition and cooperation. Finally, section 10 offers some conclusions.

2. ABOUT MODELS AND PRINCIPLES OF MODEL BUILDING

MODELS

Models are approximate representation of systems used to facilitate better understanding of system functioning, making changes in systems or managing systems. The fact that models are approximations is not undesirable. Just the opposite, it is rational since models need to include only the most important objects and relations of real systems in order to provide understanding of how these systems function. Moreover, for the model to be useful, it must address a specific problem (Sterman, 1991).

Why do we need computer models? Experiments "in vivo" are expensive and require a lot of time. Repeating such experiments may even be impossible. On the other hand, mental models are too simple since human can take only a few factors in account because of the limitation of our working memory.

Models can serve several purposes: the can be used as tools for understanding how system is functioning, as problem solving tools (for obtaining qualitative or quantitative results), tools for helping communication between system experts and modellers, and tools for gaining "artificial experience" by working with the model (rather than with the system).

Graphical models have shown to be of specific value because of their two-dimensional nature and ability of human visual system to easily grasp object and relations represented graphically. As an example, Petri nets are used for problem representation in discrete event simulation (Reisig, 1985), and they not only describe system operation but can also be used to run simulations.

PRINCIPLES OF MODEL BUILDING

Experience in model building led to several principles that should be followed in order to achieve sound modelling results (Pidd, 2004). First of all, it is important that models include only relevant parts of real systems that influence system behaviour. This means that the boundary that divides a system from its environment should be carefully set. Models should neither be too complex nor too detailed. If they are too complex, it would be difficult to understand and validate them. However, models should neither be too simple, i.e. they must not leave out objects or relations essential for understanding of system behaviour.

Model should be divided into rather simple modules with well defined functions, since this enables model building and validation to be much easier. In model building one should start with small models, and then adding complexities. Logical and quantitative validation of models (and their parts) has to be done throughout the whole period of model development.

3. OPTIMISATION MODELLING

Optimisation modelling (Winston, 1991) deals with seeking for the best solution of problems that include some constrained resources. Quality of the solution is measured by the objective function. An example problem is optimisation of a production system that produces several types of products. We may want to find such a production plan (i.e.

amount of production of each type of product) which doesn't spend more resources than available and gives the highest possible profit.

Many optimisation methods were developed, each for a specific kind of optimisation problem. The most popular method is by any means **linear programming** where both the objective function and the requirement for resources by different products are linear functions of quantity of products of different types. Solution of a linear programming problem includes a production plan, optimal value of objective function, as well as sensitivity analysis which investigates changes in model output as a consequence of changes in a single problem parameter. For example, if we have some additional funds that could be invested in the production, sensitivity analysis help us to find which resource is the best candidate for investment if we want to get the highest possible increase of profit.

Linear programming, as well as other optimisation methods, is used for solution of complex real problems of large dimensions. For example, optimisation of daily re-planning of production, refinement and transportation of various oil fractions tends to have thousands of variables and tenths of thousands of constraints.

Among the numerous variants of linear programming are **integer programming** where variables have integer values and **transportation problem** where we look for the cheapest transportation scheme for transporting some goods from one group of centres (e.g. manufacturing centres) to another group of centres (e.g. stores).

Genetic algorithms (Goldberg, 1989) are able to deal with optimisation problems that are not linear, that have many local optima and whose objective functions have discontinuities. An example of a problem that cannot be solved by linear programming is a travelling salesman problem in which a salesman has to travel to a number of cities and then to return home. The goal here is to find the shortest or cheapest route for the salesman.

Genetic algorithms are parallel search algorithms. They start with an initial population ("generation") of randomly selected "chromosomes" that consist of "genes". Chromosomes contain properties of objects under investigation, while fitness function is used to measure the quality of chromosomes and thus to help in selection of chromosomes. Next generation of "chromosomes" is formed using a random but structured exchange of information by crossover between the most successful chromosomes that "survive" from the previous generation. New generations of chromosomes have a fair chance to possess chromosomes with increased value of fitness function. Search for optimal solution finishes when no further chromosome improvement is made. Although genetic algorithms are heuristic algorithms that don't guarantee that the optimum will be found, they typically come quite close to the optimum value.

In optimisation problems we often have to deal with multiple criteria that have to be taken into account, like profit maximization, high resource usage, or increasing of the market share. One group of methods developed for solution of such problems are **multiple criteria decision making** (MCDM) methods. One of the most popular MCDM methods is analytical hierarchy process (Saaty, 1990), a method based on ranking of alternatives using a hierarchy of criteria and alternatives.

4. MODELLING DYNAMIC AND RANDOM SYSTEMS

We will use **simulation modelling** as a representative of dynamic and random systems modelling approach. These systems consist of objects that mutually interact and thus cause change of system state in time. They typically include random variables which lead to system behaviour that is very difficult to predict. Two main representatives of simulation modelling are discrete simulation and system dynamics.

Discrete simulation is used for a detailed analysis of queuing systems (Seila et al., 2003). An example of such system is an airport where passengers arrive with baggage and have to pass through check-in, body check control and boarding, possibly also a passport control. Their baggage is following a somewhat different route, but finally it has to reach the same flight. Simultaneously planes are landing and passengers leave them, possibly passing thorough a passport control, and wait for their baggage that has to be unloaded and sent to the passenger building. Both passengers and baggage have occasionally to wait in queues for various service resources.

System behaviour is described in a discontinuous way, as a sequence of different events (i.e. instantaneous changes of system state) and activities. Simulation models imitate real systems and processes, while objects in a model represent objects from real systems or processes with their properties. Discrete simulation models typically contain input random variables, like entities interarrival times or service times. Random variables are reproduced in simulation by random number generation and their transformation into required distributions.

As soon as at least one input variable is random, interaction between objects result in random behaviour of dependent variables (e.g. queuing length or queuing time). Multiple simulation experiments have to be accomplished in order to obtain statistical properties of output variables. Analysis of these experiments cannot be done with traditional statistics since simulation typically contain autocorrelated variables. Therefore, specific statistical methods were developed that deal with analysis of simulation output variables.

Distribution of output variables obtained by simulation enables obtaining information that helps in designing systems (e.g. determining the space required for queuing) or determining values of important system performances (e.g. queuing time or usage of system resources). Discrete simulation also gives answers on various "what-if" questions, and enables visual analysis of system behaviour via animation of simulation.

Influence of random phenomena on system behaviour is very difficult to predict, so without repetition of simulation experiments it would be virtually impossible to guess about performances of such systems. For example, even the use of similarly shaped distributions of input variables may result in quite different values of system performances (Law and Kelton, 1999).

System dynamics enables modelling of dynamic systems with feedback, and achieving appropriate control of such systems (Sterman, 2000). Feedback means that the variable is influencing itself through the chain of causes and their effects. Positive feedback leads to exponential growth or decrease, while negative feedback leads to stabilization around equilibrium state. An example of a system with feedback is birth and death process in some population, where a number of new births and new deaths in a year are proportional to population size. Higher population leads to higher birth rate per year and therefore to further increase of population (positive feedback loop). In the same time, higher population leads to higher death rate per year and so to decrease of population (negative feedback loop). Overall behaviour of population size depends on relative magnitudes of fertility and mortality rates.

Systems with a goal (e.g. a goal of attending certain number of flats in a town or region) tend to stabilize around the goal value, either by growing, by decreasing, or by fluctuation around the goal. Systems that contain several feedback loops have a complex behaviour that is virtually impossible to predict intuitively.

System dynamic models are not as detailed as discrete simulation models since they aggregate events and represent them as quasi continuous flows. For example, arrival of parts to a stock consists of a sequence of discrete events representing arrivals of separate parts to a stock. In system dynamics these arrivals are modelled as continuous inflow of parts. Processes that are modelled in system dynamics describe various resources that transit from one to another state.

StarLogo simulation system (Resnick, 1994) enables modelling and simulation of large decentralized systems consisting of many individuals. These individuals obey certain behaviour rules. These rules describe how an individual interacts with its neighbours. As a consequence, the behaviour of the whole system is emerging. An example of a system modelled with StarLogo is simulation of a colony of ants searching for a food. When an ant finds a piece of food, it carries the food back to the nest and is dropping a chemical while it moves in order to help other ants finding this piece of food. Evaporation of this chemical is also simulated, so when all the food is collected smell of the chemical becomes weaker and weaker. Animation of simulation demonstrates the changes and movements in the system, while graphs show changes of values of variables of interest over time.

StarLogo is based on the so called turtle graphics that uses a "turtle" which is moving around the plane following some programmed commands. StarLogo extends the turtle graphics approach by using hundreds of turtles rather than just one. Moreover, these turtles can interact with one another, and also interact with their environment. The area on which turtles are moving is made up of "patches" that can have different properties (this area is actually a two-dimensional cellular automaton).

5. MODELLING KNOWLEDGE

Knowledge modelling is an important part of artificial intelligence, and especially of expert systems (Negnevitsky, 2002). In order to be used by expert systems, knowledge has to be represented in a formal way. Some of the knowledge representation formats that have been developed are production rules, decision trees, frames and semantic nets.

Expert systems are computing systems that tend to achieve the same abilities of expertise in a narrow domain as human experts. Expert systems consist of a knowledge base, fact base and inference engine. Knowledge base contains knowledge from a certain area, fact database contain facts about the object that is analysed (e.g. a company), while inference engine carries out reasoning that uses both knowledge base and fact database. Reasoning may start with facts and include proving rules, depositing their conclusions as new facts, etc., until all possible new facts are derived. Alternative approach to reasoning is to start from some hypothesis and to try to prove it via rules in knowledge base – for those rules whose conclusion matches the hypothesis reasoning process tries to prove their conditions, etc., until we either prove the hypothesis or not.

Another important feature of expert systems is their ability to explain solutions that were found during the reasoning process. Explanation ability is important since it shows humans how expert system came to a conclusion, and this transparency helps humans in accepting expert systems as a valid tool for reasoning.

Typical tasks of expert systems are diagnosis of malfunctions (e.g. human diseases), configuration of complex objects (e.g. computer systems), or planning a sequence of actions (e.g. planning robot actions). One example of successful expert systems is Prospector, an expert system for mineral exploration. After geological characteristics of suspected deposit were given to the system, it makes an assessment of the suspected mineral deposit. In 1980 this system identified a molybdenum deposit near Mount Tolman in US. Drilling performed at the site confirmed the deposit exists and was worth over one hundred million US dollars.

6. MODELLING UNCERTAINTIES

Both data and knowledge can be uncertain. Uncertain data arise when we either cannot measure data more precisely or when description of data contains uncertainty. For example, we say that somebody has a "low" salary. Uncertain knowledge is knowledge that cannot describe some phenomenon precisely, but rather includes some uncertainties in description. For example, we may say that good manager will give an enterprise "a good chance" to survive.

Various methods were developed for working with uncertainties, like probability, Bayesian reasoning, uncertainty factors or fuzzy logic (Negnevitsky, 2002). We will present here main ideas of **fuzzy logic**, a method that successfully deals with systems that include uncertain behaviour. Fuzzy logic is based on the fact that humans use imprecise language to describe facts or rules, and we use these approximations to do classification. For example, when we say "high salary" we don't precisely say how high this salary is, and we imply that high salary is a category that includes certain class of salaries.

Fuzzy logic doesn't suppose that the range of "high salary" has to be fixed (e.g. between 4 and 5 thousands Euros a month) because in that case only a small differences in salary may lead to a conclusion that somebody has high salary with 4010 Euros per month and somebody else has a middle range salary with 3995 Euros a month. Instead, it supposes that object can belong to a category with certain degree. Moreover, one object can belong to several categories with different degree of membership - e.g. a salary of 3900 Euros could belong to "high salary" category with degree of membership 0.85, and to a "middle range salary" category with degree of membership 0.35. So, there is an overlap between membership functions (degrees of membership) for different categories. Sets with such characteristics are called fuzzy sets.

Fuzzy variables can be used in knowledge representation, e.g. a rule may declare that "IF a candidate for credit has a high salary THEN credit risk is low". Here "high salary" and "low credit risk" are fuzzy variables. Expert systems with fuzzy variables are simpler and have fewer rules than traditional expert systems, and knowledge expressed with fuzzy variables is more intuitive and closer to the way expert think. In fuzzy logic condition of a rule can be fulfilled to a certain degree, and a rule can be activated to the degree its condition is fulfilled. In expert systems with fuzzy rules all rules are activated simultaneously, and only those rules that are fulfilled to a nonzero degree are influencing the final value of output variables.

Fuzzy logic has many applications like in expert systems in business, medical or legal domain. They have also been used in transmission in cars, in fuzzy automatic focusing in video cameras or in air conditioners to control temperature. One of the best known examples is its use in Sendai Subway in Japan where fuzzy rules are used to control its speed of cruising, braking and switching.

7. MODELLING SYSTEMS WITH GRAPH STRUCTURE

Theory of graphs (Ore, 1990), that originated in 18th century, was able to help in solving various real problems, Until very recently graph theory was oriented in studying properties of graphs of limited size. However, in the last 5-6 years interest arose in studying properties of huge graphs that have grown through some evolutionary process (Hayes, 2000a; Hayes, 2000b).

For example, it was recently shown that the "diameter" of the Web at the time when it had about 800 million pages was 19. Web diameter is the shortest distance between the two most distant Web pages – the meaning of the term "distance" is the number of clicks needed to pass from one Web page to another one. Moreover, it was found that Web diameter is a logarithmic function of the Web size, and that a 10 times increase of Web size (number of Web pages) will cause increase of Web diameter from 19 to only 21. In order to study such giant graphs new computational techniques have to be developed.

This phenomenon of relatively small size of giant evolutionary graphs was named a "small world graphs". Another example of such graphs is a so called call graph, i.e. graph that describes calls between different phones in a telephone network. A study of one one-day call graph with more than 53 million vertices and 150 million edges has shown that it has one giant connected component whose size is almost 45 million vertices, i.e. 45 million telephones. The size of this huge connected component was found to be 20. Another interesting thing was found when studying complete graphs inside the entire call graph, i.e. graphs in which every vertex is joined to every other vertex. Each such complete graph describes a group of people that call each other at least once every day. It was found that there were about 14,000 complete graphs spanning as many as 30 vertices (i.e. group of people of size 30).

Investigation of properties of "small world graphs" was the beginning into investigation of graph structure of such graphs. Mathematical models of such structures typically take the form of an algorithm for generating graphs with some statistical properties. Several models were proposed, and one of the successful models is a combination of a lattice and a random graph, where lattice are highly regular graphs in which every vertex is connected with just a few of its neighbours.

8. MODELLING LEARNING

Learning is one of the key human abilities that help us to adapt to the environment and to perform well in our activities. This is the reason that modelling of learning was one of the important goals of artificial intelligence. Several modelling approaches were developed.

Development of **artificial neuron networks** (Skapura, 1996) started from the very beginning of the contemporary computer history, but only in 1980th they got proper theoretically background and enabled solution of complex real world problems. Artificial neuron networks (ANNs) were developed with the idea to overcome the constraints of programming, i.e. to enable computers to learn without the need to be reprogrammed by humans. Recognition of manual written text, speech or human faces were among the hard problems that are almost impossible to solve efficiently with programming.

ANNs are based on the principle of human brain operation, i.e. by simulation of a large number of very simple processors (neurons) connected with large number of interconnections. Neurons receive inputs that are generated as outputs by many other neurons, and neuron output may be input to a number of other neurons. Neuron activities are excited or inhibited by interconnections with other neurons. The whole brain thus works in a highly parallel way, with simultaneous actions of a huge number of neurons. ANN is a network of software simulated neurons located in input, hidden and output layers. Activity of the network is stimulated by a signal put on input layer of neurons. Each neuron then generates an output signal that is a function of the whole stimulation of that neuron, and these outputs become inputs for the next neuron layer. This process continues until neurons in output layer produce output signals that represent the answer of the ANN to the input stimulus. For example, we may stimulate the network with showing it some hand written letter (e.g. by scanning it), and as an output ANN presents the letter recognised during the process.

On the neuron level, output signal of a neuron is formed in such a way that the neuron first aggregates all input signals it receives as a sum of products of inputs with weights of connections between networks. Weights of connections between neurons are computed when the ANN passes through the learning cycle. Neuron activation function is then used to compute the neuron output signal on the basis of its aggregated input. Various network geometries and neuron activation function are in use.

There are two kind of learning, supervised and unsupervised learning. In unsupervised learning no guidance exists that tells the network in what classes to classify input signals, while in supervised learning class categories are defined in advance. Typical example of supervised learning is recognition of hand written letters and digits, while an example of unsupervised learning is finding similar documents in a set of documents. During the learning process weights of interconnections between neurons change until the network learns to recognise the characteristics of the input sample with an acceptable recognition error.

Another learning approach is **machine learning**. This method search through data and tries to find systematic statistical patterns or relationships. One group of machine learning methods called **induction from examples** enables extracting of knowledge from examples and forming rules or decision trees that accurately describe characteristics of data. The best known algorithm for induction from examples is an ID3 algorithm (Quinlan, 1982) that is using classification to learn essential features of a set of examples. The key problem in deriving the appropriate decision tree for a set of data under consideration is to find the variables that will separate the data into homogeneous groups, and will form as simple tree as possible. Learning is done with one part of data, while the rest of data is used for testing the obtained decision tree. Learning process shouldn't overfit the decision tree to the specific data.

The first program that demonstrated the ability of **machine learning in playing games** was Samuel's checkers player (Samuel, 1959) for which several methods of learning were developed. This program, although it used IBM 704 machine with only 10K memory, magnetic types for long-term storage, and a cycle time of about one millisecond, was playing on a very high level. Further improvement of computer models for playing checkers was done by Jonathan Schaeffer and his colleagues who developed a program Chinook that is using so called alpha-beta search technique for analysing future moves. Chinook was the first program to challenge for a world championship. In a first mach played in 1992 with dr. Marion Tinsley, a person who had been a world champion for 40 years and who lost just three games in all these years, dr. Tinsley won. However, in this match he suffered his 4th and 5th losses. In the match played in 1994 Tinsley had to withdraw from the match for health reasons after 6 draws, so that Chinook became the official world champion.

Finally, an interesting method for so called **concept learning** was developed and incorporated into the Eurisco model by Douglas Lenat (Johnson, 1986). Eurisco is a program for discovering and developing heuristics, i.e. approximate methods for solving problems. An interesting fact is that heuristic approach was used in generating these heuristics. Newly generated heuristics were evaluated by observing how they work in practice, and heuristics that performed better got higher weights. Eurisco was successful in different applications, e.g. in generalization of 2-D integrated circuit junction design to 3-D in VLSI chip design. One interesting application was in playing a very complex futuristic war game called Traveller, where each player must build a fleet from a constrained budget and obeying extremely complicated rules (described in about 200 pages long manual). Basic concepts of the game were incorporated to Eurisco, and after that it played thousands of simulated battles used to collect data for evaluation of its fleet design heuristics. On the basis of this experience Eurisco formed a strange fleet of lightly armoured and heavily armed fast ships – however this strange fleet won every battle it played and became a US Traveller champion.

9. MODELLING COMPETITION AND COOPERATION

An important class of problems include simultaneous decisions of several people or organizations where each subject is making decisions in accordance with its own desires, but in the same time he must try to judge what decisions will the other side make. So each player must think about the strategies that other players will use, and should be aware that other players also take care about the strategy he will use. Each subject must also decide with which other subjects he will cooperate or compete. These characteristics make these problems rather specific and complex.

Such kind of problems is modelled by the **theory of games** (Davis, 1997). Each player in these "games" is seeking best strategies for himself under proposition that other players will use best strategies for themselves too. All players have information on how much they will gain or loose for each combination of decisions of all players. In each game we want to find what each player must do in order to optimise his scores, and what will be the outcome of the game (i.e. how much will each player gain or loose).

Games that occur in reality and are treated by the theory of games are played in economy, business, social life, politics, military games, etc. Example of a business game is decision making of several competitive TV companies about what kind of TV program should be presented in what time period. Each company knows its strengths and weaknesses, as well as strengths and weaknesses of other companies. They also know how much will they gain or loose for all combination of decisions made by them and by other companies.

There are a variety of classes of games. Here we will only discuss two person games. Two person zero sum games with equilibrium points are games where players have conflict interests, since one player gets what the other one looses. Combination of strategies of both sides is called an equilibrium strategy, while the outcome of these two strategies is called an equilibrium point. By playing his equilibrium strategy player can get at least the value of the game and prevent the opposite party to get more that the value of the game.

However, many two person zero sum games don't have an equilibrium point. Such games cannot be analysed by pure strategies but rather with mixed strategies. Mixed strategies are using randomness, e.g. some mixed strategy can use one pure strategy in 65% of cases and another pure strategy in 35% of cases - for each specific decision selection of strategy is done in a random way.

In two person games where elements of cooperation appear both players have independent gains. In these types of games there is no universal accepted solution, i.e. there is no strategy that is clearly preferable to other strategies. Prisoner's dilemma game is an excellent example of such games. In this game equilibrium point is not the best solution for both prisoners. On the other hand, the best solution is not an equilibrium one since for each prisoner it is better to change his decision – however, if both prisoners would make such change both would be in a worse situation than in an equilibrium solution! In such games important factors are degree of communication between players, threats and negotiation.

Quite interesting empirical results on playing games were found (Axelrod, 1980). Here players familiar with the game are playing a series of the prisoner's dilemma game. Surprisingly, it was found that the best long term strategy is to never defect unless your partner defects first. This strategy includes short memories of what happened, and plays nice with defectors that later stop defecting.

Some cases of games played in nature were found too. One example is a cooperative game played by a small fish and its potential predator. Small fish eats parasites from the body and even the mouth of the large ones, while the large fish eats other smaller fishes. This works only if there is a consistent contact between the two fishes, and for that they must have an arranged meeting place (e.g. near he coast).

10. CONCLUSIONS

A number of powerful computer modelling and problem solving methods were developed in its rather brief history. These methods and their variants were invented for modelling and solving of a variety of different problems like dynamic systems simulation, linear and nonlinear optimisation, learning problems or competition problems. It was also found that a combination of these methods can give sound results. As an example, in the area of intelligent systems various combinations of methods like neural expert systems, neuro-fuzzy systems or evolutionary neural networks were developed and successfully applied in modelling of complex real problems.

Some of these modelling methods arise from analogy with operation of nature or human brain. Neural networks thus imitated architecture and operation of human brain. Various evolutionary methods like genetic algorithms or genetic programming were inspired by natural evolution. On the other hand, ant colony optimisation method (Doringo and Sttzle, 2004) was motivated by the collective behaviour of a group of species. However, although these analogies inspired the methods, a lot of research was needed in order that these methods become operational and efficient. As an example, neural networks needed several decades of hard work of many researchers to reach maturity and modelling power required for solving of complex real problems.

As we have seen, computer models enabled solution of a wide range of complex problems. These are problems from various aspects of business, engineering or social life. Such solutions help in increasing efficiency of business, providing better or more stable solutions, or solving strategic problems.

Since new type of problems appear with development of new technologies and with applications of these technologies to solution of various problems, new computer modelling methods or variants of current ones are being developed too. Thus there is no reason to expect that computer modelling will stop to provide us with new and exciting ideas and solutions.

REFERENCES

[1] Axelrod, R. (1980): *Effective Choice in the Prisoner's Dilemma*, Journal of Conflict Resolution, Vol. 24, 3-25.

[2] Davis, M. D. (1997), *Game Theory: a Nontechnical Introduction*, Dover Publications, Inc., Mineola, N.Y.

[3] Doringo, M. and Sttzle, T. (2004), *Ant Colony Optimization*, Bradford Books.

[4] Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

[5] B, Hayes (2000a): *Computing Science: Graph Theory in Practice: Part II*: American Scientist, Vol. 88, No. 1, 9-13.

[6] Hayes, B. (2000b): *Computing Science: Graph Theory in Practice: Part II*. American Scientist, Vol. 88, No. 2, 104-109.

[7] Johnson, G. (1986), *Machinery of the Mind – Inside the New Science of Artificial Intelligence*, Times Books, New York, N.Y.

[8] Law, A. M. and Kelton, W. D. (1999), *Simulation Modelling and Analysis*, 3rd Edition, McGraw-Hill, New York.

[9] Michalewicz, Z. and Fogel, D. B. (2002), *How to Solve It: Modern Heuristics*, Springer, 2002.

[10] Negnevitsky, M. (2002), *Artificial Intelligence: A Guide to Intelligent Systems*, Addison-Wesley.

[11] Oakshott, L. (1997), *Business Modelling and Simulation*, Pittman Publishing, London.

[12] Ore, O. (1990), *Graphs and Their Uses*, Revised and updated by R. J. Wilson, The Mathematical Association of America.

[13] Pidd, M. (2004), *Tools for Thinking: Modelling in Management Science*, 2nd Edition, Wiley, Chichester.

[14] Quinlan, J. R. (1982): *Semi-autonomous Acquisition of Pattern-Based Knowledge*, in *Introductory Readings in Expert Systems*, D. Michie (Ed.), Gordon and Breach, New York, N.Y.

[15] Reisig, W. (1985), *Petri Nets: An Introduction*, Springer-Verlag, Berlin.

[16] Resnick, M. (1994), *Turtles, Termites and Traffic Jams: Explorations in Massively Parallel Microworlds*, The MIT Press, Cambridge, Mass.

[17] Saaty, T. L. (1990), *Multicriteria Decision Making: The Analytic Hierarchy Process***.** 2nd Edition, RWS Publications.

[18] Seila, A., V. Čerić, V. and P. Tadikamalla, P. (2003), *Applied Simulation Modeling*, Thomson - Brooks/Cole.

[19] Skapura, D. M. (1996), *Building Neural Networks*, Addison Wesley, Reading, MA.

[20] Sterman, J. D. (1991), *A Sceptic's Guide to Computer Models*, in Barney, G. O. et al. (editors), *Managing the Nation: The Microcomputer Software Catalog*, Westview Press, Boulder, Co., 209-229.

[21] Samuel, A.L.(1959): *Some Studies in Machine Learning Using the Game of Checkers*, IBM Journal of Research and Development, Vol. 3, 211-32.

[22] Sterman, J. D. (2000),, *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Irwin/McGraw Hill, Boston, MA.

[23] Winston, W. L. (1991), *Operations Research: Applications and Algorithms*, PWS-Kent Publ. Comp., Boston, MA.