I T H E A

International Journal

INFORMATION THEORIES & APPLICATIONS

2009 Volume 16 Number 3

International Journal
INFORMATION THEORIES & APPLICATIONS
Volume 16 / 2009, Number 3

IJ ITA is official publisher of the scientific papers of the members of the ITHEA® International Scientific Society

IJ ITA welcomes scientific papers connected with any information theory or its application.

IJ ITA rules for preparing the manuscripts are compulsory.
The rules for the papers for IJ ITA as well as the subscription fees are given on *www.ithea.org* .
The camera-ready copy of the paper should be received by http://ij.ithea.org.
Responsibility for papers published in IJ ITA belongs to authors.

General Sponsor of IJ ITA is the Consortium FOI Bulgaria (www.foibg.com).

International Journal "INFORMATION THEORIES & APPLICATIONS" Vol.16, Number 3, 2009

Printed in Bulgaria

Edited by the Institute of Information Theories and Applications FOI ITHEA®, Bulgaria,
in collaboration with the V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
and the Institute of Mathematics and Informatics, BAS, Bulgaria.

Publisher: ITHEA®
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, e-mail: info@foibg.com

ISSN 1310-0513 (printed) ISSN 1313-0463 (online) ISSN 1313-0498 (CD/DVD)

GENE CODIFICATION FOR NOVEL DNA COMPUTING PROCEDURES

Angel Goni Moreno, Paula Cordero, Juan Castellanos

Abstract: The aim of the paper is to show how the suitable codification of genes can help to the correct resolution of a problem using DNA computations. Genes are the income data of the problem to solve so the first task to carry out is the definition of the genes in order to perform a complete computation in the best way possible. In this paper we propose a model of encoding data into DNA strands so that this data can be used in the simulation of a genetic algorithm based on molecular operations. The first problem when trying to apply an algorithm in DNA computing must be how to codify the data that the algorithm will use. With preciseness, the gene formation exposed in this paper allows us to join the codification and evaluation steps in one single stage. Furthermore, these genes turn out to be stable in a DNA soup because we use bond-free languages in their definition. Previous work on DNA coding defined bond-free languages which several properties assuring the stability of any DNA word of such a language. We prove that a bond-free language is necessary but not sufficient to codify a gene giving the correct codification. That is due to the fact that selection must be done based on a concrete gene characterization. This characterization can be developed in many different ways codifying what we call the fitness field of the gene. It is shown how to use several DNA computing procedures based on genes from single and double stranded molecules to more complex DNA structures like plasmids.

Keywords: DNA Computing, Bond-Free Languages, Genetic Algorithms, Gene Computing.

ACM Classification Keywords: I.6. Simulation and Modeling, B.7.1 Advanced Technologies, J.3 Biology and Genetics

Introduction

Since the beginning of computation, John Von Neumann held that the different machine models should try to imitate the functions which take place in living beings. Recently, two paradigms of biological inspiration are being applied very satisfactorily to the resolution of problems: neural nets and genetic algorithms. Nowadays, computer scientist try to go a little bit further by working with the same row material the nature does. That is the case of Leonard Adleman who is the pioneer in this field and solved a problem using real DNA strands. In a short period of time DNA based computations have shown lots of advantages compared with electronic computers. DNA computers could solve combinatorial problems that an electronic computer cannot like the well known class of NP complete problems. That is due to the fact that DNA computers are massively parallel [Adleman, 1994].

Computing using a DNA molecule is a modern approach to a massive parallel paradigm. This concept is based on the work made by Leonard Adleman [Adleman, 1994], where the first implementation of a computer based on DNA operations solved a hard combinatorial problem using deoxyribonucleic acid molecules. The problem which Adleman solved was the well known Hamiltonian Path Problem (HPP). This problem consists of finding, given an undirected graph, weather there is a Hamiltonian Path or not in the graph. A Hamiltonian Path is a path which visits each vertex exactly once. Adleman showed the results of solving a HPP of seven vertexes. This problem belongs to the class of the problems named NP-complete. These kinds of problems present a great complexity and there is no polynomial algorithm known that solves them. A year later Richard J.Lipton [Lipton, 1995] wrote a

paper in which he discusses, in detail, many operations that are useful in working with a molecular computer. After this moment many others followed them and started working on this new way of computing.

Molecular computing consists of representing the information of the problem with organic molecules and to make them react within a test tube in order to solve a problem. The fundamental characteristics of this type of computations are, mainly, the massive parallelism of DNA strands and the Watson-Crick [Watson-Crick, 1953] complementarity. The speed of calculation, the small consumption of energy and the big amount of information which DNA strands are able to store are the best advantages that DNA computing has. Nevertheless one of the problems is the massive calculation space needed, which limits the size of the problems.

Despite all the impressive benefits that DNA computations have, they also have several drawbacks. The biggest disadvantage is that until now molecular computation has been used with exact and "brute force" algorithms. It is necessary for DNA computation to expand its algorithmic techniques to incorporate aproximative and probabilistic algorithms and heuristics so the resolution of large instances of NP complete problems will be possible. Without algorithms DNA computing has linear time solving NP-Complete problems but exponential space.

Genetic Algorithms (GA's) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem.

John Holland in 1975 was the first one to study an algorithm based on an analogy with the genetic structure and behaviour of chromosomes. The structure of a basic genetic algorithm includes the following steps. (1) Generate the initial population and evaluate the fitness for each individual, (2) select the best individuals, (3) cross and mutate selected individuals, (4) evaluate and introduce the new created individuals in the initial population. All those steps together are called a generation.

Before generating the initial population, individuals need to be coded. That is the first thing to be done when deal with a problem so that it can be made combinations, duplications, copies, quick fitness evaluation and selection. Nature is a big genetic algorithm in which we are the individuals of the problem. Each of us is coded as a base sequence. We are all different form each other thanks to that sequence. A concrete code must have some characteristics that identify the individual to be more or less qualified.

Previous work on molecular computation for genetic algorithms [J.Castellanos, 1998] shows the possibility of solving optimization problems without generating or exploring the complete search space. A recent work produced a new approach to the problem of fitness evaluation declaring that the fitness of the individual should be embedded in his genes (in the case of the travelling salesman problem in each arch of the path). In both cases the fitness will be determined by the content in G+C (cytosine + guanine) which implies that the fitness of an individual will be directly related with the fusion temperature and hence would be identifiable by spectophotometry and separable by electrophoresis techniques [Macek 1997] or centrifugations.

Physico-Chemical Properties of DNA

DNA, deoxyribonucleic acid, is the main motor which moves this new computer paradigm. A DNA molecule consists of two single strands twisted. Each strand is a long polymer of bases. Four different bases are presented in DNA: adenine (A), thymine (T), cytosine (C) and guanine (G). It is the sequence of these four bases that encodes information.

In 1950, Erwin Chargaff analyzed the base composition of DNA composition in a number of organisms. He reported that DNA composition varies from one species to another. Such evidence of molecular diversity, which had been presumed absent from DNA, made DNA a more credible candidate for the genetic material than proteins are. [Chargaff, 1950]

Chargaff found that a peculiar regularity in the ratios of nucleotide bases. In the DNA of each species he studies, the number of adenines approximately equaled the number of thymine, and the number of guanines approximately equaled the number of cytosine. In human DNA, for example, the four bases are present in these percentages: A=30.9% and T=29.4%; G=19.9% and C=19.8%. The A=T and G=C equalities, later known as Chargaff's rules, helped Watson and Crick to discover the structure of DNA.

Chargaff's Rules:

(a) [A+G]=[T+C] The sum of the purines (A+G) equals that of the pyrimidines (T+C). The (C+G) and (A+T) ratio equals 1, (A+G)/(T+C)=1.

(b) The molar ratio of adenine to thymine equals 1, (A/T = 1)

(c) The molar ratio of guanine to cytosine equals 1, (G/C=1).And, as a direct consequence of these relationships.

(d) The (C+G) and (A+T) ratio varies from organism to organism.

Different variations can be observed in this relation for each type of organism, from 0.37 to 3,16. This causes that this relation is no good to express the composition of bases of a DNA molecule. The composition of bases of a DNA molecule, usually is expressed by the fraction of bases that are in pairs G.C, is [G] + [C]/[total bases]. This fraction is known like the content in G+C.

There is a linear relation between the content of G+C and the density of the DNA determined in a density gradient. A bigger content of G+C will make stronger the density of the DNA. Due to multiple studies about the density of DNA molecules taken from different organisms and their composition on nitrogenous bases, it has been established an empiric formula that relates the density of flotation (p) to the content of G+C expressed in mole percentage. This formula is the following:

$$p = 1{,}660 + 0{,}00098(G+C)$$

Watson and Crick proposed a structural model for the DNA known as the “Model of the double helix” based on Chargaff work. The structure of a DNA molecule is a double-helix, in which one strand runs from the 5' to the 3' end, and the other one goes in the opposite direction. The two strands interweave, giving the whole molecule a right-handed helical twist. The twist of the helix makes the whole molecules to have a new turn every 34 Amstrong.

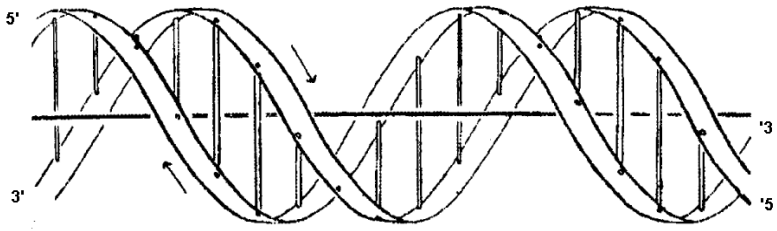


Fig. 1

In a laboratory, the set of DNA molecules is within a test tube, like a 'soup'. Single nucleotides are linked together end-to-end to form DNA strands. Under favorable physical tube conditions two single DNA strand composed by complementary nucleotides pairs, AT and G-C, are join together to form the double helix in a process called hybridization.

The Adenine of a helix matches the Thymine of the complementary helix by creating two hydrogen bonds. Also, the Guanine of a helix matches the Cytosine of the complementary three hydrogen bonds. Therefore, the bases of one strand are united by hydrogen bonds to the bases of the other strand, forming the base pairs A-T and G-C.

Fig. 2. Vertical lines represent bonds (2 bonds A-T and 3 bonds G-C) between complementary nucleotides of two single DNA sequences: 5'-AACCTTACGTTAGG-'3 and 3'-TTGGAATGCAATCC-'5.

In Figure2, the bonds formed by the two single DNA strands are complete due to perfect complementarity of the nucleotides that compose each DNA molecule; nevertheless this does not approach closer to reality of a laboratory. In many cases (Figure 3) it is possible that two parts of molecules will bind together even though some of their corresponding nucleotides are not complementary to each other.

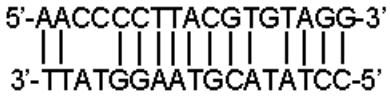


Fig. 3. Some of the corresponding nucleotides of the DNA sequence are not complementary to each other.

Gene Characterization

In DNA computing is very important to know that instability of DNA strands can cause undesirable reactions. When facing a problem of any kind, one of the most important things to do is assuring that the data the problem will use is stable. It does not matter whether we are working with DNA or not to carry out this task. During the next sections we will tackle the problem of data encoding in DNA computing problems, concretely, in a simulation of a genetic algorithm with DNA.

The input data for DNA computing must be encoded into single or double DNA strands. Many conditions can cause loss of DNA bases or strand breakage and due to the Watson-Crick complementarity's parts of single DNA strands can bind together forming a double-stranded DNA sequence. Also, several DNA operations like electrophoresis or isopycnic centrifugation, which are absolutely essential for a correct DNA computation, are based on certain characteristics of the DNA strands. Those characteristics can not be altered if we want carry out a problem with DNA. For example, in the case of genetic algorithms, electrophoresis helps us to select the better adapted individuals. That operation is essential for the process of selection of the fittest and we have to take it into account we generating our data: the genes.

We must take care of all these conditions and characteristics so that we can assure the stability of every data of our problem. We can not choose our data randomly making long sequences of bases (A, C, G, and T) because as bigger is our initial data set, more mistakes we will find during the computation.

Taking all into account, we can distinguish two different problems: codification of a stable DNA language and codification of the genes.

Codification of a DNA Language

First of all we have to recall a list of known properties of DNA languages which are free of certain types of undesirable bonds and give a solution as a uniform formal language inequation [Lila Kari, 2004]. That is to create a language from which you can choose any word and be sure of the stability of that DNA molecule. Such a language is called a bond-free language. The main variables to take into account when describing a bond-free language are the length of the words of such a language (d) and the number of words that compose the language (w). The second variable is very important if we try to give the alphabet the most stability possible. Obviously, if we want to create an alphabet of twenty words (w=20) of five nucleotides each (d=5) we will be able to give them more stability (no complementarities between them) than if the alphabet has forty words (w=40) of the same length. That is due to the fact that there are only 4 nucleotides to combine in sequences (a, c, g, t) and the lower the number w is the bigger Hamming Distance we can get among them. Here, we understand the Hamming Distance between two different words as how complement the words are. If the words are not complement at all, the Hamming Distance would be rather high. There must be the highest distance possible among the words w of the stable alphabet.

Codification of the Genes

We want to highlight the possibilities that offer the storage of the information in genes, one word is saved in a different gene, and these genes possess numerous properties (weight, size, ability). Some of the most precise operations that we can realize with the DNA are based on these properties. In our simulation each gene has a different amount of C+G bases. That condition identifies each gene.

When simulating a genetic algorithm, the individuals are formed by several genes and each gene has its own information or characteristic. This characteristic is the content of Cytosine and Guanine they have. An individual would have this aspect:

PCR-primer Np Rep XY RE0 XY RE1 … REn-1 XY Rep Np-1 PCR-primer

XY (gene) is better evaluated as more C+G content

Were the beginning of the individual and the end of it is the almost the same sequence of bases (PCR-primer Np Rep) and the different genes are separated by restriction enzymes (RE).

In this way the individuals are already evaluated. Once they are evaluated we must select them. By isopycnic centrifugation we can select the best suited to their environment. This technique is used to isolate DNA strands basing on the concentration of Cytosine and Guanine they have. The relationship between this concentration and the density (θ) of the strand is:

$$\Theta = 0{,}100[\%(G+C)] + 1{,}658$$

To begin the analysis, the DNA is placed in a centrifuge for several hours at high speed to generate certain force. The DNA molecules will then be separated based primarily on the relative proportions of AT (adenine and thymine base pairs) to GC (guanine and cytosine base pairs), using θ to know that proportion [Gerald Karp, 2005]. The molecule with greater proportion of GC base pairs will have a higher density while the molecule with grater proportion of AT base pairs will have a lower density. In this way the different individuals (different paths or solutions of TSP) are separated and can be easily selected.

With this technique we can identify one gene from the rest. But this work would be useless unless we codify the rest of the gene using a stable DNA language. If not, the genes we define in a genetic algorithm could be altered due to DNA instability.

The first problem can be solved using bond-free languages [Bo Cui, 2007] but those kind of languages do not allow us to codify the genes. We can not choose a word of that language, a sequence of nucleotides, to make a gene because it won't be different from the rest. These methods give a language which assures that any word w_1 of such a language is stable and won't bind together another word w_2 of the same language , but does not assign a proportional weight to the different words of the language. This implies that the genes could not be arranged by weight, preventing from realizing operations with DNA of that properties of the language could take advantage directly.

Every genetic algorithm will need different genes. The genes are the initial data and they must be well defined in order to obtain the correct solution to the problem. However, all the genes would have a similar format. This format must solve both of the problems, codification of a DNA language and codification of the genes. We will use a bond-free language to define most part of the gene but we will add some other characteristic (the fitness of the gene) that must be suitable for a concrete problem. This would be the aspect of a gene:

Bond-Free DNA language (w1) –- Fitness(w3) –- Bond-Free DNA language(w2)

Gene Encoding Language

Words w1, w2, w3: nucleotide sequences

For the language used to surround the fitness we will use a bond-free language. This language assure stability taking care of several conditions like temperature, complementarity, sequences of the same base, concentration of G+C, etc. Any word we take from a bond-free language can be optimal for this task, taking into account that all the words must be different to create different genes. A concrete problem will tell us how long this word must be and how many nucleotides we will use to make this words.

That kind of language could be useful in the resolution of DNA problems that uses 'brute force'. That is the case of Adleman's experiment. But if we try to go further in exploring the possibilities of DNA computing we must use algorithms like, for example, genetic algorithms. To complete the Gene Encoding Language we use a certain characteristic (fitness) which is outside 'stable' languages but are necessary to give a weight to each gene.

This characteristic which makes a gene different from the rest of the genes of the problem is based on the concentration of Cytosine and Guanine they have. Here also the problem will tell us how long this word must be. Anyway, this word should be much smaller than the other parts preserving the stability.

Example

Now we are going to establishing the notation that would allow us to describe the formalizations. Specifically, we define the terms *node, fitness, gene and language of genes*. For this example we will use the well known Travelling Salesman Problem (TSP), see figure 4. If the salesman starts at city X1, and it he distances between every city are known, what is the shortest path which visits all cities and returns to city X1?

We define *Xi* as a node of the graph (a city), and *Wij* as the length or fitness of a archers in the graph (the distance of the road).

We consider a gene as the minimal data unit of our problem and it is denoted by *Yi*. *Yi* is based on three parameters *{ Xi + Wij + Xj}* which are three different nucleotides sequence. The number of different genes in a problem is always the same as the number of arches in the graph.

$$Yi = \{ Xi + Wij + Xj \} \qquad I,j = 1..n$$

Once we codify all the genes, we have an alphabet of genes which can be used to form paths. A path is composed by a sequence of genes and they are possible solutions of the problem. We represent a path by Zi.

We codify *Xi* using a bond-free language, and *Wij* using a special language that preserves the weight of the gene. In this case, that special language gives each gene a different concentration of C+G.

More concentration of guanine and cytosine represent a shorter way. On the other hand, the lack of these nucleotides means that the gene represents a long way between two cities.

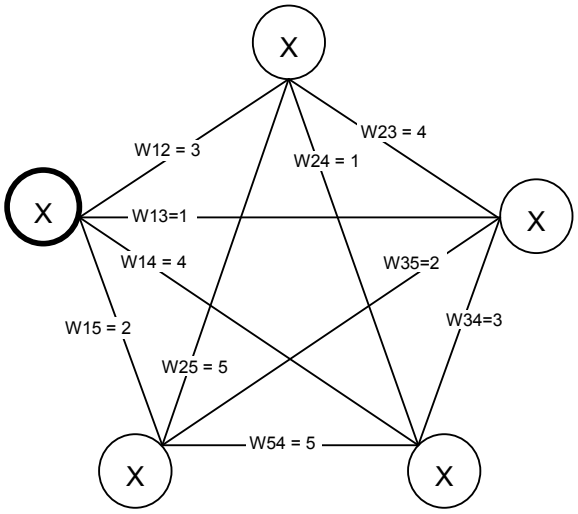


Fig. 4

The graph shown in figure 4 represents a map with five cities. All the cities are connected by roads of length W_{ij}. The city X_1 is the initial city. Five nodes are represented $\{X_1, X_2, X_3, X_4, X_5\}$ and, as the graph is fully connected, ten roads are defined $\{W_{12}, W_{13}, W_{14}, W_{15}, W_{25}, W_{24}, W_{23}, W_{35}, W_{34}, W_{45}\}$.In this example of TSP the roads (edges of the graph) have values between 1 and 5. The first step when trying to solve the problem with DNA is to assign a concrete DNA word to each data we have.

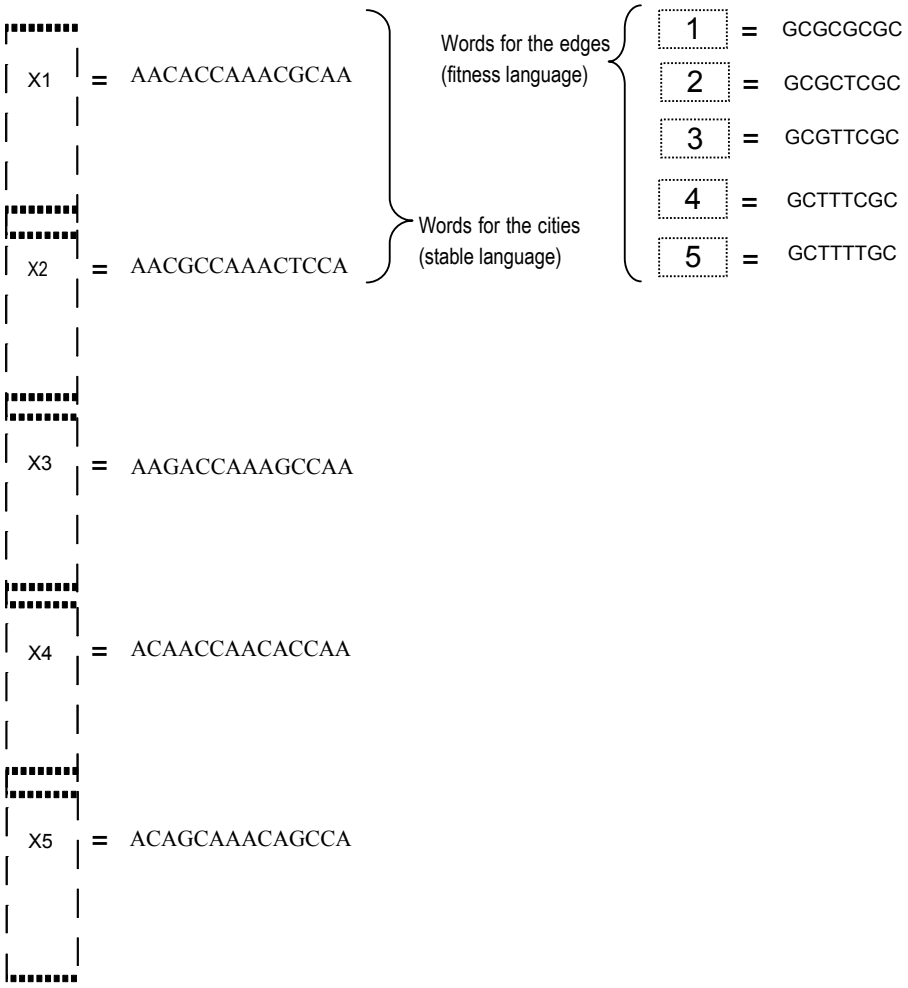


Fig. 5

As it is shown in figure 5 we assign each city a different nucleotide sequence based on a bond-free language in order to make our data set stable. The words have been composed by the software presented by Bo Cui and Stavros Konstantinidis [Bo Cui, 2007] using a constant GC Ratio and a Hamming distance equal to zero. Also we assign sequences to the roads. The sequences of the roads are not chosen randomly from a certain language. They are chosen by the rule explained before: as shorter a road is, more concentration of C+G that gene would have. Only five values are possible for the roads as they are one, two, three, four or five kilometres long. To sum up, we can distinguish between the stable language (vertexes) and the fitness language (edges).

To represent the cities we use a 14-base sequence and to represent the roads we use an 8-base sequence. As long as this is an example developed in-info the sequences result to be long enough but if we carry out the experiment in-vitro is important to notice that with longer chains the results obtained in stability are better. A gene is the conjunction of a departure city, a road and the arrival city. In this case there are ten different genes. Those genes are shown in figure 6.

Y1	=	X1 +	W12 +	X2	=	AACACCAAACGCAA +	GCGTTCGC +	AACGCCAAACTCCA
Y2	=	X1 +	W13 +	X3	=	AACACCAAACGCAA +	GCGCGCGC +	AAGACCAAAGCCAA
Y3	=	X1 +	W14 +	X4	=	AACACCAAACGCAA +	GCTTTCGC +	ACAACCAACACCAA
Y4	=	X1 +	W15 +	X5	=	AACACCAAACGCAA +	GCGCTCGC +	ACAGCAAACAGCCA
Y5	=	X2 +	W23 +	X3	=	AACGCCAAACTCCA +	GCTTTCGC +	AAGACCAAAGCCAA
Y6	=	X2 +	W24 +	X4	=	AACGCCAAACTCCA +	GCGCGCGC +	ACAACCAACACCAA
Y7	=	X2 +	W25 +	X5	=	AACGCCAAACTCCA +	GCTTTTGC +	ACAGCAAACAGCCA
Y8	=	X3 +	W35 +	X5	=	AAGACCAAAGCCAA +	GCGCTCGC +	ACAGCAAACAGCCA
Y9	=	X3 +	W34 +	X4	=	AAGACCAAAGCCAA +	GCGTTCGC +	ACAACCAACACCAA
Y10	=	X4 +	W45 +	X5	=	ACAACCAACACCAA +	GCTTTTGC +	ACAGCAAACAGCCA

Fig. 6

In figure 6 there are described all the possible genes in the TSP of figure 4. These genes are denoted by Y_i where i is a natural number between 1 and 5. These sequences are defined before starting the resolution of the problem as single stranded sequences. The possible solutions of the problem are represented by a sequence of genes which we call a path. The paths are double stranded DNA molecules so it is necessary a process to form those molecules from the single stranded sequences of genes. Such a process is based on the complementarity of bases in order to reach long sequences of genes. This process is shown in figure 7.

In figure 7A) and figure 7B) the process of how two different genes are joined together is detailed in high and low level. The main factor which allows this formation is the presence of linker sequences. We call linker sequences to those complementary sequences of the second part of a gene and the first part of other gene. In the example of figure 7A) genes Y_1 and Y_6 form a double helix thanks to one of this linker sequences. Now we will explain how to form these linkers. Gene Y_1 has a fitness field with value W_{12} so there will be called W_{12}^1 to the first part of that fitness and W_{12}^2 to the second part. The complementary chains of W_{12}^1 and W_{12}^2 are denoted by W'_{12}^1 and W'_{12}^2. It is necessary to conclude the formation of the linker to add the complementary chain of city X_2 (X'_2). The result of pouring those three chains into a DNA soup can be seen in figure 7B).

In the way explained above the different paths *Z* of the problem are generated spontaneously like the one of figure 7C) but it is before this generation when a suitable encoding of the genes can help us very much with the stability of our problem. It is when single stranded molecules are free when we can achieve the goal of making our data stable.

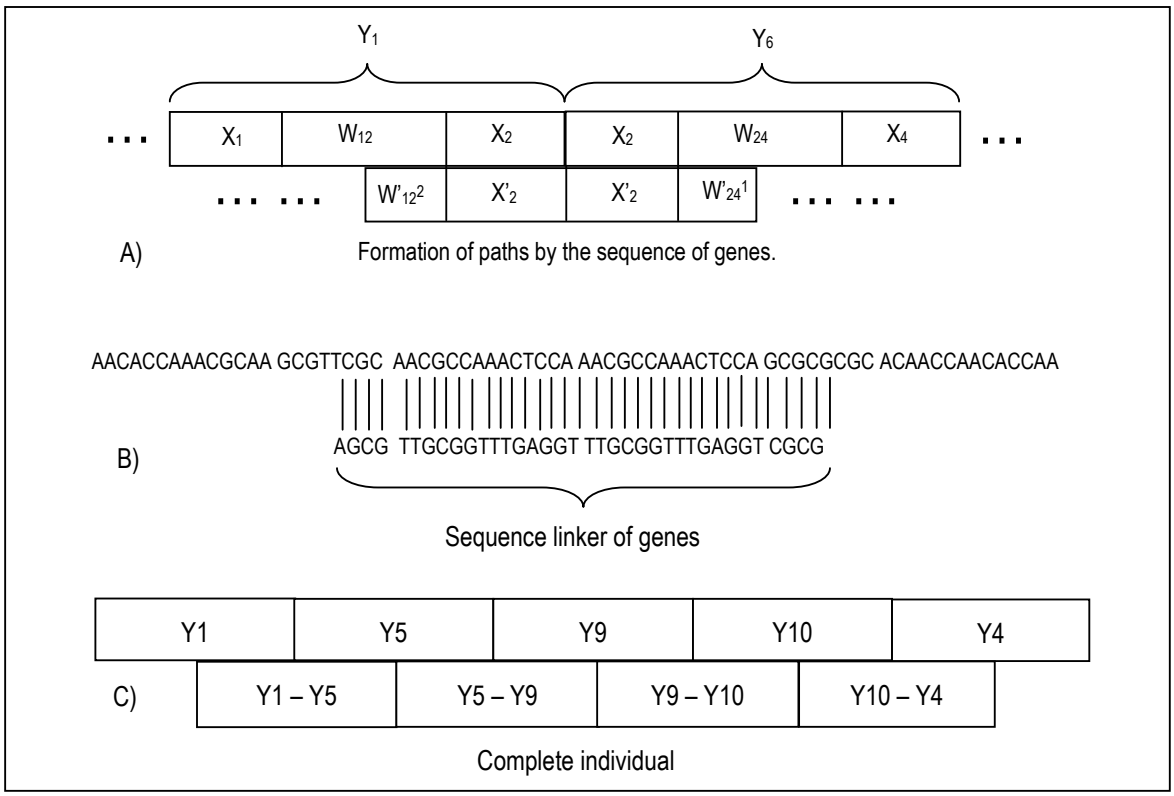


Fig. 7

Irregularities Due to Instability

There must be considered several instability factors before codifying the data (like those explained in previous sections). When we have all our genes ready to be mixed in a single test tube in-vitro in order to begin the computation a big advantage of a good codification can be seen: the uniform resistance to temperature. One of the clues that allow DNA computations is that the average amount of every data is about the same. As we do not have any power over the processes that happen in the test tube we should make sure before starting the experiment that this average value will be uniform during the computation. One of the main characteristics to take into account is the variation of temperature. A bad codification could lead us to the situation shown in the figure 8 where the initial set of data (several copies of two genes) suffer from a variation of temperature giving as a result a different set of data in which the second gene has lost most of its copies.

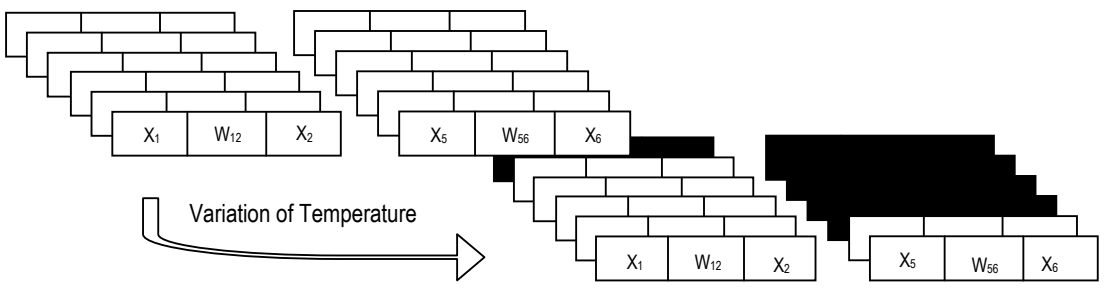


Fig. 8

The other big advantage of the codification using bond-free languages is to preserve the genes of our problem to make unwanted bonds and avoid the renaturalization process between the initial data. There are different forms of renaturalization if we do not use a bond-free language to define the genes.

First of all it is important to notice that a single gene can bond to itself in two different ways. The first one is shown in figure 9A) where both of the cities that surround the fitness of one gene are complementary. This complementarity does not have to be necessarily between the whole chain (X_2 or X_1). A high percentage of complementary bases can be enough to coil the gene and transform it into an invalid gene. The second way of losing a gene is exposed in figure 9B) where the chain of a single city (X_1 in the figure) is long enough to contain complementary sequences inside it. The result of making arbitrary sequences instead of using a bond-free language to define the gene is the loss of data. If this happens frequently the experiment will be doomed to failure.

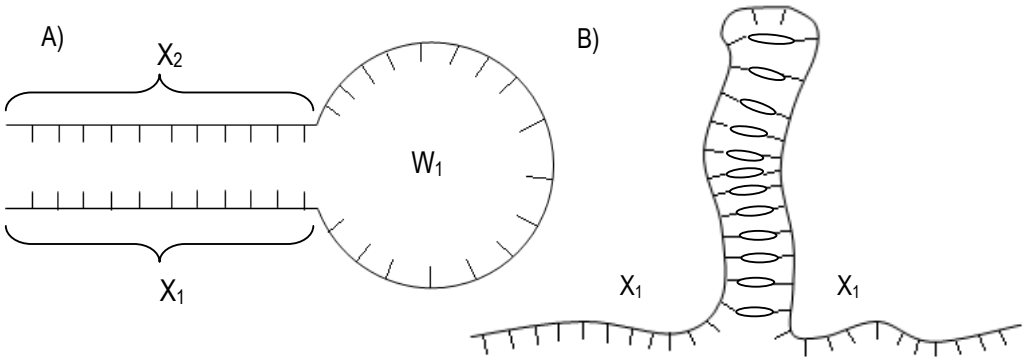


Fig. 9

Apart from the self-complementarity in genes, there also can be complementarities between two different genes like the illustration of figure 10. In this concrete example cities X_2 and X_5 show a partial complementarity between them. The result would be a double helix composed by both of the gens. That is not a correct structure for the resolution of the problem

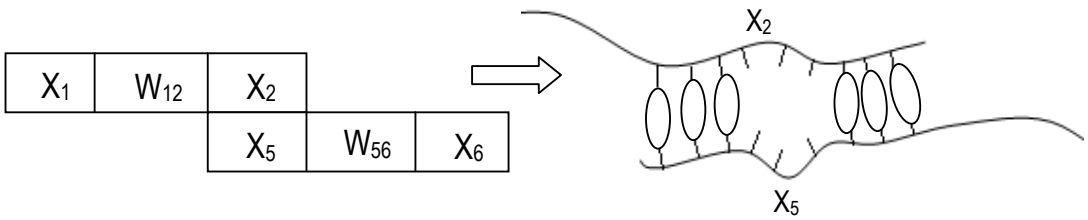


Fig. 10

As it has been explained and illustrated sufficiently, the codification of the initial data is a vital process for DNA computation and bond-free languages are necessary in this task. A probabilistic demonstration of this fact would be very illustrative. Let's call A to the phenomenon of renaturalization between randomly defined sequences and B to the phenomenon of renaturalization between sequences defined by bond-free languages. It is important that gene codification is used in a genetic algorithm simulation (GA) instead of using the brute force (F) applied in previous experiments. We can verify the following statements:

- P (A|GA) << P (A|F). The probability of the phenomenon A in a GA is much lower than using F due to de lower amount of DNA used.
- P (A) >> P (B) as explained in this section.
- P (B|GA) << P (A|GA). The probability of the phenomenon B in a GA is much lower than the probability of the phenomenon A in a GA. The benefits of a good codification are worthy.

Computations Based on Genes

The formation of genes like the ones used in solving the TSP shown in previous sections can lead us to a novel computation paradigm based on the computations of genes. It is important to emphasize that the DNA sequence which is between the bond-free sequences is the main field of the gene in order to complete the computations needed. That field is called the "fitness field" and it is used to carry out the selection step of a genetic algorithm. As we have seen, the condition of selection must be taken into account when the codification of the initial population is done. The reason for such a necessity is the time saved when the codification and evaluation steps of a genetic algorithm are done in a single step.

In the example showed above we considered this fitness field to consist of a specific sequence where a concrete number of G-C base pairs where codified. This codification allowed us to identify a concrete gene by using different techniques like isopycnic centrifugation which can separate molecules based primarily on the relative proportions of AT to GC base pairs. But this fitness field can be codified using a lot of different strategies to perform de selection step. For instance, there are several molecular procedures which use a specific sequence of antibiotic resistance to select genes. Imagine we have only three kinds of genes, those with an antibiotic resistance 1 (class 1), those with an antibiotic resistance 2 (class 2) and finally those with an antibiotic resistance 3 (class 3). If we ant to select the genes of class 1 then we only have to pour a solution of antibiotic 1 to the soup. The result would be that those genes without the resistance 1 will die and we will get in the soup only the genes wanted.

Gene oriented computations lead us to a different computation algorithm from the algorithms showed in the example of TSP above. This algorithm would be reflected in the illustration of figure 11.

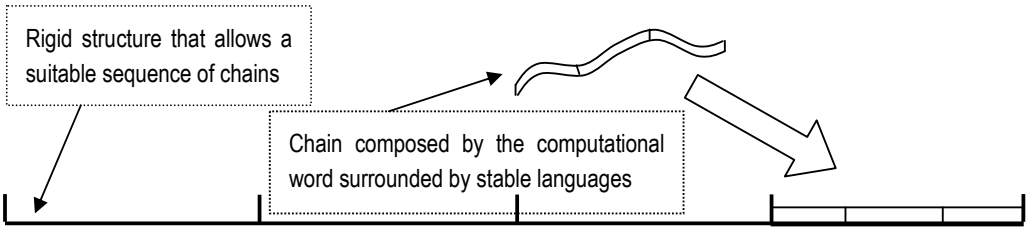


Fig. 11

This computation based on a rigid structure tries to give an answer to different problems (also NP-problems) like the Hamiltonian Path Problem (HPP). Instead of pouring all the genes in a soup and let them react guided by their codification, we codify a concrete sequence called rigid structure which represents the solution of the problem. This method is oriented to those kinds of problems in which we have to search for the solution in order to give a positive answer (YES) if the solution is achieved or a negative (NOT) if it is not. The positive answer would mean that the rigid structure is completed with genes.

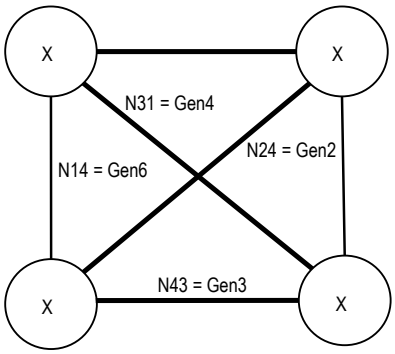
In the next section we will explain how to use more difficult DNA structures to perform these computations.

Computation by Using Plasmid

In this section it will be explained, in general, a possible computational method of one step included in the resolution of a complete algorithm that solves the well-known Hamiltonian Path Problem (NP-Complete Problem).

In graphs theory, an instance of the Hamiltonian Path Problem tries to determine if there is a cycle in a concrete graph that fulfills certain conditions. A Hamiltonian cycle is that cycle which passes through all the nodes or vertices of the graph exactly once. When we talk about directed graphs with a different weight on each edge, the Hamiltonian cycle (Cy) with a smaller cost (Co = $\sum$ edge_weigh $\forall$ edges of Cy) is also known as the solution to the Traveling Salesman Problem.

As this investigation advance we part from a codified population of ways using plasmids or vectors, as we will see next. This population must have been created following successive steps of a determined algorithm in such a form that finally we could get all the possible combinations of edges of the graph (figure 4) in order to obtain all the possible solutions of the problem. Concretely, we will focus the problem on the graph illustrated in the previous section (figure 4). In the initial set of paths we could find valid, invalid, complete and incomplete solutions. That is why the procedure explained next is so important. It is based on making possible the detection of an existent Hamiltonian way.



Each problem is represented on a certain graph. Not all the graphs will have a Hamiltonian Path (figure 13B). The purpose of final algorithm to solve HPP is to tell us if there is a solution in our graph or not.

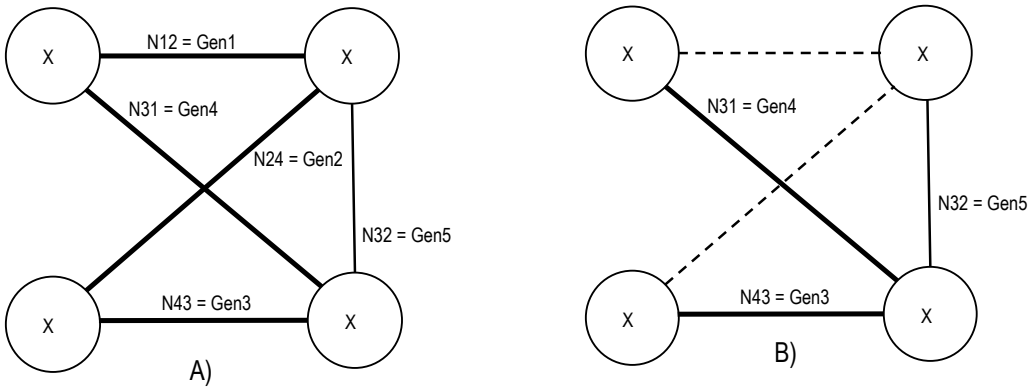


Fig. 13. The graph 13A would have a positive solution. A negative solution would be given under graph 13B.

The theoretical development of the presented proposal is based on a specific codification of the existing edges in the graph of the problem using nucleotides sequences which codify concrete genes. These genes will be expressed in fluorescents proteins when transcribed. Each edge of the graph will be codified as shown in figure 14. The fitness field representing the fluorescent gene is surrounded by bond-free sequences which represents half of the vertex.

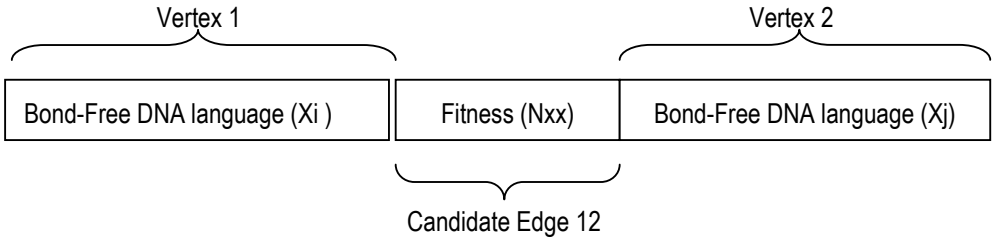


Fig. 14

Next there is a table which represents the correspondence between the existing edges in the graph (figure 13A) and the genes associated. The genes represented in the explanatory example are normal nucleotide sequences chosen randomly in order to simplify and clarify the method applied. However, there could be taken into account several possibilities to select the family of genes whose expression result to be fluorescent proteins to carry out the experiment. In this case we could design the experiment with proteins of the type GFP (the green fluorescent protein) and work with the variants that GFP produces like the RFP protein (red fluorescent protein) or YFP (yellow fluorescent protein).

An example of edge codification:

EDGE CODIFICATION	*FITNESS GENES*
N12: B-F DNA language (X1) - AAT-TGG-*CGA*-TTA-AAC - B-F DNA language (X2)	TTAACCG*C*TAATTTG AATTGGC*G*ATTAAAC
N24: B-F DNA language (X2) - TTA-CCA-*TGC*-TGA-CCC - B-F DNA language (X4)	AATGGTA*G*GACTGGG TTACCAT*C*CTGACCC
N43: B-F DNA language (X4) - GGT-CAG-*CTG*-ACG-TCA - B-F DNA language (X3)	CCAGTCG*A*CTGCAGT GGTCAGC*T*GACGTCA
N35: B-F DNA language (X3) - AGT-CGA-*TTC*-GAA-GGC - B-F DNA language (X5)	TCAGCTA*A*GCTTCCG AGTCGAT*T*CGAAGGC
N51: B-F DNA language (X5) - CGT-AGC-*TGA*-TCGA-TCT - B-F DNA language (X1)	GCATCGA*C*TAGCTAGA CGTAGCT*G*ATCGATCT
N45: B-F DNA language (X4) - GGC-TGA-*TCG*-TAA-AGT - B-F DNA language (X5)	CCGACTA*G*CATTTCA GGCTGAT*C*GTAAAGT
N14: B-F DNA language (X1) - CCG-TAG-*CTG*-ATC-GTC - B-F DNA language (X4)	GGCATCG*A*CTAGCAG CCGTAGC*T*GATCGTC

The possible solutions of the problem come expressed into vectors or plasmids as it is illustrated in figure 14.

Example of edges codification after the formation of the solution-set of the problem:

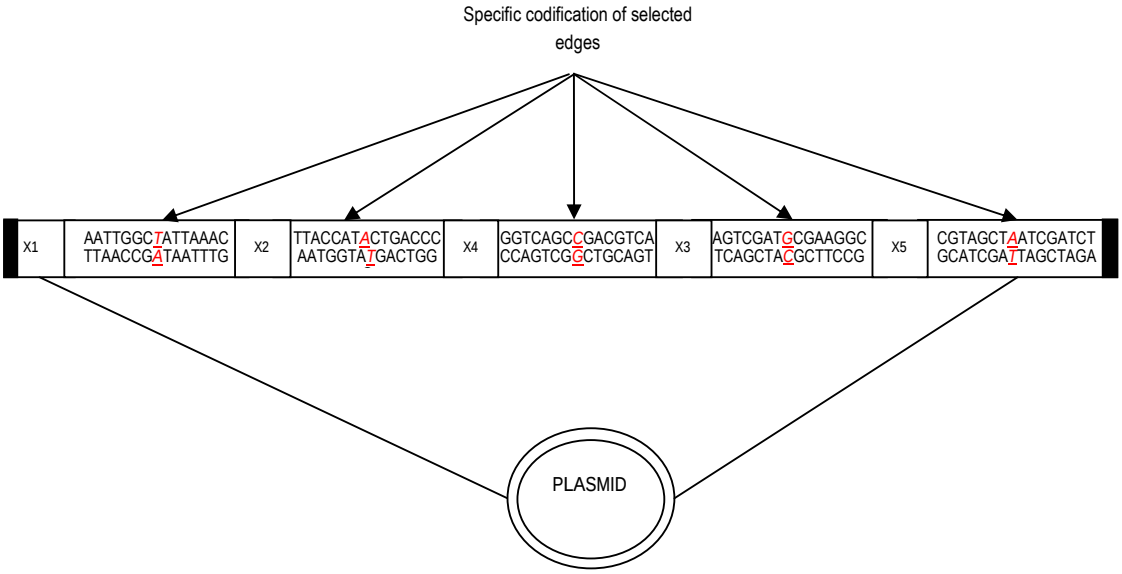


Fig. 14

Each plasmid contains sequentially the possible vertexes which form the Hamiltonian path we are looking for. As it has been mentioned before, within the set of plasmids there will be incomplete sequences and also nonvalid chains. For that reason the valid solution must have been searched. The proposal presented in this paper is based on this situation. In figure 14 is shown the plasmid of the initial set which contains the solution sequence to our problem in order to facilitate the later understanding of the theoretical development.

It is necessary to emphasize an important detail in the codification of the plasmid candidates. By observing figure 14 it can be seen that between each pair of vertexes the candidate edges are codified. These edges are codified carefully with a constant length of N-nucleotides and for the case of the existing edges in the initial graph a modification of the central nucleotide of the sequence which corresponds to the fitness gene take place. This modification consists on:

- Central sequence of the original edge N12: AATTGGCGATTAAAC
- Central sequence of the edge codified in the plasmid N12: AATTGGCTATTAAAC
  TTAACCGATAATTTG

It has been already explained the formation of the candidates which could be a solution of the problem. Next it is detailed the rest of the method which detects a correct solution. This technique is based on running n-cycles of mutagenesis in such a way that the solutions that contain the sequence of vertexes corresponding to the Hamiltonian Path will mutate until reach the moment in which the union between their vertexes turns into the fitness gene. This procedure is shown in figure 15 where a cycle of mutagenesis for the N12 edge is done. In the case that the edges do not exist in the initial graph this operation will not be carried out so the plasmid won't get the corresponding fitness field.

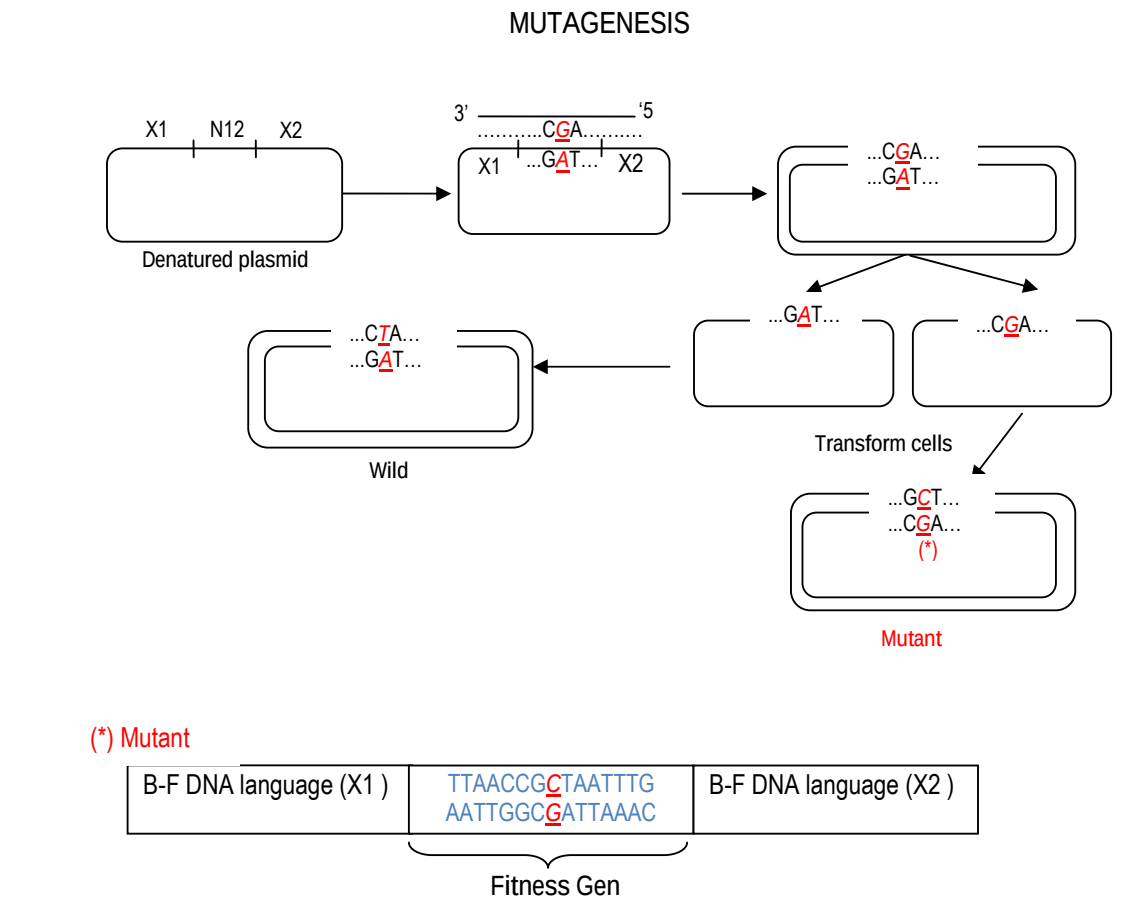


Fig. 16

When the last step has already been done, if we selected only the plasmids with the fluorescence conferred by the five genes of specific representations corresponding to the edges of the graph, we could affirm that the HPP proposed has a positive solution. Remember we are facing the problem of figure 13A.

In the same way, for that case in which a Hamiltonian Path does not exist in the given graph we would get a negative solution. That is due to the fact that we could notice after putting sequentially those bacteria which contain the vectors (once for each edge or protein) under fluorescent light of the absorption wavelength of the protein at issue that these ones do not emit.

In order to carry out these processes efficiently it is especially important to emphasize the use of bond-free languages for the codification of vertexes and edges providing stability and assuring the expected results during the formation and mutation steps of possible solutions of our problem.

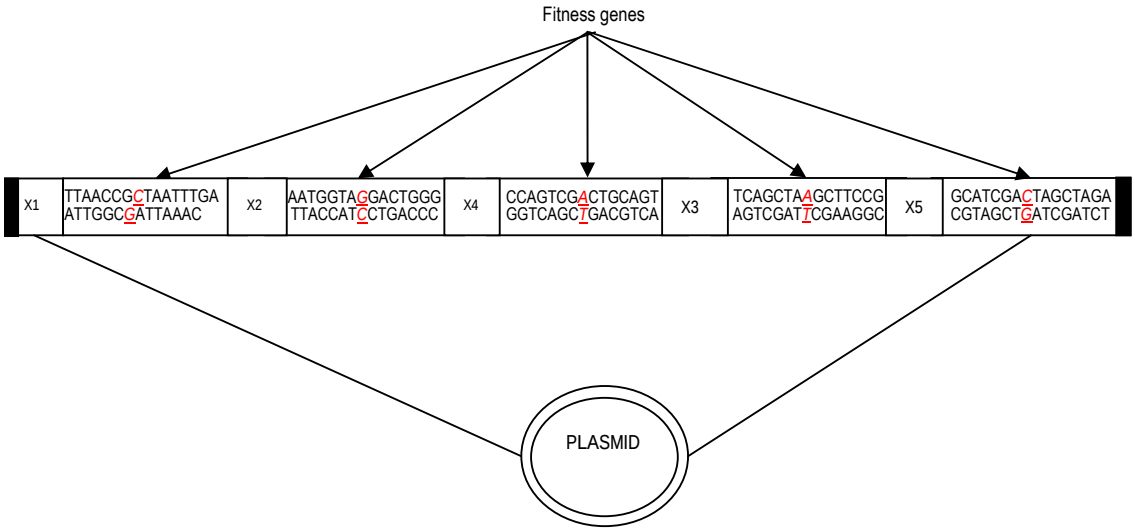


Fig. 17

Conclusion

In conclusion, we hope to have given a clarification of what gene-oriented computations are and how important is to code the gen in an appropriate way related to the problem to solve. There are as many gen codifications as problems are, but they have all the same structure which has been described in this paper. The general fields that all possible gens have in common are those that represent the stable sequences of the gen. One of the problems DNA computing has is the instability of the DNA strands. Many conditions can cause loss of DNA bases or strand breakage and because of that the problem will probably be doomed to failure. In the codification of the stable sequences of the gen we take care of the conditions that can cause DNA instability by means of bond-free languages.

However, the most important field in the gen is what we call the fitness field which is used to preserve the identity of each gene. As a result we propose a DNA codification forming genes that assures stability of DNA and give a concrete property to each gene. That property, which makes a gene different from the rest of the genes of the

problem, is based on the concentration of Cytosine and Guanine they have in the resolution of the TSP proposed. Furthermore, it is explained how to perform different DNA computations based on other kinds of fitness fields like antibiotic resistances. To sum up, the codification of the minimal computational unit (the gen) presented in this paper represents a strong structure that allow us to solve problems in a powerful and stable way.

Bibliography

[Adleman, 1994] Leonard M. Adleman. Molecular Computation of Solutions to Combinatorial Problems. Science (journal) 266 (11): 1021-1024. 1994.

[Adleman, 1998] Leonard M. Adleman. Computing with DNA. Scientific American 279: 54-61. 1998

[Lipton, 1995] Richard J.Lipton. Using DNA to solve NP-Complete Problems. Science, 268:542-545. April 1995

[Holland, 1975] J.H.Holland. Adaptation in Natural and Artificial Systems. MIT Press. 1975.

[J.Castellanos, 1998] J.Castellanos, S.Leiva, J.Rodrigo, A.Rodríguez Patón. Molecular computation for genetic algorithms. First International Conference, RSCTC'98.

[Macek, 1997] Milan Macek M.D. Denaturing gradient gel electrophoresis (DGDE) protocol. Hum Mutation 9: 136 1997.

[Dove, 1998] Alan Dove. From bits to bases; Computing with DNA. Nature Biotechnology. 16(9):830-832; September 1998.

[Mitchell, 1990] Melanie Mitchell. An Introduction to Genetic Algorithms. MIT Press, Boston. 1998.

[Lee, 2005] S.Lee, E. Kim. DNA Computing for efficient encoding of weights in the travelling salesman problem. ICNN&B'05. 2005.

[SY Shin, 2005] SY Shin, IH Lee, D Kim, BT Zhang. Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing. IEEE Transactions, 2005.

[Bo Cui, 2007] Bo cui, Stavros Konstantinidis. DNA Coding using the Subword Closure Operation. DNA 13. 13th Internacional Meeting on DNA Computing

[Kari, 2005] Lila Kari, Stavros Konstantinidis, and Petr Sosík. Bond-Free Languages: Formalizations, Maximality and Construction Methods. DNA10, LNCS 3384, pp. 169–181, 2005

[Konstantinidis, 2007] Bo Cui and Stavros Konstantinidis. DNA Coding using the Subword Closure Operation, DNA13, pp. 65–74, 2007.

[Kari, 2005] Kari, L., Konstantinidis, S., and Sosík, P. (2005) Preventing undesirable bonds between DNA codewords. Lect. Notes Comput. Sc., 3384, 182-191.

[Jonoska] Jonoska, N., Mahalingam, K.: Languages of DNA based code words. In: [4], 58–68

[Yurke, B.], Turberfield, A.J., Mills, A.P., Simmel, F.C., and Neumann, J.L. 2000 August 10. A DNA-fuelled molecular machine made of DNA. Nature 406: 605-608

[Head, 2000] T. Head, G. Rozenberg, R.S. Bladergroen, C.K.D. Breek, P.H.M. Lommerse, H.P. Spaink. Computing with DNA by operating on plasmids. BioSystems 57 (2000) 87-93.

[Wakabayashi, 2005] Kenichi Wakabayashi, Masayuki Yamamura .Natural Computing, A Design for Cellular Evolutionary Computation by using Bacteria Vol. 4, No. 3. (September 2005), pp. 275-292.

[Lui, 2005] Wenbin Liu, Xiangou Zhu, Guandong Xu, Quiang Zhang and Lin Gao. A DNA based evolutionary algorithm for the minimal set cover problem. Volume 3645/2005, Advances in Intelligent Computing, 2005

Authors' Information

Angel Goñi Moreno – *Natural Computing Group. Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain: e-mail: ago@alumnos.upm.es*

Paula Cordero – Natural Computing Group. Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain: e-mail: p.cordero@alumnos.upm.es

Juan Castellanos – Natural Computing Group. Artificial Intelligence Department. Facultad de Informática. Universidad Politécnica de Madrid, Boadilla del Monte, 28660. Madrid, Spain. e-mail: jcastellanos@fi.upm.es

FAST LINEAR ALGORITHM FOR ACTIVE RULES APPLICATION IN TRANSITION P SYSTEMS

Francisco Javier Gil, Jorge Tejedor, Luis Fernández

Abstract: Transition P systems are computational models inspired on basic features of biological membranes and the observation of biochemical processes. In these models, membrane contains objects multisets, which evolve according to given evolution rules. The basis on which the computation is based cellular membranes is the basic unit for the structure and functioning of all living beings: the biological cell. These models called P systems or membrane systems, are caused by the need to find new forms of calculation that exceed the limits set by the complexity theory in conventional computing, drawing mainly distributed operation, non-deterministic and massively parallel with which the cells behave.

In the field of Transition P systems implementation, it has been detected the necessity to determine whichever time are going to take active evolution rules application in membranes. In addition, to have time estimations of rules application makes possible to take important decisions related to the hardware / software architectures design.

In this paper we propose a new evolution rules application algorithm oriented towards the implementation of Transition P systems. The developed algorithm is sequential and, it has a linear order complexity in the number of evolution rules. Moreover, it obtains the smaller execution times, compared with the preceding algorithms. Therefore the algorithm is very appropriate for the implementation of Transition P systems in sequential devices.

Keywords: Natural Computing, Membrane computing, Transition P System, Rules Application Algorithms

ACM Classification Keywords: D.1.m Miscellaneous – Natural Computing

Introduction

Membrane computing is a branch of natural computing which tries to abstract computing models from the structure and the functioning of living cells. The main purpose of these investigations consists of developing new computational tools for solving complex, usually conventionally-hard problems. Being more concrete, Transition P systems are introduced by Gheorghe Păun derived from basic features of biological membranes and the observation of biochemical processes [Păun, 1998]. This computing model has become, during last years, an influential framework for developing new ideas and investigations in theoretical computation. Many theoretical papers have been published in different workshops, congresses and scientific journals.

P systems are hierarchical (see Figure 1), as the region defined by a membrane may contain other membranes. Membranes define compartments or regions. The basic components of the Transition P systems are the membranes that contain chemical elements (multisets of objects, usually represented by symbol strings) which are subject to chemical reactions (evolution rules) to produce other elements (another multiset). Multisets generated by evolution rules can be moved towards adjacent membranes (parent and children). This multiset transfer feeds back the system so that new multisets of symbols are consumed by further chemical reactions in the membranes.

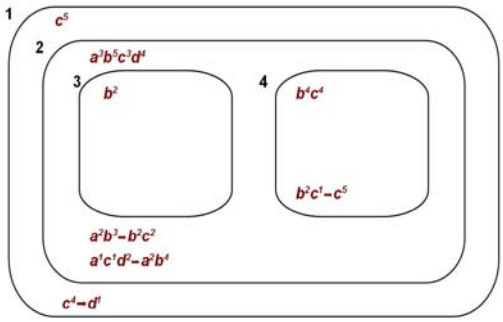


Figure 1.- Graphic representation of an abstract cell structure defined by a membrane composed of four membranes including object multisets and the evolution rules

The P system changes from a configuration to another one making a computation. P systems performs a computation starting with an initial configuration and changing to another configuration by applying evolution rules to objects inside regions until reaching a halting configuration, where there is no rule applicable to the objects in the membrane. Each transition or evolution step goes through two sequential steps: rules application and communication between membranes. First, the evolution rules are applied simultaneously to the objects multiset in each membrane. This process is performed by all membranes at the same time. Then, also simultaneously, all the membranes communicate with their neighbors, transferring symbol multisets.

Most membrane systems are computationally universal: "P systems with simple ingredients (number of membranes, forms and sizes of rules, controls of using the rules) are Turing complete" [Păun, 2005]. This framework is extremely general, flexible, and versatile. Several classes of P systems with an enhanced parallelism are able to solve computationally hard problems (typically, NP complete problems) in a feasible time (polynomial or even linear) by making use of an exponential space.

In this paper we propose a new algorithm for evolution rules application oriented towards the implementation of Transition P systems. The developed algorithm is sequential and, it has a linear order complexity in the number of evolution rules. Moreover, it obtains the smaller execution times, compared with the preceding algorithms. Therefore, due to these characteristics, the algorithm is very appropriate for the implementation of Transition P systems in sequential devices. After this introduction, other related works appear, where the problem that is tried to solve is covered. Next are exposed the formal definitions related to the rules application in Transition P systems. Later the fast linear algorithm for rules application appears developed, finally including the comparison tests and conclusions.

Related Work

In Transition P systems, each evolution step is obtained through two consecutive phases within each membrane: in first stage the evolution rules are applied, and at the second, the communication between membranes is made. This work is centered in the first phase, the application of active rules. It exists several sequential algorithms for rules application in P systems at this moment [Ciobanu, 2002], [Fernández, 2006a] and [Tejedor, 2007], but the obtained results can be improved. In the last mentioned work is introduced an algorithm based on the elimination of active rules: this algorithm is very interesting because is the first algorithm whose time is only limited by the number of rules, not by the objects multiset cardinality.

Additionally, in [Tejedor, 2006] is proposed a software architecture for attacking the bottleneck communication in P systems denominated "partially parallel evolution with partially parallel communications model" where several

membranes are located in each processor, proxies are used to communicate with membranes located in different processors and a policy of access control to the network communications is mandatory. This obtains a certain parallelism yet in the system and an acceptable operation in the communications. In addition, it establishes a set of equations that they allow to determine in the architecture the optimum number of processors needed, the required time to execute an evolution step, the number of membranes to be located in each processor and the conditions to determine when it is best to use the distributed solution or the sequential one. Additionally it concludes that if the maximum application time used by the slowest membrane in applying its rules improves N times, the number of membranes that would be executed in a processor would be multiplied by the square root of N, the number of required processors would be divided by the same factor, and the time required to perform an evolution step would improve approximately with the same factor.

Therefore, to design software architectures it is precise to know the necessary time to execute an evolution step. For that reason, algorithms for evolution rules application that they can be executed in a delimited time are required, independently of the object multiset cardinality inside the membranes. Nevertheless, this information cannot be obtained with most of the algorithms developed until now since its execution time depends on the cardinality of the objects multiset on which the evolution rules are applied.

They have been proposed also parallel solutions - [Fernández, 2006b] and [Gil, 2007] -, but they do not obtain the required performance. The first algorithm is not completely useful, since its run time is not time delimited, and both solutions present efficiency problems due to the competitiveness between the rules, the high number of collisions with the requests and delays due to the synchronization required between processes.

Formal Definitions Related to Rules Application in P Systems

Firstly, this section formally defines the required concepts of objects multisets, evolution rules, evolution rules multiset, and applicability benchmarks (maximal and minimal) of a rule over an objects multiset. Secondly, on the basis of these definitions, requirements are specified for the new algorithm for rule evolution application.

Multisets of Objects

Definition 1: *Multiset of object. Let a finite and not empty set of objects be* O *and the set of natural numbers* N*, is defined as a multiset of object* m *as a mapping*:

$$m : \mathrm{O} \to \mathrm{N}$$
$$o \to n$$

Possible notations for a multiset of objects are:

$$m = \{(o_1, n_1), (o_2, n_2), \ldots, (o_m, n_m)\}$$
$$m = o_1^{n_1} \cdot o_2^{n_2} \cdot \ldots \cdot o_m^{n_m}$$

Definition 2: *Set of multisets of objects over a set of objects. Let a finite set of objects be* O*. The set of all the multisets that can be formed over set* O *is defined:*

$$\mathrm{M}(\mathrm{O}) = \{m : \mathrm{O} \to \mathrm{N} \mid m \textit{ is a Multiset over } \mathrm{O}\}$$

Definition 3: *Multiplicity of object in a multiset of objects. Let an object be* $\mathrm{o} \in \mathrm{O}$ *and a multiset of objects* $\mathrm{m} \in \mathrm{M}(\mathrm{O})$*. The multiplicity of an object is defined over a multiset of objects such as:*

$$|\ |_o : \mathrm{O} \times \mathrm{M}(\mathrm{O}) \to \mathrm{N}$$
$$(o, m) \to |m|_o = n \mid (o, n) \in m$$

Definition 4*: Weight or Cardinal of a multiset of objects. Let a multiset of objects be* $m \in M(O)$. *The weight or cardinal of a multiset of objects is defined as:*

$$|\ |: M(O) \to N$$
$$m \to |m| = \sum_{\forall o \in O} |m|_o$$

Definition 5: *Multiset support. Let a multiset of objects be* $m \in M(O)$ *and* $P(O)$ *the power set of* O. *The support for this multiset is defined as:*

$$Supp: M(O) \to P(O)$$
$$m \to Supp(m) = \{ o \in O \ / \ |m|_o > 0 \}$$

Definition 6: *Empty multiset. This is the multiset represented by* $\varnothing_{M(O)}$ *and which satisfies:*

$$\varnothing_{M(O)} \Leftrightarrow |m| = 0 \Leftrightarrow Supp(m) = \varnothing$$

Definition 7: *Inclusion of multisets of objects. Let two multisets of objects be* $m_1,\ m_2 \in M(O)$. *The inclusion of multisets of objects is defined as:*

$$m_1 \subset m_2 \Leftrightarrow |m_1|_o \le |m_2|_o \quad \forall o \in O$$

Definition 8*: Sum of multisets of objects. Let two multisets of objects be* $m_1,\ m_2 \in M(O)$. *The sum of multisets of objects is defined as:*

$$+: M(O) \times M(O) \to M(O)$$
$$(m_1, m_2) \to \{(o, |m_1|_o + |m_2|_o) \ \forall o \in O\}$$

Definition 9*: Subtraction of multisets of objects. Let two multisets of objects be* $m_1,\ m_2 \in M(O)$, *and* $m_2 \subset m_1$. *The subtraction of the multisets of objects is defined as:*

$$-: M(O) \times M(O) \to M(O)$$
$$(m_1, m_2) \to \{(o, |m_1|_o - |m_2|_o) \ \forall o \in O\}$$

Definition 10: *Intersection of multisets of objects. Let two multisets of objects be* $m_1,\ m_2 \in M(O)$. *The intersection of multisets of objects is defined as:*

$$\bigcap: M(O) \times M(O) \to M(O)$$
$$(m_1, m_2) \to m_1 \bigcap m_2 = \{(o, \min(|m_1|_o, |m_2|_o) \ \forall o \in O\}$$

Definition 11*: Scalar product of multiset of objects by a natural number. Let a multiset be* $m_2 \in M(O)$ *and a natural number* $n \in N$. *The scalar product is defined as:*

$$\cdot : M(O) \times N \to M(O)$$
$$(m, n) \to m \cdot n = \{(o, |m|_o \cdot n) \quad \forall o \in O\}$$

Evolution Rules

Definition 12*: Evolution rule over a set of objects with target in* T *and with no dissolution capacity. Let a set of objects be* O, $a \in M(O)$ *a multiset over* O, $T = \{here,\ out\} \cup \{inj / 1 \le j \le p\}$ *a set of targets and* $c \in M(O \times T)$ *a multiset over* $O \times T$. *An evolution rule is defined like a tuple*:

$$r = (a, c)$$

Definition 13: *Set of evolution rules over a set of objects and targets in* T. *This set is defined as:*

$$R(O, T) = \{r \mid r \text{ is a rule over } O \text{ and } T\}$$

Definition 14: *Antecedent of Evolution Rule. Let an evolution rule be* $r \in R\ (O, T)$. *The antecedent of an evolution rule is defined over a set of objects as:*

$$input : \mathrm{R(O, T)} \to \mathrm{M(O)}$$
$$(a, c) \to input(r) = a \mid r = (a, c) \in R(\mathrm{O, T})$$

Definition 15: *Evolution rule applicable over a multiset of objects. Let an evolution rule be* $r \in R\ (O, T)$ *and a multiset of objects* $m \in M\ (O)$, *it is said that an evolution rule is applicable over a objects multiset if and only if:*

$$\Delta_r(m) \Leftrightarrow input(r) \subset m$$

Definition 16: *Set of evolution rules applicable to a multiset of objects. Let a set of evolution rules be* $R \in P(R\ (O, T))$ *and a multiset of objects* $m \in M\ (O)$. *The set of evolution rules applicable to a multiset of objects is defined as:*

$$\Delta^* : P(\mathrm{R(O, T)}) \times \mathrm{M(O)} \to P(\mathrm{R(O, T)})$$
$$(\mathrm{R}, m) \to \Delta_R^*(m) = \{r \in \mathrm{R} \mid \Delta_R(m) = true\}$$

Property 1: *Maximal applicability benchmark of evolution rule over a multiset of objects*. Let an evolution rule be $r \in R\ (O, T)$ and a multiset of objects $m \in M\ (O)$. The maximal applicability benchmark of a rule in a multiset is defined as:

$$\Delta\lceil\ \rceil : \mathrm{R(O, T)} \times \mathrm{M(O)} \to \mathrm{N}$$
$$(r, m) \to \Delta_r\lceil m \rceil = \min\left\{ \frac{|m|_o}{|input(r)|_o} \forall o \in Supp(m) \wedge \left|input(r)\right|_o \neq 0 \right\}$$

Property 2: *Minimal applicability benchmark of evolution rule over a multiset of objects and a set of evolution rules*. Let an evolution rule be $r \in R\ (O, T)$, a multiset of objects $m \in M\ (O)$ and a set of evolution rules $R \in P(R\ (O, T))$. The minimal applicability benchmark is defined as the function:

$$\Delta\lfloor\ \rfloor : \mathrm{R(O, T)} \times \mathrm{M(O)} \times P(\mathrm{R(O, T)}) \to \mathrm{N}$$
$$(r, m, R) \to \Delta_r\lfloor m \rfloor = \Delta_r\left\lceil m - \left(m \cap \sum_{\forall ri \in R - \{r\}} input(r_i) \cdot \Delta_{ri}\lceil m \rceil \right) \right\rceil$$

Property 3: *An evolution rule* $r \in R\ (O, T)$ *is applicable to a multiset of objects* $m \in M\ (O)$ if and only if the maximal applicability benchmark is greater or equal to 1.

$$\Delta_r(m) \Leftrightarrow \Delta_r\lceil m \rceil \geq 1$$

Property 4: The maximal applicability benchmark of a rule $r \in R\ (O, T)$ *over an object multiset* $m \in M\ (O)$ *is greater than or equal to the maximal applicability benchmark of the rule in a subset of the object multiset.*

$$\Delta_r\lceil m_1 \rceil \geq \Delta_r\lceil m_2 \rceil \ \ \forall m_1, m_2 \in M(\mathrm{O}) \mid m2 \subset m1$$

Property 5: If the maximal applicability benchmark of a rule $r \in R\ (O, T)$ over a multiset of objects $m \in M\ (O)$ is 0, then the maximal applicability benchmark of the rule r over the sum of input (r) and m is equal to the maximal applicability benchmark of the input (r) and equal to 1.

$$\Delta_r\lceil m \rceil = 0 \Rightarrow \Delta_r\lceil input(r) + m \rceil = \Delta_r\lceil input(r) \rceil = 1$$

Multisets of Evolution Rules

Definition 17: *Multiset of evolution rules*. Let a finite and not empty set of evolution rules be $R(O, T)$ and the set of natural numbers N, a multiset of evolution rules is defined as the mapping:

$$M_{R(O,T)} : \mathrm{R(O,T)} \to \mathrm{N}$$
$$r \to n$$

All definitions related to multisets of objects can be extended to multisets of rules.

Definition 18: *Linearization of evolution multiset of rules*. Let a multiset of evolution rules be $m_R = r_1^{k1} \cdot r_2^{k2} \cdot \cdots \cdot r_q^{kq} \in M_{R(O,T)}$ linearization of m_R is defined as:

$$\sum_{i=1}^{q} r_i \cdot k_i \in \mathrm{R(O,T)}$$

Requirements of Application of Evolution Rules over Multiset of Objects

Application of evolution rules in each membrane of P Systems involves subtracting objects from the objects multiset by using rules antecedents. Rules used are chosen in a non-deterministic manner. The process ends when no rule is applicable. In short, rules application to a multiset of object in a membrane is a process of information transformation with input, output and conditions for making the transformation.

Given an object set $O = \{o_1, o_2, \ldots, o_m\}$ where $m > 0$, the input to the transformation process is composed of a multiset $\omega \in M(O)$ and $R \in R(O, T)$, where:

$$\omega = o_1^{n_1} \cdot o_2^{n_2} \cdot \ldots \cdot o_m^{n_m}$$
$$\mathrm{R} = \{ r_1, r_2, ..., r_q \} being\ q > 0$$

In fact, the transformation only needs rules antecedents because this is the part that acts on ω. Let these antecedents be:

$$input(r_1) = o_1^{n_1^i} \cdot o_2^{n_2^i} \cdot \ldots \cdot o_m^{n_m^i} \quad \forall i = \{ 1, 2, ..., q \}$$

The output of the transformation process will be a objects multiset of $\omega' \in M(O)$ together with the multiset of evolution rules applied $\omega_R \in M_{R(O,T)}$.

$$\omega' = o_1^{n'_1} \cdot o_2^{n'_2} \cdot \ldots \cdot o_m^{n'_m}$$
$$\omega_R = r_1^{k_1} \cdot r_2^{k_2} \cdot \ldots \cdot r_q^{k_q}$$

Conditions for making the transformation are defined according to the following requirements:

Requirement 1: The transformation process is described through the following system of equations:

$$n_1 = n_1^1 \cdot k_1 + n_1^2 \cdot k_2 + \ldots + n_1^q \cdot k_q + n_1^{'}$$
$$n_2 = n_2^1 \cdot k_1 + n_2^2 \cdot k_2 + \ldots + n_2^q \cdot k_q + n_2^{'}$$
$$\ldots$$
$$n_m = n_m^1 \cdot k_1 + n_m^2 \cdot k_2 + \ldots + n_m^q \cdot k_q + n_m^{'}$$

That is:

$$\sum_{j=1}^{q} n_i^j \cdot k_j + n_i^{'} = n_i \quad \forall i = \{ 1, 2, ..., m \}$$

or

$$\sum_{i=1}^{q} input(r_i) \cdot k_i + \omega' = \omega$$

The number of equations in the system is the cardinal of the set O. The number of unknowns in the system is the sum of the cardinals of the set O and the number of rules of R. Thus, the solutions are in this form:

$$(n_1', n_2', ..., n_m', k_1, k_2, ..., k_q) \in N^{m+q}$$

Meeting the following restrictions:

$$0 \le n_i' \le n_i \quad \forall i = \{1, 2, ..., m\}$$

Moreover, taking into account the maximal and minimal applicability benchmarks of each rule, the solution must satisfy the following system of inequalities:

$$\Delta_{r_j} \lfloor \omega \rfloor \le k_j \le \Delta_{r_j} \lceil \omega \rceil \quad \forall j = \{ 1, 2, ..., q \}$$

Requirement 2: No rule of the set R can be applied over the multiset of objects ω', that is:

$$\Delta_r(\omega') = false \quad \forall r \in R$$

Having established the above requirements, the system of equations may be incompatible (no rule can be applied) determinate compatible (there is a single multiset of rules as the solution to the problem) or indeterminate compatible (there are many solutions). In the last case, the rule application algorithm must provide a solution that is randomly selected from all possible solutions in order to guarantee non-determinism inherent to P systems.

Fast Linear Algorithm for Active Rules Application in Transition P Systems

This section describes the fast linear algorithm for active rules application to a multiset of objects whose execution time depends on the number of rules. The initial input is a set of active evolution rules for the corresponding membrane -the rules are applicable and useful- and the initial membrane multiset of objects. The final results are the complete multiset of applied evolution rules and the obtained multiset of objects after rules application.

The algorithm is based on the one by one elimination of rules: when a rule has been applied to its maximal applicability benchmark, this rule lets be active, and therefore it is eliminated. The algorithm finishes when all rules have been eliminated. The algorithm is made up of two phases:

1. At the first phase all rules belonging to the set of active rules -except one- are applied a random number of times between 0 and its maximal applicability benchmark. In this way, each active rule has a possibility of being applied.
2. In the second phase, all the rules -beginning by the one excluded of the previous phase- are applied to its maximal applicability benchmark. Consequently, there it is not left any rule applicable, and the algorithm finishes generating like result the multiset of rules applied and the final multiset of objects.

In order to facilitate the explanation of the algorithm, the set of initially active rules is represented like an ordered sequence R and an auxiliary structure called A. The position of any rule r_i in the R sequence is i. $A[i]$ indicates if the rule r_i continues active. The rule excluded in first stage is the one that is in the last position of the sequence (it can be any rule of the set of active rules). The pseudo code of the algorithm is as follows:

(1) $\omega' \leftarrow \omega$
(2) $\omega_R \leftarrow \emptyset_{M_R(U)}$
(3) **FOR i = 1 TO** $|R|$ **- 1 DO** // Phase 1
(4) **BEGIN**
(5) $Max \leftarrow \Delta_{R[i]}(\omega')$
(6) **IF** ($Max \neq 0$) **THEN**
(7) **BEGIN**
(8) $K \leftarrow random(0, Max)$
(9) $\omega_R \leftarrow \omega_R + \{R[i]^K\}$
(10) $\omega' \leftarrow \omega' - input(R[i]) \cdot K$
(11) $A[i] = (K < Max)$
(12) **END**
(13) **ELSE** $A[i] = false$
(14) **END**
(15)
(16) $A[|R|] = true$
(17) **FOR i =** $|R|$ **DOWNTO 1 DO** // Phase 2
(18) **IF** ($A[i]$) **THEN**
(19) **BEGIN**
(20) $Max \leftarrow \Delta_{R[i]}(\omega')$
(21) $\omega_R \leftarrow \omega_R + \{R[i]^{Max}\}$
(22) $\omega' \leftarrow \omega' - input(R[i]) \cdot Max$
(23) **END**

As it has been previously indicated, the algorithm is made up of two phases. In first stage is offered the possibility to all the rules -except one- to be applied between 0 and their maximum applicability benchmark. In addition is determined if a rule lets be active. Rules can let be active in this stage due to two possible reasons: a) the rule has been applied to its maximum applicability, or b) other preceding rules have consumed the necessary objects so that the rule can be applied.

The second phase begins supposing like active the last rule (observe that this is not necessarily certain). Next, beginning by the one excluded of the first phase, all the supposedly active rules are applied to its maximum applicability. After this step all the rules let be inactive, and the application algorithm finished their execution. As it can be seen, the algorithm executes a finite and well-known number of operations, which only depends on the initial number of active rules.

In the next sections we are going to demonstrate the correctness of the exposed algorithm, as well as the efficiency analysis.

Algorithm Correctness

The presented algorithm is correct because:

Lemma 1: *The algorithm is finite.*

Proof: The first two lines are basic operations. The first loop -from line (3) to (14)- is exactly executed $|R|-1$ times, and its body only contains simple operations. The second loop -from line (16) to (23)- is exactly executed $|R|$ times, and also its body only contains simple operations.

Lemma 2: *No evolution rule is applicable to ω'.*

Proof: The sequence R initially contains all rules applicable to *ω'*. Owing to property 3 we know that a rule with a maximal applicability benchmark equal to zero is not applicable. After the execution of the second phase of the algorithm, the maximal applicability of all the rules is zero. Therefore, at the end of the algorithm execution, it is not left any rule applicable to *ω'*.

Lemma 3: *Any result generated is a possible solution.*

Proof: The multiset of rules applied ω_R is obtained by the multiple applications of the active rules in both phases. In addition, since in the second phase each active rule is applied to its maximal applicability benchmark, after the execution of the algorithm no rule is applicable over *ω'* (requirement 2), and the result generated is a possible solution.

Lemma 4: *Any solution possible is generated by the algorithm*

Proof: Phase 1 of the algorithm -from line (3) to (14)- guarantees that any possible solution can be generated. It is enough whereupon the appropriate number is generated in line 8, when the number of applications of a rule is determined. In the second phase it would be only needed to apply the last rule the appropriate number of times.

Lemma 5: *The algorithm is not determinist*

Proof: This occurs when a rule is not the last one in the set, it is applied a randomly determined number of times (sentence 8) between zero and its maximal applicability value.

Efficiency Analysis

Examining the algorithm it is possible to observe that in the two phases, the heaviest operations are those for calculating the maximal applicability benchmark (sentences 5 and 20), the scalar product of the $input$ of a rule by a whole number and the difference of two multisets (sentences 10 and 22). These operations are made in both phases in the worse case. All these operations are linearly dependant on the cardinal of the *multiset support* ω.

$$\#operations\_per\_iteration \approx 3 \cdot Supp(\omega)$$

Moreover, the worst case of the fast linear algorithm occurs when sentences 6 and 11 are evaluated always affirmatively, or what is the same, when no rule becomes inactive after the execution of first stage. In this case, there is no improvement in the behavior of the algorithm and the number of iterations executed is:

$$\#iterations = (|R|-1)+|R| = 2 \cdot |R| - 1$$

Therefore, the number of operations executed at worst case by the algorithm is:

$$\#operations = (2 \cdot |R| - 1) \cdot 3 \cdot Supp(\omega)$$

So the execution time of the algorithm at worst is linear dependant of the number of rules.

Comparison Tests

The experimental tests have compared the execution time of the *Fast Linear algorithm* (FLA) with the one that was fastest until now, that is *Active Rules Elimination* (ARE) algorithm [Tejedor, 2007]. The experimental trial game used to test both algorithms has taken into account 3 parameters:

- *Number of objects of the multiset*. In comparative the value of this parameter is 16
- *Number of rules (q)*. The value of q has taken all the values from the set {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024}
- *Relationship between the cardinal of the multiset and the cardinal of the sum of inputs of the active rules set (r)*. The value of r has taken all the values from the set {1, 10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7}

Figure 2 shows a graphic with the evolution of the execution times difference obtained in the tests between the REA and the FLA. Each curve of the graphic represents the difference of execution time of the ARE algorithm with regards to the execution time of FLA algorithm for each value of the relationship of cardinals (r). In this graphic can be seen that FLA algorithm is always better than ARE algorithm independently of the number of rules and the relationship between cardinals.

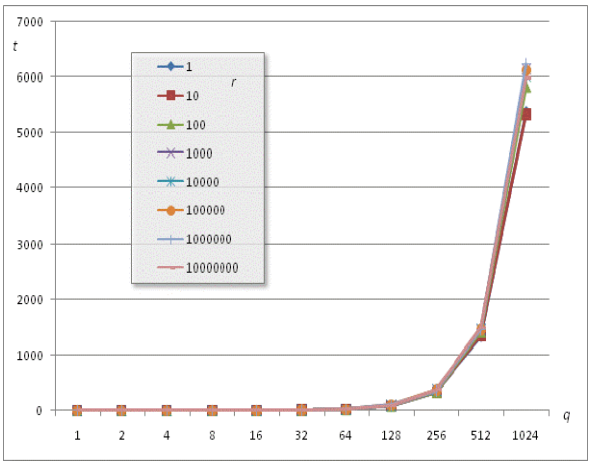


Figure 2.- Evolution of the execution times difference

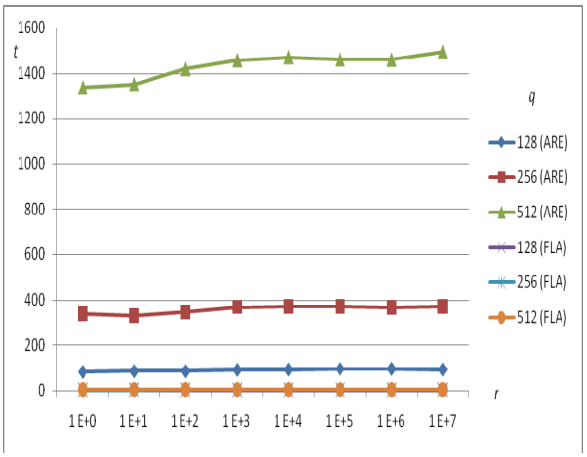


Figure 3.- Execution time of ARE and FLA

In Figure 3 it can be observed that the time execution grows modestly -with both algorithms- with the value of r. The parameter that more influences in the results is the number of rules (q). The results obtained are logical considering that the complexity of the ARE algorithm is order of square of q, and the complexity of the FLA is q linear.

Conclusions

This paper introduces a new algorithm for active rules application to a multiset of objects based on rules elimination in transition P systems. This algorithm attains a certain degree of parallelism, as a rule can be applied a great number of times in a single step. The number of operations executed by the algorithm is time delimited, because it only depends on the number of rules of the membrane. The number of rules of the membrane is well known static information studying the P system, thus allowing determining beforehand the algorithm execution

time. This information is essential to calculate the number of membranes that have to be located in each processor in distributed implementation architectures of P systems to achieve optimal times with minimal resources.

We think that the presented algorithm can represent an important contribution in particular for the problem of the application of rules in membranes, because it presents high productivity and it allows estimate the necessary time to execute an evolution step. Additionally, this last one allows making important decisions related to the implementation of P systems, like the related ones to the software architecture.

Bibliography

[Ciobanu, 2002] G. Ciobanu, D. Paraschiv, "Membrane Software. A P System Simulator". Pre-Proceedings of Workshop on Membrane Computing, Curtea de Arges, Romania, August 2001, Technical Report 17/01 of Research Group on Mathematical Linguistics, Rovira i Virgili University, Tarragona, Spain, 2001, 45-50 and Fundamenta Informaticae, vol 49, 1-3, 61-66, 2002.

[Ciobanu, 2006] G. Ciobanu, M. Pérez-Jiménez, Gh. Păun, "Applications of Membrane Computing". Natural Computing Series, Springer Verlag, October 2006.

[Fernández, 2006a] Fernández, L. Arroyo, F. Castellanos, J. et al (2006) "New Algorithms for Application of Evolution Rules based on Applicability Benchmarks". BIOCOMP 06, Las Vegas (USA)

[Fernández, 2006b] L. Fernández, F. Arroyo, J. Tejedor, J. Castellanos. "Massively Parallel Algorithm for Evolution Rules Application in Transition P System". Seventh Workshop on Membrane Computing, WMC7, Leiden (The Netherlands). July, 2006

[Gil, 2007] Gil, F. J. Fernández, L. Arroyo, F. et al "Delimited Massively Parallel Algorithm based on Rules Elimination for Application of Active Rules in Transition P Systems" i.TECH-2007. Varna (Bulgaria).

[Păun, 1998] G. Păun. "Computing with Membranes". In: Journal of Computer and System Sciences, 61(2000), and Turku Center of Computer Science-TUCS Report nº 208, 1998.

[Păun, 2005] G. Păun. "Membrane computing. Basic ideas, results, applications". In: Pre-Proceedings of First International Workshop on Theory and Application of P Systems, Timisoara (Romania), pp. 1-8, September, 2005.

[Tejedor, 2006] J. Tejedor, L. Fernández, F. Arroyo, G. Bravo. "An Architecture for Attacking the Bottleneck Communications in P systems". In: Artificial Life and Robotics (AROB 07). Beppu (Japan), January 2007.

[Tejedor, 2007] J. Tejedor, L. Fernández, F. Arroyo, A. Gutiérrez. "Algorithm of Active Rules Elimination for Evolution Rules Application". In 8th WSEAS Int. Conf. on Automation and Information, Vancouver (Canada), June 2007.

Authors' Information

F. Javier Gil Rubio – Dpto. de Organización y Estructura de la Información, E.U. de Informática. Natural Computing Group, Universidad Politécnica de Madrid, Spain;

e-mail: jgil@eui.upm.es

Jorge A. Tejedor Cerbel – Dpto. de Organización y Estructura de la Información, E.U. de Informática. Natural Computing Group, Universidad Politécnica de Madrid, Spain;

e-mail: jtejedor@eui.upm.es

Luis Fernández Muñoz – Dpto. de Lenguajes, Proyectos y Sistemas Informáticos, E.U. de Informática. Natural Computing Group, Universidad Politécnica de Madrid, Spain;

e-mail: setillo@eui.upm.es

EXTENDED NETWORKS OF EVOLUTIONARY PROCESSORS

Luis Fernando de Mingo, Nuria Gómez Blas, Francisco Gisbert, Miguel A. Peña

Abstract: This paper presents an extended behavior of networks of evolutionary processors. Usually, such nets are able to solve NP-complete problems working with symbolic information. Information can evolve applying rules and can be communicated though the net provided some constraints are verified. These nets are based on biological behavior of membrane systems, but transformed into a suitable computational model. Only symbolic information is communicated. This paper proposes to communicate evolution rules as well as symbolic information. This idea arises from the DNA structure in living cells, such DNA codes information and operations and it can be sent to other cells. Extended nets could be considered as a superset of networks of evolutionary processors since permitting and forbidden constraints can be written in order to deny rules communication.

Keywords: Networks of Evolutionary Processors, Membrane Systems, and Natural Computation.

ACM Classification Keywords: F.1.2 Modes of Computation, I.6.1 Simulation Theory, H.1.1 Systems and Information Theory

Introduction

Natural sciences, and especially biology, represent a rich source of modeling paradigms. Well-defined areas of artificial intelligence (genetic algorithms, neural networks), mathematics, and theoretical computer science (L systems, DNA computing) are massively influenced by the behavior of various biological entities and phenomena. In the last decades or so, new emerging fields of so-called "natural computing" [1,2,3] identify new (unconventional) computational paradigms in different forms. There are attempts to define and investigate new mathematical or theoretical models inspired by nature [8], as well as investigations into defining programming paradigms that implement computational approaches suggested by biochemical phenomena. Especially since Adleman's experiment [4] these investigations received a new perspective. One hopes that global system-level behavior may be translated into interactions of a myriad of components with simple behavior and limited computing and communication capabilities that are able to express and solve, via various optimizations, complex problems otherwise hard to approach.

The origin of networks of evolutionary processors is a basic architecture for parallel and distributed symbolic processing, related to the Connection Machine [7] as well as the Logic Flow paradigm [5], which consists of several processors, each of them being placed in a node of a virtual complete graph, which are able to handle data associated with the respective node. All the nodes send simultaneously their data and the receiving nodes handle also simultaneously all the arriving messages, according to some strategies, see, e.g., [6,7].

Networks of evolutionary processors (NEP) [9,11] are language-generating device, if we look at the strings collected in the output node. We can also look at them as doing some computation. If we consider these networks with nodes having filters defined by random context conditions, which seems to be closer to the recent possibilities of biological implementation, then using these simple mechanisms we can solve NP-complete

problems in linear time. Such solutions are presented for the *Bounded Post Correspondence Problem* in [10] for the *3-Colorability Problem* in [9] and for the {\it Common Algorithmic Problem} in [12]. As a further step, in [12] the so-called hybrid networks of evolutionary processors are considered. Here deletion node or insertion node has its own working mode (performs the operation at any position, in the left-hand end or in the right-hand end of the word) and different nodes are allowed to use different ways of filtering. Thus, the same network may have nodes where the deletion operation can be performed at arbitrary position and nodes where the deletion can be done only at right-end of the word.

In this paper, we present some results regarding a network of evolutionary processor based on an extended behavior from the biological point of view. This is a preliminary work in which rules pass through the net, they are not associated to a fix processor.

Networks of Evolutionary Processors

Connectionist models consist of several processors which are located in the nodes of a virtual graph and are able to perform operations in that node, according to some predefined rules. Information is passed through connections in order to obtain a collaborative solution to a given problem. All processors work in a parallel way and they only perform simple operations.

A network of evolutionary processors is a tuple

$$\Gamma = (V, U, G, \mathcal{N}, \alpha, x_I, x_O)$$

- V, U are the net alphabet and input alphabet respectively.
- G is an undirected graph in which each node is a processor. Processors have a set of objects/strings and a set of evolution rules.
- N is a mapping that associates each processor with a set of filters.
- α is a mapping that defines the behavior of filters (weak or strong conditions).
- x_I, x_O are the input and output processors.

Objects in processors can evolve and communicated to other connected processors. That is, rules can be applied (evolution) or objects can pass filter conditions (communication). These two steps could be sequential (evolution and then communication) or parallel (evolution and communication at same time).

Evolution

A given string *x* can evolve provided there is some rule to apply it. Taking into account that there are an arbitrary large number of copies of string *x* in processor, several rules can be applied in parallel to different copies in one evolutionary step.

Therefore, the set of objects in a processor *i* after an evolution step, denoted by L'_i, are those before the evolution (L_i) adding objects obtained after rules in *i* are applied. That is,

$$L'_i = L_i \cup r_k(x),\ x \in A_i, r_k \in R_i$$

Communication

A given string/object *x* can pass filters in processor *i*, iff the following constraint is satisfied:

$$\varphi_i^{(\beta)}(x;\mathcal{N}(i)), \beta = \{s, w\}$$

Where, $\mathcal{N}(i)$ is the filter set associated to a given processor *i*. This set can contain just input and output filters, or forbidden context filters as well. That is, $\mathcal{N}(i) = \{PI, PO\}$, or $\mathcal{N}(i) = \{FI, PI, FO, PO\}$. Constraints are defined as follows:

- Constraints conditions with permitting filters (*PI, PO*); where *P* is either input filter or output filter, it depends if the string is sent out or received with strong conditions (s) or weak condition (w):

$$\varphi_i^{(s)}(x;\mathcal{N}(i)) \equiv P \subseteq alph(x)$$
$$\varphi_i^{(w)}(x;\mathcal{N}(i)) \equiv P \cap alph(x) \neq \emptyset$$

- Constraints conditions with permitting filters (*PI, PO*) and forbidden context (*FI, FO*); where *P, F* is either input filter or output filter, it depends if the string is sent out or received with strong conditions (s) or weak conditions (w):

$$\varphi_i^{(s)}(x;\mathcal{N}(i)) \equiv P \subseteq alph(x) \wedge F \cap alph(x) = \emptyset$$
$$\varphi_i^{(w)}(x;\mathcal{N}(i)) \equiv P \cap alph(x) \neq \emptyset \wedge F \cap alph(x) = \emptyset$$

Therefore, the set of objects in a processor *i* after a communication step, denoted by L'_i, are those before the communication (L_i) removing objects sent out and adding objects from other processors connected to *i*. That is,

$$L'_i = L_i - \{w|\varphi^k(x;\mathcal{N}(N_i))\} \bigcup_{\{N_i,N_j\}\in E} \{x|\varphi^k(x;\mathcal{N}(N_j)) \wedge \varphi^k(x;\mathcal{N}(N_i))\}$$

The most important result of such networks of evolutionary processors is that they can solve NP-complete problems in linear time and linear resources.

Extended Networks of Evolutionary Processors

The communication step is only applied to objects in traditional nets. An extended version is proposed in order to be able to send rules from one processor to other ones. This property provides a more realistic behavior since operations are not fixed in processors, they can pass through the net in the same way objects do.

A rule can pass filter conditions provided:

$$\varphi_i^{(\beta)}(r_j : x \to y;\mathcal{N}(i)) \equiv \varphi_i^{(\beta)}(x;\mathcal{N}(i)) \wedge \varphi_i^{(\beta)}(y;\mathcal{N}(i))$$

That is, given rule r_j belonging to processor *i* can be sent out if both antecedent and consequent strings can pass filters in processor *i*, and can be received by other processors if the rule pass their input filters (weak or strong conditions).

A network of evolutionary processors can be transformed into an equivalent extended network of evolutionary processors just choosing the right filters. For example, antecedent belonging to rules can be added to forbidden filter in order to avoid rules communication.

Following theorems regarding computational power of non-extended networks of evolutionary processors can be also applied to extended networks of evolutionary processors.

Theorem 1. A complete NEP of size 5 can generate each recursively enumerable language. [9]

Theorem 2. A star NEP of size 5 can generate each recursively enumerable language. [9]

Theorem 3. The bounded PCP can be solved by an NEP in size and time linearly bounded by the product of K and the length of the longest string of the two Post lists. [12]

Theorem 4. The families of regular and context-free languages are incomparable with the family of languages generated by simple NEPs. [10]

Theorem 5. The 3-colorability problem can be solved in O(m + n) time by a complete simple NEP of size 7m+2, where n is the number of vertices and m is the number of edges of the input graph. [10]

Conclusions and Future Work

This paper presents an extended behavior in networks of evolutionary processors. Now, rules can pass from one processor to another one provided filter constraints are satisfied. This mechanism allows operations and data to pass through the net, not only data like in networks of evolutionary processors. This idea tries to model DNA behavior in which information combines data and operations for living cells, the information is a whole, does not matter if it is data or operations. Rules can pass filters as well as objects do, according to filter specifications.

It is clear that this model is a superset of networks of evolutionary processor and therefore it can solve NP-complete problems in linear time and linear resources. Main advantage of such extended model is that the time to solve a problem is lower than non-extended models since rules can travel to transform objects at different processors.

There are some other possibilities when defining conditions of rules in order to pass filters,

$$\varphi_i^{(\beta)}(r_j : x \to y; \mathcal{N}(i)) \equiv \varphi_i^{(\beta)}(x \cup y; \mathcal{N}(i))$$
$$\varphi_i^{(\beta)}(r_j : x \to y; \mathcal{N}(i)) \equiv \varphi_i^{(\beta)}(x \cap y; \mathcal{N}(i))$$
$$\varphi_i^{(\beta)}(r_j : x \to y; \mathcal{N}(i)) \equiv \varphi_i^{(\beta)}(x \setminus y; \mathcal{N}(i))$$

A lot of open problems that can be taken into account to probe computational power of extended networks of evolutionary processors. First step will consist on solving same problems than non-extended net in order to compare time and resources.

Bibliography

[1] Zandron, C. (2002). A Model for Molecular Computing: Membrane Systems, Universita degli Studi di Milano, Italy.

[2] Paun, G. (2002). Membrane Computing. An Introduction. Springer-Verlag, Berlin.

[3] Ciobanu, G., Paun, G., and Perez-Jimenez, M. (2005). Applications of Membrane Computing. Springer-Verlag, Berlin.

[4] Adleman, L. (1994). Molecular computation of solutions to combinatorial problems. Science, 226:1021–1024.

[5] Errico, L. and Jesshope, C. (1994). Towards a new architecture for symbolic processing. Artificial Intelligence and Information Control Systems of Robots, pages 31–40.

[6] Fahlman, S., Hinton, G., and Seijnowski, T. (1983). Massively parallel architectures for ai: Massively parallel architectures for ai: Netl, thistle and boltzmann machines. In AAAI National Conference on Artificial Intelligence, pages 109–113.

[7] Hillis, W. (1985). The Connection Machine. MIT Press, Cambridge.

[8] Robinson, D. A. (1992). Implications of neural networks for how we think about brain function. Behavioral and Brain Sciences, 15(4): 644–655.

[9] J. Castellanos, C. Martın-Vide, V. Mitrana, and J. Sempere (2003). Networks of evolutionary processors. Acta Informatica, 39:517–529,

[10] J. Castellanos, C. Martın-Vide, V. Mitrana, and J. Sempere (2001). Solving np-complete problems with networks of evolutionary processors. Lecture Notes in Computer Science, 2084:621–628.

[11] E. C. Varju and V. Mitrana (2000). Evolutionary systems, a language generating device inpired by evolving communities of cells. Acta Informatica, 36:913–926,

[12] C. Martin-Vide, V. Mitrana, M. Perez-Jimenez, and F. S. Caparrini (2003). Hybrid networks of evolutionary processors. Lecture Notes in Computer Science, 2723:401–412.

Authors' Information

Luis Fernando de Mingo López – Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: lfmingo@eui.upm.es

Nuria Gómez Blas – Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: ngomez@dalum.eui.upm.es

Francisco Gisbert – Dept. Lenguajes, Sistemas Informáticos e Ingeniería del Software, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Madrid, Spain; e-mail: fgisbert@fi.upm.es

Miguel Angel Peña – Dept. Lenguajes, Sistemas Informáticos e Ingeniería del Software, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Madrid, Spain; e-mail: fgisbert@fi.upm.es

TRAINED NEURAL NETWORK CHARACTERIZING VARIABLES FOR PREDICTING ORGANIC RETENTION BY NANOFILTRATION MEMBRANES

Arcadio Sotto, Ana Martinez, Angel Castellanos

Abstract: Many organic compounds cause an irreversible damage to human health and the ecosystem and are present in water resources. Among these hazard substances, phenolic compounds play an important role on the actual contamination. Utilization of membrane technology is increasing exponentially in drinking water production and waste water treatment. The removal of organic compounds by nanofiltration membranes is characterized not only by molecular sieving effects but also by membrane-solute interactions. Influence of the sieving parameters (molecular weight and molecular diameter) and the physicochemical interactions (dissociation constant and molecular hydrophobicity) on the membrane rejection of the organic solutes were studied. The molecular hydrophobicity is expressed as logarithm of octanol-water partition coefficient. This paper proposes a method used that can be used for symbolic knowledge extraction from a trained neural network, once they have been trained with the desired performance and is based on detect the more important variables in problems where exist multicolineality among the input variables.

Keywords: Neural Networks, Radial Basis Functions, Nanofiltration; Membranes; Retention.

ACM Classification Keywords: K.3.2 Learning (Knowledge acquisition)

Introduction

Phenolic compounds are commonly used as raw materials in the manufacture of polymers, plasticizers, hydraulic fluids and various industrial chemicals. Therefore, there are many wastewater effluents contain discharges amounts of these recalcitrant organic compounds.

Nanofiltration (NF) is a viable treatment for the removal of dissolved organic pollutants for production of drinking water and as combined method with advanced and traditional water treatment process [Van der Bruggen et al 2003] [Hellebrand et al 1997]. Many reported studies indicate that several physical phenomena can play a role in the solute transport through nanofiltration membranes: solution-diffusion, convection (sieving), electrostatic (charge) repulsion and dielectric exclusion. In addition, NF strongly depends on the feed water composition, membrane and solute properties, and operational conditions [Bellona et al 2004]. Therefore retention of organic compounds is influenced either by pore size and charge of membrane, or by the molecular size, hydrophobicity and ionization constant of solutes [Boussu K. et al 2008] [Arsuaga et al 2008].

Neural network is proposed as suitable tool to prediction the membrane performance on solute retention and detect the more important variables when the input variables exist high correlation. Artificial Neural networks perform adaptative learning. This advantage can be used to improve the knowledge acquisition in knowledge engineering .This paper proposes extracting knowledge from a neural network that has learned using sensitivity analysis used to determinate which are the most important variables for the prediction. These will guide the process of create one model for prediction with a few variables, at least the most important variables.

Characteristics about the Forecast Method

Neural networks [Anderson, James A. and Edward Rosenfield., 1988] are non-linear systems whose structure is based on principles observed in biological neuronal systems [Hanson, Stephen J. and David J. Burr. 1990]. A neural network could be seen as a system that can be able to answer a query or give an output as answer to a specific input. The in/out combination, i.e. the transfer function of the network is not programmed, but obtained through a training process on empiric datasets. In practice the network learns the function that links input together with output by processing correct input/output couples. Actually, for each given input, within the learning process, the network gives a certain output that is not exactly the desired output, so the training algorithm modifies some parameters of the network in the desired direction. Hence, every time an example is input, the algorithm adjusts its network parameters to the optimal values for the given solution: in this way the algorithm tries to reach the best solution for all the examples. These parameters we are speaking about are essentially the weights or linking factors between each neuron that forms our network.

There is a great number of Neural Networks [Anderson, James A. 1995] which are substantially distinguished by: type of use, learning model (supervised/non-supervised), learning algorithm, architecture, etc. Multilayer perceptrons (MLPs) are layered feed forward networks typically trained with static backpropagation. These networks have found their way into countless applications requiring static pattern classification. Their main advantage is that they are easy to use, and that they can approximate any input-output map. In principle, backpropagation provides a way to train networks with any number of hidden units arranged in any number of layers.

The research community has developed several different neural network models, such as, radial basis function, growing cell structures and self-organizing feature maps. A common characteristic of the mentioned models is that they distinguish between learning and a performance phase. Neural networks with radial basis functions have proven to be an excellent tool in approximation with few patterns. Most relevant research in theory, design and applications of radial basis function neural networks is due to Moody and Darken [Moody and Darken, 1989]. Radial basis function (RBF) neural networks provide a powerful alternative to multilayer perceptron (MLP) neural networks to approximate or to classify a pattern set. RBFs differ from MLPs in that the overall input-output map is constructed from local contributions of Gaussian axons, require fewer training samples and train faster than MLP. The most widely used method to estimate centers and widths consist on using an unsupervised technique called the k-nearest neighbour rule. The centers of the clusters give the centers of the RBFs and the distance between the clusters provides the width of the Gaussians.

The object of the present study is to ascertain whether the membrane type NF90 has a quantitative effect on the values of the retention for different components analyzed, and affects the relationships between the different variables considered as input in the model propose for prediction retention. Retention behavior of the phenolic compounds by NF90 membrane was investigated in order to clarify the influence of the molecular weight (MW), size (diameter), acid dissociation constant (pk_a) and molecular hydrophobicity ($\log P$) of selected compounds on membrane performance. This paper proposes a method in order to detect the importance of the input variables. In multivariate analysis problems, when there exists correlation among different variables of forecasting, the importance and the sequence when adding variables in the model, can be detected from the knowledge stored in NN, and must be taken into account when the study of the correlations detect relationships among a set of variables

Neural networks can predict any continuous relationship between inputs and the target; artificial neural networks develop a gain term that allows prediction of target variables for a given set of input variables., we use neural networks models with analysis of sensibility, this model predict more accurately the relationship existing between variables, and is a suitable way to find the individual effects of forecasting variables over the variable to forecast , and the way to find a set of forecasting variables to include in the new model.

The addition of a given variable into a forecasting model does not implies that this variable will have an important effect over the response of the model, that is, if a researcher identifies a set of forecasting variables, he must check if they really affect the response. A frequent problem is that some of the forecasting variables are correlated. If the correlation is small, then consequences will be less important. However, if there is a high correlation between two or more forecasting variables, then the model results will be ambiguous but not for obtain a bad prediction, the problem is the high correlation between variables (high lineal association) decrease in a drastic way the individual effect over the response for each correlation variable and sometimes is difficult to detect and is not possible measure the real effect for each variable over the output.

The process of finding relevant data components is based on the concept of sensitivity analysis applied to trained neural networks. Two ANN models predict changes for certain combinations of input variables, detecting the most important influence in the output variable. We have studied different analysis for detecting relationships between molecular diameter, molecular weight, log*P* and pk_a in the two membranes during the process of nanofiltration. Retention organic compounds by correlated with characteristics of membrane and also with phisico-chemical properties of organic solutes. In order to study the relationships between different variables it has been used neural networks models with a single hidden layer and Tanh as transfer function in both cases. One ANN model uses MLP (multilayer perceptron) and the other ANN model uses a normal radial basis function (RBF) for model development.

Two ANNs models have been implemented with four input neurons: molecular weight, molecular diameter, pka and logP to estimate the membrane solute retention. The MLP network uses a sigmoid activation function with a single hidden layer with four neurons. The general form of a feed- forward neural network expresses a transformation of the expected target as a linear combination of no-linear functions of linear combinations of the inputs. A normalized radial basis function (RBF) network is a feed-forward network with a single hidden layer using in this case, the same function sigmoid (Tanh), in the hidden layer with 15 clusters and one output layer. In contrast to MLP, each basis function is the ratio of a bell-shaped Gaussian surface. For all the learning process has been performed with the momentum algorithm. Unsupervised learning stage is based on 100 epochs and the supervised learning control uses as maximum epoch 10000, and threshold 0.001. We have performed an initial study using 17 patterns in training set.

Materials and Methods

Seventeen phenolic compounds were selected to carry out membrane retention experiments. Table 1 summarizes the most important properties of selected compounds.

Thin-film composite polyamide membrane, NF90 supplied by Dow/Filmtec was evaluated in this study. It's classified as nanofiltration membrane. According to the manufacturers, NF90 membrane is polyamide thin-film composite with a microporous polysulfone supporting layer. A cross flow system (SEPA CF II, Osmonics) was used for membrane retention measurements. Organic solution concentrations were fixed at 100 mg L^{-1} and

system temperature was maintained constant in all experiments at 25ºC. It was controlled by circulating feed water through a stainless-still coil immersed in the thermostatic bath. Quantitative analysis of the organic compounds was carried out by means of their respective absorptions in the ultraviolet region, using a Varian Cary 500 Scan UV-VIS-NIR spectrophotometer. Concentration of PEGs and saccarides were measured with a Total Organic Carbon (TOC) analyzer (model TOC-V CSN Shimadzu). Regression factor (R2) obtained for calibrations within the range of experimental concentration used was greater than 0.99.

Retention *R* (%) of a solute was calculated using the expression:

$$R = 1 - \frac{C_p}{C_r} \times 100\% \qquad (1)$$

where Cp and Cr are the concentrations for the permeate and retentate, respectively.

Compound	Formula	Molecular diameter (nm)	Molecular Weight ($gmol^{-1}$)	pK_a	log*P*
Phenol	C_6H_6O	0.1945	94.11	9.86	1.48
Resorcinol	$C_6H_6O_2$	0.1948	110.11	9.45	0.76
Hydroquinone	$C_6H_6O_2$	0.1908	110.11	10.33	0.66
Cathecol	$C_6H_6O_2$	0.2160	110.11	9.5	0.88
3-Nitrophenol	$C_6H_5NO_3$	0.2142	139.11	8.33	1.93
3-Chlorophenol	C_6H_5ClO	0.2134	128.56	9.00	2.40
2-Chlorophenol	C_6H_5ClO	0.2157	128.56	8.5	2.04
2-Nitrophenol	$C_6H_5NO_3$	0.2112	139.11	7.14	1.71
4-Chlorophenol	C_6H_5ClO	0.1915	128.56	9.47	2.43
4-Nitrophenol	$C_6H_5NO_3$	0.1849	139.11	7.23	1.57
Pirogallol	$C_6H_6O_3$	0.2154	126.11	9.12	0.29
Phloroglucinol	$C_6H_6O_3$	0.2331	126.11	7.97	0.06
Oxalic acid	$C_2H_2O_4$	0.1148	90.04	1.38	-0.24
Maleic acid	$C_4H_4O_4$	0.1291	116.07	3.15	0.04
Malonic acid	$C_3H_4O_4$	0.1378	104.06	2.92	-0.31
Acetic acid	$C_2H_4O_2$	0.1218	60.05	4.79	-0.17
Formic acid	CH_2O_2	0.1335	46.03	3.74	-0.37
Ribose	$C_5H_{10}O_5$	0.20856	150.13	12.46	-2.39
Glucose	$C_6H_{12}O_6$	0.28356	180.16	12.45	-3.169
Sucrose	$C_{12}H_{22}O_{11}$	0.38956	342.3	12.81	-3.484
Raffinose	$C_{18}H_{32}O_{16}$	0.50256	504.42	12.81	-6.76

The solute permeation (*B*) was calculated from retention values and defined as follows:

$$B = \frac{1 - R}{R}$$

Results and Conclusions: Determining the Important Inputs for the Model

This example is based on detect the more important variables when exist multicolineality.

Multilayer feedforward networks are often used for modeling complex relationships between the data sets. Deleting unimportant data components in the training sets could lead to smaller networks and reduced-size data vectors. The process of finding relevant data components is based on the concept of sensitivity analysis applied to a trained neural network. ANN models predict changes for certain combinations of input variables, detecting the most important influence in the output variables.

After work with both neural network MLP and RBNF, in both case the variable Mw is the less signification above the model which propose for prediction of the retention *B*, and is consequence of high correlation between Diameter and MW .If we are looking for a model for prediction the retention of the membrane, the most important is the variable diameter as the first to include in the model forward the variable log*P* .

Analysis of the results obtained about the weight importance in percent is listed in the tables. . MLP results are in Table 2, and in table 3 have been shown the RBF results.

Table 2 Multilayer Perceptron results (MLP)

Sensitivity of criterion %			
Variables			
MW	Diameter	pK_a	log*P*
10.182	42.634	18.170	29.014
16.357	42.664	–	38.979
–	50.497	17.565	31.938
Diameter		log*P*	
51.689		48.311	

Table 3 Radial Basis Function results (RBF)

Sensitivity of criterion %			
Variables			
MW	Diameter	pK_a	log*P*
11.709	34.042	13.720	40.529
18.939	56.182	–	24.879
–	36.545	18.174	45.281
Diameter		log*P*	

65.599	34.401

Tables 2 and 3 show how there is correspondence between the importance of the variables in percentage and the membrane retention for both variable and trained neural network.

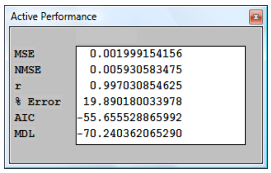
It can be seen that, from tables 2 and 3, how the most important variable in percent % is the diameter followed by the log*P*. The pK_a is not very important and finally the MW has no influence, but this last variable is correlated with the diameter and in some type of membrane it is possible obtain confuse measure about the importance over the output. It can be also seeing how the diameter is the most important variable through the different possible combinations of models, and error is softly decreasing.

The General performance probe displays the Mean Squared Error (MSE), the Normalized Mean Squared Error (NMSE), the Correlation Coefficient (r), and the Percent Error.

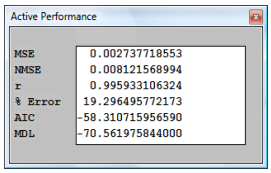
Table 3 General performance probe

MLP

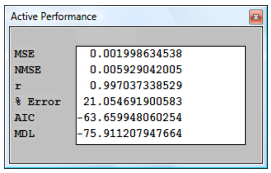
All variables



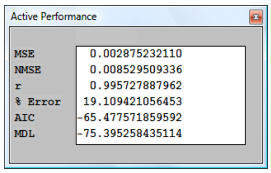
without MW



without pk_a

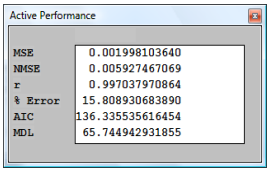


without MW and pk_a

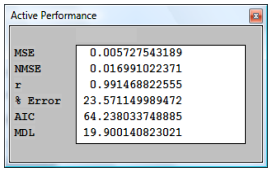


RBF

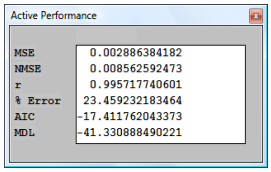
All variables



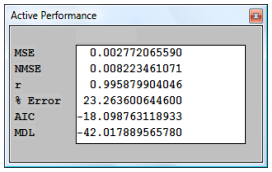
without MW



without pk_a



without MW and pk_a



Once the most important variables for the model have been determined, we can train again the neural network with three or two variables, in this case with diameter and log*P* we obtained a very good results Squared Error SME less 0.001 for prediction solute retention.

This paper presents a method for prediction. In this method, firstly the global problem is obtain the most important variables, extracted and finally the solution is globalize with a model or prediction. Two stages have been judiciously combined, which allow selected to be a more efficient, effective and easy to control process. The obtained results show that this mixed system could be applied to different situations other than the one considered in this paper, due to the general nature of the proposed solution.

Bibliography

[Anderson 1995] Anderson, James A, 1995. An Introduction to Neural Networks Cambridge, MA: MIT Press.

[Arsuaga et al 2008] Arsuaga J.M., Lopez-Muñoz M. J., Aguado J., Sotto A., 2008. Temperature pH and concentration effects on retention and transport of organic pollutants across thin-film composite nanofiltration membranes, Desalination 221. 253-258.

[Bellona C. et al 2004] Bellona, Drewes J., P. Xu and G. Amy, 2004. Factors affecting the rejection of organic solutes during NF/RO treatment. Water Res. 38. 2795-2809.

[Boussu K. et al 2008] K. Boussu C. Vandecasteele and B. Van der Bruggen, 2008. Relation between membrane characteristics and performance in nanofiltration, J. Membr. Sci. 310. 51-65.

[Hellebrand R. et al 1997] Hellebrand R., D. Mantzavinos, I. S. Metcalfe y A.G. Livingston, 1997. Integration of Wet Oxidation and Nanofiltration for Treatment of Recalcitrant Organics in wastewater, Ind. Eng. Chem. Res. 36. 5054-5062.

[Moody, J. and Darken C. (1989)]. Moody, J. and Darken C., 1989. Fast learning in networks of locally-tuned processing units. Neural Computation, 1:281-294 .

[Van der Bruggen et al 2003] Van der Bruggen B., Vandescasteele C. 2003. Removal of pollutants from surface water and groundwater by nanofiltration: overview of possible applications in the drinking water industry, Environ. Pollut. 122. 435.

Authors' Information

Arcadio Sotto - Department of Chemical and Environmental Technology, University Rey Juan Carlos. C/ Tulipán s/n, 28933-Móstoles, Madrid, Spain. arcadio.sotto@urjc.es

Ana Martinez - Natural computing group. Universidad Politécnica de Madrid, Spain. ana.martinez@upm.es

Angel Castellanos - Departamento de Ciencias Básicas aplicadas a la Ingeniería Forestal. Escuela de Ingeniería Técnica Forestal. Universidad Politécnica de Madrid, Avda. de Ramiro de Maeztu s/n 28040 Madrid, Spain. angel.castellanos@upm.es

THE CASCADE NEO-FUZZY ARCHITECTURE USING CUBIC–SPKINE ACTIVATION FUNCTIONS

Yevgeniy Bodyanskiy, Yevgen Viktorov

Abstract: in the paper new hybrid system of computational intelligence called the Cascade Neo-Fuzzy Neural Network (CNFNN) is introduced. This architecture has the similar structure with the Cascade-Correlation Learning Architecture proposed by S.E. Fahlman and C. Lebiere, but differs from it in type of artificial neurons. CNFNN contains neo-fuzzy neurons, which can be adjusted using high-speed linear learning procedures. Proposed CNFNN is characterized by high learning rate, low size of learning sample and its operations can be described by fuzzy linguistic "if-then" rules providing "transparency" of received results, as compared with conventional neural networks. Using of cubic-spline membership functions instead of conventional triangular functions allows increasing accuracy of smooth functions approximation.

Keywords: artificial neural networks, constructive approach, fuzzy inference, hybrid systems, neo-fuzzy neuron, cubic-spline functions.

ACM Classification Keywords: I.2.6 Learning – Connectionism and neural nets.

Introduction

At the present time artificial neural networks are widely applied for solving identification, prediction, and modeling problems of significantly nonlinear processes when information given as time-series or numerical "object-property" tables generated by stochastic or chaotic systems. However in real conditions data processing often must be performed simultaneously with the plant functioning and therefore timing budget becomes quite valuable. So called "optimization-based networks" such as Multilayer Perceptron, Radial Basis Functions Network (RBFN), Normalized Radial Basis Functions Network (NRBFN) in most cases can be ineffective to solve mentioned above problems because of their low convergence rate during learning procedure, curse of dimensionality, and impossibility to learn in on-line mode.

Traditionally by the learning we understand the process of the neural network's synaptic weights adjustment accordingly to selected optimization procedure of the accepted learning criterion [Cichocki, 1993; Haykin, 1999]. Quality of the received results can be improved not only by adjusting weight coefficients but also by adjusting architecture of the neural network (number of nodes). There are two basic approaches of the neural network architecture adjustment: 1) "constructive approach" [Platt, 1991; Nag, 1998; Yingwei, 1998] — starts with simple architecture and gradually adds new nodes during learning; 2) "destructive approach" [Cun, 1990; Hassibi, 1993; Prechelt, 1997] — starts with initially redundant network and simplifies it throughout learning process.

Obviously, constructive approach needs less computational resources and within the bounds of this technique the cascade neural networks (CNNs) [Fahlman, 1990; Schalkoff, 1997; Avedjan, 1999] can be marked out. The most efficient representative of the CNNs is the Cascade-Correlation Learning Architecture (CasCorLA) [Fahlman, 1990]. This network begins with the simplest architecture which consists of a single neuron. Throughout a learning procedure new neurons are added to the network, producing a multilayer structure. It is important that during each learning epoch only one neuron of the last cascade is adjusted. All pre-existing neurons process information with "frozen" weights. The CasCorLA authors, S.E. Fahlman and C. Lebiere, point

out high speed of the learning procedure and good approximation properties of this network. But it should be observed that elementary Rosenblatt perceptrons with hyperbolic tangent activation functions are used in this architecture as nodes. Thus an output signal of each neuron is non-linearly depended from its weight coefficients. Therefore it is necessary to use gradient learning methods such as delta-rule or its modifications, and operation speed optimization becomes impossible. In connection with the above it seems to be reasonable to synthesize the cascade architecture based on the elementary nodes with linear dependence of an output signal from the synaptic weights. It allows to increase a speed of synaptic weights adjustment and to reduce minimally required size of training set.

In [Bodyanskiy, 2007a] the ortho-neurons were proposed as such nodes. Also it was shown how simply and effectively an approximation of sufficiently complex function can be performed using this technique. In [Bodyanskiy, 2004a; Bodyanskiy, 2004b; Bodyanskiy, 2006a; Bodyanskiy, 2006b; Bodyanskiy, 2007b; Bodyanskiy, 2008a; Viktorov, 2008] the orthogonal and the cascade orthogonal neural networks were introduced. These architectures have shown quite good results during simulation modeling, significantly exceeding the conventional cascade neural networks in training speed.

It is well known the main ANN's disadvantage is a non-interpretability of received results, i.e. trained neural network is a "black box", and often their usage is restrained because of this reason. An interpretability and transparency together with the learning capabilities are the main properties of the neuro-fuzzy systems [Jang, 1997], which can be trained using backpropagation and in consequence the time required for weights tuning and the size of a training set are significantly increase. The neural network which allows to avoid these disadvantages was introduced in [Bodyanskiy, 2008b]. It has the cascade architecture and uses neo-fuzzy neurons [Yamakawa, 1992; Uchino, 1997; Miki, 1999] as nodes. Traditionally triangular functions are used as membership functions in neo-fuzzy neuron. Therefore when we have deal with a process described by smooth function we should either increase quantity of membership functions, what leads to increasing of the time required for weight coefficients adjustment, or quality of obtained results would be reduced. At this paper an attempt to get over this difficulty is taken.

The Neo-Fuzzy Neuron

Neo-fuzzy neuron is a nonlinear multi-input single-output system shown in Fig.1.

It realizes the following mapping:

$$\hat{y} = \sum_{i=1}^{n} f_i(x_i) \tag{1}$$

where x_i is the *i*-th input (*i* = 1,2,…,*n*), $\hat{y}$ is a system output. Structural blocks of neo-fuzzy neuron are nonlinear synapses NS_i which perform transformation of *i*-th input signal in the from

$$f_i(x_i) = \sum_{j=1}^{h} w_{ji}\mu_{ji}(x_i).$$

Each nonlinear synapse realizes the fuzzy inference

IF x_i IS x_{ji} THEN THE OUTPUT IS w_{ji}

where x_{ji} is a fuzzy set which membership function is μ_{ji}, w_{ji} is a singleton (synaptic weight) in consequent [Uchino, 1997]. As it can be readily seen nonlinear synapse in fact realizes Takagi-Sugeno fuzzy inference of zero order.

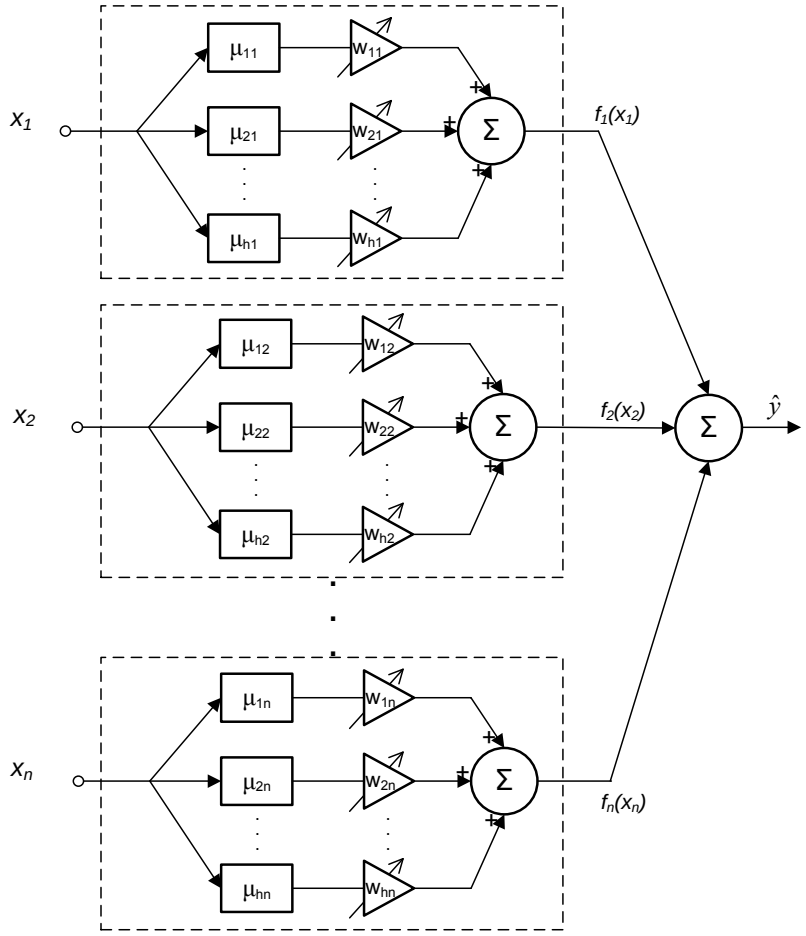


Figure 1. The Neo-Fuzzy Neuron

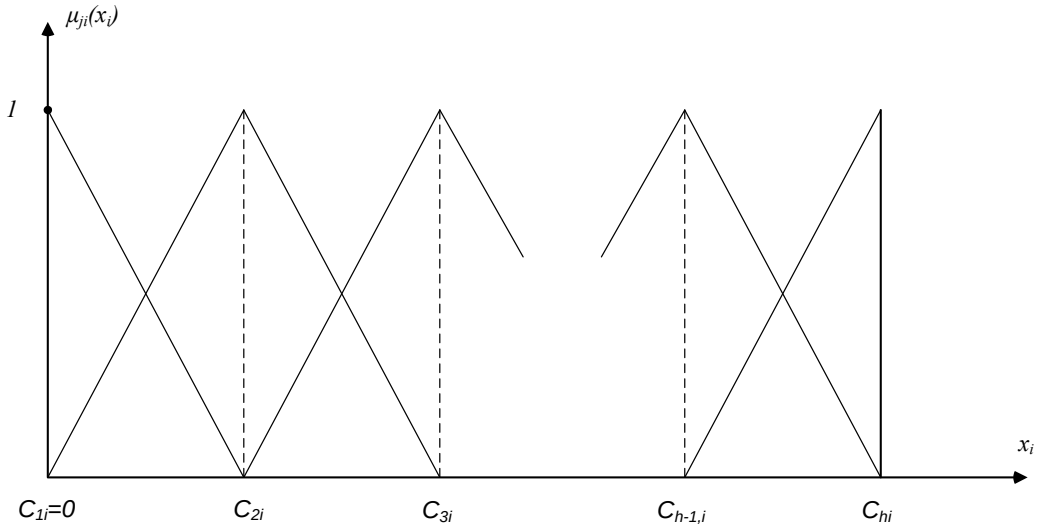


Figure 2. Triangular membership functions

Conventionally the membership functions $\mu_{ji}(x_i)$ in the antecedent are complementary triangular functions as shown in Fig. 2.

For preliminary normalized input variables x_i (usually $0 \le x_i \le 1$), membership functions can be expressed in the form:

$$\mu_{ji}(x_i) = \begin{cases} \dfrac{x_i - c_{j-1,i}}{c_{ji} - c_{j-1,i}}, x \in [c_{j-1,i}, c_{ji}], \\ \dfrac{c_{j+1,i} - x_i}{c_{j+1,i} - c_{ji}}, x \in [c_{ji}, c_{j+1,i}], \\ 0 - otherwise \end{cases}$$

where c_{ji} are arbitrarily selected centers of corresponding membership functions. Usually they are equally distributed on interval [0, 1]. This contributes to simplify the fuzzy inference process. That is, an input signal x_i activates only two neighboring membership functions simultaneously and the sum of the grades of these two membership functions equals to unity (Ruspini partitioning), i.e.

$$\mu_{ji}(x_i) + \mu_{j+1,i}(x_i) = 1. \tag{2}$$

Thus, the fuzzy inference result produced by the Center-of-Gravity defuzzification method can be given in the very simple form:

$$f_i(x_i) = w_{ji}\mu_{ji}(x_i) + w_{j+1,i}\mu_{j+1,i}(x_i). \tag{3}$$

By summing up $f_i(x_i)$, the output $\hat{y}$ of Eq. (1) is produced.

When a vector signal $x(k) = (x_1(k), x_2(k), ..., x_n(k))^T$ ($k = 1,2,...$ is a discrete time) is fed to the input of the neo-fuzzy neuron, the output of this neuron is determined by both the membership functions $\mu_{ji}(x_i(k))$ and tunable synaptic weights $w_{ji}(k-1)$, which were obtained at the previous training epoch.

$$\hat{y}(k) = \sum_{i=1}^{n} f_i(x_i(k)) = \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}(k-1)\mu_{ji}(x_i(k))$$

and thereby neo-fuzzy neuron contains $h \times n$ synaptic weights which should be determined.

The authors of the NFN note [Yamakawa, 1992; Uchino, 1997; Miki, 1999] among its most important advantages, the high rate of learning, computational simplicity, the possibility of finding the global minimum of the learning criterion in real time and also that it is characterized by fuzzy linguistic "if-then" rules.

The learning criterion (goal function) is the standard local quadratic error function:

$$E(k) = \frac{1}{2}\left(y(k) - \hat{y}(k)\right)^2 = \frac{1}{2}e(k)^2 = \frac{1}{2}\left(y(k) - \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}\mu_{ji}(x_i(k))\right)^2$$

minimized via the conventional gradient stepwise algorithm, resulting in the following weight update procedure:

$$w_{ji}(k+1) = w_{ji}(k) + \eta e(k+1)\mu_{ji}(x_i(k+1)) =$$
$$= w_{ji}(k) + \eta\left(y(k+1) - \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}(k)\mu_{ji}(x_i(k+1)) \right)\mu_{ji}(x_i(k+1))$$

where $y(k)$ is the target value of the output (learning signal), η is the scalar learning rate parameter which determines the speed of convergence and is chosen empirically.

For the purpose of increasing training speed [Bodyanskiy, 2003; Kolodyazhniy, 2005] Kaczmarz-Widrow-Hoff optimal one-step algorithm [Kaczmarz, 1937; Kaczmarz, 1993; Widrow, 1960] can be used in the following form:

$$w(k+1) = w(k) + \frac{y(k+1) - w^T(k)\mu(x(k+1))}{\left\|\mu(x(k+1))\right\|^2}\mu(x(k+1)) \tag{4}$$

where

$\mu(x(k+1)) = \left(\mu_{11}(x_1(k+1)),...,\mu_{h1}(x_1(k+1)),...,\mu_{h2}(x_2(k+1)),...,\mu_{ji}(x_i(k+1)),...,\mu_{hn}(x_n(k+1))\right)^T$, $w(k) = \left(w_{11}(k),...,w_{h1}(k),...,w_{h2}(k),...,w_{ji}(k),...,w_{hn}(k)\right)^T$ are $(hn)\times 1$ vectors, generated by the corresponding variables. Also exponentially weighted modification of procedure (4) can be used:

$$\begin{cases} w(k+1) = w(k) + r^{-1}(k+1)\left(y(k+1) - w^T(k)\mu(x(k+1))\right)\mu(x(k+1)), \\ r(k+1) = \alpha r(k) + \left\|\mu(x(k+1))\right\|^2, 0 \le \alpha \le 1 \end{cases} \tag{5}$$

which possesses both smoothing and following properties.

In case we have priori defined data set training process can be performed in a batch mode for one epoch using conventional least squares method.

On basis of neo-fuzzy neurons in [Kolodyazhniy, 2004a; Kolodyazhniy, 2004b; Bodyanskiy, 2005a; Bodyanskiy, 2005b; Bodyanskiy, 2005c; Kolodyazhniy, 2006] two-layer feedforward neuro-fuzzy network was synthesized. It possesses improved approximation capabilities in comparison with conventional multilayer perceptron. Given ANN utilized backpropagation for weight adaptation and obviously it results in decreasing rate of convergence in the hidden layer. This circumstance also was a reason for developing cascade neo-fuzzy neural network and improvement of its approximation possibilities.

Cubic-Spline Activation Functions

As stated above conventionally triangular membership functions are used as activation functions in neo-fuzzy neuron. It entails some difficulties during modeling or forecasting processes which are described by smooth functions. At this case piecewise-linear approximation performed by conventional neo-fuzzy neuron can bring us to decreased accuracy of received results. To minimize its negative effect we can increase number of membership functions. But it results in increasing of synaptic weight coefficients quantity and therefore complexity of our architecture is rising as well as time required for its learning.

To avoid this disadvantage cubic-spline membership functions (6) can be used. Commonly they are given in the following form:

$$\mu(x)=\begin{cases}0,\ if\ x\le a,\\ 1-3\left(\dfrac{x-b}{a-b}\right)^2+2\left(\dfrac{x-b}{a-b}\right)^3,\ if\ a<x\le b,\\ 1,\ if\ b<x\le c,\\ 1-3\left(\dfrac{x-c}{d-c}\right)^2+2\left(\dfrac{x-c}{d-c}\right)^3,\ if\ c<x\le d,\\ 0,\ if\ x>d\end{cases} \tag{6}$$

and shown in the Fig. 3.

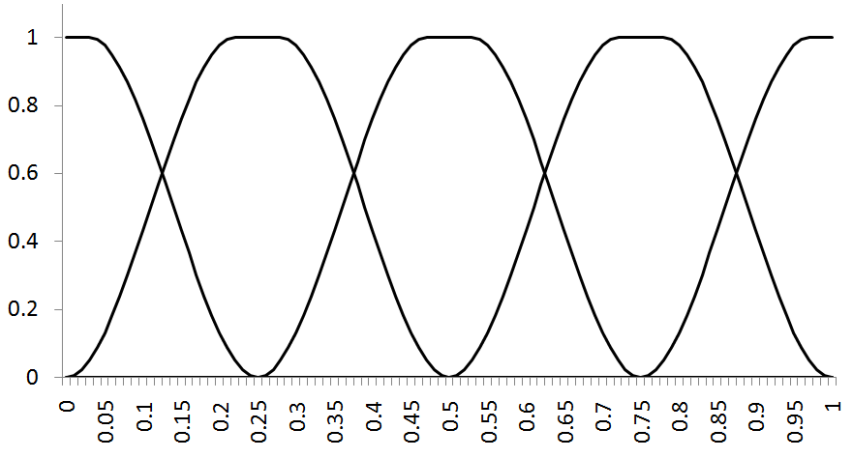


Figure 3. Cubic-spline membership functions (in case $b \neq c$)

It can be readily seen from the Fig. 3 that input signal activates only two neighboring functions simultaneously just like in case with triangular membership functions. But given system of functions doesn't meet the requirements to satisfy the Ruspini partitioning (2). It brings us to impossibility of using Center-Of-Gravity deffuzification method in a simple form (3) and so overall computational complexity is increases. To avoid this imperfection we can use cubic spline membership functions with $b = c$ and also in this case expression (6) can be replaced by another expression (7) which gives completely identical results but its usage is more suitable.

$$\mu(x)=\begin{cases}0.25\left(2+3\dfrac{2x-x_i-x_{i-1}}{x_i-x_{i-1}}-\left(\dfrac{2x-x_i-x_{i-1}}{x_i-x_{i-1}}\right)^3\right),\ x\in[x_{i-1},x_i],\\ 0.25\left(2-3\dfrac{2x-x_{i+1}-x_i}{x_{i+1}-x_i}+\left(\dfrac{2x-x_{i+1}-x_i}{x_{i+1}-x_i}\right)^3\right),\ x\in(x_i,x_{i+1}].\end{cases} \tag{7}$$

Proposed cubic spline membership functions shown in Fig. 4.

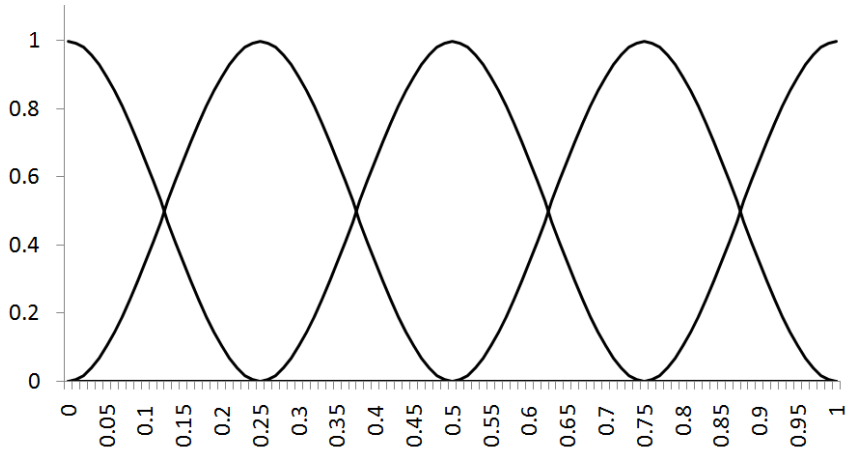


Figure 4. Cubic spline membership functions (in case $b = c$ or in case of notation (7))

Utilizing such recorded membership functions (7) we've got all requirements to meet the Ruspini partitioning (2) and it is considerably contributes to simplify the fuzzy inference process. On the other hand, using of cubic spline activation functions provides smooth polynomial approximation instead of piecewise-linear approximation and makes possible to perform a high quality modeling of significantly nonlinear nonstationary signals and processes.

The Cascade Neo-Fuzzy Architecture Using Cubic-Spline Activation Functions and Its Learning Algorithm

The Cascade Neo-Fuzzy Neural Network architecture shown in Fig.5 and mapping which it realizes has the following form:

- neo-fuzzy neuron of the first cascade

$$\hat{y}^{[1]} = \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}^{[1]}\mu_{ji}(x_i),$$

- neo-fuzzy neuron of the second cascade

$$\hat{y}^{[2]} = \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}^{[2]}\mu_{ji}(x_i) + \sum_{j=1}^{h} w_{j,n+1}^{[2]}\mu_{j,n+1}(\hat{y}^{[1]}),$$

- neo-fuzzy neuron of the third cascade

$$\hat{y}^{[3]} = \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}^{[3]}\mu_{ji}(x_i) + \sum_{j=1}^{h} w_{j,n+1}^{[3]}\mu_{j,n+1}(\hat{y}^{[1]}) + \sum_{j=1}^{h} w_{j,n+2}^{[3]}\mu_{j,n+2}(\hat{y}^{[2]}),$$

- neo-fuzzy neuron of the *m*-th cascade

$$\hat{y}^{[m]} = \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}^{[m]}\mu_{ji}(x_i) + \sum_{l=n+1}^{n+m-1}\sum_{j=1}^{h} w_{jl}^{[m]}\mu_{jl}(\hat{y}^{[l-n]}). \qquad (8)$$

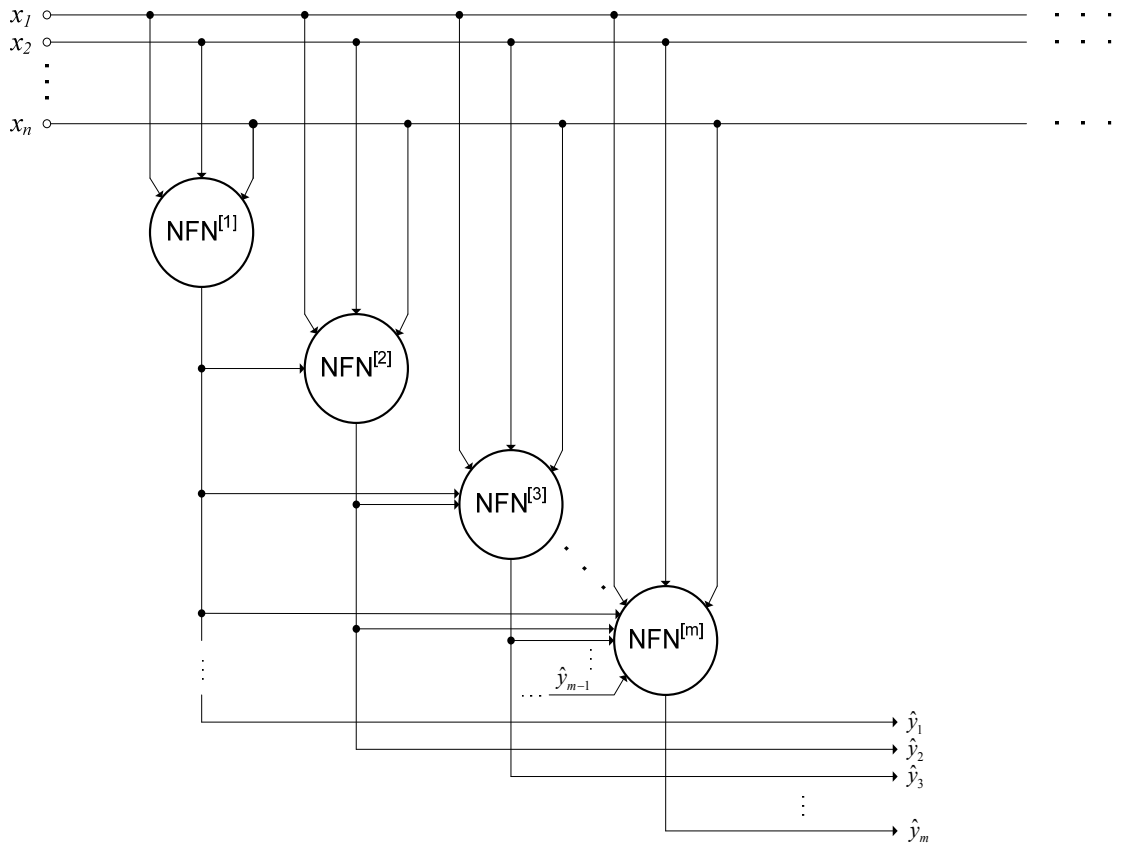


Figure 5. The Cascade Neo-Fuzzy Neural Network Architecture.

Thus cascade neo-fuzzy neural network contains $h\left(n+\sum_{l=1}^{m-1} l\right)$ adjustable parameters and it is important that all of them are linearly included in the definition (8).

Let us define $h(n+m-1)\times 1$ membership functions vector of *m*-th neo-fuzzy neuron $\mu^{[m]} = (\mu_{11}(x_1),...,\mu_{h1}(x_1),\mu_{12}(x_2),...,\mu_{h2}(x_2),...,\mu_{ji}(x_i),...,\mu_{hn}(x_n),\mu_{1,n+1}(\hat{y}^{[1]}),...,\mu_{h,n+1}(\hat{y}^{[1]}),$ $,...,\mu_{h,n+m-1}(\hat{y}^{m-1}))^T$ and corresponding vector of synaptic weights $w^{[m]} =$ $= (w_{11}^{[m]}, w_{21}^{[m]},..., w_{h1}^{[m]}, w_{12}^{[m]},..., w_{h2}^{[m]},..., w_{ji}^{[m]},..., w_{hn}^{[m]}, w_{1,n+1}^{[m]},..., w_{h,n+1}^{[m]},..., w_{h,n+m-1}^{[m]})^T$ which has the same dimensionality. Then we can represent expression (8) in vector notation:

$$\hat{y}^{[m]} = w^{[m]T}\mu^{[m]}.$$

The cascade neo-fuzzy neural network learning can be performed in both batch mode and mode of sequential information processing (adaptive weights tuning).

Firstly, let us examine situation when all training dataset priori defined, i.e. we have a set of points $x(1), y(1); x(2), y(2);...; x(k), y(k);...; x(N), y(N).$ For the neo-fuzzy neuron of the first cascade NFN[1] a set of membership level values $\mu^{[1]}(1), \mu^{[1]}(2),..., \mu^{[1]}(k),..., \mu^{[1]}(N)$ is evaluated, where

$\mu^{[1]}(k) = \left(\mu_{11}(x_1(k)),..., \mu_{h1}(x_1(k)), \mu_{12}(x_2(k)),..., \mu_{h2}(x_2(k)),..., \mu_{ji}(x_i(k)),..., \mu_{hn}(x_n(k))\right)^T$. Then using direct minimization of the learning criterion

$$E_N^{[1]} = \frac{1}{2}\sum_{k=1}^{N}(e^{[1]}(k))^2 = \frac{1}{2}\sum_{k=1}^{N}(y(k) - \hat{y}^{[1]}(k))^2$$

vector of synaptic weights can be evaluated

$$w^{[1]}(N) = \left(\sum_{k=1}^{N}\mu^{[1]}(k)\mu^{[1]T}(k)\right)^{+}\sum_{k=1}^{N}\mu^{[1]}(k)y(k) = P^{[1]}(N)\sum_{k=1}^{N}\mu^{[1]}(k)y(k) \tag{9}$$

where $(\bullet)^{+}$ - symbol of Moore-Penrose pseudoinversion.

In case data are proceed sequentially more suitable to use a recurrent form of least squares method instead of (9)

$$\begin{cases} w^{[1]}(k+1) = w^{[1]}(k) + \dfrac{P^{[1]}(k)(y(k+1) - w^{[1]T}(k)\mu^{[1]}(k+1))}{1 + \mu^{[1]T}(k+1)P^{[1]}(k)\mu^{[1]}(k+1)}\mu^{[1]}(k+1), \\ P^{[1]}(k+1) = P^{[1]}(k) - \dfrac{P^{[1]}(k)\mu^{[1]}(k+1)\mu^{[1]T}(k+1)P^{[1]}(k)}{1 + \mu^{[1]T}(k+1)P^{[1]}(k)\mu^{[1]}(k+1)}, P^{[1]}(0) = \beta I \end{cases} \tag{10}$$

where β is a large positive number, I is a unity matrix with corresponding dimensionality.

Usage of adaptive algorithms (4) or (5) is also possible and leads to reducing of computational complexity of learning process. In any case utilization of procedures (4), (5), (9), (10) essentially reduces learning time in comparison with gradient algorithms underlying delta-rule and backpropagation.

After the first cascade learning completion, synaptic weights of the neo-fuzzy neuron NFN_1 become 'frozen', all values $\hat{y}^{[1]}(1), \hat{y}^{[1]}(2),..., \hat{y}^{[1]}(k),..., \hat{y}^{[1]}(N)$ are evaluated and second cascade of network which consists of a single neo-fuzzy neuron NFN_2 is generated. It has one additional input for the output signal of the first cascade. Then procedure (5) again applied for adjusting vector of weight coefficients $w^{[2]}$, which dimensionality is $h(n+1)$.

In serial mode neurons are trained sequentially, i.e. on basis of input signal $x(k)$ synaptic weights $w^{[1]}(k)$ are evaluated and vector of outputs $\hat{y}^{[1]}(k)$ is obtained, then using vector of second cascade inputs $\left(x^T(k), \hat{y}^{[1]}(k)\right)$ weights $w^{[2]}(k)$ and outputs $\hat{y}^{[2]}(k)$ are calculated. For this purpose algorithms (2), (3) and (6) can be used equally well.

The neural network growing process (increasing quantity of cascades) continues until we obtain required precision of the solved problem's solution, and for the adjusting weight coefficients of the last *m*-th cascade following expressions are used:

$$w^{[m]}(N) = \left(\sum_{k=1}^{N}\mu^{[m]}(k)\mu^{[m]T}(k)\right)^{+}\sum_{k=1}^{N}\mu^{[m]}(k)y(k) = P^{[m]}(N)\sum_{k=1}^{N}\mu^{[m]}(k)y(k)$$

in a batch mode or

$$\begin{cases} w^{[m]}(k+1) = w^{[m]}(k) + \dfrac{P^{[m]}(k)(y(k+1) - w^{[m]T}(k)\mu^{[m]}(k+1))}{1 + \mu^{[m]T}(k+1)P^{[m]}(k)\mu^{[m]}(k+1)}\mu^{[m]}(k+1), \\ P^{[m]}(k+1) = P^{[m]}(k) - \dfrac{P^{[m]}(k)\mu^{[m]}(k+1)\mu^{[m]T}(k+1)P^{[m]}(k)}{1 + \mu^{[m]T}(k+1)P^{[m]}(k)\mu^{[m]}(k+1)}, P^{[m]}(0) = \beta I \end{cases}$$

or

$$\begin{cases} w^{[m]}(k+1) = w^{[m]}(k) + (r^{[m]}(k+1))^{-1}\left(y(k+1) - w^{[m]T}(k)\mu^{[m]}(k+1)\right)\mu^{[m]}(k+1), \\ r^{[m]}(k+1) = \alpha r^{[m]}(k) + \left\|\mu^{[m]}(k+1)\right\|^2, 0 \le \alpha \le 1 \end{cases}$$

in a serial mode.

Proposed CNFNN significantly excels its prototype – cascade-correlation architecture – in learning speed and can be trained in both batch mode and serial (adaptive) mode. Linguistic interpretation of received results considerably extends functional facilities of cascade neo-fuzzy neural network. Using of cubic spline functions as activation functions in neo-fuzzy neurons significantly increases approximation qualities of network when we have deal with processes which described by highly non-linear signals.

Simulation Results

In order to confirm the efficiency of proposed approach for smooth functions approximation we solved two test problems. First was the 'breast cancer in Wisconsin' benchmark classification problem. It is evident that when we have deal with such problem piecewise-linear separating hyperspace which can be obtained using conventional neo-fuzzy neuron gives almost the same quality of classification like smooth shaped separating hyperspace. So at this case advantage of proposed method wouldn't be significant, what shown below. On the other hand if we model some process which described by the smooth function proposed at this paper technique allows considerably reduce quantity of weight coefficients and therefore overall computational complexity of neuro-fuzzy architecture and also increase quality of received results. So the second test problem was the dynamic object identification problem.

Let us have a look at the first test problem – 'breast cancer in Wisconsin' benchmark classification problem. We used dataset which contained 699 points and can be found at the address ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/cancer/cancer1/datacum. 16 points had parameters with missed values so they were eliminated from the dataset and remaining 683 points were separated on training set – 478 points (70%) and test set – 205 points (30%).

Each point has 9-dimensional feature vector and 1 class parameter which should be determined and identifies either benign or malignant tumor has examined patient. Feature values were normalized on interval [0; 1]. Normalization procedure preceding synaptic weights adjustment process is very important when we train the Cascade Neo-Fuzzy Neural Network because used set of cubic spline activation functions gives zero output if input signal lies outside specified interval. Also normalization procedure is necessary between cascades.

For comparison two architectures were trained and simulated. First of them used conventional neo-fuzzy neurons with triangular activation functions and the second one utilized neo-fuzzy neurons with cubic spline activation functions. Both architectures consist of 3 cascades and each cascade consists of a single neo-fuzzy neuron with 5 activation functions. For synaptic weight coefficients adaptation in each cascade least squares method was utilized. Obtained results of classifications can be found in table 1.

When output signal be found within the range [0.3; 0.7] it is lesser probability that classification were correct. We quantify and marked out such classified samples as points outside the 'belief zone'.

Table 1 – Classification results for Cascade Neo-Fuzzy Neural Networks with different types of neo-fuzzy neurons.

Type of neo-fuzzy neuron	Correctly classified examples	Incorrectly classified examples	Outside the 'belief zone'
Conventional NFN	197	6	2
NFN with cubic spline activation functions	197	3	5

From the table we can see that Cascade Neo-Fuzzy Neural Network which utilizes neo-fuzzy neurons with cubic spline activation functions gives slightly better results than architecture with triangular activation functions. As it was already stated above difference between two examined architectures at this case isn't considerable. But let us further pay attention on the class of problems which can be solved better and faster using proposed at this paper cascade architecture.

The second test problem was the dynamic plant identification problem. Proposed dynamic plant [Patra, 2002; Narendra, 1990] can be defined by following equation:

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + f(u(k))$$

where

$$f(u) = 0.6\sin u + 0.3\sin 3u + 0.1\sin 5u.$$

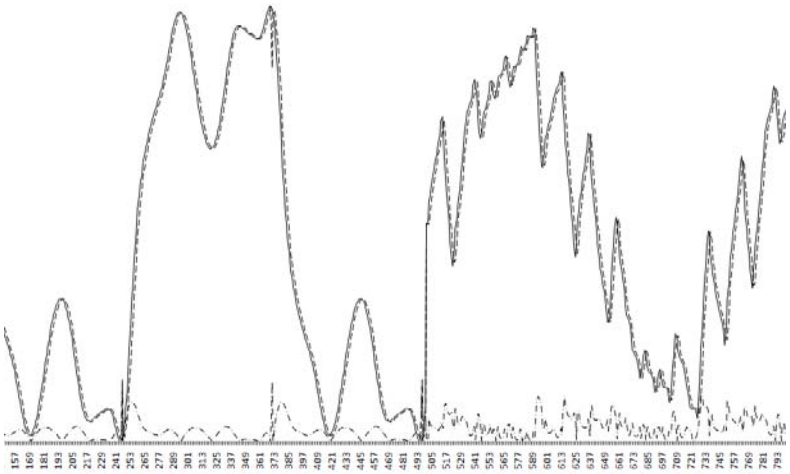
There was generated a sequence which contained 1500 values of signal for k=1,2,...,1500. On training set signal $u(k) = \sin 2k/250$ (k=1,...,500) have been used and on the testing set $u(k) = \sin 2k/250$ (k=501,...,1000), $u(k) = 0.5\sin 2k/250 + 0.5\sin 2k/25$ (k=1001,...1500). It means that on testing set sinusoidal component of the dynamic plant is changing and therefore output signal changes its form too. Obtained set was normalized on interval [0, 1] because of reasons mentioned above.

For estimation of received results we used normalized mean square error:

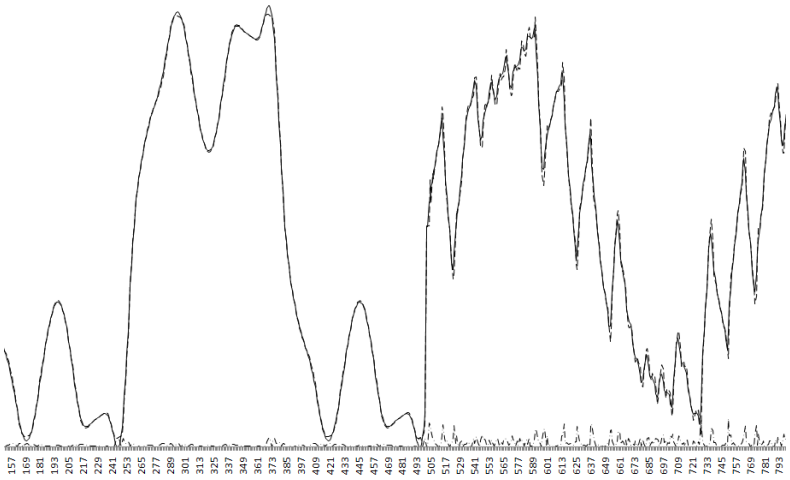
$$NRMSE(k,N) = \frac{\sum_{q=1}^{N} e^2(k+q)}{N\sigma}$$

where σ is a mean square deviation of the predicted process on the training set.

Three architectures were trained and simulated. First architecture used conventional neo-fuzzy neuron with 6 activation functions in each cascade and had eight cascades. Second and third architectures were the same except the type of activation functions. One of them used triangular functions and other cubic spline functions. Each nonlinear synapse contained 4 activation functions and quantity of cascades for both architectures was 4. For adjusting weight coefficients in all cases LSM was used. Obtained results are given in table 2 and shown in Fig. 6 (dynamic object identification results using second and third architecture are shown).



a)



b)

Figure 2. Dynamic plant identification using
the Cascade Neo-Fuzzy Neural Network with a) conventional neo-fuzzy neurons;
b) neo-fuzzy neurons with cubic spline activation functions in non-linear synapses;
normalized plant output – solid line; network output – dashed line; identification error – chain line.

Table 2. Results of the dynamic plant identification.

Type of neo-fuzzy neuron in the cascade architecture	Quantity of activation functions	Quantity of cascades	Total number of adjustable parameters in architecture	NRMSE
Conventional NFN	6	8	102	0.0008
Conventional NFN	2	4	16	0.0063
NFN with cubic spline activation functions	2	4	16	0.0005

As it can be seen from the table 2 usage of neo-fuzzy neurons with cubic spline activation functions provides decreasing of architecture complexity as well as increasing of obtained results quality. In concerned test problem, when we applied two identical architectures, obtained error for that which utilized conventional neo-fuzzy neurons was appreciably greater and therefore quality of output signal was noticeably worse, what can be seen in Fig. 6. To obtain quality of output signal comparable with provided by Cascade Neo-Fuzzy Neural Network with cubic spline activation functions, complexity of architecture which utilized conventional neo-fuzzy neurons was considerably increased. And approximately identical result was obtained when total number of adjustable parameters was about six times greater than in architecture with cubic spline activation functions.

Conclusion

The Cascade Neo-Fuzzy Neural Network which utilized neo-fuzzy neurons with cubic spline activation functions is proposed. It differs from the known cascade networks in increasing of operation speed, real-time processing possibility and transparency due to linguistic interpretability of received results. Used type of activation functions allows to increase accuracy of smooth functions approximation, reduce time required for adjusting weight coefficients and decrease overall architecture complexity. Theoretical justification and experiment results confirm the efficiency of developed approach to cascade neo-fuzzy systems synthesis.

Bibliography

[Cichocki, 1993]. Cichocki A., Unbehauen R. Neural Networks for Optimization and Signal Processing. Stuttgart, Teubner, 1993.

[Haykin, 1999] Haykin S. Neural Networks. A Comprehensive Foundation. Upper Saddle River, N.J, Prentice Hall, 1999.

[Platt, 1991] Platt J. A resource allocating network for function interpolation. In: Neural Computation, 3. 1991. P. 213-225.

[Nag, 1998]. Nag A., Ghosh J. Flexible resource allocating network for noisy data. In: Proc. SPIE Conf. on Applications and Science of Computational Intelligence. 1998. P. 551-559.

[Yingwei, 1998] Yingwei L., Sundararajan N., Saratchandran P. Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm. IEEE Trans. on Neural Networks, 9. 1998. P. 308-318.

[Cun, 1990] Cun Y. L., Denker J. S., Solla S. A. Optimal brain damage. In: Advances in Neural Information Processing Systems, 2. 1990. P. 598-605.

[Hassibi, 1993] Hassibi B. Stork D. G. Second-order derivatives for network pruning: Optimal brain surgeon. In: Advances in Neural Information Processing Systems. Eds. Hanson et al. San Mateo, CA: Morgan Kaufman, 1993. P. 164-171.

[Prechelt, 1997]. Prechelt L. Connection pruning with static and adaptive pruning schedules. Neurocomputing, 16. 1997. P. 49-61.

[Fahlman, 1990] Fahlman S.E., Lebiere C. The cascade-correlation learning architecture. In: Advances in Neural Information Processing Systems. Ed. D.S. Touretzky. San Mateo, CA: Morgan Kaufman, 1990. P. 524-532.

[Schalkoff, 1997] Schalkoff R.J. Artificial Neural Networks. N.Y.: The McGraw-Hill Comp., 1997.

[Avedjan, 1999] Avedjan E.D., Barkan G.V., Levin I.K. Cascade neural networks. Avtomatika i Telemekhanika, 3. 1999. P. 38-55.

[Bodyanskiy, 2007a] Bodyanskiy Ye., Viktorov Ye., Slipchenko O. Orthosynapse, ortho-neurons, and neuropredictor based on them. Systemi obrobki informacii, Issue 4(62). 2007. P. 139-143.

[Bodyanskiy, 2004a] Bodyanskiy Ye., Kolodyazhniy V., Slipchenko O. Artificial neural network with orthogonal activation functions for dynamic system identification. Synergies between Information Processing and Automation. Eds. O. Sawodny and P. Scharff. Aachen: Shaker Verlag, 2004. P. 24-30.

[Bodyanskiy, 2004b] Bodyanskiy Ye., Kolodyazhniy V., Slipchenko O. Structural and synaptic adaptation in the artificial neural networks with orthogonal activation functions. Sci. Proc. of Riga Technical University. Comp. Sci., Inf. Technology and Management Sci, 20. 2004. P. 69-76.

[Bodyanskiy, 2006a] Bodyanskiy Ye., Pliss I., Slipchenko O. Growing neural networks based on orthogonal activation functions. Proc. XII-th Int. Conf. "Knowledge–Dialog–Solution". Varna, 2006. P. 84-89.

[Bodyanskiy, 2006b] Bodyanskiy Ye., Slipchenko O. Ontogenic neural networks using orthogonal activation functions. Prace naukowe Akademii Ekonomicznej we Wroclawiu, 21. 2006. P. 13-20.

[Bodyanskiy, 2007b] Bodyanskiy Ye., Pliss I., Slipchenko O. Growing neural network using nonconventional activation functions. Int. J. Information Theories & Applications, 14. 2007. P. 275-281.

[Bodyanskiy, 2008a] Bodyanskiy Ye., Dolotov A., Pliss I., Viktorov Ye. The cascade orthogonal neural network. Advanced Research in Artificial Intelligence. Bulgaria, Sofia: Institute of Informational Theories and Applications FOI ITHEA, 2. 2008. P. 13-20.

[Viktorov, 2008] Viktorov Ye., Bodyanskiy Ye., Dolotov A. Solving approximation and forecasting problems using double ortho-neuron. Komp'juterni nauky ta informacijni tekhnologhiji, Issue 598. 2008. P. 70-77.

[Jang, 1997] Jang Jr. S. R., Sun C. T., Mizutani E. Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence. N.J.: Prentice Hall, 1997.

[Bodyanskiy, 2008b] Bodyanskiy Ye., Viktorov Ye., Pliss I. The cascade neo-fuzzy neural network and its learning algorithm. Visnyk Uzhghorods'kogho Nacional'nogho universytetu. Serija "Matematyka i informatyka", Issue 17. 2008. P. 48-58.

[Yamakawa, 1992] Yamakawa T., Uchino E., Miki T., Kusanagi H. A neo fuzzy neuron and its applications to system identification and prediction of the system behavior. Proc. of 2-nd Int.Conf. on Fuzzy Logic and Neural Networks "IIZUKA-92". Iizuka, Japan, 1992. P. 477-483.

[Uchino, 1997] Uchino E., Yamakawa T. Soft computing based signal prediction, restoration and filtering. Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks and Genetic Algorithms. Ed. Da Ruan. Boston: Kluwer Academic Publisher, 1997. P. 331-349.

[Miki, 1999] Miki T., Yamakawa T. Analog implementation of neo-fuzzy neuron and its on-board learning. Computational Intelligence and Applications. Ed. N. E. Mastorakis. Piraeus: WSES Press, 1999. P. 144-149.

[Bodyanskiy, 2003] Bodyanskiy Ye., Kokshenev I., Kolodyazhniy V. An adaptive learning algorithm for a neo-fuzzy neuron. Proc. 3rd Int. Conf. of European Union Soc. for Fuzzy Logic and Technology (EUSFLAT 2003). Zittau, Germany, 2003. P. 375-379.

[Kolodyazhniy, 2005] Kolodyazhniy V., Bodyanskiy Ye., Otto P. Universal approximator employing neo-fuzzy neurons. In: Computational Intelligence: Theory and Applications. Ed. by B. Reusch. Berlin-Heidelberg: Springer, 2005. P. 631-640.

[Kaczmarz, 1937] Kaczmarz S. Angenaeherte Ausloesung von Systemen Linearer Gleichungen. Bull. Int. Acad. Polon. Sci, Let. A. 1937. S. 355-357.

[Kaczmarz, 1993] Kaczmarz S. Approximate solution of systems of linear equations. Int. J. Control, 53. 1993. P. 1269-1271.

[Widrow, 1960] Widrow B., Hoff Jr. M. E. Adaptive switching circuits. 1960 IRE WESCON Convention Record, Part 4. N.Y.: IRE, 1960. P. 96-104.

[Kolodyazhniy, 2004a] Kolodyazhniy V., Bodyanskiy Ye. Fuzzy neural network with Kolmogorov's structure. Proc. East West Fuzzy Coll. Zittau/Gorlitz: HS, 2004. P. 139-146.

[Kolodyazhniy, 2004b] Kolodyazhniy V., Bodyanskiy Ye. Fuzzy Kolmogorov's network. In: Lecture Notes in Computer Science. Heidelberg: Springer Verlag, V.3214, 2004. P. 764-771.

[Bodyanskiy, 2005a] Bodyanskiy Ye., Gorshkov Ye., Kolodyazhniy V. Neuro-fuzzy Kolmogorov's network with a hybid learning algorithm. Proc. XI-th Int. Conf. "Knowledge-Dialog-Solution", V.2. Varna, Bulgaria, 2005. P. 622-627.

[Bodyanskiy, 2005b] Bodyanskiy Ye., Kolodyazhniy V., Otto P. Neuro-fuzzy Kolmogorov's network for time-series prediction and pattern classification. In: Lecture Notes in Artificial Intelligence, V.3698. Heidelberg: Springer Verlag, 2005. P. 191-202.

[Bodyanskiy, 2005c] Bodyanskiy Ye., Gorshkov Ye., Kolodyazhniy V., Poyedintseva V. Neuro-fuzzy Kolmogorov's network. In: Lecture Notes in Computer Science, V.3697. Berlin-Heidelberg: Springer Verlag, 2005. P. 1-6.

[Kolodyazhniy, 2006] Kolodyazhniy V., Bodyanskiy Ye., Poedintseva V., Stephan A. Neuro-fuzzy Kolmogorov's network with a modified perceptron learning rule for classification problems. In: Computational Intelligence: Theory and Applications. Ed. by B. Reusch. Berlin-Heidelberg: Springer-Verlag, 2006. P. 41-49.

[Patra, 2002] Patra J.C., Kot A.C. Nonlinear dynamic system identification using Chebyshev functional link artificial neural network. IEEE Trans. on Systems, Man, and Cybernetics, 4, Part B. 2002. P. 505-511.

[Narendra, 1990] Narendra K.S., Parthasarathy K. Identification and control of dynamic systems using neural networks. IEEE Trans. on Neural Networks, 1. 1990. P. 4-26.

Authors' Information

Yevgeniy Bodyanskiy – Doctor of Technical Sciences, Professor of Artificial Intelligence Department and Scientific Head of the Control Systems Research Laboratory; Kharkiv National University of Radio Electronics, Lenina av. 14, Kharkiv, 61166, Ukraine. Tel: +380577021890, e-mail: bodya@kture.kharkov.ua.

Major Fields of Scientific Research: Hybrid Systems of Computational Intelligence

Yevgen Viktorov – PhD Student of Artificial Intelligence Department; Kharkiv National University of Radio Electronics, Lenina av., 14, Kharkiv, 6116, Ukraine. Tel: +380686163429, e-mail: yevgen.viktorov@gmail.com.

Major Fields of Scientific Research: Artificial Neural Networks: Nonconventional Cascade and Multilayer Architectures; Fuzzy Logic; Neuro-Fuzzy Hybrid Systems

DISTANCE MATRIX APPROACH TO CONTENT IMAGE RETRIEVAL

Dmitry Kinoshenko, Vladimir Mashtalir, Elena Yegorova

Abstract: As the volume of image data and the need of using it in various applications is growing significantly in the last days it brings a necessity of retrieval efficiency and effectiveness. Unfortunately, existing indexing methods are not applicable to a wide range of problem-oriented fields due to their operating time limitations and strong dependency on the traditional descriptors extracted from the image. To meet higher requirements, a novel distance-based indexing method for region-based image retrieval has been proposed and investigated. The method creates premises for considering embedded partitions of images to carry out the search with different refinement or roughening level and so to seek the image meaningful content.

Keywords: content image retrieval, distance matrix, indexing.

ACM Classification Keywords: H.3.3 Information Search and Retrieval: Search process

Introduction

For the image retrieval from large scale database traditionally queries 'ad exemplum' are used. There are many approaches developed considering similarities of the query and images in database based on the distance between feature vectors which contain image content descriptors, such as color, texture, shape, etc [Greenspan et al., 2004; Yokoyama, Watanabe, 2007]. The most effort at present is put on the problem of putting the image retrieval on a higher semantic level to perform the search based on the image meaningful content, and not just on image own properties.

There are many metrics developed and widely used in image processing applications: Minkowski-type metric (including Euclidean and Manhattan distances), Mahalanobis metric, EMD, histogram metric, metric for probability density functions, sets of entropy metrics, pseudo metrics for semantic image classification [Rubner et al., 2000; Cheng et al., 2005; Wang et al., 2005]. Yet, by virtue of their limitations these metrics cannot give the desirable results, so a new metric was introduced and extended for considering the embedded partitions and so it was effectively used for the content image retrieval [Kinoshenko et al., 2007]. Due to the embedded structure it will become possible to perform the search with different level of refinement or roughening.

As volume of multimedia databases grows exponentially a great need of means of fast search arises. Many of multidimensional indexing methods used in the field of text retrieval were modified and improved in order to index high-dimensional image content descriptors. Among them X-trees, VA-file and l-Distance approaches are the most promising [Bohm et al., 2001]. However, in case of comparing images as embedded partitions we do not have the features to describe complex objects and only information about distances between them is available, and so-called 'distance-based' indexing methods come to the aid [Chavez et al., 2001; Hjaltason, Samet, 2003]. In this work existing 'distance-based' indexing methods are analyzed and improved and their possible application for the region-based image retrieval is considered.

Theoretical background of distance matrix based content image retrieval

Let $X = \{x_1, x_2, ..., x_n\}$ be a set characterizing images in the database. Each element of this set can represent as an image $B(z), z \in D \subset R^2$ (D is a sensor field of view); as feature vector $p \in R^k$ (Z^k) ; as some combination of image processing results and features (e.g. segmentation, detected edges, shape features).

Further, keeping the generality of consideration, $X \subseteq U$ where U is some universum ensuring introducing of similarity functional (specifically metric), we shall understand as an image database putting aside the indexing. The task consists in the search for correspondence of elements $x_i \in X$ in the best way to the query $y \in U$. Under 'the best way' we shall understand the minimum of distance $\rho(y, x), y \in U$, $x \in X$. Using metric as a similarity criterion provides adequacy of the search result to the query and the triangular inequality makes premises for excluding from the consideration whole sets of images without calculating distances to them.

We shall note, that there are 2 ways to perform the search with limited matches number: either by using preliminary clustering in image or feature spaces, or based on methods analyzing values of pre-calculated distance matrix for all images collection elements. Ex altera parte, all search algorithms can be classified as follows: search of k most similar images ordered according to their similarity; search of the images which differ from the query on not more than given δ (range queries), and combination of these two approaches.

Definition 1. The result of (δ)- search on query $y \in U$ is any element (all the elements) $x_i \in X$, if $\rho(y, x_i) \le \delta$ for given $\delta \ge 0$, called as the search radius.

It is clear that choice of range δ is a non-trivial task. Moreover, choice of rational value δ much depends on the database objects configuration (mutual location regarding the chosen metric). However often the choice of this value is dictated by the practical application, i.e. required extent of image similarity.

Definition 2. The result of (k)- search on query $y \in U$ are elements of set $X^k = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\} \subseteq X$, for which $\forall x_{i_j} \in X^k, \forall x \in X \setminus X^k, \forall y \in U \ \ \rho(y, x_{i_j}) \le \rho(y, x), \ \rho(y, x_{i_j}) \le \rho(y, x_{i_{j+1}}), \ j = \overline{1, k-1}$.

It is necessary to indicate a rather important special case: $y \in X, k = 1$. It means that it is needed to find exact coincidence of query with a database element, i.e. practically identify query and detect image characteristics connected to it, e.g. to identify a person according to his fingerprints.

Definition 3. The result of (δ, k)-search on query $y \in U$ are elements of set $X^m = \{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\} \subseteq X$, $m \le n$, for which $\forall x_{i_j} \in X^m, \forall x \in X \setminus X^m, \forall y \in U, \rho(y, x_{i_j}) \le \delta \ \Rightarrow \rho(y, x_{i_j}) \le \rho(y, x), \rho(y, x_{i_j}) \le \rho(y, x_{i_{j+1}}), j = \overline{1, m-1}$.

We shall call a search successful if there are elements satisfying definitions 1, 2 and 3. Otherwise, the feedback coupling is needed i.e. query object or search parameters (for instance radius δ) refinement what is closely connected to the image presentation by feature descriptions and their matching. We shall emphasize that formally (k)- search is always successful as query refinement decision should always being made on the base of obtained distances analysis and solving task requirements. Notice that each successive search type is more

complicated to solve than the previous one. Thus procedures of handling the (δ)- search are to be used as pre-processing during (k)- search, (δ, k)- search should exploit (δ)- search and (k)- search results.

We shall analyze the possibility of reducing the number N of calculative values $\rho(y, x_i)$ which can be rather computationally expensive especially in the image space. With that purpose a symmetrical matrix of all pairwise distances of all database elements

$$d(\mathrm{X}) = \begin{pmatrix} 0 & \rho(x_1, x_2) & \rho(x_1, x_3) & \dots & \dots & \rho(x_1, x_n) \\ & 0 & \rho(x_2, x_3) & \dots & \dots & \rho(x_2, x_n) \\ & & 0 & \dots & \dots & \dots \\ & & & \dots & \dots & \dots \\ & & & & 0 & \rho(x_{n-1}, x_n) \\ & & & & & 0 \end{pmatrix} \tag{1}$$

is created. Let $y \in \mathrm{U}$ be a query image. We shall fix some image $x^* \in \mathrm{X}$ called pivot object or vantage point or simply pivot and consider the triangular inequality involving one more image $x_i \in \mathrm{X},\ i \in \{1, 2, ..., n\}$ (remind that the distance $\rho(x^*, x_i)$ is known)

$$\rho(y, x_i) \le \rho(y, x^*) + \rho(x^*, x_i) \tag{2}$$

$$\rho(x_i, x^*) \le \rho(y, x^*) + \rho(y, x_i) \tag{3}$$

$$\rho(y, x^*) \le \rho(y, x_i) + \rho(x_i, x^*) \tag{4}$$

From inequalities (1) – (3) it follows that knowing two distances, notably $\rho(y, x^*)$ and $\rho(x^*, x_i)$, it is not hard to obtain low and upper distance bounds

$$\rho(x_i, x^*) - \rho(y, x^*) | \le \rho(y, x_i) \le \rho(y, x^*) + \rho(x_i, x^*) \tag{5}$$

Thus the implication $\forall y \in \mathrm{U},\ \forall x_i, x^* \in \mathrm{X} : \rho(x_i, x^*) \ge 2\rho(y, x^*) \Rightarrow \rho(y, x_i) \ge \rho(y, x^*)$ is true what can be used in the (k)- search. Let exact value of distance $\rho(x^*, x_i)$ be unknown and it can be evaluated as

$$\varepsilon_{min} \le \rho(x^*, x_i) \le \varepsilon_{max} \tag{6}$$

Then if for objects x^*, x_i the inequality (6) is fulfilled the evaluation of low and upper distance bounds

$$max\{\rho(y, x^*) - \varepsilon_{max}, \varepsilon_{min} - \rho(y, x^*), 0\} \le \rho(y, x_i) \le \rho(y, x^*) + \varepsilon_{max} \tag{7}$$

is true. Indeed, according to the triangular inequality rule and taking into consideration (6) we have

$$\rho(y, x^*) \le \rho(y, x_i) + \rho(x_i, x^*) \le \rho(y, x_i) + \varepsilon_{max}$$

then

$$\rho(y, x^*) - \varepsilon_{max} \le \rho(y, x_i) \tag{8}$$

On the other hand, for object s x_i and x^* it is true that $\varepsilon_{min} \le \rho(x_i, x^*) \le \rho(y, x^*) + \rho(y, x_i)$ from where

$$\varepsilon_{min} - \rho(y, x^*) \le \rho(y, x_i) \tag{9}$$

Inequalities (8) and (9) are the low bounds evaluations $\rho(x, x_i)$, and for narrowing the inequality conditions we chose the maximal value. It also should be considered that both evaluations can simultaneously become negative

what is reflected in formula (7). Finally, evaluation of the upper bound $\rho(y, x_i)$ directly follows from the triangular inequality (2) and condition $\rho(x^*, x_i) \le \varepsilon_{max}$.

Let an object x_i be situated 'closer' to the pivot x_1^* than to x_2^*, i.e.

$$\rho(x_i, x_1^*) \le \rho(x_i, x_2^*) \tag{10}$$

Then the following inequality takes place

$$max\{(\rho(y, x_1^*) - \rho(y, x_2^*))/2,\ 0\} \le \rho(y, x_i) \tag{11}$$

Indeed, from the triangular inequality we have $\rho(y, x_1^*) \le \rho(y, x_i) + \rho(x_i, x_1^*)$, hence $\rho(y, x_1^*) - \rho(y, x_i) \le$ $\le \rho(x_i, x_1^*)$. Then $\rho(x_i, x_2^*) \le \rho(y, x_2^*) + \rho(y, x_i)$ is hold. Using condition (10), from the inequalities above we get $\rho(y, x_1^*) - \rho(y, x_i) \le \rho(y, x_2^*) + \rho(y, x_i)$ what with account of the expression $\rho(y, x_1^*) - \rho(y, x_2^*)/2$ possible negativity gives relationship (11).

Metric search models

Let us consider some approaches for (δ)- search support, which make premises for creating effective indexing system for reducing calculation operations on the search stage.

Suppose the distance matrix $d(\mathrm{X})$ is calculated on the pre-processing stage. We shall denote set X as X_0 and its cardinality $card(\mathrm{X}_0) = n_0$. On the initial search stage we calculate the distance $\rho(y, x^{(0)})$ between the searched object y and some object $x^{(0)} \in \mathrm{X}_0$ which is chosen arbitrary or using some criterion. If $\rho(y, x^{(0)}) \le \delta$ we add object $x^{(0)}$ to a resulting set Y. From inequality (5) it directly follows that the distance $\rho(y, x_j^{(0)})$, $j = \overline{1, n_0}$, $x_j^{(0)} \ne x^{(0)}$ does not satisfy the search criterion δ if $|\rho(x_j^{(0)}, x^{(0)}) - \rho(y, x^{(0)})| > \delta$. Applying this criterion to all elements X_0 we get a set $\mathrm{X}_1 = \{x_j^{(0)} \in \mathrm{X}_0 : |\rho(x_j^{(0)}, x^{(0)}) - \rho(y, x_j^{(0)})| \le \delta\} \subseteq \mathrm{X}_0$. If $n_0 > card(\mathrm{X}_1) = n_1$ we shall chose by analogy element $x^{(1)} \in \mathrm{X}_1$ and repeat the procedure of evaluating $\rho(y, x_j^{(1)}) \le \delta$ and distance filtration for all $x_j^{(1)} \in \mathrm{X}_1$, $x_j^{(1)} \ne x^{(1)}$ getting in result $\mathrm{X}_2 \subseteq \mathrm{X}_1$, $card(\mathrm{X}_2) = n_2$. The procedure is carried out recursively till step l when $n_{l-1} - 1 = card(X_l)$, $X_l \subseteq X_{l-1}$. In this case distances $\rho(x_j^{(l)}, y)$, $j = \overline{1, n_l}$ are calculated and evaluated directly. Thus the matches number will be equal to $N(\delta) = l + n_l$ where $n_l = card(\mathrm{X}_l)$.

In practice storing distance matrix $d(\mathrm{X})$ 'in whole' is insufficient due to considerable preprocessing time and especially quadratic memory space requirements. One of the ways to solve this problem is to use its 'sparse' form where for some limited set of paired indices $\{k, l\}$, $0 \le k, l \le n, k \ne l$ value $\rho(x_k, x_l)$ is calculated on the pre-processing stage. The natural demand of the methods choosing a set of given combinations is a compromise between storage expenses and number of distance calculating operations on the search stage, which should tend to $N(\delta) = l + n_l$. At the same time one should note that value $N(\delta)$ is a random one in a sense of being dependent on objects space configuration, pivots $x^{(1)}, x^{(2)}, \ldots, x^{(l)}$ choice order and location of the query object

y. For instance $N(\delta)$ can be decreased if at first the point $x^{(2)}$ and then the point $x^{(1)}$ are chosen. Thus, indexing methods using the sparse form $d(\mathrm{X})$ (and, therefore operating is limited by information volume), theoretically can perform less matching operations than methods on 'complete' distance matrix $d(\mathrm{X})$.

We shall introduce a set of pivots $\mathrm{X}^* = \{x_1^*, x_2^*, ..., x_k^*\}$. From (5) it follows that low distance value is $\rho(y, x_i) \geq \rho_{\mathrm{X}^*}(y, x_i)$ where $\rho_{\mathrm{X}^*}(y, x_i) = \max\limits_{x^* \in \mathrm{X}^*} |\rho(y, x^*) - \rho(x^*, x_i)|$. This is the simplest filtering method for the sparse distance matrix, where $d(\mathrm{X})$ after corresponding index rearrangement of indexes takes form

$$d(\mathrm{X})_{\mathrm{k}}^* = \begin{pmatrix} 0 & 0 & 0 & ... & \rho(x_1, x_{k+1}) & ... & \rho(x_1, x_n) \\ & 0 & 0 & ... & \rho(x_2, x_{k+1}) & ... & \rho(x_2, x_n) \\ & & 0 & ... & ... & ... & ... \\ & & & 0 & \rho(x_k, x_{k+1}) & ... & \rho(x_k, x_n) \end{pmatrix} = \begin{pmatrix} \rho(x_1, x_{k+1}) & ... & \rho(x_1, x_n) \\ \rho(x_2, x_{k+1}) & ... & \rho(x_2, x_n) \\ ... & ... & ... \\ \rho(x_k, x_{k+1}) & ... & \rho(x_k, x_n) \end{pmatrix} \quad (12)$$

We will emphasize that we do not store distances between pivots in $d(\mathrm{X})_{\mathrm{k}}^*$ since $\rho(y, x_j^*), j = \overline{1,k}$ are calculated directly during the search. It also should be noted that the introduced approach can be interpreted as a mapping $(\mathrm{X}, \rho) \to (\mathrm{R}^k, L_\infty)$ and search in k -dimensional space.

Another way of creating index structure without calculating and storing $d(\mathrm{X})$ 'in whole' is to analyze the structure of the data set on the base of distances between objects and then create a partition (possibly nested) of $\mathrm{X} = \{\mathrm{X}^{(j)}\}$, $j = \overline{1,m}$, where

$$\left.\begin{array}{r} \forall j, j' \in \{1, \ldots, m\} : j \neq j' \Rightarrow \mathrm{X}^{(j)} \bigcap \mathrm{X}^{(j')} = \varnothing, \\ \mathrm{X}^{(1)} \bigcup \ldots \bigcup \mathrm{X}^{(m)} = \mathrm{X}, \end{array}\right\} \quad (13)$$

is fulfilled. Here the equivalence relation built on the base of function ρ can be exploited on the search stage: we do not consider those sets $\mathrm{X}^{(j)}$ which element $x_i^{(j)} \in \mathrm{X}^{(j)}$ is not equivalent to the query object y. Here important role plays determination of low and upper bounds (7), (12) $\rho(y, x_i)$ which allow to estimate the distance from y to the elements of $\mathrm{X}^{(j)}$, $j = \overline{1,m}$, separately or using distances to other sets $\mathrm{X}^{(j')}$, $j' = \overline{1,k}$, $j' \neq j$.

Let us consider another way of partitioning X. We shall choose pivot x_j^* which has index j in matrix $d(\mathrm{X})$, and determine the distance to all the rest of the objects $d(\mathrm{X})_{j,1}, d(\mathrm{X})_{j,2}, \ldots, d(\mathrm{X})_{j,n}$. We shall sort values of raw j in ascending order reassign indices $d^*(\mathrm{X})_{j,1}, d^*(\mathrm{X})_{j,2}, \ldots, d^*(\mathrm{X})_{j,n}$ and define the distance to the median object $\rho(x_j^*, d^*(\mathrm{X})_{j,k}) = \mathrm{M}$, $k = \lceil n/2 \rceil$. We shall introduce partition of X into two equivalence classes $\mathrm{X}_{\leq}$ and $\mathrm{X}_{>}$ where $\mathrm{X}_{\leq} = \{x_i \in \mathrm{X} : \rho(x_j^*, x_i) \leq \mathrm{M}\}$, $\mathrm{X}_{>} = \{x_i \in \mathrm{X} : \rho(x_j^*, x_i) > \mathrm{M}\}$. In this case on the search stage under $\rho(x_j^*, y) \leq \mathrm{M} - \delta$ it is necessary to search only class $X_{\leq}$ and under $\rho(x_j^*, y) > \mathrm{M} + \delta$ only class $\mathrm{X}_{>}$. Thus producing such a partition can allow us excluding from the consideration half of the set elements. But in the worse case when $\mathrm{M} - \delta < \rho(x_j^*, y) \leq \mathrm{M} + \delta$ is true, search algorithm has to consider both

branches $\mathrm{X}_{\leq}$ and $\mathrm{X}_{>}$. It should be emphasized that this method is to be used recursively.

We shall introduce into consideration equivalence relation based on the closeness to the pivot. As before let us chose set $\mathrm{X}^* = \{x_1^*, x_2^*, ..., x_k^*\}$. Then elements x_j^* produce partition $\mathrm{X} = \{\mathrm{X}_s^*\}$, $s = \overline{1,k}$ such that

$$\mathrm{X}_s^* = \{x_i \in \mathrm{X} : \forall t = \overline{1,k}, t \neq s \; \rho(x_i, x_s^*) < \rho(x_i, x_t^*)\} \tag{14}$$

Such criteria coincide with definition of Voronoi cell in Euclidean space. Partition introduced this way allows to use evaluation (11) of the low distance bound. Indeed, the criterion $\rho(x_i, x_s^*) \leq \rho(x_i, x_t^*)$ for $x_i \in \mathrm{X}_s^*$, $t \neq s$ is fulfilled by (14). Then from (11) for k pivots it follows that the low bound of $\rho(y, x_i)$ for $x_i \in \mathrm{X}_s^*$ will be

$$\rho_{min}^{(1)}(s) = max\Big\{ \max_{t=\overline{1,k}(t\neq s)} \{(\rho(y, x_s^*) - \rho(y, x_t^*)/2)\}, 0\Big\} \tag{15}$$

Let $\varepsilon_{max}(s) = \max\limits_{x_i \in \mathrm{X}_t^*} \rho(x_s^*, x_i)$ be a cover radius for the partition X_s^*. Then according to (7) the evaluation low distance bound $\rho_{min}^{(2)}(s) = \rho(y, x_s^*) - \varepsilon_{max}(s)$ is true. Let minimal and maximal distance evaluations between partitions be calculated on the preprocessing stage for $s, t = \overline{1,k}$.

$$\varepsilon_{min}(s,t) = \min_{x_i \in \mathrm{X}_t^*} \rho(x_s^*, x_i), \; \varepsilon_{max}(s,t) = \max_{x_i \in \mathrm{X}_t^*} \rho(x_s^*, x_i)$$

Then $\varepsilon_{min}(s,t)$ and $\varepsilon_{max}(s,t)$ under $s \neq t$ are evaluations of ε_{min} and ε_{max} distance $\rho(x_s^*, x_t^*)$ in (7). For $s = t$ $\varepsilon_{max}(s,s) = \max\limits_{x_i \in \mathrm{X}_s^*} \rho(x_s^*, x_i) = \varepsilon_{max}(s)$. Hence it is legitimate to claim that evaluation $\rho_{min}^{(2)}(s)$ is a special case of evaluation $\rho_{min}^{(3)}(s) = max\{\max\limits_{t=\overline{1,k}} \{\rho_{min}^{(3)}(s,t)\}, 0\}$ where

$$\rho_{min}^{(3)}(s,t) = max\{\rho(y, x_s^*) - \varepsilon_{max}(s,t), \varepsilon_{min}(s,t) - \rho(y, x_s^*)\} \tag{16}$$

The final maximal low bound of distance $\rho(y, x_i)$, $x_i \in \mathrm{X}_s^*$ is defined as $\rho_{min}(s) = max\{\rho_{min}^{(1)}(s), \rho_{min}^{(3)}(s)\}$.

To make search algorithm on partition index structure more optimal, we propose to carry out the following steps during the search. Let E be a set of not processed pivots x_s^* which form corresponding regions X_s^*, $s = \overline{1,n}$, and T be a set of regions X_s^* which cannot be dropped by partition index structure. Then iteratively get the first pivot $x_s^* \in \mathrm{E}$, calculate $\rho(x_s^*, y)$, estimate the low bound of all $\rho(x_t^*, y)$, $x_t^* \in \mathrm{E}$ using (16) and remove from E those $x_t^* : \rho_{min}^3(s,t) > \delta$ or put corresponding to x_t^* set X_t^* to T. Also after each iteration remove considered pivot x_s^* from E, thus iterative procedure stops when some pivots $x_s^*, s = \overline{1,t}$ are eliminated by (16) and distance to the rest of pivots is calculated. After it we argue to apply (15) criteria for all pairs $x_s^*, x_t^* \in \mathrm{E}$, $s \neq t$ since a range $(\varepsilon_{min}(s,t), \varepsilon_{max}(s,t))$ could be large and therefore could produce large low bound of $\rho(y, x_i)$, $x_i \in \mathrm{X}_t^*$ or $x_i \in \mathrm{X}_s^*$.

Results of experiments and conclusion

A number of tests have been performed on set of points in R^2 space with L_2 metric. We used two configurations of data distribution: uniform and with formed clusters. We implemented the following index structures to preprocess the initial data set: i) indexation on full matrix (1); ii) indexation on sparse matrix (12); iii) indexation via binary tree with branches $\mathrm{X}_{\leq}$ and $\mathrm{X}_{>}$; iv) indexation via compact partition using criteria (14) and iterative procedure which exploits (15) and (10) low bounds.

The purpose was to calculate the matches' number during the search on uniform and clusters distribution of objects and its dependency on the query object position and data set configuration.

Below the results of the tests with the following experiment parameters are presented. The data for uniform distribution consisted of 1024 points, for the clusters one there were 16 clusters, with cluster cardinality mean equal to 64 and variance equal to 10 of (1024 elements in total). Both sets of points coordinated were within the range [0;256] (data square here and after). Variance of single cluster points location was set to 6 for both coordinates. It was allowed that clusters could overlap. We used $k = \sqrt{n}$ parameters for indexation ii), and created one-level partition with $k = \sqrt{n}$ number of pivots in indexation iv).

First experiment examined dependency of the index structure on the position of the query object. We generated uniform and cluster data structures, created all index structures and randomly chose query object from the data square 500 times, tracking the number of matches of each data structures for all queries. We then calculated the mean and variance of matches count for each index structure (Table 1).

Table 1. Dependency of the matches' number on different data configurations

Data configuration	Full matrix		Sparse matrix		Partition search		Binary search	
	μ	σ	μ	σ	μ	σ	μ	σ
uniform	23.56	4.82	50.98	4.58	107.78	36.18	90.08	23.7
clusters	30.63	29.73	55.19	26.27	71.2	71.39	74.362	48.49

As it was expected indexation which exploits a full distance matrix i) outperforms the other ones while indexation on sparse matrix ii) keeps the second place. Indexations iii) and iv) have approximately equal efficiency. We can claim that when objects of database tend to form clusters the number of matches does not vary dramatically for all indexing methods, they only differ notably on variance and therefore in some cases performance of methods iii) and iv) can take much more time than it was expected. On the other hand, when distribution of objects is uniform then indexation i) perform 4 times better results than iii) and iv) while indexation ii) performs 2 times better results.

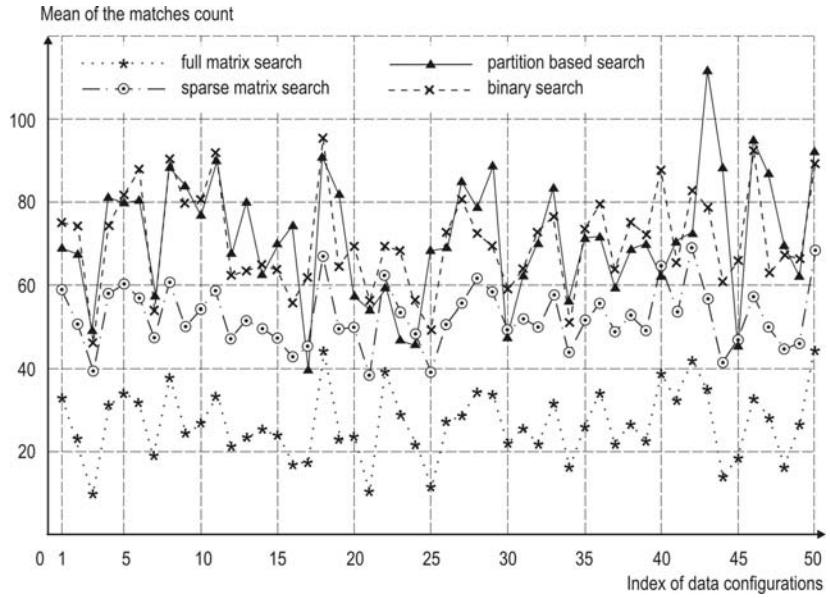


Figure 1. Results of matches count dependency on change of cluster data configuration

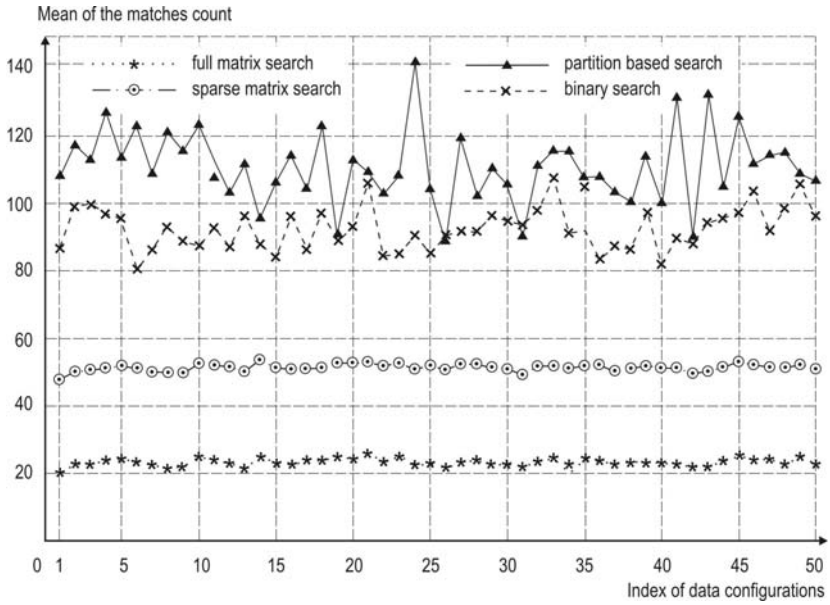


Figure 2. Results of matches count dependency on change of uniform data configuration

In the second experiment we tried to track dependency of index structures efficiency on different data configurations. We created 50 random configurations of uniform and cluster data distributions and ran routines of the first experiment changing query object position 20 times. We found out that statistics of the indexations efficiency was almost the same compared to first experiment results, only variance grew a bit. On Figures 1, 2 an average number of distance computations on 50 iterations for indexation algorithms i) – iv) is given for uniform and cluster data configuration. From here we can conclude that only index structures i) and ii) can guarantee almost/rather constant low bound of matches count on uniformly distributed data set solely.

Conclusion

We have proposed a novel indexing structure using sparse distance matrices for the image search with queries 'ad exemplum' which considering embedded partitions of the images. The experimental exploration of the method has proved it to be fast and efficient. The future work will be directed for investigation of the pivots choice method and possibility to use clustering methods for distance matrices analysis which would provide effective using of the partition metric for content image retrieval.

Bibliography

[Greenspan et al., 2004] H. Greenspan, G. Dvir, Y. Rubner. Context-Dependent Segmentation and Matching in Image Database. Computer Vision and Image Understanding. Vol. 93, No. 1, 2004, pp. 86-109.

[Yokoyama, Watanabe, 2007] T. Yokoyama, T. Watanabe. DR Image and Fractal Correlogram: A New Image Feature Representation Based on Fractal Codes and Its Application to Image Retrieval. In: Lecture Notes in Computer Science. Berlin Heidelberg: Springer-Verlag. Vol. 4351, 2007, pp. 428-439.

[Rubner et al., 2000] Y. Rubner, C. Tomasi, L.J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. International Journal of Computer Vision. Vol. 40, No 2, 2000, pp. 99-121.

[Cheng et al., 2005] W. Cheng, D. Xu, Y. Jiang, C. Lang. Information Theoretic Metrics in Shot Boundary Detection. In: Knowledge-Based Intelligent Information and Engineering Systems. Khosla R., et al. (eds.). Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg. Vol. 3683, 2005, pp. 388-394.

[Wang et al., 2005] D. Wang, X. Ma, Y. Kim. Learning Pseudo Metric for Intelligent Multimedia Data Classification and Retrieval. Journal of Intelligent Manufacturing. Vol. 16, No 6, 2005, pp. 575–586.

[Kinoshenko et al., 2007] D. Kinoshenko, V. Mashtalir, V. Shlyakhov A Partition Metric for Clustering Features Analysis // International Journal 'Information Theories and Applications'. Vol. 14, No 3, 2007, pp. 230-236.

[Bohm et al., 2001] C., Berchtold S., Keim D. Searching in High-Dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. ACM Computing Surveys. Vol. 33, No. 3, 2001, pp. 322–373.

[Chavez et al., 2001] Chavez E., Navarro G., Baeza-Yates R., Marroquin J.L. Searching in Metric Spaces. ACM Computing Surveys. Vol. 3, No. 3, 2001, pp. 273–321.

[Hjaltason, Samet, 2003] Hjaltason G., Samet H. Index-driven Similarity Search in Metric Spaces. ACM Transactions on Database Systems. Vol. 8, No. 4, 2003, pp. 517-580.

Authors' Information

Dmitry Kinoshenko – Researcher; Kharkov National University of Radio Electronics, Lenin ave. 14, Kharkov 61166, Ukraine; e-mail: Kinoshenko@kture.kharkov.ua

Vladimir Mashtalir – Professor, Dean of Computer Science faculty; Kharkov National University of Radio Electronics, Lenin ave. 14, Kharkov 61166, Ukraine; e-mail: Mashtalir@kture.kharkov.ua

Elena Yegorova – Senior researcher; Kharkov National University of Radio Electronics, Lenin ave. 14, Kharkov 61166, Ukraine; e-mail: Yegorova@kture.kharkov.ua

ANALOGICAL MAPPING USING SIMILARITY OF BINARY DISTRIBUTED REPRESENTATIONS

Serge V. Slipchenko, Dmitri A. Rachkovskij

Abstract: We develop an approach to analogical reasoning with hierarchically structured descriptions of episodes and situations based on a particular form of vector representations – structure-sensitive sparse binary distributed representations known as code-vectors. We propose distributed representations of analog elements that allow finding correspondence between the elements for implementing analogical mapping, as well as analogical inference, based on similarity of those representations. The proposed methods are investigated using test analogs and the obtained results are as those of known mature analogy models. However, exploiting similarity properties of distributed representations provides a better scaling, enhances the semantic basis of analogs and their elements as well as neurobiological plausibility. The paper also provides a brief survey of analogical reasoning, its models, and representations employed in those models.

Keywords: analogy, analogical mapping, analogical inference, distributed representation, code-vector, reasoning, knowledge bases.

ACM Classification Keywords: I.2 ARTIFICIAL INTELLIGENCE, I.2.4 Knowledge Representation Formalisms and Methods, I.2.6 Learning (Analogies)

Introduction

Example-based reasoning is a powerful approach extensively used by humans for everyday and expert reasoning with incomplete, inaccurate, contradictory input information. For many domains and problems it is essential to rely on examples that include complex relational structures (systems of hierarchical relations). Such information is also present in some knowledge bases and ontologies. Analogical reasoning is a kind of example-based reasoning with special attention paid to the similarity of relational structures, i.e. systems of hierarchical relations between objects, situations, episodes, domains, etc. [Gentner, 1983; Hummel & Holyoak, 1997].

Analogical thinking is one of the most commonly encountered and important cognitive processes. Analogy is closely related to such phenomena as memory, remembering, perception, learning, conceptual change, knowledge formation, similarity, etc. That is why a lot of research is devoted to its study and modeling. For an introduction, see sections Processes of Analogical Reasoning and Models of Analogical Reasoning below, as well as [Holyoak & Thagard, 1989; Hummel & Holyoak, 1997; Markman, 1997; Thagard et al., 1990; Gentner & Markman, 2003; Gentner & Markman, 2000] and references therein. The work of analogy researchers continuously gives new interesting results about the peculiarities of human intellectual activity (for some recent examples, see [Gentner et.al., 2009; Taylor & Hummel, 2009]).

Historically, most advanced models employed symbolic-localist representations, see the Internal Representations and Similarity section below. Such representations are able to deal with the hierarchically structured descriptions of episodes or situations that are used as analogs in analogical reasoning. However, those models have a number of drawbacks that are often attributed to the shortcomings of the symbolic-localist representations. They are computationally complex and so do not scale well, use hand-crafted representations, have poor account of semantics, memory and generalization, are not concerned with neurobiological plausibility.

To some extent, those drawbacks could be overcome by distributed (vector) representations. However, distributed representations were considered incapable to represent structure (for a review, see [Rachkovskij &Kussul 2001] and references therein). Some time ago, structure-sensitive distributed representations have appeared where binding operations are used for structure representation, and have been immediately applied to modeling analogical reasoning [Eliasmith & Thagard, 2001; Hummel & Holyoak, 1997; Kanerva, 1998; Plate, 2003]. Previous attempts to model analogical reasoning with structure-sensitive distributed representations mainly demonstrated "proof of concept": they worked with simple analogies and used distributed representations only at some model stages, see also the Models of Analogical Reasoning section below.

In analogical reasoning, the following basic processes are usually considered and modeled: retrieval (finding in memory most close base analog given a target situation), mapping (finding correspondences between the elements of two analogs), and inference (knowledge transfer from base analog to target). In this paper, we describe a particular scheme for distributed representations and apply it to modeling of analogical mapping and inference. Mapping is established on the basis of similarity of distributed representations. We show that the distributed representation scheme of and its application to analogical reasoning are simple in use, have low computational complexity, and provide the same result as advanced traditional models.

The rest of the paper is organized as follows. In the first three sections we provide a brief survey of the analogical reasoning processes, local and distributed internal representations and similarity, and the models of analogy. Then our representational scheme is introduced and representations of analog elements are described. In the next sections we describe mapping and inference techniques on the basis of the introduced representations, and study them in the experiments. Future research directions are outlined together with the discussion.

Processes in Analogical Reasoning

Analogical reasoning theories deal with the following processes [Gentner, 1983; Kokinov & French, 2003; Falkenhainer et.al., 1989; Gentner & Markman 2003; Thagard et.al. 1990; Holyoak & Thagard, 1989; Hummel & Holyoak, 1997].

Building representations. Analogs are considered as hierarchically structured descriptions of situations and domains.

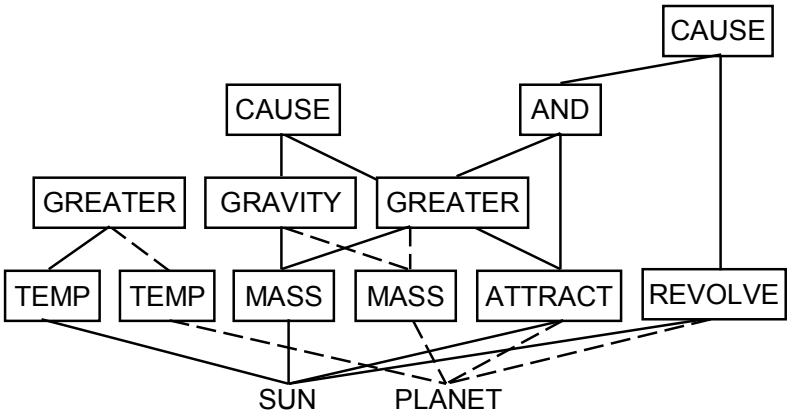


Fig. 1. Graph sketch description of the Solar System analog. Re-representation code-vector of the Sun is constructed from the roles marked by the solid lines

Descriptions. Analog descriptions (Fig. 1) include descriptions of their elements – entities, attributes and relationships. Objects are entities of subject domains (for example, the Sun, Planet, etc.). Attributes describe

properties of the objects (for example, mass, temperature, etc.). Relations determine relationships between the elements of analogs (for example, attract, more, cause, etc.). Arguments of relations may be objects, attributes and other relations. Attributes may be considered as relations with single argument, so thereafter we will mainly speak about relations.

Are such commonly used analog descriptions appropriate for modeling human analogical reasoning and using in artificial reasoning systems? Probably, only to some degree, – if one considers a rather conservative expert estimate of recent progress in both computer reasoning systems and models of human analogical reasoning. The reason may be explained by the fact that nobody knows how such information is represented in human brain whose ability to use analogies we are trying to reproduce.

Humans describe analogs using natural language or pictures, whether in everyday life or participating in experiments. The usual way to convert verbal or visual descriptions to the formal descriptions of analogs (like in Fig. 1) is to do it manually, by human-expert. The resulting formalized description itself lacks a lot of information present in the initial "natural" descriptions. On the other hand, the formalized descriptions carry a lot of information for humans (especially for the experts who work in this area and encode those descriptions) because of the knowledge associated with the formal description in their brains. However, the system or model dealing with the description lacks such common sense knowledge, at least at the level comparable to a human. Also, the process of manual construction of the formalized description is very laborious and expensive.

So, the question is how to form at least those formalized descriptions, thus overcoming, to some degree, the knowledge acquisition bottleneck. Current efforts in this direction are dedicated to the automation of this process. [Mostek, Forbus, & Meverden, 2000] consider techniques for automatically constructing representations of analogs from knowledge bases. This work is very interesting and allows a more efficient use of knowledge bases and reasoning. However, a much more essential and difficult problem is automatic knowledge acquisition from texts.

The manual transformation of textual descriptions to the formal analog descriptions by humans-experts exploits their previous knowledge of language, of the world, of the particular analogy, of the task to be solved by analogy, and, often, of the analogy model that will work with the produced formalized description. All this knowledge is lacking in automatic systems, making the problem very challenging – one of the Holy Grails of AI. Knowledge discovery in texts or Internet became a hot topic in 90s [Hahn & Schnattinger, 1998; Schubert, 2002; Agichtein & Gravano, 2000]. At present, some progress has been made in extracting simple objects and relations, but not complex episodes. Or, a human editorial staff is used to validate knowledge as part of the process. Recent advances in this direction are known as Learning by Reading [Bobrow et al., 2009; Forbus et al., 2007; Kim & Porter, 2009], but there still is insufficient progress in the formation of the episodes at the knowledge base level of complexity. [Forbus et al., 2008] consider generating analogical representations from sketches.

Another important question is how to enhance the semantic basis of representations. Some approaches exist that create representations of objects from similarity data obtained in experiments, e.g. with humans [Navarro & Lee, 2003; Shepard, 1962; Tenenbaum, 1996]. Another approach is to form representations as "context vectors" using frequencies of joint occurrences in text corpus (e.g., [Ramscar & Yarlett, 2003; Misuno, Rachkovskij, & Slipchenko, 2005; Sahlgren, Holst, & Kanerva, 2008]. However, those approaches have several shortcomings. The representations produced may be still far from those of the brain. Also, context-dependency of representations is not considered within those approach, otherwise that single representation set is produced per experimental domain or text corpus. So far, mainly simple vector representations are formed, much simpler than the episodes used in analogical research.

Once the formalized descriptions are formed, they are usually used in the models of analogy without associated knowledge. Attempts are made to eliminate this drawback in the models of analogy as follows. WordNet is used

in [Thagard et al., 1990; Holyoak & Thagard, 1989] to estimate similarity between the entities. [Forbus et al., 2008] use Cyc for generating representations and reasoning in the domain of interactive sketch understanding. [Forbus, Mostek, & Ferguson, 2002] consider integration of analogical and first-principles reasoning.

Internal representations. Given input descriptions of analogs, they should be represented in the internal format. For human analogical reasoning, those are internal brain representations still unknown to us. For modeling, those are internal models representations. The purpose of this transformation to internal representation can be: compatibility with the format and methods of model; performance issues; a more adequate representation of some aspects poorly supported by the input description (e.g., semantic aspects).

Internal representations may differ from the input descriptions in a varying degree. For example, the relational descriptions of analogs mentioned above (Fig. 1) can immediately be used in the symbolic models. Or, they can be rather straightforwardly transformed to localist connectionist representations. Transformations to fully distributed representations are more radical (see the Internal Representations and Similarity section below). Re-representation may also occur in the process of model functioning (see the end of this section).

Retrieval and memory. When the analogical episodes are represented internally, and the target episode is given at the input, the task is to find the closest analog. Usually retrieval models find the closest analog by similarity (see [Rachkovskij & Slipchenko, under revision] and references therein). Both semantic/featural content and relational/propositional organization of analogs are important for the estimation of similarity, though the structural similarity in analogical retrieval is considered less important than in mapping. Retrieval returns the closest analog, or several candidates. Retrieval models are usually computationally expensive, since they some manner handle the whole base of known analogs (albeit with optimizations).

The issue related to retrieval is where and how the episodes are stored. In humans, the obvious answer is long term memory. Most of the analogical retrieval models ignore neurobiological aspects of memory and store isolated representations of episodes. Those representations do not change in the process of storage and recall retrieval.

Few analogical models consider generalization [Kuehne et al., 2000; Hummel & Holyoak, 2003], but again the generalized representations are stored separately and memory processes are not considered. We believe that a very promising direction is to study storage, retrieval, and generalization in a distributed associative memory [Frolov, Rachkovskij, & Husek 2002, Frolov et al., 2007]. However, we know of no work in this direction, except for [Kokinov & Petrov, 2001] that uses a kind of associative memory to context-sensitive reconstruction of the stored episodes in interaction with mapping.

Mapping. When both the base and the target analogs are known, the task is to find their corresponding elements. This is performed by mapping that is considered as the most important part of analogical reasoning. Mapping pays more attention to structure similarity, than retrieval (see also the Models of Analogy section). Structural constraints on mapping include parallel connectivity and one-to-one mapping [Markman & Gentner, 2000]. Parallel connectivity requires the arguments of corresponding predicates to be placed into correspondence. One-to-one mappings limit any element in one analog to correspond to at most one element in the other analog.

Inference. After mapping is established, knowledge from the base analog has to be transferred to the target. As an outcome of the mapping process, we obtain correspondences between the elements of analogs. So, we observe relations present in the base but absent in the target. They are the candidate inferences about the target analog projected from the base.

To make inferences, copying with substitution and generation is usually used [Holyoak, Novick, & Melz, 1994; Markman, 1997]. It takes any element in the target for which there is a corresponding element in the base and copies to the target all relational structure connected to the element in the base. In this structure, the elements

copied from the base are replaced by their corresponding elements of the target. If some entities are absent in the target but are required to complete the structure from the base, they are generated anew. All this looks as a symbolic operation, and hardly lends itself to a more natural implementations. Most of the models use some kind of implementation of this process for analogical inference [Hummel & Holyoak, 1997; Hummel & Holyoak, 2003].

Because analogy is not a deductive mechanism, these candidate inferences are only hypotheses and must be evaluated and checked (see e.g. [Gentner & Colhoun, 2010] and references therein).

Re-representation. In the process of analogy-making, some modelers of analogical reasoning allow re-representation of initial internal representations – e.g., when the initial representations are not adequate for the task [Yan, Forbus & Gentner, 2003]). In [Rachkovskij, 2004] identical representations were substituted for some mapped analogical elements in order to ensure the correct mapping of the rest of the episode. [Yan, Forbus, & Gentner, 2003] proposed a theory of re-representation in analogical mapping. They consider the problems of detecting opportunities for re-representation, generating re-representation suggestions based on libraries of general methods, and controlling the re-representation process. In "fluid analogy" models ([Kokinov & Petrov, 2001; Hofstadter, 1995; French, 1995], see also the Models of Analogy section) representations are formed depending on the context of particular episodes, and representation construction and mapping proceed in parallel and are inseparable.

As we have seen, representation and similarity are the most important factors of analogy. Let us consider them in some more detail.

Internal representations and similarity

Implementation and efficiency of operations on relational structures, which are required by the models of analogical reasoning and other cognitive models, depend essentially upon the representation scheme employed. Estimation of similarity of representations is one of the most important operations with the models. Retrieval of relevant analogs is mainly similarity-based. Mapping can also be considered as finding and purring into correspondence the most similar elements of analogs.

Processing of analogies requires taking into account both types of representation and similarity – structural and semantic. Structural similarity reflects how the elements of analogs are arranged with respect to each other. It is based on the notion of "structural consistency" [Falkenhainer et al., 1989; Gentner & Markman, 2003] or "isomorphism" [Thagard et al., 1990; Hummel & Holyoak, 1997; Eliasmith & Thagard, 2001]. Analogs are also matched by the "surface" or "superficial similarity" [Gentner 1983, Forbus et al., 1995] based on common analog's elements or a broader "semantic similarity" [Thagard et al., 1990; Hummel & Holyoak, 1997; Eliasmith & Thagard, 2001], based on some prior semantic similarity (e.g., joint membership in a taxonomic category) of those elements.

In early similarity models, objects or some other items were represented as points in a space of some dimensions [Shepard, 1962], or by feature sets [Tversky, 1977]. Similarity was estimated as the distance between those points or by the set-theoretical operations. Note that both mentioned representations can be considered as vector ones, with gradual or binary components, and so similarity of items can readily be calculated by some combination of vector(dis)similarity measures (dot product, Euclidean distance, etc.). Since the early naive versions of the vector representations did not represent relations, they were considered incompatible with the inherent structure of real world objects and situations.

Instead of repairing those drawbacks of naive vector representations, novel emerged similarity accounts were based on structured representations. They used graph-like representations, where elements of varying complexity are represented as vertices of the graph (Fig. 1). Edges provide information about the structural

organization of elements (how the elements are grouped or bound together). However, processing of graphs is complicated compared to vectors, and semantic, feature similarity of elements is rather difficult to take into account when processing graphs.

We think that the drawbacks of both feature/vector and structure/graph similarity models and representations stem from their being instances of symbolic and localist representations [Thorpe, 2003; Page, 2000]. In such representations, each element, such as a feature or attribute, entity, relation, episode, etc., has a representation in the form of symbol, node, or single vector component. Each new item or combination of items requires allocating a new node or memory location. Such representations are semantically "brittle" (all-or-none similarity). It is necessary to decide for each input item, whether it is “sufficiently” similar to (and so can be encoded by) some existing symbol or node, or requires a new representation. Thus, the system's knowledge does not generalize naturally to newly represented situations. Also, symbol or higher-level node is just a label, and so it does not carry immediate information about the structural organization or semantics of its elements. Sequential link or pointer following is needed to traverse the structure corresponding to the vertex, and to retrieve its semantic description, if any.

Comparison of structures is difficult, since it requires alignment and finding correspondences between element substructures. Popular graph similarity measures based on sub-graph isomorphism are very expensive computationally (NP-complete) and so practically inapplicable to large graphs. However, isomorphism does not take into account the semantic similarity of objects and does not meet the requirements that humans apply to analogical similarity [Markman & Gentner, 2000]. So, some heuristic measures are proposed in the models instead. Nevertheless, for localist and symbolic representations, an estimation of structure similarity requires computationally very expensive procedures that include construction and processing of virtual or real constraint-satisfaction networks (as in [Falkenhainer et al., 1989; Holyoak & Thagard, 1989; Keane et al., 1994], see also the Models of Analogical Reasoning section below).

Here, we are especially interested in fully distributed representations [Thorpe, 2003], where any item (e.g., feature or attribute, entity, relation, analog, etc.) is represented as a vector of fixed dimensionality. Each vector component does not have clear semantics, i.e. does not correspond to single item, in contrast to localist representations. It is natural to represent similar items by correlated distributed representations and to estimate their similarity by dot product. A high information capacity is provided by the possibility to represent exponentially many items by different vectors of the same dimensionality N (e.g., (M from N) binary vectors vs N for localist representations with single component per item).

However, many thought that distributed representations cannot represent nested (recursive) structures because of the superposition catastrophe (losing the information concerning item arrangements in structures, see [Rachkovskij & Kussul, 2001] for discussion and references). In fact, a number of connectionist schemes have been proposed capable of forming recursive structured representations (e.g., RAAMs [Pollack, 1990], tensor products [Smolensky, 1990], etc. – for a review and comparison, see [Plate, 2000, 2003; Browne & Sun, 2001; Rachkovskij & Kussul, 2001] and references therein). In those schemes, binding procedures (an analog of grouping brackets in symbolic representations) were proposed to avoid the superposition catastrophe. In RAAMs [Pollack, 1990], binding is realized using the weight matrix formed by training a multilayer perceptron. This requires a challenging retraining using all the items when new items to be bound are introduced. In tensor products [Smolensky, 1990], the dimensionality of the resulting tensor grows with the number of bound code-vectors. In dynamic bindings [Shastri & Ajjanagadde, 1993; Hummel & Holyoak, 1997], the representations are bound by synchronous activation, but vector measures of similarity cannot be immediately employed for resulting dynamic representations.

Some time ago, a number of schemes for structure-sensitive fully distributed representations with vectors of various formats have appeared, where binding operations that does not change dimensionality are used and bindings are produced on-the-fly (without any training). Holographic Reduced Representations (HRRs) of [Plate, 2003] use real-valued vectors and circular convolution for binding. Binary Spatter Codes (BSCs) of [Kanerva, 1996] use Boolean vectors and component-wise exclusive-or. [Gayler, 1998] uses vectors with components from {−1, +1} and component-wise multiplication. Associative-Projective Neural Networks (APNNs) use sparse binary vectors with {0,1} components and special context-dependent thinning (combination of vector component-wise AND, OR, and permutations) procedure for binding (Rachkovskij and Kussul 2001, see also this paper below). For a more detailed comparison of those binding schemes, see [Browne & Sun, 2001; Rachkovskij, 2001; Rachkovskij & Kussul, 2001].

Similarity of resulting bound representations is influenced both by the set of components and their arrangements. Thus, similar structures are encoded by similar code-vectors. So, it is not necessary to search for the match between the elements of two structures in order to estimate their overall similarity by dot product. As discussed above, this property is important for many cognitive models and systems, including those of analogical reasoning considered in the following section.

Models of analogical reasoning

Let us consider models of analogical reasoning that employ different internal representations and methods of their processing, with the emphasis on mapping models. Since, till recently, representation and processing of structures were achievable only with hierarchical symbolic or localist representations, it is natural that those representations are used in the most influential computational models of analogy.

Symbolic and localist models. Structure Mapping Theory (SMT) [Gentner, 1983] is probably the first and best-known theory of analogical reasoning that explicitly underlined the importance of structure similarity defined as the common systems of relations between analogs. In this theory, the processes involved in analogy-making – representation-building, retrieval, mapping, etc. – are separated from one another. The model of mapping in SMT is instantiated by Structure Mapping Engine (SME) [Falkenhainer et al., 1989].

At the input, SME takes propositional representations of two analogs (symbolic version of representation of the type shown in Fig.1). It uses a local-to-global alignment process to determine correspondences between their elements as follows. First, SME finds all possible local correspondences between the elements of the two representations by putting into correspondence identical relations, as well as the arguments of identical relations. Then, SME produces one or few globally consistent interpretations by integrating consistent local correspondences.

All interpretations generated by SME strictly impose the structural constraints of parallel connectivity and one-to-one mapping ([Markman & Gentner, 2000] and the Processes in Analogical Reasoning section above). Finally, SME calculates scores reflecting the quality of each interpretation using an algorithm that favors mappings preserving interconnected, higher-order relational structure (systematicity principle). The interpretation with the highest score is selected as the preferred interpretation.

SME's drawback, as a symbolic model, is a rather poor account of semantic similarity. Also, SME structure matching is computationally expensive. It makes prohibitive using it during retrieval for a structure-sensitive comparison of the input to each of (many) potential analogs stored in memory. So, SMT-based model of retrieval, MAC/FAC [Forbus et al., 1995] uses a two-stage process. The first stage MAC uses feature vectors of analogs and a computationally cheap vector similarity measures to select the candidates based on the surface similarity

only. The second stage FAC uses SME to take structure of the candidates into account. The drawback of MAC/FAC is that top-scored candidates of MAC may miss some useful analogs.

The Analogical Constraint Mapping Engine (ACME; [Holyoak & Thagard, 1989] is a localist connectionist model that determines analogical mappings using a parallel constraint satisfaction network. ACME takes the input representations of two analogs and constructs a network where each node corresponds to a possible match of analog elements, and the between-node connections represent the constraints to be satisfied. Inhibitory connections are used between competing hypotheses, while excitatory connections are used between consistent hypotheses. A spreading-activation relaxation algorithm runs until the network settles to a final state. In this state, active nodes represent winning matches corresponding to the maximally consistent mapping hypothesis.

Unlike SME, ACME relies not only on structural information, but also takes into account semantic and pragmatic constraints. Unlike strict constraints posited by SME, all those types of constraints are soft and together drive the system to decision. Pragmatic and semantic constraints, as well as representations of analogs, are hand crafted. ACME can map structures that do not involve semantic similarities, and this may be considered as its shortcoming. Also, it can produce mappings that violate the structural constraint of one-to-one correspondence, that can lead to inconsistencies in analogical inferences [Markman, 1997]. Usually, it is even more computationally expensive than SME. So, the retrieval model ARCS [Thagard et al., 1990] is a two-stage scheme, as well as MAC/FAC. It uses ACME, but with greater emphasis on semantic constraints.

Connectionist Analogy Builder (CAB) [Larkey & Love, 2003] is a localist connectionist model that takes as input two directed graphs and determines mappings via a dynamic process of interactive activation among correspondences that are represented as network nodes. Nodes are excited by other nodes that represent parallel correspondences. Nodes also compete for excitation and are inhibited by other nodes. Resulting activations establish one-to-one mappings. CAB makes time-course predictions as well as predictions concerning the role of working memory in the comparison process. Besides the common shortcomings of all localist models, its specific drawbacks are not clear.

The Incremental Analogy Machine (IAM) [Keane, Ledgeway, & Duff, 1994] is an incremental model of analogical mapping that takes into account the effects of the order of material presentation. A problem with IAM is that people's comparisons are incremental, but in a different way that in IAM [Larkey & Love, 2003]. Another limitation of IAM is that the fully serial nature of its processing prevents scaling up. Also, IAM can map unnatural analogies that do not involve semantic commonalities.

In solving analytical problems by analogy, Gladun and his colleagues [Gladun, 2000; Gladun et al., 2000] used structural-attributive models. They employ localist connectionist representations, such as growing pyramidal networks, where generalization of analogs is used as a prerequisite for inference. Classification of new objects is implemented by comparison of their representations to the class concepts. The model is mainly elaborated for classification tasks, where only the class label is transferred, not relational systems.

Unlike the models discussed above, the models such as CopyCat [Hofstadter & Mitchell, 1994; Hofstadter, 1995], Tabletop [French, 1995], Associative Memory Based Reasoning (AMBR) [Kokinov, 1988; Kokinov, 1994; Kokinov & Petrov, 2001] consider representation-building and reasoning as parallel, interacting sub-processes. In the process of analogy-making, the interactions between micro-agents construct in working memory context-sensitive representations of the episodes using semantic knowledge in memory and the current problem task. Elements of analogs and the whole episodes are represented by coalitions of agents that may be considered as a kind of distributed representation.

Copycat solves proportional letter-string analogies such as "abc is to abd as mrrjjj is to ?" (most people answer either mrrkkk, mrrjjk, or mrrjjjj). It gradually builds its own representations and simultaneously explores different interpretations of analogy. The drawback is that Copycat does not provide a domain-general account of analogy.

Each new micro-domain requires encoding new system knowledge. The model extension to more complex types of analogies is also unclear.

AMBR integrates retrieval, mapping and transfer. These processes run in parallel and can influence each other. In the system's long-term memory coalitions of agents allow for distributed representations of objects, concepts and episodes. Mapping is performed by gradually building and relaxing a constraint satisfaction network that is built incrementally and in a distributed way by the independent operation of many agents which make their decisions using only local information. The AMBR model demonstrates insertions from general memory and blending of episodes, priming order and context effects. Some of those effects were met in the experiments with humans.

Analogy models including distributed internal representations. The quest to enhance a semantic basis of representations, their scaling, and degree of neurological relevancy, led to the attempts to augment the models of analogy with some share of distributed representations.

Learning and Inference with Schemas and Analogies (LISA) is an integrated theory of analogical access and mapping [Hummel & Holyoak, 1997, 2003], as well as similarity [Taylor & Hummel, 2009]. It is a connectionist model that stresses the importance of working memory capacity limitations. Elements of analogy are represented in long-term memory as hierarchies of localist nodes. However, entities are semi-distributed and represented by micro-features of the bottom level semantic nodes. The more similar two entities are - the more semantic nodes they share. Those distributed representations come to work in working memory. Structure sensitivity is achieved through dynamic binding by synchronous oscillation of distributed representations. LISA posits limitations on the number of different oscillating activation patterns that can be simultaneously active in working memory, thus modeling its span. The drawback of LISA is its limited ability to scale up to human performance in processing analogies involving large representations.

STAR2 [Gray et al., 1997] uses the tensor-product representations to represent and to store relational structures of analogs. Tensor representations [Smolensky, 1990] can potentially represent items in fully distributed fashion. Their dimensionality grows exponentially vs the number of relational arguments that is used by the authors to limit complexity of reasoning according to the pattern demonstrated by humans. For mapping, localist constraint satisfaction networks are built and evolve in parallel, similar to ACME. Sequential focusing on different parts of analogs allows working with more complex analogies than proportional analogies of the previous model versions.

Distributed Representation Analogy MApper (DRAMA) [Eliasmith & Thagard, 2001] uses distributed representations of analogs – HRRs of [Plate, 2003]. However, those representations are only used at the first stage to construct and initialize a "semantically driven mapping network" based on ACME that then produces the mappings.

So, application of distributed representations in the described models is not systematic. At some stage, they use structure handling techniques conceptually similar to localist networks. Emerged structure-sensitive fully distributed representations mentioned in the Internal Representations and Similarity section have been applied to modeling of analogical retrieval and mapping.

HRRs [Plate, 2000; 2003] and APNNs (Rachkovskij, 2001; Rachkovskij & Slipchenko, under revision] were tested on the analogical retrieval tasks. It was shown that for analogical episodes with different similarity types of (see the Experiments section below), the retrieval results obtained by vector similarity measures are consistent with the experimental results observed for people and modeling results reported for MAC/FAC and ARCS (where different episodes were used). In [Rachkovskij & Slipchenko, under revision], a modified representation scheme was proposed that provided results comparable to MAC/FAC using MAC/FAC's knowledge base of analogs.

The models of mapping based on HRRs [Plate, 2000; 2003] and BSCs [Kanerva, 1998; 2009] were proposed that used the technique based on "unbinding" (binding with the inverse of a distributed representation) operations. In [Rachkovskij, 2001], similarity of distributed APNN representations of analogical elements was used for their mapping. However, those techniques worked only for the most straightforward mappings cases. In [Rachkovskij, 2004], a number of approaches for mapping with the APNN distributed representations were proposed (including direct similarity mapping, re-representation by substitution of identical code-vector, parallel traversing of structures, using higher-level roles) and some of them were demonstrated on complex analogies. However, the methods were rather complex and used sequential operations. In [Gayler & Levi, 2009] an interesting scheme was proposed for finding graph isomorphism with HRRs and associative memory.

In the rest of the paper, we consider the scheme for binary distributed representation of analog elements that allows a simple mapping by similarity of rather difficult and complex analogies and consider mapping and inference using this scheme.

Distributed representations of relational structures

Let us consider APNN-style of analogs. Each item *x* (element of analog – attribute, object, relation) is represented by a code-vector X ($x \rightarrow$ X). Code-vector is a form of vector representation that is binary ($X \in \{0,1\}^N$) and sparse (the fraction of non-zero vector components *M* in code-vector X with dimensionality *N* is small: $M/N << 1$). Similar items (in context of the application problem) should have similar code-vectors, whereas items with undefined similarity should have dissimilar code-vectors.

Similarity of items *sim*(*x*,*y*) is estimated by similarity of their code-vectors *sim*(X, Y). Code-vector similarity is defined using dot product, which for binary code-vectors is equal to the number of common unit components ("1s"). Similarity *sim*(X, Y) is a relative overlap of code-vector:

$$sim(x,y) = sim(X, Y) = \sum_{i=1,N} X_i Y_i / \sum_{i=1,N} X_i = |X \wedge Y| / |X|, \quad (1)$$

where X_i is a component of X, $\wedge$ is component-wise conjunction, |Z| is the number of non-zero components in Z.

We consider relational structures of analogs or episodes in declarative knowledge bases as labeled directed acyclic graph [Frasconi, Gori, & Sperduti, 1998], where child vertices (arguments) of parent vertices (relations) can be objects (entities) or relations (Fig. 1). Previously [Rachkovskij & Kussul, 2001; Rachkovskij 2001], we have proposed schemes for representation of relations *R*(*A*,*B*,....), (where *R* is the label of relation, *A*,*B*.... are its arguments), that corresponds to the *role-filler* schemes traditionally used in symbolic representations. In the scheme we use bellow, code-vector of the relation is formed as $R(A,B,....) \rightarrow \langle\langle R_a, A\rangle, \langle R_o, B\rangle, ... \rangle$, where A, B, ... are code-vectors of the arguments (fillers), R_a, R_o, ... are code-vectors of the roles (agent, object,...), and $\langle ... \rangle$ denote the binding procedure for code-vectors.

We consider the binding procedure as a functional analog of grouping brackets traditionally used in symbolic notation. In the examples of relation given above, $\langle R_a, A\rangle$ should preserve the information that R_a is bound with A, not with B or R_o. We will use the Context Dependent Thinning (CDT) binding procedure described in [Rachkovskij & Kussul, 2001]. It is implemented as follows.

First, code-vector Z is formed as component-wise disjunction $\vee$ of element code-vectors X_i:

$$Z = \vee_i X_i, \quad (2)$$

For the example above, $Z = R_a \vee A$ (and also another $Z = R_o \vee B$). Then the result $\langle Z\rangle$ of binding $X_1, X_2, ..., X_S$ (for the example above, $\langle Z\rangle = \langle R_a \vee A\rangle$ as well as $\langle Z\rangle = \langle R_o \vee B\rangle$) is formed as

$$\langle Z\rangle = Z \wedge \vee_{k=1,K} Z^*(k). \quad (3)$$

Here $\wedge$ is component-wise conjunction, $Z^*(k)$ is Z with permuted components. For each k, random independent permutation is used, fixed for this k (there can be versions with single fixed permutation iteratively applied to Z^* [Kussul et.al., 2006]. The number of 1s in $\langle Z \rangle$ is controlled by K. Dimensionality of $\langle Z \rangle$ and X_i is the same. Subset of 1s of each component code-vector X_i preserved in $\langle Z \rangle$ depends on Z and therefore on each and all X_i, thus preserving information on the particular subset of elements that produced it, and so providing binding property. Also, similar code-vectors X_i produce similar $\langle Z \rangle$.

So, the dimensionality of all code-vectors (roles, fillers, relations) in the considered representation scheme is the same. Therefore, its recursive application allows forming code-vectors of episodes with hierarchical relational structures, where higher order relations (relations over relations) are present.

To construct the code-vectors of episode's elements and episode itself, it is necessary to have the code-vectors of its terminal elements, such as roles, attributes, names, labels, constants. In this paper, randomly and independently generated and memorized code-vectors are used for the terminal elements. The same code-vector is always used for the initial representation of any occurrence of the particular terminal element. From those code-vectors, other elements are constructed using the role-filler scheme described above.

Note that the code-vector of the whole episode is just the disjunction of code-vectors of its top-level relations (i.e., those relations that are not arguments of other relations), see Fig 2.

SOLAR_SYSTEM =
$\langle$ **CAUSE_1** $\vee \langle$**GRAVITY_1** $\vee \langle$**MASS** $\vee$ **SUN**$\rangle \rangle \vee \langle$**GRAVITY_2** $\vee \langle$**MASS** $\vee$ **PLANET**$\rangle\rangle \rangle$
$\vee \langle$**CAUSE_2** $\vee \langle$**ATTRACTS_1** $\vee$ **SUN**$\rangle \vee \langle$**ATTRACTS_2** $\vee$ **PLANET**$\rangle \rangle$
$\vee$
$\langle$ **GREATER_1** $\vee \langle$**TEMPERATURE** $\vee$ **SUN** $\rangle\rangle$
$\vee \langle$ **GREATER_2** $\vee \langle$**TEMPERATURE** $\vee$ **PLANET**$\rangle \rangle$
$\vee$
$\langle$ **CAUSE_1** $\vee$
$\langle$ **AND** $\vee \langle$**GREATER_1**$\vee\langle$**MASS**$\vee$**SUN**$\rangle\rangle \vee \langle$**GREATER_2**$\vee\langle$**MASS**$\vee$**PLANET**$\rangle\rangle \rangle$
$\vee \langle$ **AND** $\vee \langle$**ATTRACTS_1** $\vee$ **SUN**$\rangle \vee \langle$**ATTRACTS_2** $\vee$ **PLANET**$\rangle\rangle \rangle$
$\vee \langle$ **CAUSE_2** $\vee$
$\langle$ **REVOLVE-AROUND_1** $\vee$ **PLANET**$\rangle \vee \langle$**REVOLVE-AROUND_2**$\vee$**SUN**$\rangle \rangle$

Fig. 2. The role-filler code-vector representation of the Solar System analogical episode

If different episodes include some identical elements, and therefore identical code-vectors, the code-vectors of the episodes will be similar. The more identical relations and their arguments the episodes have, the more is the similarity of their code-vectors. Note also, that this approach allows the episode code-vectors to be similar not only if they include identical code-vectors, but also for similar code-vectors of similar elements, such as, e.g., produced as context code-vectors mentioned in section Processes in Analogical Reasoning above.

The role-filler representation scheme has been used earlier both for analogical retrieval [Rachkovskij, 2001; Rachkovskij & Slipchenko, under revision] and mapping [Rachkovskij, 2001, 2004]. For analogical retrieval by similarity of code-vector episodes as wholes, the representation schemes of this type appeared adequate. However, [Rachkovskij, 2001, 2004] showed that mapping using the direct similarity of code-vectors of analog's elements obtained by this scheme is not always correct This is due to the fact that in the simple role-filler scheme the element codevecor includes only code-vectors of its sub-elements. However, the mapping also depends, and to a greater degree, on the similarity of relational systems to which the mapped elements belong.

So, in order to develop mapping techniques based on the similarity of element code-vectors, we developed a scheme for re-representation of analog's elements (based on the ideas from [Rachkovskij, 2004; Slipchenko & Rachkovskij, 2009]). The re-representation idea we employ consists in combining the element code-vector

obtained by the simple role-filler scheme described above, with the code-vectors of the higher-levels roles in that element. Thus, using the code-vectors of analog elements, generated using the role-filler scheme (that are adequate for retrieval and always the same for identical elements), for mapping we construct re-representations that depend on the episode.

Re-representation code-vector Cx* of analog element *x* is the combination of its "lower" representation, that is the code-vector CxLower of *x* generated by the role-filler scheme, and its "higher" representation CxHigher, that contains the information about the roles r in the higher-level relations to which x belongs (immediately as their filler, or recursively through other higher level relations):

$$Cx^* = CxLower \vee CxHigher,\ CxHigher = \vee_r C_r. \quad (4)$$

Here C_r may be the role code-vector itself, or binding with the initial role-filler code-vector of *x*, i.e., CxLower.

Example of the roles used for the re-representation of the Sun element of the Solar System – Atom analogy is shown in Fig. 1 by the solid lines,. and the re-representation code-vector is formed as follows (repeated code-vectors are omitted):

$$SUN^* = SUN \vee TEMPERATURE \vee MASS \vee ATTRACT\_1 \vee REVOLVE\_1 \vee GREATER\_1 \vee GRAVITY\_1 \vee GREATER\_1 \vee CAUSE\_1 \vee CAUSE\_2 \vee AND. \quad (5)$$

Depending on implementation, here RELATION_1 means RELATIONagent or ⟨RELATIONagent ∨ ELEMENT⟩.

Mapping and inference techniques

Mapping. Mapping using the re-representation code-vectors is done as follows.

Step 1. For each top-level relation $t \in T_{top}$ (i.e., those relations that are not arguments of other relations) of the target analog *T*, the best mapping *b*'(*t*) is found by the maximal similarity between its code-vector C_t^* and code-vectors C_b^* of the elements of the base analog $b \in B$:

$$b'(t) = \text{argmax}_{b \in B}\ sim(C_t^*, C_b^*). \quad (6)$$

Step 2. Taking into account the parallel connectivity constraint (the arguments of corresponding relations must correspond), for each element pair mapped at step 1 similarity of their sub-elements of all levels is calculation as follows. All pairs (ti, bj) of children of the mapped elements are considered, where ti ∈ T are children of the target element t, bi ∈ B are children of the base element b, and similarity of their re-representation code-vectors (4) is calculated by (1). All pairs whose similarity is less that the threshold value (i.e. similarity of random code-vectors with the same number of 1s) are ignored. So, at the output we get the list of remained triples (t, b, sim(t,b)), where x and y are elements of the target and the base, correspondingly, and sim is their similarity.

Note that since each element can belong to different top-level relations identified at step 1, there could be multiple triples (*t*, *b*, *sim*(*t,b*)) for the same pair (*t*, *b*).

Step 3. For each element of the target t the corresponding mapping b' is obtained as follows.

$$b'(t) = \text{argmax}_{b \in B}\ \Sigma\, sim(t,b), \quad (7)$$

where summation stands for the triples that occur several times after step 2.

Note that strict one-to-one mapping can be achieved, if necessary, by elimination of the already mapped elements with maximal similarity.

This mapping technique takes into account both the similarities of the elements themselves reflected in the similarity of their corresponding code-vector representations ("lower" ones), as well as the similarity of roles of those elements in relations or attributes to which the elements belong ("higher" code-vector representations).

Analogical Inference. As many analogical reasoning models, we consider inference as an extension of mapping and use copying with substitution and generation (section Processes in Analogical Reasoning and [Markman 1997]) for its implementation. The inference is realized based on the mapping results. Specifically, the elements of the base which have not been mapped are identified and transferred to the target. Those are hypotheses, the quality of which is evaluated as follows. For each element transferred from the base, starting from the highest-level elements, similarity of its code-vector is calculated to both the code-vectors of the target sub-elements (i.e., child elements or arguments or fillers) of its transferred mapping hypotheses and to the code-vector of the entire target episode. The code-vectors used here are the "lower" ones, i.e., the initial role-filler representations. If the similarity to the code-vector of the whole episode is larger than the similarity to each of its sub-elements, then the hypothesis is accepted, otherwise it is rejected. The same procedure is repeated for the lower-level hypotheses. Computational complexity of the proposed mapping method is O(n2M), where n is the number of analog elements, and M is the average number of 1s in the element code-vectors (M may be considered constant). The developed techniques were implemented and studied experimentally on analogical episodes that are commonly used for model testing.

Experiments

Experiment 1. In this experiment, we consider the episodes adapted by [Plate 2000, 2003] from [Thagard et al., 1990]. The following entities are involved: dogs (Fido, Spot, Rover), people (Jane, John, Fred), a cat (Felix), a mouse (Mort). Relations are bite, flee, cause. The probe (base) episode P is "Spot bit Jane, causing Jane to flee from Spot". Other episodes (Table 1) have the same relations as the probe, but different types of similarity - mainly according to Gentner's classification [Gentner, 1983; Forbus et al., 1995].

Table 1. The animal analogical episodes with various types of similarity to the Probe

Similarity type	Episode							Comments (description)
Probe P	Spot	bit	Jane	causing	Jane	to flee from	Spot	all episodes are described compared to Probe
Literal similarity LS	Fido	bit	John	causing	John	to flee from	Fido	both structural and superficial similarity
Surface features SF	John	fled from	Fido	causing	Fido	to bite	John	superficial but not structural similarity
Cross-mapped CM	Fred	bit	Rover	causing	Rover	to flee from	Fred	structural and superficial but entities switched
Analogy AN	Mort	bit	Felix	causing	Felix	to flee from	Mort	structural but not superficial similarity
First order relations FOR	Mort	fled from	Felix	causing	Felix	to bite	Mort	neither structural nor superficial similarity

Dimensionality of code-vectors was $N=10^5$ if not stated otherwise. The average number of 1s M(attribute) = M(object) = 1000 was chosen as in [Rachkovskij, 2001], whereas M(role) = 2000 > M(attribute) = M(object)=1000 was chosen to reflect importance of relations. Instantiations of those parameters and the thinning factor 0.2 (defined as $|\langle Z\rangle|/|Z|$, see (2) and (3)) were selected from the interval that provided correct retrieval (not mapping yet!) scores for the animal episodes, as in [Rachkovskij & Slipchenko, 2009; under revision].

Correct mapping of the Probe (Base) episode to each of the other episodes is shown in columns of the Table 1. It may seem rather non-evident, but it is ensured by the same roles in the higher-order relation "cause". If we use the naive mapping by the direct similarity of code-vectors of the initial role-filler representation schemes (used for retrieval and represented by the "lower" code-vector of (2)), results of [Rachkovskij 2001, 2004] show that only the LS episode can be correctly mapped to the Probe, and for the AN episode, the mapping was correct for all elements, despite the entities. For the SF, the CM, the FOR episodes, the correct mapping can not be established by the similarity of the "lower" code-vectors. On the other hand, the results obtained using the re-representation code-vectors and the techniques described in this paper are correct. This is illustrated for the Probe-FOR episodes in Table 2 that shows the similarity values calculated for the re-representation code-vectors of the elements of those episodes. The correct mapping requires the values at main diagonal to be the largest, and that is actually observed.

Table 2. The similarity matrix for the element code-vectors (re-representation) of the Probe and the FOR episode

	Probe	Bite	Flee	Spot	Jane
E_FOR	0.48	0.38	0.39	0.64	0.65
Flee	0.31	0.47	0.30	0.66	0.67
Bite	0.30	0.30	0.47	0.66	0.66
Mort	0.23	0.31	0.31	0.81	0.45
Felix	0.24	0.31	0.31	0.45	0.81

Experiment 2. We investigated our mapping and inference model using rather complex analogical episodes that are usually used to test analogical models, such as various versions of "Water Flow – Heat Flow", "Solar System – Atom", "Schools", etc. The code-vector parameters and representation scheme are the same as in Experiment 1.

In figures below, dashed lines show the found correspondences between the elements of the base and the target analogs, the numbers on the arrows indicate the similarity value of the mapped elements, and the dotted analog elements of the target show the hypotheses.

In the experiment with the Water Flow – Heat Flow analogy (Fig. 3), the similarity between clean(beaker), liquid(water), flat-top(water), greater(diameter(beaker), diameter(vial)) and any element of the other analog is less than similarity of random code-vectors, so they do not map. The mappings for the other elements of the Heat Flow analog established by the similarity of corresponding code-vectors, as described in the techniques proposed in this paper, are one-to-one and correct The similarity between the unmapped cause (...) code-vector from the base analog and the elements of the target analog flow(…) and greater(…) is less than its similarity to the code-vector of the target episode, so the hypothesis of cause (...) transfer to the target becomes the valid inference.

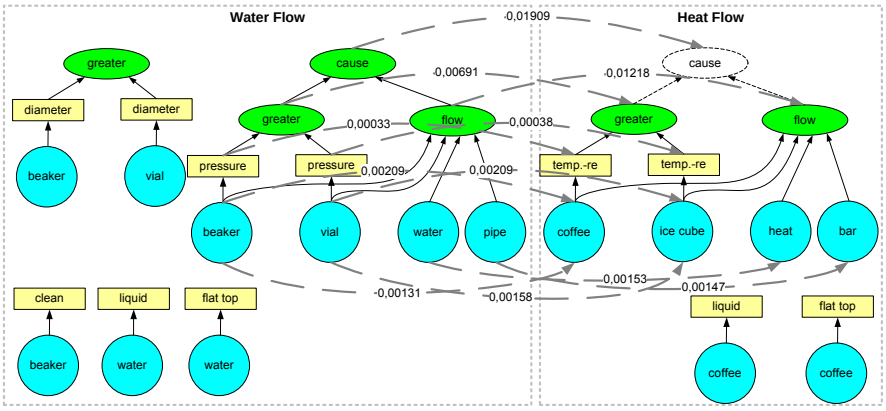


Fig. 3. Mapping and inference for the Water Flow – Heat Flow analogy

For the "Solar System – Atom" analogy (Fig. 4), the algorithm generates and checks hypotheses at several levels (see cause(…) and and(…) in the base). Unlike the previous example, where there was only one group of mappings, and all non-essential mappings and inferences were ignored due to the low similarity value, here after generation of inference hypotheses we observe three mapping groups.

The first group establishes the correspondence between the difference of mass, gravity and rotation of bodies. In this group, the causal relation between the attraction of the nucleus and the electron, and the rotation of the electron around the nucleus has been transferred to the Atom from the Solar System as a hypothesis and confirmed by similarity of appropriate code-vectors.

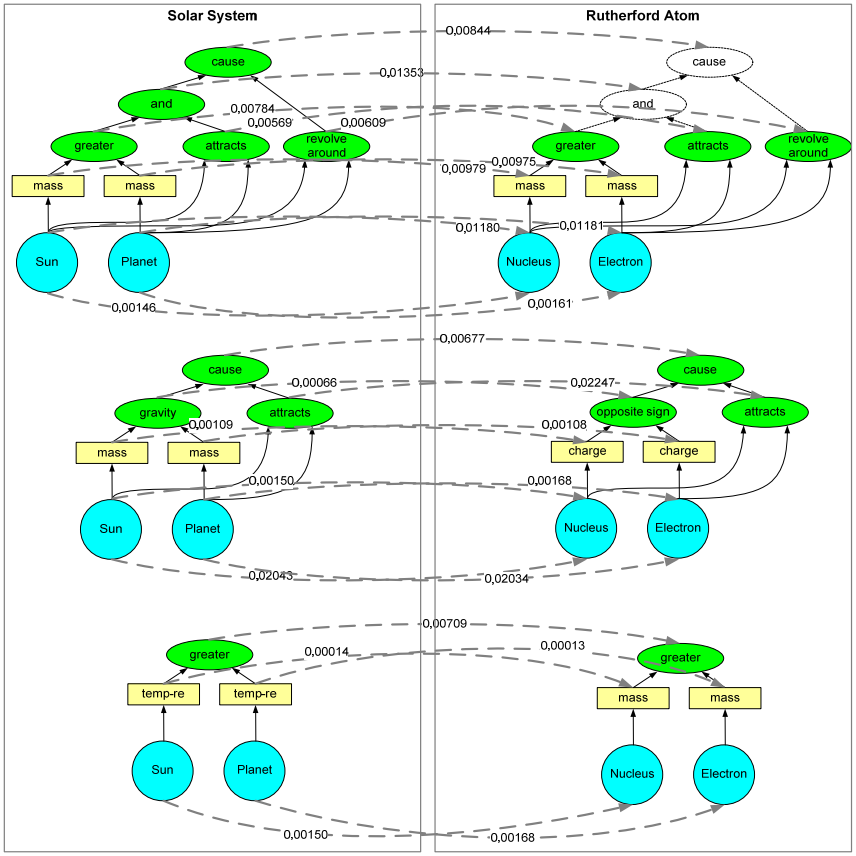


Fig. 4. Mapping and inference for the Solar System – Atom analogy

The second group matches the causation between the gravity and attraction in the Solar System analog and the causation between the charge sign difference and the attraction in the Atom analog.

The third group matches the temperature differences between the Sun and the Planet, as well as the mass difference of the Nucleus and the Electron, but the combined weight of those matches is very small (less than a random overlap), so they should be ignored.

For the "Schools" analogy (Fig. 5) [Markman 1997] all four cause() relations present in the Old School and Absent in the New School were correctly transferred to the target, and were accepted as valid inferences.

Thus, the proposed approach allows processing of fairly complex analogs and creates mappings and inferences consistent with the results of psychological tests and with the results of other analogical models.

```
"Old-School"
; English Department
(CAUSE (OBTAIN ENGLISH-FACULTY GRANTS) (HIRE ENGLISH-FACULTY RESEARCH-
ASSISTANTS))
(CAUSE (INFIGHTING ENGLISH-FACULTY) (AVOID ENGLISH-FACULTY OFFICES))
; Biology Department
(CAUSE (EXCELLENT BIOLOGY-FACULTY TEACH) (OVERSUBSCRIBED CLASSES))
(CAUSE (SMALL-NUMBER BIOLOGY-FACULTY) (NOT (PERFORM BIOLOGY-FACULTY ADVISING)))

"New-School"
; Computer Science Department
(OBTAIN CS-FACULTY GRANTS)
(INFIGHTING CS-FACULTY)
; Music Department
(EXCELLENT MUSIC-FACULTY TEACH)
(SMALL-NUMBER MUSIC-FACULTY)
```

Fig. 5. The Old School – New School analogy.

Experiment 3. The computational complexity of the implementation of the proposed code-vector-based approach to analogical mapping and inference largely depends on the code-vector dimensionality. We investigated the quality of mapping and inference for varying N at constant p = N / M (equal to 0.01 or 0.02 for some terminal code-vectors as described above). Fig. 6. shows the percentage of correct mappings vs N for mapping of the animal episodes. Fig. 7 shows the standard measures of recall, precision, and integrated F1-measure of inference vs N for the Schools analogy. For each *N*, the results reported were averaged over 100 instances of random terminal code-vectors used to construct code-vectors of analogs. Both figures show reliable results at *N* > 1000...10000.

In general, the experiments have shown the adequacy of the proposed approach to modeling mapping and inference based on the similarity of re-representations of analog elements and a lower computational complexity of its implementation compared to the traditional symbolic methods.

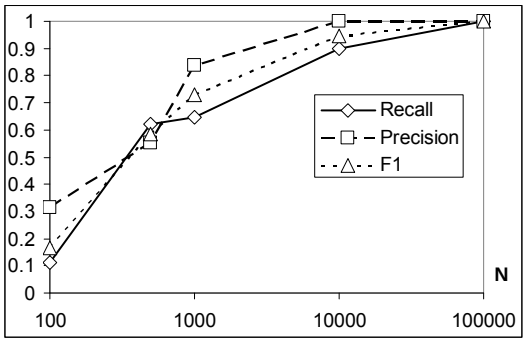


Fig. 6. Mapping quality vs code-vector dimensionality N

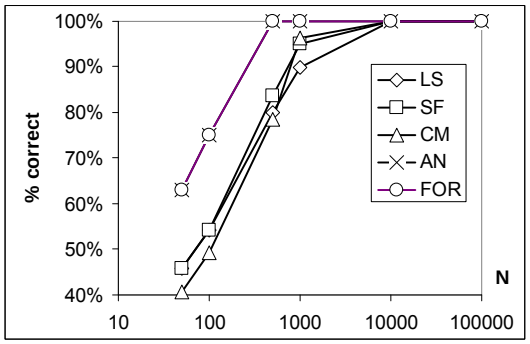


Fig. .7. Inference quality vs code-vector dimensionality N

Discussion

Analysis of analogical models developed on the basis of symbolic or localist neural representations (SME, ACME, etc.) shows that, although such models are well suited for processing and comparison of complex hierarchical structures required for analogical reasoning, estimation of analogical similarity requires mapping of their elements implemented by computationally expensive procedures. Computational complexity of SME varies from $O(n^2)$ to $O(n!)$, and that of ACME [Holyoak & Thagard, 1989] is $O(n^4)$.

In quest for a more adequate account of analog semantics, increasing the neurobiological relevance of analogy models and their scaling potential, some new models of analogical reasoning employed distributed representations. However, in previous models the use of distributed representations was fragmented or inconsistent, or they were able to work only with simple analogies.

We propose a scheme for constructing distributed re-representations (code-vectors) of analog elements that allow finding correspondences between the elements (mapping) and justification of new knowledge (relations) transfer to the target (inference validation) to be done based on the similarity of the re-representation code-vectors. Code-vectors are binary distributed vectors that are similar for similar elements in terms of vector similarity measures based on dot-product. Those re-representation code-vectors are constructed from two parts – the "lower" part (that may be considered as more semantically grounded) and the "upper part" (mainly reflecting structure). This creates a context-dependent representation of analogs' elements – the same element gets different representations depending on the systems of relations in which it participates (reflected in the "upper" representation code-vector). Also, the same relational system gets different contexts to which it applies. (a kind of grounding reflected in the "lower" representation code-vector).

Experiments with the developed mapping and inference techniques using analogies that were previously used for testing known symbolic models (SME, ACME, etc.) and psychological experiments showed appropriate results and have confirmed the adequacy of the proposed approach and methods of analogical reasoning.

The computational complexity of the proposed mapping method is $O(n\ n')$–$O(n^2)$, where n' is the number of elements for which mapping should be established, n is the number of elements of another analog, with whom mappings are established. This is at the level of the lowest known computational complexity of mapping, and lower than the complexity of traditional methods $O(n^4)$–$O(n!)$. So the proposed methods are promising for mapping fragments of knowledge bases consisting of a large number of elements.

The proposed combined representation consisting of two parts lends itself to further elaboration. For example, the lower part can be made more grounded, if the terminal code-vectors are taken not random, but are formed by and

reflect the semantic basis of the outside world and the semantic similarity of corresponding objects and relations. The upper part can be made more structure-sensitive by using not only the set of higher-level roles (which may not distinguish different relational structures including identical relations in different arrangements) but also binding them with each other. We think that such representations would be useful not only in models of analogy, but also in other (cognitive) models, such as structure-sensitive models of similarity [Taylor & Hummel, 2009], etc.

We think that interesting and promising directions of future research of structure-sensitive distributed representations might include the following:

- The mechanisms for various possible mappings of the same analogical element.

- Making mapping with code-vector similarity more dynamic by introducing a process by which alternative mappings compete with each other, e.g. in the spirit of [Gayler & Levy, 2009].

- Mapping of large knowledge base fragments consisting of many elements.

- Possible code-vector representations in long-term memory. Whether only code-vector representations for retrieval are stored in memory and after retrieval are deployed and re-represented for mapping, or, if re-re-representations and mappings are stored in memory.

- How storing code-vector distributed representations in associative memory influenced retrieval, generalization, and other processes of analogical reasoning.

Bibliography

[Agichtein & Gravano, 2000] E.Agichtein & L.Gravano (2000). Snowball: Extracting relations from large plain-text collections. Proceedings of the fifth ACM conference on Digital libraries, San Antonio, Texas, US, pp. 85 - 94.

[Bobrow et al., 2009] D.G.Bobrow, C.Condoravdi, L.Karttunen, & A. Zaenen (2009). Learning by reading: normalizing complex linguistic structures onto a knowledge representation. AAAI Symposium 2009: Learning by Reading and Learning to Read; 2009 March 23-25; Stanford, CA.

[Browne & Sun, 2001] A.Browne & R.Sun (2001). Connectionist inference models. Neural Networks, 14(10): 1331-1355.

[Eliasmith & Thagard, 2001] C. Eliasmith & P. Thagard (2001). Integrating Structure and Meaning: A Distributed Model of Analogical Mapping. Cognitive Science, 25(2), 245-286.

[Falkenhainer et al., 1989] B. Falkenhainer, K.D. Forbus, & D. Gentner (1989). The Structure-Mapping Engine: Algorithm and Examples. Artificial Intelligence, 41, 1-63.

[Forbus, Mostek, & Ferguson, 2002] K.D.Forbus, T.Mostek, & R.Ferguson (2002). An analogy ontology for integrating analogical processing and first-principles reasoning, Proceedings of the 14th conference on Innovative applications of artificial intelligence, p.878-885, July 28-August 01, 2002, Edmonton, Alberta, Canada.

[Forbus et al., 1995] K.D.Forbus, D.Gentner, & K.Law (1995). MAC/FAC: a model of similarity-based retrieval. Cognitive Science, 19(2):141-205.

[Forbus et al., 2007] K.Forbus, C.Riesbeck, L.Birnbaum, K.Livingston, A.Sharma, & L.Ureel (2007). Integrating Natural Language, Knowledge Representation and Reasoning, and Analogical Processing to Learn by Reading. Proceedings of AAAI-07: Twenty-Second Conference on Artificial Intelligence, Vancouver, BC.

[Forbus et al., 2008] K.Forbus, J.Usher, A.Lovett, K.Lockwood, & J. Wetzel (2008). CogSketch: Open-domain sketch understanding for cognitive science research and for education. In the Proceedings of the Fifth Eurographics Workshop on Sketch-Based Interfaces and Modeling. Annecy, France.

[Frasconi, Gori, & Sperduti] P.Frasconi, M.Gori, & A.Sperduti (1998). A general framework for adaptive processing of data structures. IEEE Transactions on Neural Networks, 9(5):768-786.

[French, 1995] R.M.French (1995). The Subtlety of Sameness. Cambridge, MA: The MIT Press, 320 pages.

[Frolov, Rachkovskij, & Husek] A.A.Frolov, D.A.Rachkovskij, & D.Husek (2002). On Information Characteristics of Willshaw-Like Auto-Associative Memory. Neural Network World , Issue 2 , p. 141-157.

[Frolov et.al, 2007] A.A.Frolov, D.Husek, I.P.Muraviev, & P.Y.Polyakov (2007). Boolean factor analysis by attractor neural network. IEEE Trans. Neural Netw. 18(3), pp. 698-707.

[Gayler, 1998] R. Gayler (1998). Multiplicative binding, representation operators, and analogy. In Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences. Edited by K. Holyoak, D. Gentner, and B. Kokinov. New Bulgarian University. Sofia, Bulgaria, pp. 405.

[Gayler & Levy, 2009] R.W. Gayler & S.D. Levy (2009). A distributed basis for analogical mapping. In Proceedings of the Second International Analogy Conference. NBU Press, Sofia, Bulgaria, pp. 165-174.

[Gentner, 1983] D. Gentner (1983). Structure-mapping: A theoretical framework for analogy. Cognitive Science, 7, pp. 155-170.

[Gentner & Markman, 2003] D.Gentner & A.B.Markman (2003). Analogy-based reasoning and metaphor. In The Handbook of Brain Theory and Neural Networks. Edited M.A Arbib. The MIT Press, Cambridge, MA, pp. 106-109.

[Gentner & Colhoun, 2010] D.Gentner & J.Colhoun (2010). Analogical processes in human thinking and learning. In A. von Müller & E. Pöppel (Series Eds.) & B. Glatzeder, V. Goel, & A. von Müller (Vol. Eds.), On Thinking: Vol. 2. Towards a Theory of Thinking. Springer-Verlag Berlin Heidelberg. p. 35-48.

[Gentner et.al., 2009] D.Gentner, J.Loewenstein, L.Thompson, K.D.Forbus (2009) Reviving Inert Knowledge: Analogical Abstraction Supports Relational Retrieval of Past Events. Cognitive Science 33 (2009), pp.1343–1382

[Gladun, 2000] V.P.Gladun (2000). Partnership with Computers. Kiev: Port-Royal, 128 p. (in Russian).

[Gladun et al., 2000] V.P.Gladun, V.Ju.Velichko, N.N.Kiselyova, N.N.Moskalkova (2000). Inference of Hypothesis on Composition and Properties of Objects on the Basis of Analogy. Artificial Intelligence, 1, pp.44-52 (in Russian).

[Gray et al., 1997] B.Gray, G. S. Halford, W. H. Wilson, & S.Phillips, (1997). A Neural Net Model for Mapping Hierarchically Structured Analogs, In Proceedings of the Fourth conference of the Australasian Cognitive Science Society. University of Newcastle, NSW, Australia.

[Hahn & Schnattinger, 1998] U. Hahn & K. Schnattinger (1998). Towards Text Knowledge Engineering, In: AAAI'98 - Proceedings of the 15th National Conference on Artificial Intelligence. Madison, Wisconsin, USA, July 26-30, 1998. AAAI Press - MIT Press, 1998, pp.524-531.

[Hofstadter, 1995] D.Hofstadter (1995). Fluid Concepts & Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought, NY: Basic Books.

[Holyoak & Thagard, 1989] K.J. Holyoak, & P. Thagard (1989). Analogical mapping by constraint satisfaction. Cognitive Science, 13, 295-355.

[Hummel & Holyoak, 1997] J.E. Hummel & K.J Holyoak (1997). Distributed representations of structure: A theory of analogical access and mapping. Psychological Review, 104, pp. 427-466.

[Hummel & Holyoak, 2003] J. E.Hummel & K.J.Holyoak (2003). A symbolic-connectionist theory of relational inference and generalization. Psychological Review, 110, 220–264.

[Kanerva, 1996] P.Kanerva (1996). Binary Spatter-Coding of Ordered K-tuples. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, & B. Sendhoff (Eds.). Proceedings of the International Conference on Artificial Neural Networks - ICANN'96, Bochum, Germany. Lecture Notes in Computer Science, 1112, 869-873. Berlin: Springer.

[Kanerva, 1998] P.Kanerva (1998). Dual role of analogy in the design of a cognitive computer. In: K. Holyoak, D. Gentner, and B. Kokinov (Eds.), Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences (Proc. Analogy'98 workshop). NBU Series in Cognitive Science, Sofia, Bulgaria: New Bulgarian University, pp. 164-170.

[Kanerva, 2009] P.Kanerva (2009). Hyperdimensional computing: An introduction to computing in distributed representation with highdimensional random vectors. Cognitive Computation, 1:139-159.

[Keane, Ledgeway, & Duff, 1994] M.T.Keane, T.Ledgeway, & S.Duff (1994). Constraints on Analogical Mapping: A comparison of Three Models. Cognitive Science, 18: 387-438.

[Kim & Porter, 2009] D.S.Kim & B.Porter (2009). Kleo: A Bootstrapping Learning-by-Reading System. Symposium 2009: Learning by Reading and Learning to Read; 2009 March 23-25; Stanford, CA.

[Kokinov, 1988] B. Kokinov (1988). Associative memory based reasoning: How to represent and retrieve cases. In Artificial intelligence III: Methodology, systems, applications. Edited by T. O'Shea and V. Sgurev. Elsevier Science Publishers B.V., Amsterdam, North Holland, pp. 51-58.

[Kokinov & French, 2003] B.Kokinov & R.M.French (2003). Computational Models of Analogy-making. In Nadel, L. (Ed.) Macmillan Encyclopedia of Cognitive Science. Vol. 1, pp.113 - 118. London: Nature. Publishing Group.

[Kokinov & Petrov, 2001] B.Kokinov & A.Petrov (2001). Integrating memory and reasoning in analogy-making: The AMBR model. In D. Gentner, K. Holyoak, & B. Kokinov (Eds.), The analogical mind: Perspectives from cognitive science (pp. 59-124). Cambridge, MA: MIT Press

[Kuehne, 2000] S.Kuehne, K.Forbus, D.Gentner, & B.Quinn (2000). SEQL: Category Learning as Progressive abstraction using structure mapping Proc. 22nd Ann. Conf. Cognitive Science Soc. (Cogsci 00), Lawrence Erlbaum Associates, 2000, pp. 770–775.

[Kussul, 1992] E.M.Kussul (1992) Associative neuron-like structures. Kiev: Naukova Dumka. 144 p. (In Russian).

[Kussul et al., 2006] E.M.Kussul, T.N. Baidyk, D.C. Wunsch, O. Makeyev, & A. Martin (2006). Permutation coding technique for image recognition systems. IEEE Transactions on Neural Networks, 17(6):1566-1579.

[Larkey & Love, 2003] L.B.Larkey & B.C.Love (2003). CAB: Connectionist Analogy Builder [find similar] [try Google] Cognitive Science, 27(5), pp. 781-794, 2003.

[Markman, 1997] A.B. Markman (1997). Constraints on Analogical Inference. Cognitive Science, 21(4), 373-418.

[Markman & Gentner, 2000] , A.B.Markman & D.Gentner (2000) Structure mapping in the comparison process. American Journal of Psychology, 113, 501-538.

[Markman et al., 2003] A.B.Markman, D.A.Rachkovskij, I.S.Misuno,& E.G.Revunova (2003). Analogical reasoning techniques in intelligent counterterrorism systems // Informational Theories & Applications, 10(2), pp. 139-146.

[Misuno, Rachkovskij, & Slipchenko, 2005] I.S. Misuno, D.A.Rachkovskij, & S.V.Slipchenko (2005). Vector and distributed representations reflecting semantic relatedness of words. Mathematical Machines and Systems, Issue 3, p. 50–66 (in Russian).

[Mostek, Forbus, & Meverden, 2000] T. A.Mostek, K. D. Forbus,.& C.Meverden (2000). Dynamic case creation and expansion for analogical reasoning. In Proceedings of the 17th National Conference on Artificial Intelligence.and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 323 - 329.

[Navarro & Lee, 2003] D.J. Navarro & M.D. Lee (2003). Combining Dimensions and Features in Similarity-based

Representations, Advances in Neural Information Processing Systems. Ed. by S. Becker, S. Thrun, K. Obermayer. N.Y.: MIT Press, P. 6774. No. 15.

[Plate, 2000] T.A.Plate (2000). Analogical Retrieval and Processing with Distributed Vector Representations. Expert Systems: The International Journal of Knowledge Engineering and Neural Networks, 17(1), pp. 29-40.

[Plate, 2003] T.A. Plate (2003). Holographic Reduced Representation: Distributed Representation for Cognitive Structures. Stanford: CSLI Publications, 300 p.

[Pollack, 1990] J. B.Pollack (1990). Recursive distributed representations. Artificial Intelligence, 46: 77-105.

[Rachkovskij, 2001] D.A. Rachkovskij (2001). Representation and processing of structures with binary sparse distributed codes. IEEE TKDE 13(2), pp. 261-276.

[Rachkovskij, 2004] D.A. Rachkovskij (2004). Some Approaches to Analogical Mapping with Structure Sensitive Distributed Representations, Journal of Experimental and Theoretical Artificial Intelligence, 16(3), pp. 125-145.

[Rachkovskij & Kussul, 2001] D.A. Rachkovskij & E.M. Kussul (2001). Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning. Neural Computation, 13(2), pp. 411-452.

[Rachkovskij & Slipchenko, under revision] D.A.Rachkovskij & S.V. Slipchenko. Similarity based retrieval with structure-sensitive sparse binary distributed representations

[Ramscar & Yarlett, 2003] M.Ramscar & D.Yarlett (2003). Semantic grounding in models of analogy: an environmental approach. Cognitive Science 27(1): 41-71.

[Sahlgren, Holst, & Kanerva, 2008] M.Sahlgren, A.Holst, & P.Kanerva (2008). Permutations as a Means to Encode Order in Word Space. Proceedings of the 30th Annual Meeting of the Cognitive Science Society (CogSci'08), July 23-26, Washington D.C., USA.

[Schubert, 2002] L.K. Schubert (2002). Can we derive general world knowledge from texts?, M. Marcus (ed.), Proc. of the 2nd Int. Conf. on Human Language Technology Research (HLT 2002), March 24-27, San Diego, CA, pp. 94-97.

[Shastri & Ajjanagadde, 1993] L.Shastri, & V.Ajjanagadde (1993). From simple associations to systematic reasoning: connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. Behavioral and Brain Sciences, 16: 417-494.

[Shepard, 1962] R. N. Shepard (1962). The analysis of proximities: Multidimensional scaling with an unknown distance function: Part I. Psychometrika, 27, 125–140. Part II. Psychometrika, 27, 219–246.

[Slipchenko & Rachkovskij, 2009] S.Slipchenko & D.Rachkovskij (2009). Mapping and analogical inference based on neural distributed representations. 15-th Intern. Conf. Knowledge-Dialogue-Solution, Varna, Bulgaria. 9, p. 95–101 (in Russian).

[Smolensky, 1990] P.Smolensky (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artificial Intelligence, 46: 159-216.

[Taylor & Hummel, 2009], E.G.Taylor, J.E.Hummel (2009). Finding similarity in a model of relational reasoning Cognitive Systems Research, Vol. 10, No. 3, pp. 229-239.

[Tenenbaum, 1996] J.B.Tenenbaum (1996). Learning the structure of similarity. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), Advances in Neural Information Processing Systems (Vol. 8, p. 3-9). Cambridge, MA: MIT Press.

[Thagard et al., 1990] P.Thagard, K.J.Holyoak, G.Nelson, D.Gochfeld (1990). Analog Retrieval by Constraint Satisfaction Artificial Intelligence 46, pp. 259-310.

[Thorpe, 2003] S.Thorpe (2003). Localized versus distributed representations. In The Handbook of Brain Theory and Neural Networks. Edited by M.A Arbib. The MIT Press, Cambridge, MA, pp. 643-646.

[Tversky, 1977]. A. Tversky (1977). Features of Similarity Psychological Review 84(4), pp. 327--352.

[Yan, Forbus, & Gentner, 2003] J.Yan, K.Forbus, & D. Gentner (2003). A theory of rerepresentation in analogical matching. In R. Alterman & D. Kirsch (Eds.), Proceedings of the Twenty-Fifth Annual Conference of the Cognitive Science Society. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Authors' Information

Serge V. Slipchenko – PhD, Senior Research Scientist, International Research and Training Center for Information Technologies and Systems, NAS and MES of Ukraine; Pr. Acad. Glushkova, 40, Kiev, 03680, Ukraine; e-mail: serge.slipchenko.irtcits (at) gmail.com

Major Fields of Scientific Research: Computational and Artificial Intelligence

Dmitri A. Rachkovskij – PhD, DSc, Leading Research Scientist, International Research and Training Center for Information Technologies and Systems, NAS and MES of Ukraine; Pr. Acad. Glushkova, 40, Kiev, 03680, Ukraine; e-mail: dar (at) infrm.kiev.ua

Major Fields of Scientific Research: Computational and Artificial Intelligence

A GENETIC AND MEMETIC ALGORITHM FOR SOLVING THE UNIVERSITY COURSE TIMETABLE PROBLEM

Velin Kralev

Abstract: In this paper genetic and memetic algorithms as an approach to solving combinational optimization problems are presented. The key terms associated with these algorithms, such as representation, coding and evaluation of the solution, genetic operators for the crossing, mutation and reproduction, stopping criteria and others are described. Two developed algorithms (genetic and memetic) with defined computational complexity for each of them, are presented. These algorithms are used in solving the university course timetable problem. The methodology and the object of study are presented. The main objectives of the planned experiments are formulated. The conditions for conducting experiments are specified. The developed prototype and its functionality are briefly presented. The results are analyzed and appropriate conclusions are formulated. The future trends of work are presented.

Keywords: genetic algorithm, memetic algorithm, university course timetable problem.

1. Introduction

1.1. Genetic Algorithms

Evolutionary algorithms (EA) are based on ideas borrowed from natural processes [1]. These algorithms use a set (called population) of possible solutions (called individuals of the population) which are subject to future changes in order to obtain the optimal solution, according to a predefined optimality criterion.

The idea of evolutionary algorithms was presented for the first time in [2] while the idea of genetic algorithms (GAs) was presented for the first time in [3]. Later in 1992 the beginning and the genetic programming were established [4]. Evolutionary computations date back even earlier [5] where a review of earlier approaches was made.

A particularly efficient subclass of the evolutionary algorithms is a class of the genetic algorithms. In these algorithms, the variable is called a gene or allele, while the solution is called an individual or chromosome. Solutions must be coded in such a way that genetic operators can be applied. An encoded solution is called genotype while a decoded solution is called phenotype. To measure the quality of a solution the term fitness is used i.e. a solution with high fitness is a better solution. The set of all solutions is called population. At each iteration the genetic algorithm creates a new generation. This process is called reproduction.

To solve an optimization problem initially several possible solutions are constructed. These solutions are coded properly according to the problem being solved, but so that the genetic operators can be applied. The genetic algorithms start with a set of initialized population of solutions, which are most often generated randomly. The two basic genetic operators are crossover and mutation. The genetic operator "crossover" most often takes two solutions (parents) and combines them after which one or more new solutions (children) come as a result. The parents are selected from all solutions of the current population. However, the choice of solutions is stochastic and is based on solutions with better value of the fitness function. The genetic operator "mutation" changes one solution and forms a new one [6].

The space of the possible solutions is a set of points where each point presents a possible solution. The searching of solution is equivalent to finding the extremes in this space (minimum or maximum). Since it is not always possible to predict the actual optimum, then the best solution found so far, is often regarded as a near-optimal.

After performing a number of genetic operations "crossover" and "mutation", some of the solutions in the old population with new ones are replaced. Thus the creation of a new population of the algorithm ends. The process of creating new generations is repeated until the stopping criterion is satisfied (eg, creating a specific number of generations or a certain number of generations after which a better solution than the last best has not been found).

Typically, the process of seeking a solution by genetic algorithm is independent from the scope. The genetic operators "crossover" and "mutation" do not know which solution is better and which not. This is determined by the fitness function which evaluates each solution. However, it is shown that good results using this approach can be obtained [7, 8].

1.2. Memetic Algorithms

There is still a debate associated with genetic algorithms a whether the genetic operator "crossover" plays a greater role or the local search process. To solve this dispute, usually the genetic algorithms that use techniques for local search are called mimetic algorithms [9].

The concept of memetic algorithms was presented for the first time by Moscato and Norman [10]. They describe an evolutionary algorithm which uses local search. This idea was later formulated by Radcliffe and Surrey [11] who made a comparison between genetic algorithms and mimetic ones. In the mimetic algorithms "meme" is regarded as a unit of information that can be self-replicating in a way in which people exchange ideas. The "meme" is different from the gene in that when it passes between the individual solutions, any solution adapts "meme" in the best possible way, while the gene remains unchanged in this transition [9].

The main advantage in using the memetic algorithms is that the space of possible solutions is reduced to the subspace of acceptable solutions with local optimum. After the execution of an operation such as mutation in the genetic algorithms the new solution may be worse than the original estimate. In the mimetic algorithms through the use of local the best location for a given gene (meme) will be always found. Thus the best possible cost for a solution will be achieved [9].

2. A Genetic and Memetic Algorithms for Solving the University Course Timetable Problem

In developing the genetic algorithm, the basic techniques used are the crossover and mutation operators. In the mimetic algorithm besides the above-mentioned two techniques the method of local search is used as well. Both algorithms are similar and therefore in this section they will be presented together showing only the differences between them.

2.1. Generating a Solution Based on Constructive Heuristics

The basis of the developed genetic algorithm is generating a solution based on constructive heuristics. The main goal of this algorithm is given in advance events so that they be placed on the weekly schedule. The pseudo-code of this algorithm is presented in Fig. 1.

For each event the algorithm seeks a free timeslot. The aim is to put each event in the weekly schedule so that no violation of soft constraints to occur. The computational complexity of the algorithm is square, i.e. $\Theta\left(N^2\right)$, where N is the number of events.

```
for each event
  if the event is fixed then continue to the next event
  for each timeslot
    current event set the current timeslot
    if not possible then continue to the next timeslot
    if any of the soft constraints is not violated
    then continue to the next event
  end
end
```

Figure 1. Pseudo-code for generating a solution based on constructive heuristics.

2.2. Generating a Solution Based on Local Search

The basis of the developed memetic algorithm is generating a solution based on a local search. The main goal of this algorithm is with given in advance a ordinance of events to be placed on a weekly schedule in such a way so that to obtain the best costs of the generated solution. The pseudo-code of this algorithm is presented in Figure 2.

```
for each event
  if the event is fixed then continue to the next event
  for each timeslot
    current event set the current timeslot
    if not possible then continue to the next timeslot
    if not violated any of the soft constraints
    then calculate the cost of the solution
    and continue to the next timeslot
  end
  put the current event at this timeslot
  in which the cost of the solution was best
end
```

Figure 2. Pseudo-code for generating a solution based on local search.

For each event and each timeslot the algorithm calculates the cost of the generated solution. Once all the timeslots have been verified the algorithm sets the current event at this location for the cost of the solution which was best. The computational complexity of the algorithm is cubic, i.e. $\Theta\left(N^3\right)$, where N is the number of events.

2.3. Generating Initialized Population. Genetic Operators of Selection, Combination, Crossing and Mutation

The initialized population of genetic algorithm is created by constructing q solutions through the algorithm described above which is based on constructive heuristics. You will note that q is the size of the population that remains constant in the process of reproduction.

The initialized population of mimetic algorithm is created similarly. Using the method of constructing the solutions by local search we generate q solutions.

At stage "selection" for both algorithms $\frac{q}{2}$ (i.e. 50%) of solutions in the current population which will play the role of parents are chosen. Then $\frac{q}{4}$ (i.e. 25%) pairs form at random. They will be crossed in order to generate new solutions - children.

The operator of the crossing for both algorithms is the same and plays an essential role. For each pair of parents a crossover operator is applied in the following manner: choose two elements (genes) k and r, such that k < r and k <> r. They represent the intersection of the parents. By rotation of a range of genes $[1,..,k-1]$, $[k,..,r]$ and $[r+1,..,N]$ the permutations of events for the two new children are formed of them. In this way of combining and coding it is likely to have missing and/or repetitive genes in descendants. The problem is solved by the repeated genes which replace the missing ones.

The mutation operators for both algorithms genetic and memetic are similar. Choose randomly two genes k and r which exchanged places. For the algorithms of consideration it was adopted 10% of the children to mutate, i.e. $\left\lceil \frac{q}{2}10\% \right\rceil = \left\lceil \frac{q}{20} \right\rceil$.

2.4. Reproduction Operators of the GA and MA

Reproduction operators of genetic and memetic algorithm are similar and will be considered together. Pseudo-code for these operators is presented in Fig. 3.

```
generate initialize population [genetic or memetic]
for I := 1 to reproduction count do
begin
  delete worst solutions from population
  distribute parents to pairs
  crossover [genetic or memetic]
  mutate [genetic or memetic]
  sort population
end
```

Figure 3. Pseudo-code for reproduction operators of GA and MA.

First an initialized population of genetic and memetic algorithm is generated. Then a sequence of operations is performed until it reaches a predetermined number of reproductions. From each population 50% will be deleted of those solutions that have worse costs. Then the parents are combined in pairs and crossed.

Solutions of the children are generated with the method based on constructive heuristics for the GA and the method based on local search for the MA. At each step, the relevant method is called $\frac{q}{2}$ times as the number of new solutions. The next step: the mutation operators are executed. The reproduction cycle ends with sorting out the solutions with increasing cost.

In general, the complexity of the GA is square, and the MA is cubic. The complexity of the algorithms depends linearly on the number of reproductions (GenerationCount), i.e. the number of generations that will be generated.

3. Experimental Results

Three experiments with the following objectives were made:

1. To verify the efficiency of the prototype.

2. To determine the influence of the size of the input data and of the population size on the execution time of algorithms.

3. To determine the impact of population size and number of reproductions on the quality of the solution of the different sets of input data.

4. To make a comparative analysis between the proposed algorithms by comparing the performance quality of the generated solutions and the execution time for all sets of input data.

5. To demonstrate the effectiveness of the prototype and the proposed algorithms as compared to the estimates of the solutions generated by the prototype and the user (expert).

3.1. Development of Prototype

For the purposes of the experiment a prototype was developed. Through it planned experiments were conducted. From the obtained results appropriate conclusions were made. Algorithms that were discussed in the previous section are implemented in the prototype and can be used to generate schedules. Fig. 4 shows an example session using the prototype.

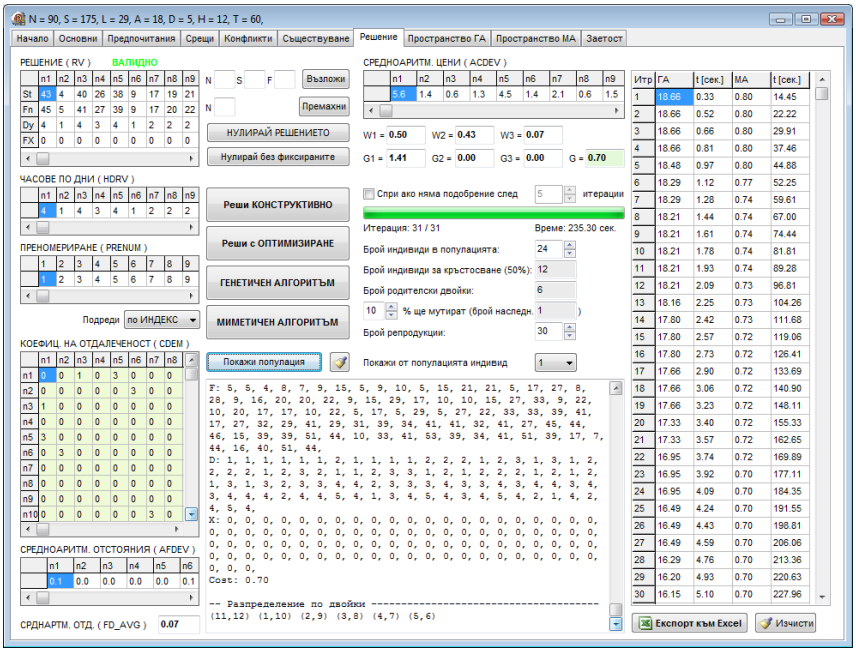


Figure 4. Work session with the prototype.

3.2. Conditions for the Experiment

Experiments were conducted on a PC with 32 bit operating system Windows VistaTM Enterprise (Service Pack 2) and the following hardware configuration:

1) Processor: Intel(R) Core(TM)2 Duo CPU T7500 @2.20GHz 2.20GHz

2) RAM memory: 2.00 GB

3.3. Methodology of the Experiment

For the purposes of the experiment 5 sets of input data were prepared, which are presented in Table 1.

Table 1. Sets of input data used in the experiments.

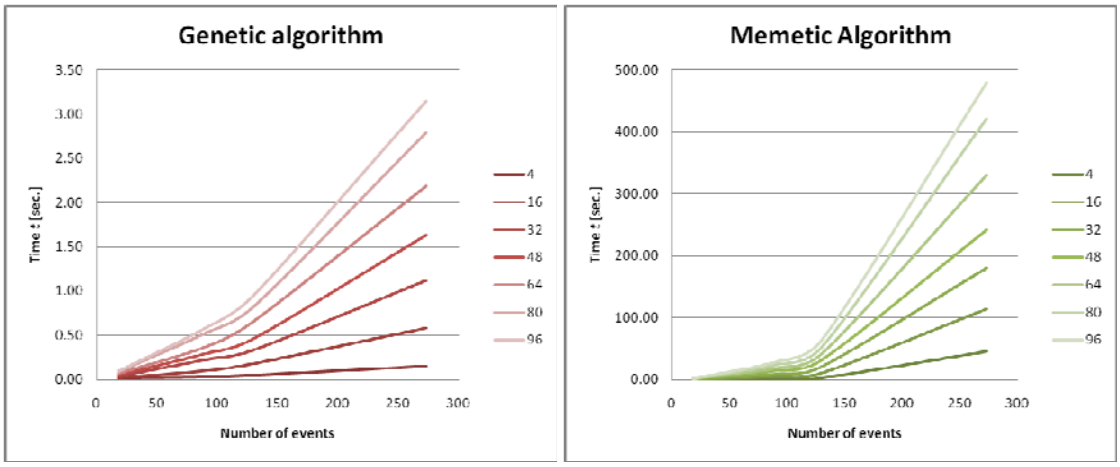
Dataset	DS1	DS2	DS3	DS4	DS5
Events (N)	18	90	130	273	30
Students (S)	52	175	274	549	45
Groups (Gr)	4	14	21	43	-
Lecturers (L)	10	29	37	62	10
Auditoriums (A)	10	18	22	39	10
Total (N+S+L+A)	90	312	463	923	95

The input datasets were chosen with certain reasons. The dataset 1 corresponds to one course. The dataset 2 corresponds to one subject. The dataset 3 is a combination of two subjects which have a common block courses, common lecturers and common auditoriums. The dataset 4 corresponds to approximately half a faculty and is a combination of three subjects sharing common resources. The dataset 5 is a set of 45 students who may not be distributed in different groups of students because they chose different curricula.

3.4. Conclusions from the Experiments

After conducting the experiments the following conclusions were made:

1. With the increase in the number of events, exponentially the execution time for GA and MA increases. This behavior is similar to both algorithms for all values of the parameter q (see Fig. 5a and 5b).

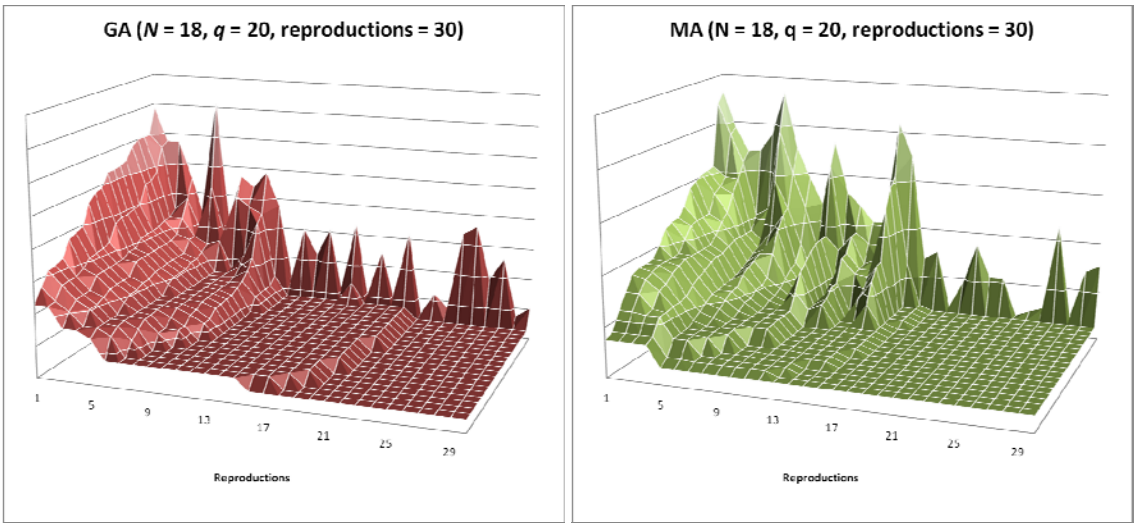


a) b)

Figure 5. Execution time of algorithms with increasing number of events: a) GA, b) MA.

2. With the increase in the number of solutions in the population the execution time for GA and MA increases linearly. This behavior is similar to all tested datasets (1 - 5).

3. With the increase in the number of solutions in the population the quality of solutions for both algorithms, improve. The level of convergence (i.e. overpopulation of the population with similar solutions) occurs differently for each of the algorithms (see Fig. 7a and 7b).



a) b)

Figure 7. Behavior of the GA and MA for a dataset 1 (N = 18, q = 40) a) GA, b) MA.

4. With the increase in the number of solutions in the population, the intensity of improving the quality of solution decreases for both algorithms.

5. In terms of quality of the decisions, MA provides significantly better results for all values of the parameters q and N, compared with GA. In terms of execution time, however, MA runs significantly slower than the GA.

6. GA is able to generate schedules with better cost which are commensurate with the schedules constructed by the user-expert (for the same input data). MA generates solutions with much better estimates than the other two methods, aiming to combine the events in such a way as to obtain a schedule with an optimal cost (see Fig. 8).

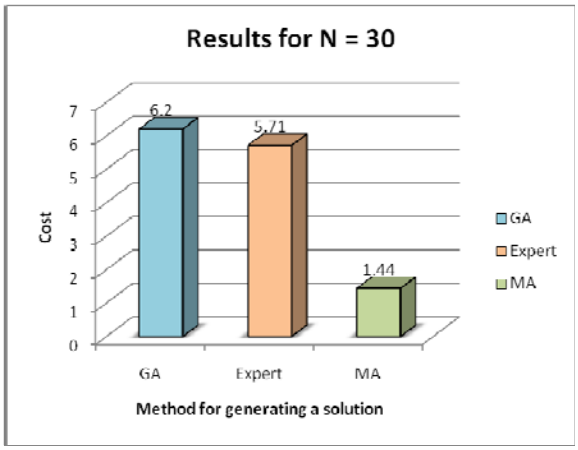
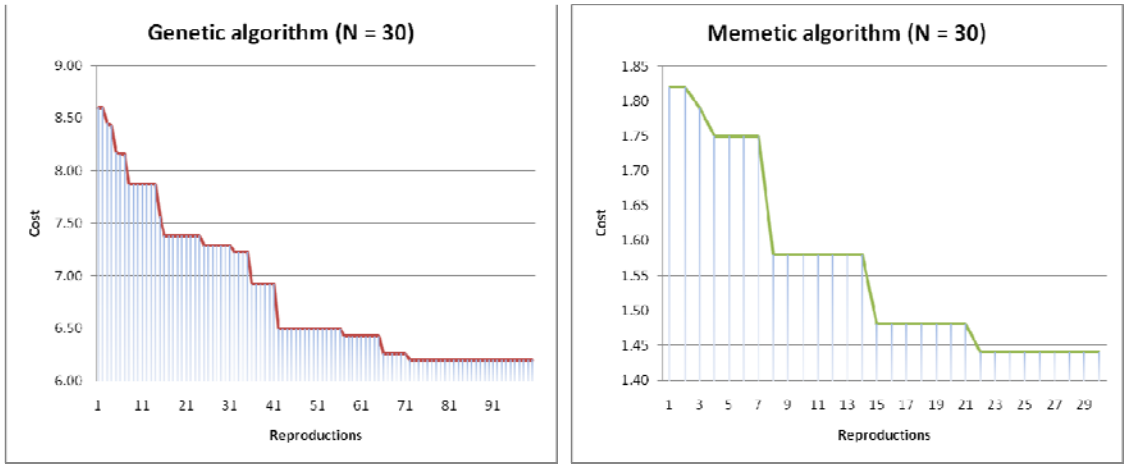


Figure 7. Costs of schedules generated by the GA, user-expert and MA (for a dataset 5 (N = 30)).

In Fig. 8 the behavior of the GA and MA for a set of input data 5 (N = 30) is shown.



а) б)

Figure 8. Behavior of the GA and MA for a dataset 5 (N = 30).

4. Conclusion

In this paper genetic and memetic algorithms as an approach to solving combinational optimization problems are presented. The key terms associated with these algorithms, such as representation, coding and evaluation of the solution, genetic operators for the crossover, mutation and reproduction, stopping criteria and others are described. Two developed algorithms (genetic and memetic) are presented for each of them the computational complexity is defined. These algorithms are used in solving the university course timetable problem. The methodology and the object of study are presented. The main objectives of the planned experiments are formulated. The conditions for conducting experiments are specified. The developed prototype and its functionality are briefly presented. The results are analyzed and appropriate conclusions are formulated. The future trends of work are presented.

Bibliography

1. Fogel D. B. (1992). A brief history of simulated evolution. Proceedings of the First International Conference on Evolutionary Programming, Evolutionary Programming Society, La Jolla, CA.
2. Rechenberg I. (1973) Evolutionsstrategie: Optimierung technischer Systeme und Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart.
3. Holland J. H. (1975) Adaptation in Natural and Artificial Systems. The U. of Michigan Press.
4. Koza J. H. (1992) Genetic programming: on the programming of computers by means of natural selection. Massachusetts Institute of Technology Press.
5. Fogel D. (1998) Evolutionary Computation. IEEE Press.
6. Burke E. K. (1994) A Genetic Algorithm based University Timetabling System. Proceedings of the 2nd East-West International Conference on Computer Technologies in Education, vol 1, pp 35-40.
7. Bruns R. (1993) Knowledge-Augmented Genetic Algorithm for Production Scheduling. IJCAI '93 Workshop on Knowledge based Production Planning, Scheduling and Control.

8. Burke E. K., Elliman D. G. and Weare R. F. (1994) A Genetic Algorithm for University Timetabling", AISB Workshop on Evolutionary Computing, Leeds.
9. Burke E. K., Newall J. P. and Weare R. F. (1996) A memetic algorithm for university exam timetabling. The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I). Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag, pp 241-250.
10. Moscato P., Norman G. M. (1992) A "Memetic" approach for the travelling salesman problem – implementation of a computational ecology for combinatorial optimisation on message-passing systems. Proceedings of the International Conference on Parallel computing and Transputer Applications, IOS Press (Amsterdam).
11. Radcliffe N. J., Surry P. D. (1994) Formal Memetic Algorithms. Lecture Notes in Computer Science 865 (Evolutionary Computing) Springer-Verlag, pp 250–263.

Author's Information

Velin Kralev – Department of Informatics, Faculty of Mathematics and Natural Sciences, South-West University "Neofit Rilski"-Blagoevgrad, Bulgaria; e-mail: velin_kralev@abv.bg

TABLE OF CONTENTS OF IJ ITA, VOLUME 16, NUMBER 3