Constraint-Based System for Genomic Analysis

Nittaya Kerdprasop and Kittisak Kerdprasop

***Abstract*—Recent advent of the new high-throughput biological technologies has brought more challenges to the computer science community in terms of the amount and variety of biological data awaiting for analysis. Computationally intensive techniques such as pattern recognition and machine learning algorithms have been applied to extract knowledge from several biological domains ranging from genomics, proteomics to system biology and evolution process. Learning techniques applied to the computational biology are mostly in the category of classification. Therefore, the sequence analysis problem has to be formulated as classification task, which is quite difficult due to the unobvious one-to-one mapping of the problem. In this paper, we propose a different setting of sequence analysis formulation based on the nucleotide patterns using a constraint logic programming paradigm, in which the sequence alignment can be performed through pattern matching techniques. With available knowledge from the field of pattern mining, we can apply the well-established techniques within the new framework of constraint programming. However, to make the system efficiently work, we need a new set of constraint solver algorithms specifically designed for the sequence analysis problem. The design and implementation of such algorithms are thus the main focus of our research project. We propose in this paper the design of a constraint-based system for genomic sequence analysis including the algorithm for the constraint solver, a major part of the proposed system.**

***Index Terms*—Genomic sequence analysis, constraint-based system, constraint solver algorithm, constraint programming.**

I. Introduction

Living organisms contain multiples cells to perform different functions. There are two basic types of cells: prokaryote cells (found in bacteria) and eukaryote cells (appeared in plants and animals). Contained within the cell membrane are several organelles and thousands of different types of molecules, the important one is DNA (deoxyribonucleic acid) that carries the entire genetic inheritance, or genes, of the cell. DNA is a long polymer molecule that contains sugar, phosphate group, and a mixture of four different nucleotide bases: adenine (A), cytosine (C), guanine (G), and thymine (T).

In 1953, Watson and Crick [1] discovered the DNA double helix structure in a complementary base pairing that A-T and G-C units always occur together, they are thus referred to as base pairs. In 1957, Crick [2], [3] described the flow of genetic information in biological systems (Fig. 1) that firstly DNA is copied to more DNA in the replication process, then DNA is transcribed into mRNA (or messenger-RNA) in a transcription process and finally mRNA is translated (by ribosome) into protein in a translation process.

This overall process of biological protein synthesis is known as gene expression. Understanding the process of gene expression in different types of cells and under different conditions is one of the fundamental research aspects of genomics, which is all the studies related to genes.

In prokaryotes, genetic information is encoded continuously on a DNA strand. But in eukaryotes, regions that code for protein (called exons) are interrupted by the non-coding regions (called introns). During the transcription of most eukaryotic genes, the primary RNA transcript (or pre-mRNA) needs additional modification step called splicing to remove introns and join exons together to make one long continuous mRNA strand (Fig. 2). The ribosome is an organelle that translates code on mRNA to different kinds of amino acids.

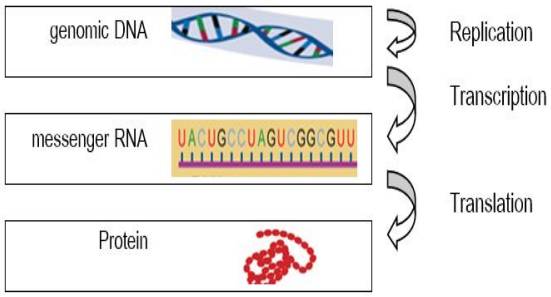

Fig. 1. Flow of genetic information in biological systems.

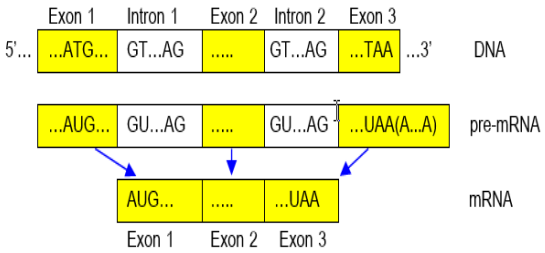

Fig. 2. Removal of introns and joining of exons in the splicing process during the DNA transcription.

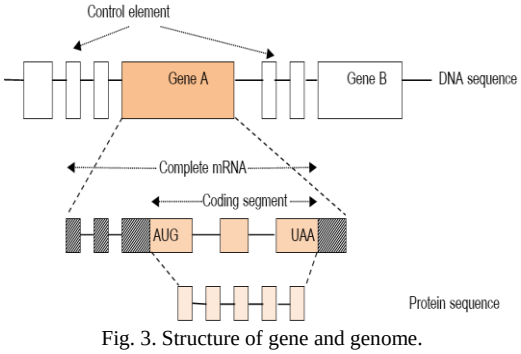

Fig. 3. Structure of gene and genome.

Manuscript received November 28, 2013; revised March 14, 2014. This work was supported in part by the National Research Council of Thailand and Suranaree University of Technology through the funding of Data Engineering Research Unit.

The authors are with the School of Computer Engineering. They are also with Data Engineering Research Unit, Suranaree University of Technology, 111 University Avenue, Muang District, Nakhon Ratchasima 30000, Thailand (e-mail: nittaya@sut.ac.th, kittisakThailand@gmail.com).

DOI: 10.7763/IJIET.2015.V5.487

In multi-cellular organisms, the DNA in each and every cell is identical. Different cells can perform different functions because different portions of the DNA molecule are active in different cells at different times. The entire DNA sequence contained in a cell, including the genes (stretches of DNA that code for a protein) and the control elements, is referred to as genome. Sketch of genome structure is presented in Fig. 3.

Genome is not only important to the life cycle of the cell, but also represents a blueprint for the life of the organism. Large genome sequencing projects have been set up by many governments and commercial organizations. The development of automated sequencing technologies such as shotgun sequencing technique allows scientists to decode genomes of many organisms at a significantly increasing rate. After a genome is reconstructed from the pieces of sequencing data, the next and most important step is to understand the content of the genome. That is, to identify the gene location and then annotate the function of each gene.

Since the announcement of the draft version of the human genomic sequence in 2001 [4] and the completion of several genome projects during the past decade, enormous amount of raw biological sequence data has been stored and awaited for interpretation. Dealing with large volume of data, efficient computational methods and intelligent techniques are a real need.

Sequence comparison is a fundamental operation in the field of computational molecular biology to detect similarity between biological sequences such as proteins and DNA sequences. The optimal match in a comparison between two sequences can be achieved through the dynamic programming technique. But it is very slow when it has to compare a sequence against many others, or compare among a group of related sequences in large database. To solve the computational time problem, approximate techniques tend to be the method of choice.

Many search tools employ a sophisticated statistical-based technique such as hidden Markov model (HMM). An HMM is a stochastic model that characterizes a coding sequence by computing probability of appearance of a nucleotide base (A, C, G, or T) based on the *k* previous nucleotides in the sequence. Computer scientists from the machine learning field prefer the neural network and support vector machines approaches rather than the statistical method of HMM. However, to characterize reliable coding sequences, these approaches require a large number of training sequences. The significant impact of such requirement is a very large search space. We, therefore, propose to tackle the sequence analysis problem through the constraint-based setting in which the search space could be reduced prior to the search for solution. Our prototype implementation is based on the constraint logic programming paradigm.

II. Literature Review

Genomic sequences are just raw biological data. To understand biological process inherent in the genomic sequences, computational and statistical techniques such as pattern recognition and machine learning algorithms are essential tools for the analysis of such large amount of genetic sequences. The analysis task over DNA, RNA and protein sequences includes several subtasks of

- searching for patterns within a sequence,
- searching for similarities between two sequences including sequence alignment for a pairs of sequences,
- searching for similarities among many sequences and performing multiple sequence alignment,
- constructing phylogenetic trees based on sequences,
- predicting and analyzing the secondary structures based on the sequences, and
- predicting and analyzing tertiary structure and folding patterns for protein and RNA sequences.

In this research we concentrate on the first three subtasks of analyses. That is, the problem of finding and parsing eukaryotic protein-coding genes.

Gene finding is generally the detection of sequence elements such as exons, introns, and the intergenic regions that separate genes. Once gens are found, their internal exon-intron structure can be predicted so that the encoded protein may be deduced. The gene finding problem in eukaryotes is difficult because the genes comprise less than 30% of the genome and once a gene is found, the locations of introns within the gene must be precisely determined in order to accurately deduce the protein product of the gene.

In the past, genes were identified with experimental validation on living cells and organisms. It is the most reliable method, but costly and labor intensive. At present, most biologists rely on the computational methods to automatically analyzed the uncharacterized genomic sequences. Some of the frequently applied computational techniques include dynamic programming, linear discriminant analysis, linguistic methods, hidden Markov model, and machine learning techniques such as neural network, decision-tree induction, support vector machines. Several successful gene-finding programs are based on the hidden Markov model algorithms.

A Markov model is a model of discrete stochastic process that evolves through the states from the set $S = \{s_1, s_2, \ldots, s_n\}$. The main assumption is that the probability of appearance of any future state depends only on the *k* preceding states, for some constant *k*. Given a learning set of sequences, a Markov model can be built by computing the probability that a certain nucleotide x_i appears after a sequence s_i, for example,

$$< P(x_i = \mathrm{A} \mid s_i = \mathrm{TTGGA}),$$
$$P(x_i = \mathrm{C} \mid s_i = \mathrm{TTGGA}),$$
$$P(x_i = \mathrm{G} \mid s_i = \mathrm{TTGGA}),$$
$$P(x_i = \mathrm{T} \mid s_i = \mathrm{TTGGA}) >$$

is a computing scheme of the Markov model of degree 5 (that is, $k = 5$). To model the codon usage that appears as a triplet of nucleotide bases, the degree *k* of the Markov model is normally set to 2, 5, 8, and so on.

The oldest gene-finding method based on Markov model is GeneMark [5]. From the success of GeneMark as an accurate tool to recognize and annotate genes in genome projects, a family of GeneMark programs have been developed including GeneMark.hmm [6], GeneMarkS [7], GeneMarkE [8], GenScan [9], EuGene [10], and GeneTack [11]. The

GeneMark family detects genes by identifying open reading frames (the regions between start and stop codons) using precomputed species-specific gene models as training data to determine parameters of the protein-coding and non-coding regions.

The major limitation of GeneMark family is the fixed-order Markov model such that models of higher order require exponentially more training data, which are usually not available for new sequences. The Glimmer gene-finding program [12] introduces a generalized hidden Markov model with variable order called the interpolated Markov model. Other gene-finding programs that based on the concept of interpolated Markov model and generalized Markov model include FGENESH [13], HMMGene [14], and AUGUSTUS [15].

Machine learning and data mining methods have been successfully applied to various kinds of prediction problems such as exon prediction [16], start codon prediction [17], and splice site prediction [18], [19]. More than 90% of nucleotides can be correctly identified as either coding, or non-coding. But the exact boundaries of the exons and their assemblies into complete coding regions are much more difficult to predict correctly using the classification-oriented formulation. DNA sequences are rather parsing-oriented in their nature.

We thus propose a novel setting of constraint logic programming to formulate a computational method toward the problem of gene searching and recognition in DNA sequences. This new scheme of DNA sequence analysis has just recently gained interest with some preliminary work appeared in the literature [20]-[22].

III. Methology

A. Constraint Programming for Computational Genome Analysis

Constraint programming is a programming paradigm normally applied to solve combinatorial search problems such as flight scheduling, crew rostering, logistic planning, and many more of this kind. The main steps of constraint programming are:

1) Users specify a problem by defining the variables together with their associated domains and constraints on these variables,
2) The search procedure and constraint solver find solutions, which are values assigned to the specified variables such that all constraints are satisfied.

It is obvious from the program structure that constraint programming has been designed to solve constraint satisfaction problems that have been extensively studied in artificial intelligence. The efficiency of constraint programs is basically due to the constraint propagator feature in a constraint solver. The function of constraint propagator is to reduce the domains of variables by inferring from the existing constraints and then to prevent the search procedure from visiting parts of the search tree that do not contain any solution.

A constraint propagator takes as input a domain D from which a variable can be assigned its value, and a set of constraints C. The output of the propagator is a reduced domain D′. For instance, given that X, Y, Z are variables, the domains

$$D(X) = \{a, c, d\},$$
$$D(Y) = \{a, b, c, d\},$$
$$D(Z) = \{c\},$$

and a set of constraints

$$C = \{ X=Y \wedge Y \neq Z \},$$

the output of the constraint propagator are

$$D'(X) = D'(Y) = \{a, d\}, \text{ and}$$
$$D'(Z) = \{c\}.$$

The repeated application of propagator can lead to increasingly stronger (that is, smaller) domains. The propagator continues until it reaches a fixed point in which the domains cannot be further reduced. At this stage, the search procedure (either global or heuristic-based) can efficiently start assigning possible value to each variable. A toy example of map coloring in Fig. 4 illustrates the constrain-and-search strategy of constraint logic programming (CLP), as opposed to the generate-and-test of logic programming (LP) scheme.

```
%% CLP style: Constrain-and-search
:- lib(fd).

map_color_CLP([A,B,C,D]) :-
          % declare variables and domains
      [A,B,C,D]:: red,green,blue,yellow],
          % constrain
      alldifferent([A,B,C,D]),
          % then search
      labeling([A,B,C,D]).
```

```
%% LP style: Generate-and-test
color(red).   color(green).
color(blue).  color(yellow).

map_color_LP([A,B,C,D]) :-
              % generate solution
     color(A), color(B),
     color(C), color(D),
              % then test for constraints
     A \= B,   A \= C,  A \= D,
     B \= C,   B \= D,  C \= D.
```

Fig. 4. Constraint logic programming versus logic programming schemes.

At present, there are several constraint systems that provide functions to specify (or model) the problems and maintain the constraint consistency efficiently. They are called constraint programming systems if they are based on procedural languages. The systems are classified as constraint logic programming systems if they are based on logic programming languages. The focus of this research is the development of constraint solvers (that is, the integration of constraint propagators and search procedures) for a specific application of genomic analysis using the constraint logic programming paradigm. The main benefits of such scheme are two folds:

1) the declarative style allows users to specify a problem itself, instead of specifying how to solve the problem, and

2) a high level of knowledge representation facilitates genomic pattern specification and the inclusion of new knowledge which is highly dynamic in the active area of genomics and computational microbiology.

Most constraint logic programming systems provide a large set of predefined constraints such as `alldifferent` and powerful search commands such as `labeling` to solve the combinatorial problems. The predefined constraints and exhaustive depth-first search procedure aim at solving a general class of constraint satisfaction problems. We argue that for a specific problem of genomic sequence analysis, a new set of built-in constraints that propagate with the already known biological relations together with alternative approximate search methods can more or less benefit the sequence analysis tool.

B. A Framework of Constraint-Based System

The main purpose of our research is the design and implementation of a computational system for genomic sequence detection and recognition of its structure and function using the declarative paradigm of constraint logic programming. The advantage of such paradigm is its powerful features to handle patterns within the DNA and RNA sequences. In addition, the heuristic search such as branch and bound, simulated annealing can be applied to speed up the computation time. The design is sketched as shown in Fig. 5.

A constraint-based approach to DNA sequence analysis starts from the modeling of sequence search and gene finding problems as constraint satisfaction problems, and also constraint optimization problems if preferences are to be numerically measured or when several solutions are generated. We name this step as "Query reformulator."

The constraints involved in the sequence analysis problems could be symbolic and numeric constraints over finite domains. The constraints formulated at this step can be either local or global constraints. Local constraints are restrictions over local patterns, whereas the global constraints address restrictions over the whole set of solutions.

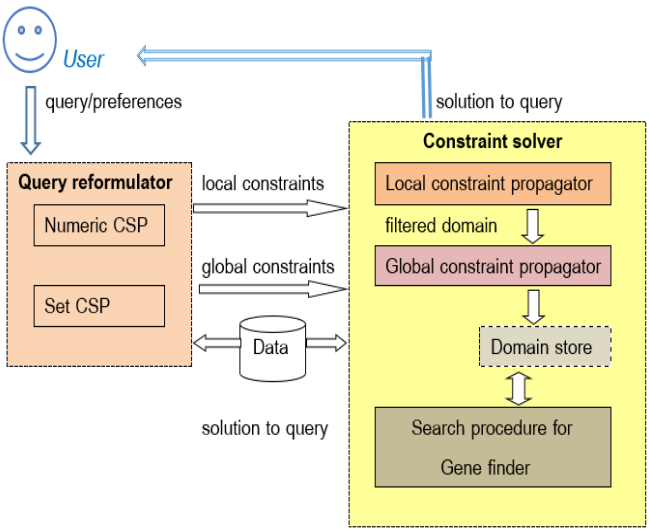

Fig. 5. Schematic overview of a constraint-based system for genomic sequence analysis.

In a subsequent phase of constraint solving, local constraints are handled prior to the global ones because local patterns under constraints can be checked independently of the other patterns holding in the data. The reduced domain of data values is then passed over to the global constraint propagator. The product of this step is the domain store to collect variable domains that their sizes have been reduced by the constraint propagators. The search procedure designed for gene-finding task can now start its process and report solutions to the user. The skeleton of constraint solver is shown in Fig. 6.

1. Develop the problem's model, in terms of variables and constraints, as a constraint satisfaction problem (or constraint optimization problem if some costs, distance measures, or other measurable metrics are specified)

2. Initialize for all variable-value pairs as modeled in step 1

3. Repeat until a termination condition is reached (this can be a maximum number of iteration, acceptable score range, or other conditions)

   3.1 Call the local constraint propagator to constrain the domain space of each variable and return a possibly smaller set of domains

   3.2 Call the global constraint propagator to further constrain the domain space

   3.3 Perform a search procedure (including a heuristic-based method)

      3.3.1 Select a variable

      3.3.2 Select a value from the domain

      3.3.3 Instantiate the variable

      3.3.4 If the instantiation fails (because constraint is violated), then backtrack to the step 3.3.2 and select another value

      3.3.5 If the sets of variables and their values are not empty yet, then repeat the steps 3.3.1-3.3.4

4. Return the solution (which is a set of variables and their values as modeled in step 1) if it exists

Fig. 6. A constraint solver algorithm for the genomic sequence analysis system.

IV. Conclusion

In the past, genes were identified with experimental validation on living cells and organisms. It is the most reliable method, but costly and labor intensive. At present, most biologists rely on the computational methods to automatically analyze the uncharacterized genomic sequences. Gene finders are programs that analyze and predict the exon-intron structures of genes using the sequences of one or more genomes as their only input. Many algorithms implement statistical and intelligent methods to represent sequence patterns and output a model of the gene structure. Some of the frequently applied techniques include dynamic programming, linear discriminant analysis, linguistic methods, hidden Markov model, and various machine learning techniques such as neural network, decision-tree induction, and support vector machines.

However, the insufficiency of known genes causes trouble to many algorithms to produce accurate prediction model. Some gene finders find most of the genes, but have a significant number of false positives. We thus propose a novel setting of constraint logic programming to formulate a

computational method toward the problem of gene searching and recognition in DNA sequences. This new scheme of DNA sequence analysis has just recently gained interest with some preliminary work appeared in the literature.

We concentrate our research study on the early biological process of gene detection and prediction because the understanding of gene structure and its function is important to the subsequent knowledge of protein analysis. Most of previous constraint-based work has based their constraint implementation on the constraint handling rules. The proposed techniques of our constraint solvers are mostly constraint propagation and search procedures embedded in the libraries of a finite and symbolic domains of the logic-based constraint system. Upon completion of this research project, we therefore expect to achieve some advancement to not only the computational gene-finding research area, but also to the constraint solving field.

ACKNOWLEDGMENT

This work has been supported by grants from the National Research Council of Thailand (NRCT) and the authors are supported by research funding from Suranaree University of Technology.

REFERENCES

[1] J. Watson and F. Crick, "A structure of deoxyribose nucleic acid," *Nature*, 1953, pp.171, 737.
[2] F. Crick, "Nucleic acids," *Scientific American*, vol. 197, 1957, pp. 188-200.
[3] F. Crick, "Central dogma of molecular biology," *Nature*, vol. 227, 1970, pp. 561-563.
[4] International Human Genome Sequencing Consortium, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, 2001, pp. 860-921.
[5] M. Borodovsky and J. McIninch, "GeneMark: parallel gene recognition for both DNA strands," *Computers & Chemistry*, vol. 17, no. 2, 1993, pp. 123-133.
[6] M. Borodovsky, A. Lomsadze, N. Ivanov, and R. Mills, "Eukaryotic gene prediction using GeneMark.hmm," *Current Protocols in Bioinformatics*, 2003, pp. 4.6.1-4.6.12.
[7] J. Besemer, A. Lomsadze, and M. Borodovsky, "GeneMarkS: a self-training method for prediction of gene starts in microbial genomes, implications for finding sequence motifs in regulatory regions," *Nucleic Acids Research*, vol. 29, no. 12, 2001, pp. 2607-2618.
[8] J. Besemer and M. Borodovsky, "GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses," *Nucleic Acids Research*, vol. 33, 2005, pp. W451-W454.
[9] C. Burge and S. Karlin, "Prediction of complete gene structures in human genomic DNA," *Journal of Molecular Biology*, vol. 268, 1997, pp. 78-94.
[10] T. Schiex, A. Moisan, and P. Rouze, "EuGene: an eukaryoric gene finder that combines several sources," in *Proc. JOBIM*, 2001, pp. 111-125.
[11] I. Antonov and M. Bodorovsky, "GeneTack: frameshift identification in protein-coding sequences by the Viterbi algorithm," *Journal of Bioinformatics and Computational Biology*, vol. 8, no. 3, 2010, pp. 535-551.
[12] A. Delcher, K. Bratke, E. Powers, and S. Salzberg, "Identifying bacterial genes and endosymbiont DNA with Glimmer," *Bioinformatics*, vol. 23, 2007, pp. 673-679.
[13] A. Rust, E. Mongin, and E. Birney, "Genome annotation techniques: new approaches and challenges," *Drug Discovery Today*, vol. 7, no. 11, 2002, pp. S70-S76.
[14] A. Krogh, "Two methods for improving performance of an HMM and their application for gene finding," in *Proc. Int. Conf. Intelligent Systems for Molecular Biology*, vol. 5, 1997, pp. 179-186.
[15] M. Stanke, O. Keller, I. Gunduz, A. Hayes, S. Waack, and B. Morgenstern, "AUGUSTUS: ab initio prediction of alternative transcripts," *Nucleic Acids Research*, vol. 34, 2006, pp. W435-W439.
[16] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," *Advances in Neural Information Processing Systems*, vol. 11, 1999, pp. 487-493.
[17] A. Zien, G. Ratsch, S. Mika, B. Scholkopf, T. Lengauer, and K. Muller, "Engineering support vector machine kernels that recognize translation initiation sites," *Bioinformatics*, vol. 16, 2000, pp. 799-807.
[18] N. Kerdprasop and K. Kerdprasop, "Recognizing DNA splice sites with the frequent pattern mining technique," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 5, no. 1, 2011, pp. 87-94.
[19] Y. Sun, X. Fan, and Y. Li, "Identifying splicing sites in eukaryotic RNA: support vector machine approach," *Computers in Biology and Medicine*, vol. 33, no. 1, 2003, pp. 17-29.
[20] H. Christiansen, "Logic-statistic modeling and analysis of biological sequence data: a research agenda," in *Proc. Int. Workshop Abduction and Induction in Artificial Intelligence*, 2007, pp. 42-49.
[21] C. Rigotti, I. Mitasiunaite, J. Besson, L. Meyniel, J. Boulicaut, and O. Gandrillon, "Using a solver over the string pattern domain to analyze gene promoter sequences," in S. Dzeroski (ed.), *Inductive Databases and Constraint-Based Data Mining*, Springer-Verlag, 2010, pp. 407-423.
[22] R. Yap, "Parametric sequence alignment with constraints," *Constraints*, vol. 6, 2001, pp. 157-172.

Nittaya Kerdprasop is an associate professor at the School of Computer Engineering, Suranaree University of Technology, Thailand. She received her bachelor degree in radiation techniques from Mahidol University, Thailand, in 1985, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, U.S.A, in 1999. Her research of interest includes knowledge discovery in databases, artificial intelligence, logic programming, and intelligent databases.

Kittisak Kerdprasop is an associate professor and chair of the School of Computer Engineering, Suranaree University of Technology, Thailand. He received his bachelor degree in mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, USA, in 1999. His current research includes data mining, artificial intelligence, functional and logic programming languages, computational statistics.