International Journal of Computing Science and Information Technology, 2013, Vol. 01 (01), 10-16
ISSN: 2278-9669, January 2013 (http://ijcsit.org)

Finite state machines implementation using DNA Techniques

Abeer ESHRA
Computer Science & Eng. Dept., Faculty of Electronic Eng., Menoufiya University, 32952 Menouf, Egypt.
abeerabdelazeez@hotmail.com

Ayman EL-SAYED (IEEE Senior Member)
Computer Science & Eng. Dept., Faculty of Electronic Eng., Menoufiya University, 32952 Menouf, Egypt.
ayman.elsayed@el-eng.menofia.edu.eg

Abstract

A finite-state machine (FSM) is an abstract mathematical model of computation used to design both computer programs and sequential logic circuits. Considered as an abstract model of computation, the finite state machine is weak; it has less computational power than some other models of computation such as the Turing machine. This paper overview the finite-state automata based on Deoxyribonucleic Acid (DNA). Such automata uses massive parallel processing offered by molecular approach for computation and exhibits a number of advantages over traditional electronic implementations. Different implementations of DNA finite state machines are discusses, such as Restriction Enzymes Finite State Machines, DNAzymes Finite State Machines, and Finite State Machines with DNA Polymers. Moreover, a comparison was made to clarify the advantages and disadvantages of each kind of presented DNA finite state machines.

Keywords:: Finite state automata, DNA computing, DNAzymes, Restriction enzymes

1. Introduction

To start with, it is necessary to point out what is DNA, its usage in deferent areas of Biocomputing in general, and specifically point outwhy it was used in finite state machines. DNA (Deoxyribonucleic Acid) is a nucleic acid [1], which contains certain genetic instructions, used for the functioning of living organisms. DNA computing is fundamentally similar to parallel computing in that it takes advantage of the many different molecules of DNA to try many different possibilities at once. For certain specialized problems, DNA computers are faster and smaller than any other computer built so far. Furthermore, particular mathematical computations have been demonstrated to work on a DNA computer. [2, 3].

DNA nanotechnology uses the unique molecular recognition properties of DNA and other nucleic acids to create self-assembling branched DNA complexes with useful properties. DNA is thus used as a structural material rather than as a carrier of biological information.Researches recently focus on utilizing biomolecules to develop nanostructures that simulates some machines functions. Some examples of those machines are tweezers, motors and walkers [4]. In the case of finite state machines, the segments of DNA are carrying the genetic information and this function is used in machine implementation [5].

Designing and further implementation of correct, robust DNA machines is rather difficult because there are quite many opportunities for unnecessaryintrusion between molecules in this system. DNA string displacement was proposed as a design paradigm for finite state machines designed with DNA, and the DNA strand displacement (DSD) [6]programming language was developed as a major means of officially programming and analyzing these finite machines to check for unnecessary interference[7].

This paper aims to comparing the different implementations of finite state machines using DNA, showing their differences and their benefits. We identify, classify, and discuss different implementations of DNA finite state machines such as Restriction Enzymes Finite State Machines (section 4.1), DNAzymes Finite State Machines (section 4.2) and Finite State Machines with DNA Polymers (section 4.3).

This paper is organized as follows. Section 2 provides a brief description of the finite state machines. The history of using DNA in computing is described in section 3.In section 4; we compare the different DNA finite state machine implementations and some examples for using DNA finite state machine in building nano-devices like tweezers. In section 5 we discuss using finite state machines in modeling and analyzing DNA. Finally we make suggestions for farther research aspects in section6.

2. Finite State Machines

A Turing machine[8], the origin of finite state automata, is a model of computation, to represent and perform a given computation. Turing machines are automatically equivalent to many other models of computation like cellular automata, neural networks, and digital computers. It is believed that Turing machines embody what is meant by something is computable. Anything can be computed by a Turing machine if a procedure or an algorithm can be written for it.

Finite state automata are significant in many different areas, including electrical engineering, linguistics, computer science, philosophy, biology, mathematics, and logic. Finite state machines are a class of automata studied in automata theory and the theory of computation[9]. In computer science, finite state machines are widely used in modeling of application behavior, design of hardware digital systems, software engineering, compilers, network protocols, and the study of computation and languages. An example of a finite state machine is given in Figure 1.

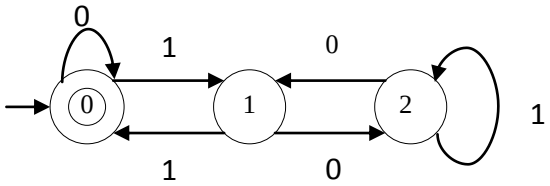

Figure 1: Example of a finite-state machine for binary divisibility by 3

This is a FSM that accepts strings formed with input alphabet {0, 1}. It accepts exactly those strings that are a numeral representing a multiple of 3 in binary, least-significant digit first. For example: the accepted strings include: 0, 11, 110, 1001, 1100, 1111, 10010 ... etc.

3. Earlier work with DNA in Computer Engineering

Adelman demonstrated how the actual mechanisms underlying recombination and separation of DNA carry computations significant to human endeavours[10], such as solving examples of the HAMILTONIAN PATH problem [10]. The technique to use them to carry out computation consists of three main steps: (1)encoding that represents the problem onto DNA strings[11], (2)hybridization/ligation that performs the processing of the main core and (3)extraction that makes the results evident and noticeable to the naked eye [12].

The biggest part of research is now being done to comprehend the reliability and feasibility of the techniques for pushing the restrictions of viable computation, a very important step in its development. One more research in this sphere attempts to characterize the power of computations of DNA. The starting point of such research was naturally, the evaluation and comparison with the standard computational framework, which is provided by Turing machines and classical computation[13]. It is clear that, even if DNA is capable to process information in ways that may not be captured by Turing's framework, in means or effectiveness[14]. Establishing the limits of DNA's power of computing requires the dual approach of mapping Turing computability and complexity into computing of DNA [15].

4. DNA Implementation of Finite-State Machines

In[16], as taken to be under discussion, the authors state that the basic information processing capabilities of DNA based reactions have been properly explored at the upper end of the computability spectrum using splicing systems and establishing computation universality. They investigated the information processing capabilities and competence of DNA computations from the other end by giving two different kinds of implementations of the simplest nontrivial information processingmodel,the finite-state machine[17]. A ligation-based approach permits input ofarbitrary length and can be readily implemented with present biotechnology, but requires sequential input feed anddifferent moleculesfordifferent machines[18]. Ina second implementation not based on ligation, transitions are representedby the reusable molecules, and the input, coded as amolecule, can be introduced at once[19].Both implementations allow optical extraction of the status of thefinite machine[12].

The authors of the article[16] and the researchers who conducted investigations state that in question of DNA designed finite machines, from the practical viewpoint, more significant still, is to recognize the variety of feasible tasks that DNA computing can perform competently and reliably under realistic assumption the chemical environments where the DNA computations are taking place. Several algorithms such as binary arithmetic [20], real-valued multiplication [21,22], breadth-first search [23]. And dynamic programming [24] were actively implemented using DNA.

Inthe article under discussion,the authorstooka different courseof action torealizingthereal power of DNA computing byinvestigating itsrelationswiththe classes of low level complexity. Inparticular, the researchersexploretherecognition of regular languages, awell-known andproperly realizedcomplexity classwith agreat variety and wide rangeofverypracticalapplications.In the article it is shown that two main implementations of DNA are

realistic and can be competently implemented in vitro[25, 26].The designs in the article under discussion are intended to serve as a generic algorithm for implementation of a deterministic finite state machine (FSM) using DNA processes[25].

4.1. Restriction Enzymes Finite State Machines

In [25], the authors managed to build programmable finite machines including DNA and DNA-manipulating enzymes that are able to solve computational problems autonomously. The hardware of machine consists of a restriction nuclease and ligase; double-stranded DNA encodes software and input, moreover, it contains programming amounts to selecting suitable software molecules. By mixing solutions containing those components, the machine processes the input molecule through a flow of restriction, hybridization and ligation cycles, thus it produces detectable output molecule, which encodes the final state of machine, and therefore computational result. The machine hardware consists of a mixture of the class IIS restriction nuclease FokI, T4 DNA ligase and ATP, while the software comprises eight short double-stranded (ds) DNA molecules, the `transition molecules', which encode all possible transition rules.

In [27] two new models are presented for finite state machine implementation with DNA. The operations used in both models are simple and easy to implement. Operations include immobilization of DNA strands onto paramagnetic beads, DNA hybridization, DNA ligation and restriction enzyme cleavage. In the first model, the size of the molecules representing the finite state control depends on the length of the input string. In the second model, obstacles caused by increasing lengths of the input string are discarded. Adding an enzymatic reaction to the operations of the first model, resulted in remaining the length of the DNA attached to the beads unchanged before and after each step of the algorithm and, therefore, it remains independent of the length of the input string.

There are certain restrictions of enzyme usage while implementation of DNA finite state machines. When two ends annealed another enzyme, DNA ligase, may be applied. The cuts in the backbone are repaired by means of DNA ligase and long piece of double-stranded DNA is created. In [28] the author proposed an encoding for a Turing machine transition table in DNA and series of restrictions and ligations. The author claims that every operation can be performed using commercially available restriction enzymes and ligases. That claim goes to the invoking imaginary enzymes to perform the state-symbol transitions in Charles Bennett's DNA based Turing machines.

4.2. DNAzymes Finite State Machines

DNA-based synthetic molecular devices are relatively simple to design and engineer, because of the predictable secondary structure of DNA nanostructures and the good biochemistry used to control DNA nanostructures. Though, ideally the designers try to minimize the use of protein enzymes in DNA-based synthetic molecular device design. Therefore, a class of DNA-based molecular devices using DNAzymes is presented in [26]. These DNAzymes-based devices are independent, programmable, and do not require protein enzymes. The DNAzymes-based designs presented in that research [26] are: finite state automaton, DNAzymes FSA (it performs finite state transitions using DNAzymes); extensions to it including probabilistic automaton and non-deterministic automaton, and its application as a DNAzymes router for programmable routing of nanostructures on a 2D DNA addressable network.

It is clear that smart nano-mechanical devices operating in an autonomous way interests numerous scientists. Recent successes in creating DNA nano-structures of large scale, in constructing DNA machines, provide a solid foundation for the next step forward: creating autonomous DNA mechanical devices capable of arbitrarily compound behavior. One prototype system in the direction of this objective can be autonomous DNA mechanical device competent for universal computation, by imitating the actions of universal Turing machine. Building on a previous work of [29]'s authors, as a theoretical design and prototype experimental construction of an autonomous unidirectional DNA walking device moving along a linear track, the authors presented the design of a nano-mechanical DNA device that autonomously mimics the operation of a 2-state 5-color universal Turing machine. The autonomous nano-mechanical device, called an Autonomous DNA Turing Machine, is thus capable of universal computation and hence complex translational motion, which is defined as universal translational motion.

4.3. Finite State Machines with DNA Polymers

In [30]the authors propose a chemical implementation of stack machines — a Turing-universal model of computation similar to Turing machines- using DNA strand displacement cascades as the underlying chemical primitive. More specifically, the mechanism they described is the addition and removal of monomers (single unjoined organic molecules) from the end of a DNA polymer, controlled by strand displacement logic. Bennett's proposed chemical Turing machine[31] is one of the most important thought experiments in the study of the thermodynamics of computation. Yet the sophistication of molecular engineering required to physically construct Ben-

nett's hypothetical polymer substrate and enzymes has prevented experimental implementations. The authors contributed also in capturing the motivating feature of Bennett's scheme: that physical reversibility corresponds to logically reversible computation, and arbitrarily little energy per computation step is required. Further, as a method of embedding logic control into chemical and biological systems, polymer-based chemical computation is significantly more efficient than geometry-free chemical reaction networks. However their construction lacks the attractive feature of material recycling. Yet, in their scheme, different fuel molecules would be used in the "compute" and "retrace" phases of the transformed Turing machine computation, and would not be regenerated.

In [32] the authors overview a series of their research on implementing finite automata in vitro and in vivo in the framework of DNA-based computing. First, they employ the length-encoding technique presented in [33, 34] to implement finite automata in test tube. In the length-encoding method, the states and state transition functions of a target finite automaton are effectively encoded into DNA sequences, a computation (accepting) process of finite automata is accomplished by self-assembly of encoded complementary DNA strands, and the acceptance of an input string is determined by the detection of a completely hybridized double-strand DNA. Secondly, they report their intensive in vitro experiments in which they have implemented and executed several finite-state automata in test tube. They have carried laboratory experiments on various finite automata of from 2 states to 6 states for several input strings.

4.4. Using DNA FSM to build nano-devices

As an application for the DNAzymes Finite state machine discussed earlier, in [26] the authors proposed a DNAzymes router for programmable routing of nanostructures on a 2D DNA addressable lattice. Furthermore, they gave a medical-related application, DNAzymes doctor that provide transduction of nucleic acid expression: it can be programmed to respond to the under-expression or over-expression of various strands of RNA, with a response by release of RNA.

DNA finite state automata can be used to build DNA tweezers. In [35] the authors built three DNA tweezers that are activated by the inputs H+/OH-; Hg2+/cysteine; nucleic acid linker/complementary anti-linker to yield a 16-states finite-state automaton. The outputs of the automata are the configuration of the respective tweezers (opened or closed) determined by observing fluorescence from a fluorophore/quencher pair at the end of the arms of the tweezers. The system exhibits a memory because each current state and output depends not only on the source configuration but also on past states and inputs.

5. Using Finite state machines in modelingand Analyzing DNA

Another direction of using DNA finite state machines is to train a FSM as a good/bad classifier for PCR (polymerase chain reaction) primers [36]. PCR primers are short sequences of DNA used in a reaction that amplifies other DNA. PCR amplification of DNA underlies a multitude of technologies from forensic DNA fingerprinting to genetic mapping. The system presented in [36] is a post-production add-on to a standard primer picking program intended to compensate for organism and lab specific factors.

In [37] a project presents an updated method for classification of PCR primers in mice using finite state classifiers. Five different evolutionary algorithms that use an incremental fitness reward are used for training these classifiers.

Recently Finite state machines can be used to improve modeling and analysis of DNA properties and protein structure. In [38] the authors extends an early study on discrete events system formulations of DNA hybridization, and focuses discussions on gene mutation in Molecular Biology. Key concepts and analyzing the process related to those phenomena can be expressed by applying FSM theory.

6. Discussion

In Shapiro's restriction enzymes model application[25], the researchers implemented 1012 automata, sharing the same software, run independently and in parallel on inputs at a combined rate of 109 transitions per second with transition fidelity greater than 99.8%, consuming less than 10-10 W. However, in Shapiro's model it is noticed that final detection remains labor intensive and it contains programming amounts to selecting suitable software molecules.

In Nowzari's models [27], the use of paramagnetic beads greatly reduces performance time and demonstrates DNA chip compatibility of the models. In one of the models, the lengths of DNA strands created during the intermediate operations are independent of the length of the input string. The three operations used in the first algorithm, ligation, hybridization and optical extraction, are all very simple. Also, the use of biotin labeled DNA and streptavidin-coated paramagnetic beads allows each cycle to be performed within minutes manually. However, a very attractive feature of the system is its automation, which would allow hundreds of thousands input sequences to be analyzed simultaneously and rapidly. Finally, the two models are simple models to implement in the laboratory. The implementations of both models can be further made fault-tolerant and can be easily used for

implementing nondeterministic models. Optical extraction in both models detects the final state which can be considered much easier than Shapiro's result detection.

In Rothemund's proposal [28], the researchers assigned real DNA sequences to their schematic that could be used with commercially available enzymes to implement a Turing machine in lab. They recognized that the Turing machine model, while useful for proving theoretical results, may be too slow to do any practical computation. This proposal can be considered as an open a door for other researchers to come up with a series of operations from which one can build a practical universal molecular computer.

The DNAzymes-based devices are independent, programmable, do not require protein enzymes, and allow for the execution of multiple state transitions. In the DNAzymes FSA, the number of DNAzymes required is proportional to the number of transitions in the automata. For binary-coded inputs the number of transitions is proportional to number of states. However, the implementation of finite state machines that do not have a planar layout might be challenging.

Different architectures for molecular computing such as algorithmic self-assembly, circuits implemented with chemical reaction networks (CRNs). Turing machines implemented with CRNs and polymer CRNs embody different tradeoffs between time, volume, energy and uniformity. In [30] the proposed construction is exponentially more efficient in terms of the required molecular counts and volume than geometry-free Turing-universal computation using strand displacement reactions, and also polynomially faster. Moreover, their polymer CRNconstruction in theory yields the correct computation output with probability 1. However, the construction lacks the attractive feature of material recycling.

Another direction takes the DNA FSM to improve primer design performance by using machine learning as a latch on post-processing to capture features of primer performance not related directly to the DNA biophysics already implanted in primer-picking packages. The technique applied in [36] appears to have yielded improved performance.In [37] updating methods of PCR primers classifications can compensate for many lab, organism and chemical specific factors that are costly. Using Finite State Classifiers can help decrease the number of primers that fail to amplify correctly. By controlling the fitness reward correctly, there is a potential to develop classifiers with a high probability of accepting only good primers.Modeling and analysis of DNA properties and protein structure can be improved also by using finite state machines. In [38]the researchers managed to mathematically represent and interpret metabolism and the effects to structures of protein macro molecule caused by gene mutation.

7. Conclusion

FSA has been challenging for conventional computers, in its programming and in its execution. The bigger the machine becomes the more execution time it costs. On the other hand, DNA has proved to have the ability of massive parallelism. So in the last few years, many approaches have been presented to use DNA in solving many computational problems including FSA. Three approaches have been discussed in this article with their pros and cons. The approaches are: Restriction Enzymes FSMs, DNAzymes FSMs, and DNA Polymers FSMs. After this comparison, we decided to go with DNAzymes FSM, make a design and experience it in the wet lab.

REFERENCES

[1] Watson, J.D. and F.H.C. Crick, Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. Nature, 1953. 171(4356): p. 737-738.

[2] Nayebi, A., Fast matrix multiplication techniques based on the Adleman-Lipton model. Matrix, 2009: p. 15.

[3] Qian, L. and E. Winfree, Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades. Science, 2011. 332(6034): p. 1196-1201.

[4] Beissenhirtz, M.K. and I. Willner, DNA-based machines. Organic & Biomolecular Chemistry, 2006. 4(18): p. 3392-3401.

[5] Gill, A., Introduction to the theory of finite-state machines / Arthur Gill. 2002, New York : McGraw-Hill.

[6] Green, S.J., D. Lubrich, and A.J. Turberfield, DNA Hairpins: Fuel for Autonomous DNA Devices. Biophysical journal, 2006. 91(8): p. 2966-2975.

[7] Lakin, M.R., et al., Visual DSD: a design and analysis tool for DNA strand displacement systems. Bioinformatics, 2011.

[8] Copeland, B.J., The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma 2004, Oxford UK: Clarendon Press (Oxford University Press).

[9] Rich, E.A., Automata, Computability and Complexity: Theory and Applications. 2007: Prentice Hall.

[10] Adleman, L.M., Molecular computation of solutions to combinatorial problems. Science, 1994. 266(11): p. 1021-1024.

[11] Liedl, T., T.L. Sobey, and F.C. Simmel, DNA-based nanodevices. Nano Today, 2007. 2(2): p. 36-41.

[12] Benenson, Y., et al., An autonomous molecular computer for logical control of gene expression. Nature, 2004. 429(6990): p. 423-429.

[13] Kohavi, Z. and N.K. Jha, Switching and finite automata theory. 3rd ed. 2010, Cambridge: Cambridge University Press.

[14] Clarke, E.M., O. Grumberg, and D.A. Peled, Model Checking. 2000, Cambridge: MA: MIT Press.

[15] Tian, Y. and C. Mao, Molecular gears: a pair of DNA circles continuously rolls against each other. J Am Chem Soc, 2004. 126(37): p. 11410-1.

[16] Rose, J.A., et al., DNA Implementation of Finite-State Machines, in Second Conference on Genetic Programming. 1997: Stanford University in Stanford, California.

[17] Seeman, N.C., From genes to machines: DNA nanomechanical devices. Trends Biochem Sci, 2005. 30(3): p. 119-25.

[18] Seelig, G., et al., Enzyme-Free Nucleic Acid Logic Circuits. Science, 2006. 314(5805): p. 1585-1588.

[19] Garzon, M. and E. Eberbach, Dynamical Implementation of Nondeterministic Automata and Concurrent Systems, in Revised Papers from the First International Workshop on Implementing Automata. 1997, Springer-Verlag. p. 35-49.

[20] Barua, R. and J. Misra, Binary Arithmetic for DNA Computers, in DNA Computing, M. Hagiya and A. Ohuchi, Editors. 2003, Springer Berlin Heidelberg. p. 124-132.

[21] Wu, G. and N. Seeman, Multiplying with DNA. Natural Computing, 2006. 5(4): p. 427-441.

[22] Nayebi, A., Fast matrix multiplication techniques based on the Adleman-Lipton model. International Journal of Computer Engineering Research, 2011. 3(1): p. 10-19.

[23] Ogihara, M., Breadth First Search 3SAT Algorithms for DNA Computers. 2004.

[24] B., E., Baum, and D. Boneh, Running dynamic programming algorithms on a DNA computer. In Proceedings of the Second Annual Meeting on DNA Based Computers, 1996: p. 141-147.

[25] Benenson, Y., et al., Programmable and autonomous computing machine made of biomolecules. Nature, 2001. 414(6862): p. 430-434.

[26] Reif, J. and S. Sahu, Autonomous programmable DNA nanorobotic devices using DNAzymes. Theoretical Computer Science, 2009. 410(15): p. 1428-1439.

[27] Nowzari-Dalini, A., et al., A New DNA Implementation of Finite State Machines. International Journal of Computer Science & Applications, 2006. 3(1): p. 51 -60.

[28] Rothemund, P.W.K. A DNA and restriction enzyme implementation of Turing Machines. in DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 2000.

[29] Yin, P., et al., Design of an autonomous DNA nanomechanical device capable of universal computation and universal translational motion, in Proceedings of the 10th international conference on DNA computing. 2005, Springer-Verlag: Milan, Italy. p. 426-444.

[30] Qian, L., D. Soloveichik, and E. Winfree, Efficient Turing-Universal Computation with DNA Polymers, in DNA Computing and Molecular Programming, Y. Sakakibara and Y. Mi, Editors. 2011, Springer Berlin / Heidelberg. p. 123-140.

[31] Bennett, C.H., et al., Thermodynamics of computation and information distance, in Proceedings of the twenty-fifth annual ACM symposium on Theory of computing. 1993, ACM: San Diego, California, United States. p. 21-30.

[32] Sakakibara, Y., Bio-molecular computing of finite-state machine, in Proceedings of the 3rd International Conference on Bio-Inspired Models of Network, Information and Computing Sytems. 2008, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Hyogo, Japan. p. 1-2.

[33] Yokomori, T., Y. Sakakibara, and S. Kobayashi, A magic pot: self-assembly computation revisited, in Formal and natural computing, B. Wilfried, et al., Editors. 2002, Springer-Verlag New York, Inc. p. 418-429.

[34] Sakakibara, Y. and T. Hohsaka, In Vitro Translation-Based Computations, in DNA Computing, J. Chen and J. Reif, Editors. 2004, Springer Berlin / Heidelberg. p. 1982-1982.

[35] Wang, Z.-G., et al., All-DNA finite-state automata with finite memory. Proceedings of the National Academy of Sciences, 2010. 107(51): p. 21996-22001.

[36] Ashlock, D., A. Wittrock, and W. Tsui-Jung. Training finite state machines to improve PCR primer design. in Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. 2002.

[37] Yadav, S.R. and S.M. Corns. Improved PCR design for mouse DNA by training finite state machines. in Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2010 IEEE Symposium on. 2010.

[38] Gao, R., W.-S. Hu, and C.-Q. Zhang, Modeling and Analysis of DNA Mutation Effects on Protein Structure with Finite State Machine, in Advances in Electronic Engineering, Commu-

nication and Management. Vol 2, D. Jin and S. Lin, Editors. 2012, Springer Berlin Heidelberg. p. 317-323.

Author Biography

Abeer ESHRA Demonstrator, of Computer Science and Engineering dept Faculty of Electronic Engineering, Menoufiya University, Egypt. She received her B.Sc. degree in computer science and engineering in 2006. She works as a demonstrator in Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufiya University, Egypt.

Ayman EL-SAYED Associate Professor, of Computer Science and Engineering dept. Faculty of Electronic Engineering, Menoufiya University, Egypt. He received his B.Sc. degree in computer science and engineering in 1994, his Master degree in computer networks in 2000 from the University of Menoufiya, Egypt, and his PhD degree in computer network in 2004 from ”Institute National De Polytechnique De Grenoble” INPG, France. He worked in Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufiya University, Egypt. Now he is working a head of Computer Science and Information System Department, Alquwayiyah Community College, Shaqra University, KSA. He is specialized in Soft Computing, Algorithms, and Data Structure. Also, his interests include multicast routing protocols, application-level multicast techniques, IP Trace back for security, multicast on both Mobile Network and Mobile IP, and Image processing techniques. Also, there are other interesting topics such as Bioinformatics, BioComputing, and BioComputer. He is an approved Supervisor for M.Sc and Ph.D Programmes in various University. He has completed various project in government and private organization. He has published more than 35 research papers in international Journals and two books about OSPF protocol and multicast protocols. Currently he is serving as Editorial Board Member in various international Journal and conferences.