26

Genetic Engineering in a Computer Science Curriculum

Nevena Ackovska[1], Liljana Bozinovska[2] and Stevo Bozinovski[2]
[1]*University Sts Cyril and Methodius, Institute of Informatics,*
[2]*South Carolina State University,*
[1]*Macedonia*
[2]*USA*

1. Introduction

Traditionally genetic engineering is understood as a molecular biology discipline. The tools used in molecular biology are specific, mostly used by people who come from biological or medical background, which made the discipline distant from classical Computer Science. In this paper we would like to address computer science auditorium and point out the importance of understanding genetic engineering.

Although the term "genetic engineering" was coined 1951 in a science fiction novel by Williamson (reprinted in (Williamson, 2002)), it was not until 1970s when the first achievements in DNA modification showed that genetic engineering is actually happening. As in all of the sciences, genetic engineering has its own milestones. In this introductory part we will describe the earliest genetic engineering achievements, using computer science terminology.

Through the prism of computer science terms, the best way to look at DNA is that it represents a string of letters, written in four letter alphabet. The string might be interpreted as a text subject to processes such as transcription and translation (Watson & Crick, 1953), or as database and software for processes in a cell (Bozinovski and Bozinovska, 1987; Bozinovski et al., 2000). The first genetic engineering achievement was the *cut-and-paste* operation of a DNA segment. Using two types of enzymes, restriction enzymes for cut operation, and DNA ligase for join operation, a segment from one DNA was cut and pasted in another DNA. Actually, the achievement was greater, since the DNA of two living forms, the bacterial phage lambda and monkey virus SV40, were (re)combined into a new DNA (Jackson et al., 1972). The second achievement, in 1973, was *prepare-and-copy* operation. A DNA fragment was inserted in a plasmid (pSC101) and put into a bacterium (Escherichia coli) and was replicated inside the bacterium. This experiment has another important point: a fragment was prepared outside a cell (in vitro) and was replicated inside a cell (in vivo). The cell machinery processed (replicated) the foreign piece of software as its own. That was the first step or cell (re)programming. In 1974 the first transgenic animal was created, by inserting a foreign DNA into a mouse embryo. In 1977 and 1978 the first human proteins (somatostatin and insulin) were produced inside a bacterium. Bacteria produced human insulin become commercially available in 1982 opening market for various types of genetically engineered products, making genetic engineering a new industry. Important

achievement happened in 2010 by which a complete synthesized genome was introduced into a bacterial cell which had no DNA (Gibson et al., 2010). Therefore, a new life form was created with complete software prepared outside the cell. The bacterium was named Synthia and is the first synthetic life form. All of these examples and many more, represent the milestones that the science and technology of genetic engineering have already made possible for humans to use.

As computer engineers, it is our view that genetic engineering is a type of (software) engineering, and the way of doing genetic engineering is a way of programming and reprogramming a DNA. As today many computer science curricula contain a bioinformatics course, we argue that Computer Science curriculum should contain courses in genetic engineering as well. Since Computer science in part is about programming, genetic reprogramming can be viewed as important part of the Computer Science education. We will also present our experience in teaching genetic engineering to computer science students.

In this paper we will first describe natural way of doing genetic engineering. That is the way the Nature was doing genetic engineering long before it became known to humans. Then, we will describe a metaphor that can be used in education of Computer Science students. Afterwards we will describe our approach to introducing genetic engineering into Computer Science curriculum, including also lab exercises.

This paper addresses mainly the computer science auditorium. We present some elementary knowledge in molecular biology, but the concepts are presented through (computer) engineering terms. However, the notion presented here might be of interest for other scientists coming from biological sciences background.

2. Genetic engineering before Genetic Engineering

Genetic engineering, by human definition, is a process of human produced genotypic effect in order to obtain some phenotypic effects. Phenotypic effects can be at molecular level, such as production of a new protein, or at higher level, producing visible effect either in the structure or in the behavior of an organism. Usually, it is achieved by planned DNA alternation, in order to (re)program behavior of a cell or a multicellular organism.

The nature has been doing genetic engineering for a very long time. Some of the life forms on the Earth survive using sort of "genetic engineering". The question we will start our presentation here is how we can help Computer Science students to understand the concept of Genetic Engineering? We believe that the following story of phage lambda is an inspirational approach toward understanding genetic engineering for computer science students and professionals alike, if we can level the terminology used in classical Genetic engineering to Computer Science. As we have stated before, the way the story is been transferred to Computer Science students enables them to understand the life cycle of phage lambda. The simplicity and the clarity of the terminology seem to be of high importance in order for these students to understand complex biological processes. Even more, the story explains one way of genetic engineering done by the Nature itself.

"Consider a life form, a bacterium named Escherichia coli. It is a prokaryote, it does not have a nucleus inside the cell. It is a life form that is capable of self reproduction. We call it a single cell organism. Its chromosome is a circular one. Every twenty minutes it replicates

itself, provided there are favorable conditions in the environment. The new bacterium is pretty much the copy of the previous one.

Now consider another life form, namely phage λ (lambda). It is a life form which cannot reproduce itself. So, the phage lambda needs a host organism to reproduce and E. coli is such a host. A phage (or bacteriophage) is a virus to a bacterium: it can replicate inside a bacterium and eventually destroy it. It is interesting that a phage is harmless to human cells and to all other eukaryotic cells (cell that do have a nucleus).

The phage lambda is a life form with head-and-tail appearance. The head contains the DNA as a single chromosome. The phage DNA is double stranded. It contains the phage genome, i.e. the set of all phage genes. The tail is used to insert the DNA into a bacterial cell. Once inside a bacterium, the phage chromosome exhibits two possible behaviors: lysogenic and lytic. Lysogenic behavior occurs if the phage chromosome remains its linear form. It then integrates into the bacterial chromosome and becomes segment of that chromosome. When bacterium replicates its chromosome, the phage segment is also replicated. However, under some condition, such as environment stress, linear phage chromosome leaves the bacterium chromosome, and exhibits lytic behavior. It is not necessary that a phage has a lysogenic behavior; it might start its lytic behavior immediately after entering a bacterial cell. The lytic behavior starts with transforming the linear phage chromosome into a circular one. That is preprogrammed into the sticky ends of both side of the linear chromosome. The site of sticking the both strands together is named a cos site. Once into its circular form phage chromosome replicates inside bacterium and creates new phage life forms. Eventually bacterium explodes releasing phages into the environment outside the bacterium.

Since the replication inside a bacterium is lethal for the bacterium, a simple immune system was designed in bacterial evolution: restriction enzymes. Restriction enzymes (or endonucleases) are enzymes able to recognize a foreign DNA and cut it, rendering it non reproducible. Restriction enzymes are probably the first form of immune system in organisms."

The story of phage lambda seems to be very simple. However, it explains one of the Nature's life cycles in a simplified manner, understandable for engineering students. From the above example the students can learn about three mechanisms how a bacterial genome can be modified:

1. by incorporating a linear DNA segment into the bacterial chromosome,
2. by adding a separate circular chromosome into the bacterium and creating a two-chromosome system inside the bacterium,
3. by producing environment stress that would activate some genetic response of inserted DNA segments.

In the following section we elaborate on other terminology modification in order to explain processes and actors in genetic engineering to (computer) engineers.

3. Leveling the terminology: Metaphors for understanding genetics

For computer science, genetic engineering and cell (re)programming can be viewed as kind of software engineering. A software sequence is written, and then inserted into genetic

machinery for compilation and execution. Now, if we "translate" the biological processes into terminology that computer science students understand, we obtain greater results and better understanding of these rather complicated processes. In the sequel we will consider some metaphors that can be used in teaching computer science students concepts of genetic engineering.

A metaphor is understood as a paradigm transformation; using knowledge from a familiar system in order to understand the phenomena in another system. The first metaphor explaining genetic processes was the biochemistry metaphor, which basically relied on the fact that DNA is an acid. Obviously the acid metaphor was not good enough to explain the life processes; the fact that the DNA is an acid cannot explain information that is stored into a DNA. Another metaphor proposed in 1953 (Watson & Crick, 1953) stated that the DNA is a sequence of letters, actually a text where information is stored. Today it is a dominant metaphor. The principal processes named transcription and translation (Crick, 1958) are linguistic, text processing terms. In 1985 the relation between genetic engineering and robotics was pointed out (Bozinovski, 1985). An observation that DNA is actually a database was first made in 1987 (Bozinovski, 1987; Bozinovski & Bozinovska, 1987; Demeester et al. 2004; Pirim, 2005). Afterwards, new metaphor for genetic engineering was proposed, the robotics and flexible manufacturing metaphor, which proposed a viewpoint that the cell is a flexible manufacturing system. According to that metaphor, some molecular structures should be viewed as cell robots, an example being the tRNA, which is a transporting robot. Related to the flexible manufacturing metaphor is the systems software metaphor (Bozinovski et al. 2000; Bozinovski et al 2001; Danchin & Noria 2004, Ackovska & Bozinovski, 2008; Ackovska et al. 2008).

The latest metaphor is very comprehendible for computer science and computer engineering students. It uses the concept of a genetic file as a logical segment on DNA. In the cell processes related to manufacturing (e.g. protein biosynthesis) the genetic files are considered existing and read-only. In this paper we are focused on files that can be altered, such as updated, created, written, inserted into another files, and otherwise manipulated. This also happens in nature, but more importantly, it is a basis of genetic engineering. In the following section we address the issue of writing in genetic files, and creating genetic file systems and genetic disks.

4. Genetic files, disks, and genetic file systems

When studying genetics engineering and genetics in general, the crucial concept is the concept of gene. Thus, a very natural question is "What is a gene?" A usual answer is that a gene is a segment of a DNA that encodes for either protein or RNA. Also, one could encounter slightly different definitions (Brown, 2002).

In computer science and engineering a usual reasoning on an information processing system considers the files of that system. So, for genetic information processing we might ask the question "what are the files of the genetic information processing system?" Is the concept of a gene corresponding to the concept of a file? Having that as a starting point, in this section we will present our understanding of DNA organization and DNA computing in terms of files and related concepts.

Looking for a concept of a file in DNA, we found that the transcription units (or scriptons (Ratner, 1975)) are analogous to cell files. A transcription unit is a segment of DNA that eventually becomes transcribed to RNA. In prokaryotes, a transcription unit often produces a transcript with several genes (so-called polycistronic RNA). In eukaryotes it produces a precursor RNA (pre-RNA), which contains the information about a single gene, but in order to obtain it, additional processing needs to be performed.

The eukaryotic files are rather complex and contain segments of a gene, interleaved with segments that do not belong to the gene. Those segments are known as introns, as opposite to exons (gene expressing segments). To the people involved with genetics, there is a standard question considering this phenomenon: how did it happen that eukaryotic genes became segmented? However, for computer engineers introduced to the concept of a file, the answer is straightforward – busy files are fragmented. Defragmentation is sometimes needed in computer file systems. Moreover, it is expected that between two fragments of a file an entire different file could be expected. This fact points to the concept of distributed file systems (Nutt, 1992; Tanenbaum, 1994; Tanenbaum & Van Steen, 2007). And indeed, this is the case in molecular genetics: After the first evidence that Tetrahymena ribozyme is actually an intron (Kruger et al., 1982), more evidence has been found that genetic files could be found within a complete different file (Been, 2006).

The cell files contain genes and other important sequences. Some files are executable ones, they will produce cell robots. Cell robots are either enzymes (protein based) or ribozymes (RNA based enzymes). Besides genes, which contain program files, the file system contains files that are not genes; they are data structures, some of which can be used as template for pattern recognition.

We believe that while the genes are the proper concept when talking about heredity, the concept of a file is very useful in describing the DNA transcription process. This makes the first step in the analogy between the computer systems and genetic systems. The cell, especially the eukaryotic cell, undergoes extensive file processing: from copying the pre-RNA file until obtaining the RNA message. This process includes operations like: cut (introns), join (exons), right append (trailer string), left append (header string), letter replacement and so on, which are standard file processing operations in every modern computer operating system (Bozinovski et al., 2001).

Under genetic disk (or cell disk) we understand a cell chromosome. For example, human genome is a distributed file system that resides on 46 (or 23 pairs) disks in each cell. There are exceptions, for example gonad cells have only 23 cell disks. In some organisms besides main genomic system there is a satellite chromosome system. An example is the bacterium E. coli which contains its chromosome, but it can also contain satellite chromosomes from life forms such as phages and plasmids. So, under the concept of genetic disk we include both main, cell-replicating chromosomes, as well as the satellite, independently replicating chromosomes, such as plasmid chromosomes or phage chromosomes. Examples of satellite chromosomes in eukaryotic cells are mitochondria DNA.

We define disk segment as a set of files that will be written on a genetic disk. Under the genetic file system we understand ordered set of genetic files. Usually, genetic disks contain files in strict order. Our approach understands genome (set of all genes in a life form) as part of the cell file system.

Therefore, this approach toward genetic engineering starts with understanding that the DNA is cell operating system which resides on cell disks (chromosomes). Set of chromosomes can be viewed as an array of genetic disks.

In the sequel we will often use the terminology of genetic disks and disk segments to refer to a chromosome or DNA sequence. Here we will first describe the phage lambda and its genetic disk. The DNA of phage lambda is double stranded and circular one. A double stranded DNA allows storing gens on both strands, so that gens can be copied and processed by reading them on both sides, in opposite directions. Figure 1 describes behaviour of a phage DNA in a cell of bacterium E. coli.

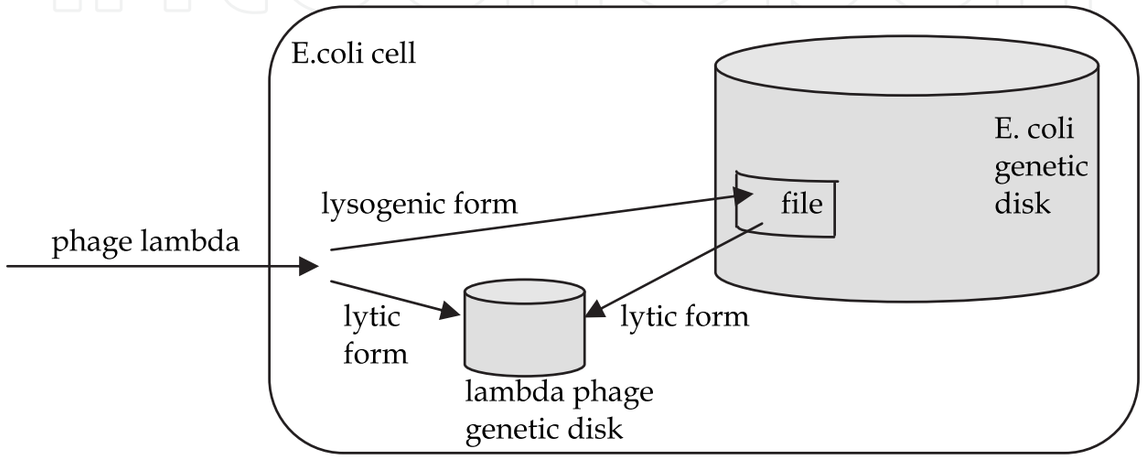


Fig. 1. Phage lambda DNA enters into an E. coli bacterium. The lambda DNA either becomes a file into the bacterium system disk or becomes own system disk.

Figure 1 shows two forms of existence of phage lambda DNA in a bacterium E. coli. It may integrate, as a disk segment (its linear form) into the E.coli DNA disk (lysogenic existence). However, it may encircle (its circular form) and form its own system disk (lytic existence). Once becoming a system disk, it replicates and forms new phages.

The phage lambda genetic disk is a file system that contains set of genes (genome) that can be divided into functional groups. According to the systems software metaphor the genes are viewed as source program files that are compiled into proteins. The phage genetic disk also contains two data files and they are not compiled.

5. Education of computer science students in genetic engineering

Contemporary education of computer science students is often related through molecular genetic through various forms of Bioinformatics courses. Bioinformatics is about genetic sequences that are stored on databases, usually available on the World Wide Web. Many institutions offer digital encyclopedias related to molecular biology. Many applications are built for using the knowledge stored in databases, such as searching various forms of similarities among genetic sequences and predicting genes in a genetic sequence (Xiong, 2006). The goal of post genome informatics (Kanehisa, 2000) is to understand the information in genetic sequences, including function of all the genes and other functional sentences. At this point, Genetic Engineering is usually not part of bioinformatics courses.

We propose that genetic engineering should be included in Computer Science curriculum. One way of doing that is through the existing bioinformatics related courses. Example is the elective undergraduate course CS495 Biocomputing and Bioinformatics which is part of the Computer Science curriculum of the South Carolina State University, or the courses Intelligent Systems (Madevska-Bogdanova & Ackovska, 2009), DNA Programming and Bioinformatics at the Institute of Informatics, University Sts Cyril and Methodius. The other approach is introducing a separate course.

In genetic engineering related course students will be able to learn programming life forms. So far, they are capable of programming robots through various type of robotics courses contemporary found in Computer Science curricula. However, Genetic engineering is a way of designing, writing, and executing programs for living beings: it is about designing new genes and genomes and consequently, their phenotypes. Therefore it is interesting for the students to learn how to program DNA in order to design life form robots.

The core of genetic engineering is creation of a file or set of files that will be written on a genetic disk. There are several ways how to obtain a genetic file. Examples are: 1) cut a file or segment from existing disk (using restriction enzymes), 2) copy a file from existing disk (copy on mRNA and then synthesize complementary DNA, cDNA), and 3) synthesize a human made (artificial) file or segment.

In this section we will be focused on genetic disk segments, and genetic tools as ways of creating genetic source programs that would be executed by the cell. We shall explain the way we represent the theoretical knowledge, including restriction enzymes, engineered genetics disks, artificial chromosomes and the process of transferring the source genes into host systems and creating genomic libraries.

5.1 Theoretical knowledge

When creating a curriculum it is always a question which topics should be primarily covered. When designing an interdisciplinary course, such as Genetics Engineering for Computer science students, it is even more difficult to make such a decision. It would depend on how much time or space the instructor has available for the course. The course can be separate, usually graduate course, for example on Physiology Engineering (Bozinovski and Bozinovska, 2011), or it can a part of an undergraduate course, for example on Bioinformatics. In any case, organization of genetic files, genetic disks, and related operating systems and robotics metaphors for understanding genetics is a good introduction to the subject. Other topics might include: sequencing, amplification, modifying enzymes, cloning, screening, applications, and state of the art. A good textbook might be Nichol's book (Nicholl, 2008).

5.1.1 A tool for genetic engineering: restriction enzymes

One of most used DNA modifying enzymes are restriction enzymes. They are tools for cutting a DNA string. Restriction enzymes, also known as endonucleases, are DNA cutting bionanorobots.

Restriction enzymes are naturally used by bacteria which use them as a natural defense mechanism to cut an invading phage DNA. Many bacterial restriction enzymes have been

found and they are named according to the bacteria they are isolated from and the order (first, second, third etc.) in which they are isolated. For example EcoRI means first isolated restriction enzyme from *Esherichia coli*, HindIII means the third isolated restriction enzyme from bacteria *Haemophilus influenzae*, and PstI means the first restriction enzyme extracted from *Providencia stuartii*.

Genetic engineering relies upon ability to cleave (cut, splice) and ligate (paste) a functional piece of DNA predictably and precisely. Example is cutting a gene from a DNA. One should look for restriction sites on both sides of the gene and then use specific restriction enzymes that will cut at the observed restriction sites. As a result a DNA fragment (genetic disk segment) is obtained, which contains the gene of interest. That fragment should be inserted into another, recipient DNA. The same restriction enzymes are also used to cut the recipient DNA into which the fragment will be inserted. This cut-and-paste operation is one of the ways of engineering a genetic disk.

Each restriction enzyme recognizes a specific nucleotide sequence in the DNA, called a restriction site, and cuts the DNA molecule at only that specific sequence. Many restriction enzymes leave a short length of unpaired bases, called a "sticky" end, at the DNA site where they cut. Other restriction enzymes make a cut across both strands creating double-stranded DNA fragments with "blunt" ends. In general, restriction sites are palindromic, meaning the sequence of bases reads the same forwards as it does backwards on the opposite DNA strand. Example of a palindromic restriction site recognized by restriction enzyme HindIII is given in Figure 2. As Figure 2 shows, HindIII cuts the DNA at last letters of the palindrome and leaves two one-stranded ends of DNA, named sticky ends.

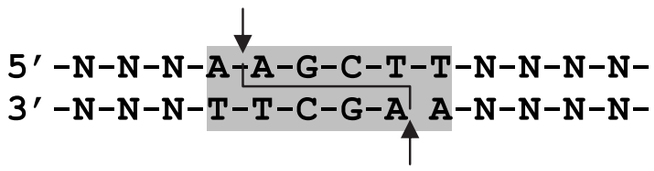


Fig. 2. A two strand palindromic string in a DNA. This particular one is recognized by a HindIII restriction enzyme. Arrows show the cut sites, leaving sticky ends at the cut.

A DNA in presence of particular restriction enzymes will be cut at all the corresponding restriction sites. For example, HindIII restriction enzymes will cut the lambda phage chromosome into 8 segments (or fragments).

5.1.2 Engineered source disks

A genetic program is a file written on a genetic disk that contains a code for specific function in a cell. The program can be created and written by a human, which is essence of cell (re) programming. The written program is called source program, and the disk the program is written on is named source disk. A source disk is usually prepared outside a target organism. There are basically two types of source disks: engineered genetic compact disks and artificial chromosome disks.

A) Engineered Genetic Compact Disk. Engineered compact disk is based either on a plasmid or a phage. Usually a natural plasmid (or phage) is loaded with an engineered DNA sequence.

For computer engineers, it can be viewed as a rather small capacity disk media such as compact disk (CD). A cell can be viewed as a computer system that also contains a separate disk replicator, so that a particular CD can be replicated by the cell information processing system. A plasmid and phage can be inserted in a cell using natural way: just put a cell and plasmids into a favorable environment, and a plasmid (or phage) will enter the cell.

A typical example is the engineered plasmid pBR322, which is used for transferring a particular disk segment into bacterium E. coli. Plasmid pBR322 was engineered out of three natural E. coli plasmids: R1 plasmid, containing amp^R gene, which provides resistance to ampicillin, R6-5 plasmid, containing tet^R gene which provides resistance to tetracycline, and pMB1 plasmid, containing replication origin (ori) segment. There is an ori part in a genetic disk which makes the disk replicable. There are specific spots on the pBR322 where restriction enzymes such as EcoRI, SalI, PstI, PvuiI, and HindIII can make the disk open for inserting a file. After insertion of the disk segment, the plasmid disk has extended its used disk space. Plasmid pBR322 has capacity of accepting disk segments of about 10 Kbp (Brown, 2001).

B) Artificial chromosome. Artificial chromosome is an engineered genetic disk which has organization of a natural chromosome, but is much shorter. An example of artificial chromosome is the Yeast artificial chromosome (YAC). Figure 3 shows a procedure of insertion of a DNA fragment into a Yeast artificial chromosome.

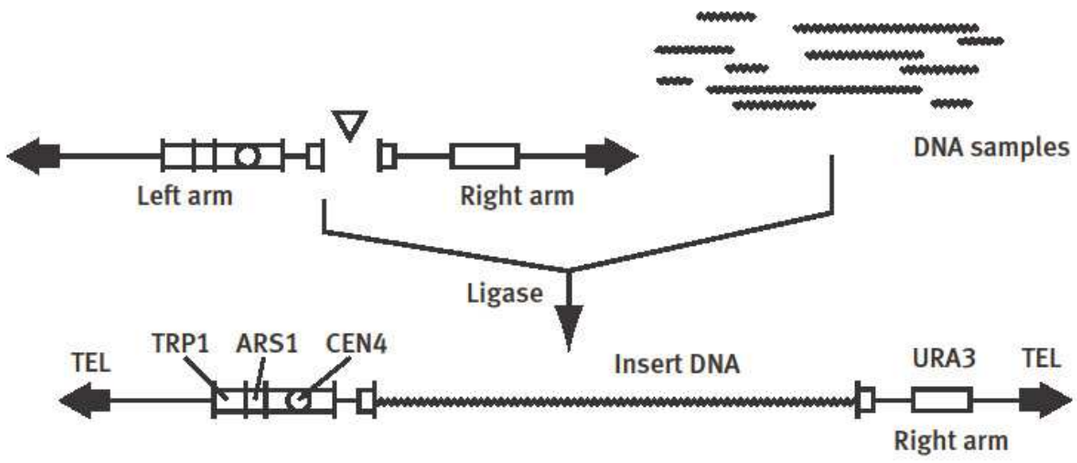


Fig. 3. File insertion into a Yeast artificial chromosome

Figure 3 shows how a DNA fragment is inserted into an artificial chromosome in order to be transferred into a cell. A chromosome of a eukaryote (yeast is a one-cell eukaryote) contains specific DNA part such as telomeric DNA (TEL1), replication origin (ori), replication sequences (ARS1), and centromeric DNA (CEN4). Centromeric DNA enables segregation of the DNA at the time of cell division. Selectable markers are genes that allow distinguishing cells that have this artificial chromosome. For example, for the pYAC2 chromosome, the genes are: amp^R, ura3, and trpI. Artificial chromosomes possess important property, they do not transfer files into the main, cell replicating disk(s) of the cell. Instead they remain as a separate disk, in addition to the cell chromosome disks. The inserted DNA is possibly an

engineered one, which acts as part of the chromosome system of the cell where the artificial chromosome is inserted.

There are several types of artificial chromosomes. One is the Bacterial Artificial Chromosome (BAC), which is based on plasmid F and can accept segments between 80-300Kbp. The Yeast Artificial Chromosome (YAC) is often used for transferring files into yeast (Burke et al, 1987). A YAC is able to accept disk segments of Mbp size. Mammalian artificial chromosomes (MAC) (Grimes & Cooke, 1998) as well as Human Artificial Chromosomes (HAC) (Larin & Mejia , 2002) were also engineered. Recently a plant artificial chromosome with length of 30 Mbp was reported (Ananiev et al, 2009).

5.1.3 Transferring source disks into cell file systems

Engineered genetic compact disks are plasmids (or phages). They are transferred into a cell by simply mixing cells and plasmids under favorable conditions; the plasmids will enter the cells. In genetic engineering those source disks are named vectors, pointing out their inherent transferring ability.

There is no natural way of inserting a rather large artificial chromosome into a cell. Usual procedure is electroporation (Rebersek & Miklavcic, 2011), by which the artificial chromosome is forced into a cell as a high energy particle. Once inside a cell, an artificial chromosome behaves in principle like other cell chromosomes; it just has much less files than the natural chromosomes.

5.1.4 Arrays of genetic disks: Genomic libraries

Genetic engineering can create new life forms or modify existing life forms. Example of modified life forms are so called transgenic organisms, which in their genome contain genes from another organism. For example, it is possible to include a gene of a human in a bacterium.

There are other applications that are more oriented toward file systems rather than the primary intention of modifying an existing mechanism. One such application is creating an array of genetic disks to store a particular genome. Figure 4 (Ackovska et al., 2010) shows creation of a human genomic library. The human genome (all human genes) are kept into array of transgenic disks, each disk is a bacterial genome in which a human gene is inserted.

In humans there are 23 pairs of disks on which the cell operating system resides. They are marked F1-F22 and M1-M22 on Figure 4, pointing out that F23 disk represents the X chromosome and M23 disk represents the Y chromosome. The capacity of those disks is between 50 and 230 Mbp. The density of files on disks is low, only about 5% of the entire DNA contains genes. The other parts are control sequences, data structures, and also areas which, by today's knowledge, contain no meaningful information. Some of human cell files are longer than 100 Kbp, so BAC disks are used (Osoegawa et al., 2001). In such a case, about 30,000 disks, which means 30,000 bacteria are needed to store the human genome.

The obtained set of files contains the whole genome, but is not the true representative of the cell file system and cell operating system, as it is in the original 23 pairs of disks. However, a transgenic disk array allows access to a particular segment and set of files faster and in a way more convenient for study. So called arrayed genomic libraries, arranged as a matrix, are built for easier access of a particular segment.

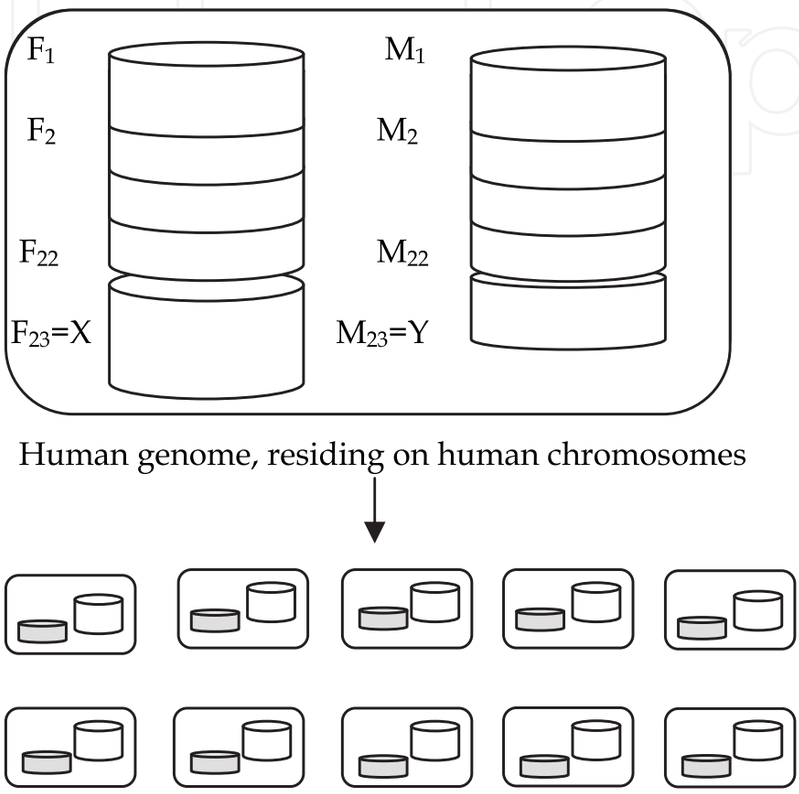


Fig. 4. Creating genomic library on a transgenic disks array

5.2 A lab experience in genetic engineering for computer science students

In addition to knowledge to be conveyed to students as lectures, lab experience is important part of every course. Working in labs with tools for genetic engineering is very different from the everyday practice for computer science students. If one is going to expose computer science students to work in a genetic lab, there are basically two ways of doing that: 1) lab work is carried out in a computer science lab, or similar, in which limited number of equipment and tools for genetic engineering can be installed; 2) lab work is carried out in a specialized lab, for example a molecular biology lab, which already has all the genetic engineering equipment and tools. The tools for genetic engineering include instruments such as water baths, dry baths, centrifuges, incubation ovens, spectrophotometers, electrophoresis chambers, polymerase chain reactors, and electroporators, among others. The lab activities can be carried out as regular lab activities

inside a Computer Science course, such as Bioinformatics course. Another way of carrying out genetic engineering labs is an extracurricular activity, for example activity funded by a research project. In short, here we describe two lab exercises and the way to organize this specific lab practice so it can become closer to Computer science students.

5.2.1 Lab example 1: Extracting DNA from human saliva cells

First, we will describe a lab activity that does not require many specialized devices, and as such this activity can be carried out inside a computer lab. As introduction to this lab activity one should mention that, DNA is most important molecule for life. It carries hereditary information and also manages production of proteins and manages processes in a cell. Human genome is organized in linear chromosomes, and somatic cells such a saliva cells have 46 chromosomes. The total length of all chromosomes is about 2 m. However, DNA is invisible, it is a nanostructure, and its total width is 2 nm.

Understanding and working with DNA is one crucial educational task in hands-on lab experience for a Computer Science student. A DNA sample can be obtained from human blood, such as in medicine or in forensics. In lab practices for Computer Science obtaining DNA sample from human saliva is preferable approach. Therefore, in this lab task a computer science student will be able to extract her/his own DNA from her/his saliva.

The important part of "hardware" needed for this exercise, not usually used by CS students, is an incubator, for example an incubation oven, which can keep a temperature of 50°C for some time. There are devices such as water bath, that keeps water on that temperature, and students insert their lab tubs into that water for some time. The simplest approach is to take any heating source, take a thermometer, heat the water until thermometer shows around the desired temperature, and put the lab tubes in such water for needed time. If conditions allow, a water bath might be purchased. However, a lab kit that contains necessary tools, such as enzymes, tubes etc, should be purchased. There are many vendors of these types of kits, one example is BioRad.

The detailed explanation of the lab activity for extraction a human DNA from human saliva cells is given in Figure 5. It shows detailed flow of the processing of the tube in which the saliva is placed up until the DNA is visible in the same tube. It is not necessary to give the students task in detailed terms. As the reader may notice, the explanation written for CS students differs significantly than the explanation found in lab manuals for biology or medical students. Computer science students are accustomed to process thinking. Given below is a description of the task.

Step 1. **Collect cells.** You can collect thousand of cells from the inside of your mouth just by gently chewing your cheeks and rinsing your mouth with water.

Step 2. **Break open (lyse) the cells.** Use provided *lysis buffer*. It will break open the cell membrane and nucleus membrane and release DNA.

Step 3. **Remove proteins.** Use provided enzymes named *proteases*. They will remove all the proteins in your solution along with proteins that keep DNA as a thread. Proteases work best at 50°C. So you incubate (in a water bath previously warmed up to 50°C) your solution together with enzymes to that temperature.

Step 4. **Condense DNA, make it visible.** Use salt and cold alcohol. It will precipate DNA out of the solution, and you can see it as a mass of white threads.

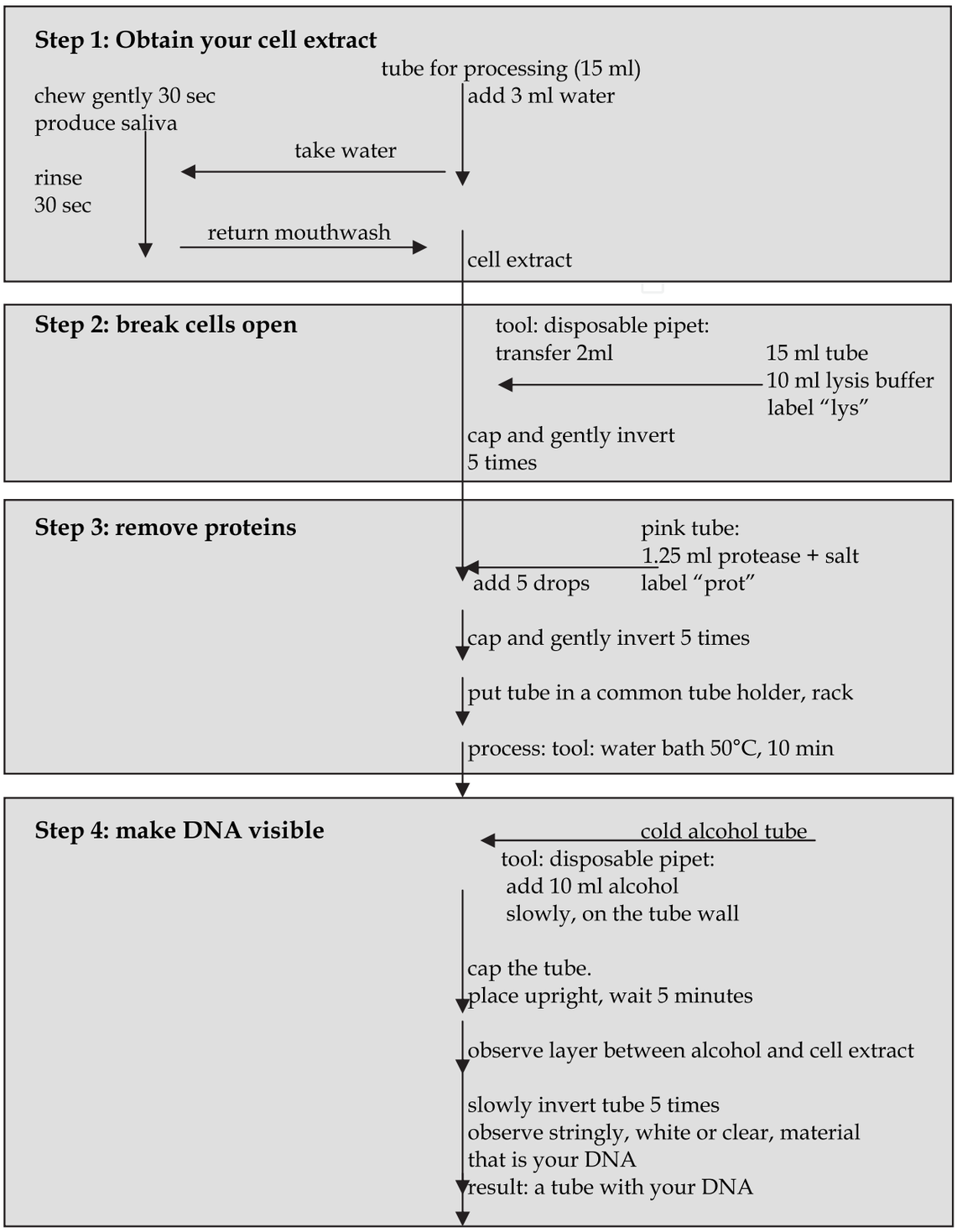


Fig. 5. Lab procedure for extracting DNA from human saliva cells

5.2.2 Organization of a lab for computer science students

Organization of the laboratory is important for any lab activity. It is especially important for computer science students working with tools such as lab tubes and pipets, which are not part of their regular computer science lab activity.

A workstation should be prepared, which can be for one student or shared by more than one student. Each student has her/his processing tube, in which she/he puts the saliva and then processes it using the provided tools such as lysis buffer and protease enzymes. The equipment, such as Water Bath, is shared by all the students. All the processing tubes are put in a rack and then placed in the Water Bath for incubation. Some tools needed for processing are kept in the refrigerator, such as alcohol. This part is very different form everything the Computer Science students are accustomed to.

5.2.3 Lab example 2: Finding shortest file segment obtained by a restriction enzyme

Here we describe a lab that introduces hands-on experience with restriction enzymes to computer science students. For Computer Science students the easiest way to understand restriction enzymes is that they are type of cell robots that are able to cut DNA at a particular point. In this lab exercise the fragments are obtained from a DNA and their length is estimated. This exercise uses a classical lab technique named agarose gel electrophoresis to estimate length of a particular DNA fragment. Since specialized equipment, electrophoresis chamber, is used, this activity might be easier if carried out in a biology lab.

The task description is as follows: DNA of a phage lambda is given. Also, given are three restriction enzymes: EcoRI, PsfI, and HindIII. Cut the lambda DNA with each enzyme. Determine which restriction enzyme will obtain DNA fragments with minimum length.

The lab procedure is performed in parallel on four processing tubes. One tube is lambda DNA, the second is lambda+EcoRI, the third is lambda+HindIII, the forth is lambda+PstI. They are centrifuged and then put into an incubator with temperature that is best for enzymes. The enzymes cut the DNA into number of fragments.

The following is more elaborated description of the lab procedure for obtaining fragments of the phage lambda genome processed by various restriction enzymes:

Step 1. Obtain DNA fragments

Put restriction enzymes into three tubes with DNA, the fourth is just DNA. Mix the tubes, possibly in a centrifuge, heat the tubes at 37°C for 30 min, possibly in a water bath. Result: each tube contains fragmented DNA.

Step 2. Distinguish fragments by their lengths

Put marker dye into tubes, mix, possibly with centrifuge, prepare agarose gel wells, put the tube content into the gel wells, and perform electrophoresis at 100V for 30 min. Result: The resulting gel contains information how far in the electrical field the fragments travelled. However, the result is not visible for human eyes.

Step 3. Visualize DNA fragments

Put colouring marker on the gel, wait, then wash the gel. Result: Visible lanes of DNA fragments with different travelling paths.

If the exercise is done correctly there is a visible result of the procedure, which can be analyzed and/or photographed. On a solid rectangle piece of gel, there are 4 visible vertical lanes. Each lane has horizontal bars, each bar representing DNA fragments with various lengths. The first lane contains uncut DNA (single fragment), the second lane contains fragments cut by PstI, the third and the fourth lanes contain fragments cut by EcoRI and HindIII respectively. It can be observed that the horizontal bar at longest distance is in the lane where fragments cut by PstI are positioned. Therefore, the answer to this lab task is: If a lambda DNA is processed with three restriction enzymes, EcoRI, PstI, and HindIII, the shortest DNA fragment will be obtained by PstI restriction enzyme.

The lab exercise is carried out in groups, each group having own lab workstation with all the tubes and materials necessary. The devices like electrophoresis chamber and water bath might be common for the groups.

6. Implementation

This approach has been successfully implemented to at least two Universities: South Carolina State University in Orangeburg, SC, USA and St. Cyril and Methodius University in Skopje, Macedonia. It is applied in the courses CS495 Biocomputing and Bioinformatics (South Carolina state University), and partly in the course DNA Programming, Intelligent Systems and Bioinformatics (Sts Cyril and Methodius University). During lab activities part of which are described here, computer science students are given opportunity to obtain hands on experience with bionanorobots and other nano structures used in genetic engineering. Each lab has a lab quiz that asks students to relate the observed knowledge with their background knowledge in robotics, flexible manufacturing, and operating systems. Students show enthusiastic interest in learning steps toward genetic engineering.

7. Conclusion

The paper describes an innovative approach toward education of Computer Science students and specialists in genetic engineering. We argue that genetic engineering is good way of evolution for classical engineering and that it should be part of computer science education. Computer science is about programming and reprogramming, and cell (re)programming is essence of genetic engineering and of creating new organisms.

For a long time computer science is involved in creating artificial creatures such as robots. However, biological robots, such as bionanorobots, were not part of computer science education. We argue that the students should be familiarized with possibility of programming DNA that will compile into a bionanorobot.

Appropriate language should be used in explaining molecular genetics to computer science students. This paper proposes use of the computer science related metaphors, such as robotics and flexible manufacturing metaphor as well as operating systems and systems software metaphor.

This approach has been successfully implemented to at least two Universities: South Carolina State University (SCSU) in Orangeburg, SC, USA and Sts. Cyril and Methodius University in Skopje, Macedonia. It is applied in the course CS495 Biocomputing and bioinformatics in SCSU, and the course Bioinformatics, Intelligent Systems, and DNA

Programming at the Faculty of Natural Sciences and Mathematics at Sts. Cyril and Methodius University. The terminology used in these courses is adapted toward computer engineering way of thinking. The addressed issues are strongly connected to the terms of files, disks, operating systems, file processing, robots etc. This enables the computer science students better understanding of some of the most complicated processes known to man, the processes of life. Students are given opportunity to obtain hands on experience with bionanorobots, such as restriction enzymes. Computer Science student have shown interest toward understanding and reprogramming DNA. In addition to theoretical lectures, the students participate in extracurricular activities which give them hands on-experience with DNA manipulation. It seems that this way of reasoning makes students curious for additional knowledge in Genetic Engineering. Many of these students have expressed interest for the upcoming master's degree program related to biorobotics at SCSU. Some of the students, who graduated at the University St. Cyril and Methodius, and have taken classes that support Genetic Engineering education, are already students in some of the Europe's Genetics Engineering masters programmes.

We believe that we should continue towards further research for appropriate metaphors in relation between computer science and genetic engineering. We also believe that we should enrich the student work with more practical implementation of the concepts presented in this paper.

8. Acknowledgment

This work was supported in part by the NSF grant EPS-0903795-2010-702 awarded to South Carolina State University in 2009.

9. References

Ackovska N., Bozinovska L. & Bozinovski S (2010) Artificial chromosomes as genetic disks: A systems software metaphor for genetic engineering, *Proceedings of IEEE SouthestCon 2010*, pp. 324-327, 978-1-4244-5853-0, Charlotte, NC, March 18-21, 2010

Ackovska N. & Bozinovski S. (2008), Next Generation Operating Systems: A Biologically Inspired Future, *Proceedings of 2nd Annual IEEE Systems Conference*, pp. 1-7,978-1-4244-2150-3, Montreal, Canada, April 7-10, 2008

Ackovska N., Bozinovski S. & Jovancevski G. (2008a). Real-Time Systems – Biologically Inspired Future, *Journal of Computers*, Vol. 3, No.3, (March 2008), pp. 56-63, 1796-203X, 2008.

Ackovska N., Bozinovski S. & Jovancevski G. (2008b). File system organization in minimal biological system, *Proceedings of the 6th International Conference on Informatics and Information Technology*, pp. 44-47, 978-9989-668-78-4, Bitola, Macedonia, February 10-14, 2008

Ackovska N., Bozinovski S. & Jovancevski G. (2007). A New Frontier for Real – Time systems – Lessons from Molecular Biology, Proceedings of IEEE SoutheastCon 2007, pp.224-228, 1-4244-1029-0, Richmond, VA, March 22-25, (2007)

Ananiev E., Wu C., Chamberlin M., Switashev S., Schwartz C., Gordon-Kamm W. & Tingey S. (1985). Artificial chromosome formation in maize, *Chromosoma,* Vol. 118, No. 2, pp. 157-177, ISSN : 0009-9515, 1985

Been M. (2006). Versatility of Self-Cleaving Ribozymes, *Science*, Vol. 313, pp. 1745-1747, ISSN: 0036-8075, 2006

Bozinovski S. (1985) Guest Editor's Introduction, *Automatika, Vol. 26 (3-4), Special Issue on Biocybernetics*, pp. 128, Zagreb, ISSN: 0005-1144, 1985

Bozinovski S. (1987) Flexible manufacturing systems: Biocybernetics approach (In Russian), *Problems in Manufacturing and Control*, Vol. 16, pp. 31-34, ISSN: 0234- 6206, 1987

Bozinovski S. & Bozinovska L. (1987), Flexible production lines in genetics: a model of protein biosynthesis process, *Proceedings of International Conference on Robotics*, pp.1-4, Dubrovnik, Yugoslavia, 1987

Bozinovski S., Mueller B. & diPrimio F. (2000). Biomimetic autonomous factories: Autonomous manufacturing systems and systems software, *GMD Report*, No. 115, German National Research Center for Information Technology, Bonn, ISSN: 1435-2702, 2000

Bozinovski S. & Bozinovska L. (2001), Manufacturing science and protein biosynthesis, In N. Calaos, W. Badawy, S. Bozinovski (eds.) *Proceedings of SCI Conference 2001, Vol. XV,* pp. 59-64, ISBN : 980-07-7555-2, Orlando, FL, 2001

Bozinovski S., Jovancevski G. & Bozinovska N. (2001). DNA as a real time, database operating system, In N. Calaos, W. Badawy, S. Bozinovski (eds.) *Proceedings of SCI Conference 2001*, VOL XV : pp. 65-70, ISBN: 980-07-7555-2, Orlando, FL, 2001

Bozinovski S. & Bozinovska L. (2011). Human Body Parts Making: Educational Challenges for Engineered Physiology, Biorobotics, and Biofabrication, *Proceedings of IEEE SoutheastCon 2011,* pp. 307-308, ISBN: 978-1-61284-737-5 , Nashville, TN, 2011

Brown T. A. (2001). *Genetics, A Molecular Approach*, Nelson Thomas, ISBN 0-7487-4370-7, 2001

Brown T.A. (2002), *Genomes*, 2-nd Ed., Willey-Liss, ISBN: 0-471-31681-0, 2002

Burke D., Carle M. & Olson M. (1987). Cloning of large segments of exogenous DNA into yeast by means of artificial chromosome vectors, *Science*, No. 236, pp. 806-812, ISSN : 0036-8075, 1987

Crick F. (1958). On protein synthesis, *Proceedings of Symposium on Society for Experimental Biology*, No.12, pp. 138-163, ISSN: 0081-1386, 1958

Demeester L. ; Eichler K & Loch C. H. (2004). Organic Production Systems: What the Biological Cell Can Teach Us About Manufacturing, *Manufacturing and Service Operation Management*, Vol. 6, No. 2, INFORMS, pp. 115-132, ISSN: 1523-4814, 2004

Danchin A. & Noria S. (2004). Genome structure, operating systems and the image of the machine in *Molecules in Time and Scpace: Bacterial Shape, Division, and Phylogeny*, Vicente M., Tamames J., Valencia A., Mingorance J. (eds.), pp. 195-208, Kluwer, ISBN: 0-306-4857-8, 2004

Gibson D., Glass J., Lartigue C., Noskov V., Chuang R., Algire M., Benders G., Montague M., Ma L., Moodie M., Merryman C., Vashee S., Krishnakumar R., Assad-Garcia N., Andrews-Pfannkoch C., Denisova E., Young L., Oi Z-O., Segall-Shapiro T., Calvey C., Parmar P., Hutchison C., Smith H. & Venter C. (2010). Creation of a bacterial cell controlled by a chemically synthesized genome, *Science,* Vol. 329 (5987): 52–56. 2010

Grimes B. & Cooke H. (1998) Engineering mammalian chromosomes, *Human Molecular Genetics*, Vol. 7, No.10, pp. 1635-1640, ISSN: 0964-6906, 1998

Jackson D., Symons R. & Berg P. (1972) Biochemical method for inserting new genetic information into DNA of Simian Virus 40: Circular SV40 DNA molecules

containing lambda phage genes and the galactose operon of Escherichia coli. *Proceedings of National Academy of Sciences,* 69 (10): 2904–2909, 1972.

Kruger K., Grabowski P. J., Zaug A. J., Sands J., Gottschling D. E. & Cech T. R. (1982). Self-splicing RNA: Autoexcision and autocyclization of the ribosomal RNA intervening sequence of tetrahymena, *Cell*, Vol.31, pp. 147-157 , ISSN: 0092-8674, 1982

Kanehisa M. (2000). *Post-genome Informatics*, Oxford University Press, ISBN: 0-19-850326-1, 2000

Larin Z. & Mejia . (2002). Advances in human artificial chromosome technology, *Trends in Genetics*, Vol. 18, No.6, pp. 313-319, ISSN: 0168-9525, 2002

Madevska-Bogdanova A. & Ackovska N. (2009), Different Approach to Information Technology – Teaching the Intelligent Systems Course in Technology, Education and Development, Lazinica A. & Calafate C. (eds), pp. 357-366, 978-953-307-007-0, In-Teh, Vukovar, Croatia, 2009

Nicholl D. (2008). *An Introduction to Genetic Engineering*, Cambridge University Press, ISBN: 051-80867-7, 2008

Nutt G. (1992). *Centralized and Distributed Operating Systems*, Prentice Hall, ISBN 0-13-122326-7, 1992

Osoegawa K., Mammoser A., Wu C., Frengen E., Zeng C., Catanese J., & de Jong P. (2001). A Bacterial Artificial Chromosome Library for Sequencing the Complete Human Genome, *Genome Research*, No. 11, pp.483–96, ISSN: 1088-9051, 2001

Pirim H. (2005). Biological Cell's production system, *Proceedings of 35th International Conference on Computers and Industrial Engineering*, pp. 1571-1575, Istabul, Turkey, 2005

Ratner V. (1975). Control Systems in Molecular Genetics, (In Russian) *Nauka*, Novosibirsk, 1975

Rebersek M. & Miklavcic D. (2011) Advantages and disadvantages of different concepts of electroporation pulse generation. *Automatika 42(1):, Special Issue on Recent Advances in Biomedical Engineering,* pp. 12-19, ISSN 0005-1114, 2011

Ren X., Tihimic C., Katoh M., Kurimasa A., Inoue T. & Oshimura M. (2006). Human artificial chromosome vectors meet stem cells, *Stem Cells Reviews and Reports*, Vol. 2, No.1, pp. 43-50, ISSN: 1558-6804, 2006

Tanenbaum A. (1995). *Distributed Operating Systems*, Prentice Hall, ISBN 10: 0321-99084, 1994

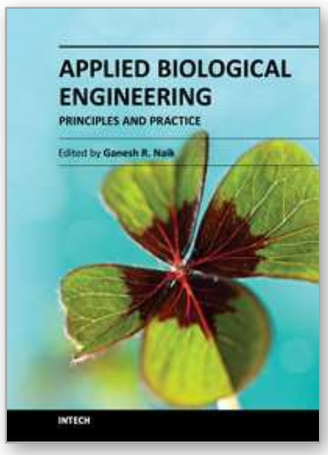
Tanenbaum A. & Van Steen M. (2007). *Distributed Systems; Principles and Paradigms.* ISBN 0-13-239227-5, Prentice Hall, 2007

Watson J. & Crick F. (1953). Molecular structure of nucleic acids: a structure of deoxyribose nucleic acid, *Nature,* No. 171, pp. 737-738, ISSN: 0028-0836, 1953

Williamson J. (2002) Dragon's island and other stories. Five Star, ISBN-10: 0786243147

Xiong J.(2006). *Essential Bioinformatics*, Cambridge University Press, ISBN-10 0-521-60082-0, 2006

Zaibak F., Kozlovski J., Vadolas J., Sarsero J., Williamson R. & Howden S. (2009). Integration of functional bacterial artificial chromosomes into human cord blood-derived multipotent stem cells, *Gene Therapy*, Vol. 16, No.3, pp. 404-414, ISSN: 0969-7128, 2009



Applied Biological Engineering - Principles and Practice
Edited by Dr. Ganesh R. Naik

ISBN 978-953-51-0412-4
Hard cover, 662 pages
Publisher InTech
Published online 23, March, 2012
Published in print edition March, 2012

Biological engineering is a field of engineering in which the emphasis is on life and life-sustaining systems. Biological engineering is an emerging discipline that encompasses engineering theory and practice connected to and derived from the science of biology. The most important trend in biological engineering is the dynamic range of scales at which biotechnology is now able to integrate with biological processes. An explosion in micro/nanoscale technology is allowing the manufacture of nanoparticles for drug delivery into cells, miniaturized implantable microsensors for medical diagnostics, and micro-engineered robots for on-board tissue repairs. This book aims to provide an updated overview of the recent developments in biological engineering from diverse aspects and various applications in clinical and experimental research.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen