Solutions to Decision-Making Problems in Management Engineering Using Molecular Computational Algorithms and Experimentations

KIM, Ikno

May 2011

Waseda University Doctoral Dissertation

Solutions to Decision-Making Problems in Management Engineering Using Molecular Computational Algorithms and Experimentations

KIM, Ikno

Graduate School of Information, Production and Systems
Waseda University

May 2011

Ikno Kim

Solutions to Decision-Making Problems in Management Engineering Using Molecular Computational Algorithms and Experimentations

To my respectable parents, Yonggui Kim and Okhee Hyun

All uncredited Figures and Tables in this dissertation are designed and drawn by Ikno Kim.

Contents

Acknowledgements

1) Since I first started studying at the Graduate School of Information, Production and Systems, Waseda University, I have been able to acquire knowledge in a great number of concepts: communications, methodologies, methods, models, perceptions, and technologies, as well as theories of information, production, and systems engineering. The school has provided me with a great opportunity to understand and take a broad view of the various aspects of the world's engineering and scientific knowledge. That experience has resulted in my desire to contribute widely to the engineering and scientific endeavours. Collaboration on new and innovative ideas will continue to make large and positive contributions to developing cooperative international relationships.

2) It has already been nine years since I first met Professor Junzo Watada. I was looking at many professors' laboratories when I first decided to study in Japan. I found Professor Watada's article on the online board of the Human Informatics laboratory in the Department of Technology Management, Osaka Institute of Technology. Its managerial and financial analyses, using soft computing techniques and methods, were different from the general concepts of industrial engineering. It introduced me to a new world of industrial engineering and management, and greatly affected my decision to study human informatics for industrial engineering and management.

   Since the courses Professor Watada taught, such as Decision Support Systems, Management Information Systems, Human Informatics, and Soft Computing, involved material that was new to me, they were quite interesting. When I was a sophomore, I heard about his transfer. He called me and suggested that join him at the Graduate School of Information, Production and Systems, Waseda University. Because I have always wished to study in his laboratory, I was very happy to hear his suggestion.

   The various research projects and warm interpersonal relationships at the Management Engineering Laboratory were very beneficial and meaningful to me. In particular, I remember helping Professor Watada with the several international conferences that he held. I still appreciate having had the several opportunities that he gave me to help him, because I learned much from helping him. I was able to meet excellent professors and students from all over the world at those conferences. Through these precious experiences, I was able to foster global perspective and become an international researcher, even though I was a graduate student studying in Japan.

   Looking back, I felt that I often did not meet Professor Watada's expectations, and that he worried about my lack of abilities and skills. Even though I felt that way, he always had faith in me and was there for me. It was truly a great help for me. Like a lighthouse that shows ships the correct direction in the middle of raging waves, he has been a compass to my life. I sincerely appreciate his teaching. Just as he was my proud professor, I will always be a proud student of his. I will never forget his sincere, welcoming, warm, and kind-hearted care.

3) I would greatly like to thank my respectable committee members, Professor Takeshi Yoshimura, Professor Tomohiro Murata, Professor Osamu Yoshie, Professor Toshitsugu Ueda, Professor Harutoshi Ogai, and Professor Masahiro Nakano of the University of Occupational and Environmental Health, for spending their time to make many meaningful comments and suggestions about my work. I will never forget their kindness and support.

4) I wish to express my sincere appreciation to Professor Ichiro Shigaki of the Osaka Institute of Technology, Japan, who taught me the theoretically important concepts and ideas of industrial engineering and management studies, and who guided me to the right way of studying in Japan.

5) I would truly like to express my gratitude to Professor Witold Pedrycz of the University of Alberta, Canada, who had a lot of important comments and ideas for me, and who took care of me during my stay there.

6) Whenever I visit Taiwan asking for help, Professor Jui-Yu Wu of the Taipei Medical University, Taiwan always welcomed me, taught me biochemistry and molecular engineering, and allowed me to stay in his laboratory. I greatly appreciate the teaching and comments that he gave to me during my research period there. I will also never forget his kindness, hospitality, goodness, and friendly treatment.

7) I greatly appreciate, with thankful hearts, my respectable parents, Yonggui Kim and Okhee Hyun, who sent me to Japan, and who always believe in me. I really appreciate my two elder brothers, Youngno Kim and Hakno Kim, who kindly treat me. I greatly appreciate my respectable parents-in-law, Joohyung Baek and Gabjin Oh, who always take care of me. I finally appreciate my sweetheart, Sunghee Baek, who always encourages and loves me.

May 2011
Ikno Kim
Graduate School of Information, Production and Systems
Waseda University

Solutions to Decision-Making Problems in Management Engineering Using Molecular Computational Algorithms and Experimentations

Ikno Kim
Doctor of Engineering
Graduate School of Information, Production and Systems
Waseda University

Abstract

Since the concept of a biologically computational paradigm was first pioneered, this novel concept has provided the ability to develop new types of computational algorithms by exploiting and implementing existing algorithms. However, we were unable to find information regarding interdisciplinary molecular types of algorithms for management engineering decision-making problems. This dissertation will present, for the first time, general molecular algorithms that have been (1) constructed by molecular engineering mechanisms and techniques, such as manipulating DNA molecular structures and characteristics; and (2) integrated with other field methods and techniques. These new types of general molecular algorithms are proposed here, and we refer to them as interdisciplinary types of molecular computational algorithms in our new studies. The computational model that makes use of interdisciplinary types of molecular computational algorithms for solving decision-making problems in management engineering is referred to, in this dissertation, as molecular decision support computation. Since interdisciplinary types of molecular computational algorithms are essentially designed and used for the development of molecular decision support computation, interdisciplinary types of molecular computational algorithms can also be simply called molecular computational algorithms from the viewpoint of developing molecular decision support computation. The aim of our studies is to show novel and various types of molecular computational algorithms implemented by novel molecular engineering experimentations, termed molecular computational experimentations. In this dissertation, there are four different configurations of novel molecular computational algorithms used for solving decision-making problems in management engineering: (1) a job-shop DNA-based algorithm used to classify the given schedules into both feasible and unfeasible schedules and determine the minimised maximum production completion time; (2) a fuzzy DNA-based algorithm used to identify cohesive subsets with their density analyses; (3) a hierarchical DNA-based algorithm used to model complicated interpretive structures; and (4) a rough DNA-based algorithm used to minimise decision rules in a rough set approach. Finally, these four molecular computational algorithms, each with its own proposed molecular computational experimentation, will illuminate significant concepts and open up ways of developing mainframe, flexible, and practical applications of molecular decision support computation for decision-making problems in management engineering.

Chapter 1

Introduction

Since the 1970s, scientists have been learning how to freely reproduce and transform deoxyribonucleic acid (DNA), a material that holds significant information about human life. Experimental and theoretical methods of molecular genetics that handle DNA have developed rapidly, particularly in the area of recombinant DNA sequencing, and these methods have been used and applied in many fields.

Adleman [1] suggested that *molecular computation* is one method that has been applied in *in vitro* approaches for implementing DNA molecular structures and characteristics when combined with biochemical tools and techniques. He first solved the *Hamiltonian path problem* (one computationally intractable problem) using DNA molecules. Since his pioneering investigation first showed molecular-level computations, many researchers, such as biochemical scientists and information engineers, have not only paid special attention to solving this computationally intractable problem, but have also realised the great potential for developing a computing machine that is propelled and controlled by free natural biochemical energies and techniques at a nanometric level.

Many computational advantages have become apparent, since the computational mechanisms of DNA molecules have been viewed as arithmetical elements. Molecular computation mainly works with high level functions by initiating biochemical reactions with nucleic acids and various enzymes in a liquid solution. The applied methods of molecular computation are executed in a significantly shorter time by DNA molecule-based arithmetical elements, which provide valuable DNA molecular functions. This permits the creation of a super-parallel computing system, which deals with a solution search space that can be enlarged exponentially in accordance with an increase in the required variables and samples.

Molecular computation does not implement integrated circuits consisting of semiconductors, such as the *von Neumann architecture-based computers* that are normally employed. Information processing by electronic computers is basically conducted with binary digits, whereas information processing by DNA molecules is conducted with four different bases: adenine, thymine, guanine, and cytosine. In addition, DNA molecules are represented as forming molecular chains with regular DNA sequences of sugar and phosphate groups. A single DNA molecule is a polymer with a string of nucleotides, and these nucleotides are bound by a method different from that of both phosphodiester and hydrogen bonds.

Since molecular computation was investigated by Adleman, it has provided significant and exploitable ideas and concepts, which have enabled us to express new types of general molecular algorithms, methods, computing paradigms, and application-based research. For instance, one method of cleaving and catalysing DNA molecular sequences using both restriction enzymes and DNA ligations is actually a molecular engineering method, which can also be used as an applied general molecular method. However, many scientists and engineers pay particular attention to solving computationally intractable problems, such as combinatorial problems or mathematical issues, because they can be implemented using biological and chemical experimental techniques.

In this dissertation, we show how to handle and implement new types of general molecular engineering experimentations (termed *molecular computational experimentations*) in order to strengthen general molecular algorithms and general molecular engineering experimentations. While implementing these novel experimental methods, we also propose several integrations of general molecular algorithms (each of which includes its own molecular computational experimentation for each purpose), advanced mathematical information engineering, and completely different field methodologies. These multiple integrated algorithms (termed *interdisciplinary types of molecular computational algorithms*) are associated with various forms in different engineering and science fields. We focus on both decision support computation and molecular engineering mechanisms and techniques associated with various areas of engineering and science fields. Hence, when we consider the computational model for solving decision-making problems in management engineering (the model makes use of interdisciplinary types of molecular computational algorithms), we refer to this model as *molecular decision support computation*. Interdisciplinary types of molecular computational algorithms are designed and used for the development of the main frame of molecular decision support computation, meaning that when we adopt the viewpoint of developing molecular decision support computation, interdisciplinary types of molecular computational algorithms can also be simply referred to as *molecular computational algorithms*. Four different molecular computational algorithms for molecular decision support computation, each with its own molecular computational experimentation, are used for management of the engineering decision-making problems discussed in this dissertation; each design was based on theoretical concepts of molecular and genetic characteristics and molecular engineering experimental functions. As shown in Figure 1.1, the rest of this dissertation is organised into eight chapters.

In Chapter 2, nanoscale molecular biochemistry is explained in general terms, describing select fundamental structures and the chemical reactions DNA and ribonucleic acid (RNA); these are all related to the theoretical concepts of designing molecular computational algorithms and experimentations. The contents of this chapter provide an important foundation regarding DNA molecules and their biological features and chemical reactions for the interdisciplinary approaches described in the following chapters.

Chapter 3 details molecular experiments, instrumentation, and manipulations to determine the final experimental methods for each of the four proposed molecular computational algorithms shown in Chapters 4, 5, 6, and 7. These notes explain the biological and chemical experimental backgrounds. The three molecular technologies, corresponding to experiments, instrumentation, and manipulations, are significant for designing molecular computational algorithms and experimentations.

In Chapter 4, a molecular computational algorithm, here termed a *job-shop DNA-based algorithm*, is described. This is composed of a job-shop scheduling method and a general molecular algorithm. The algorithm includes our proposed adapted molecular computational experimentation, and deals with minimising the maximum production completion time in job-shop scheduling. Searching for both feasible and unfea-

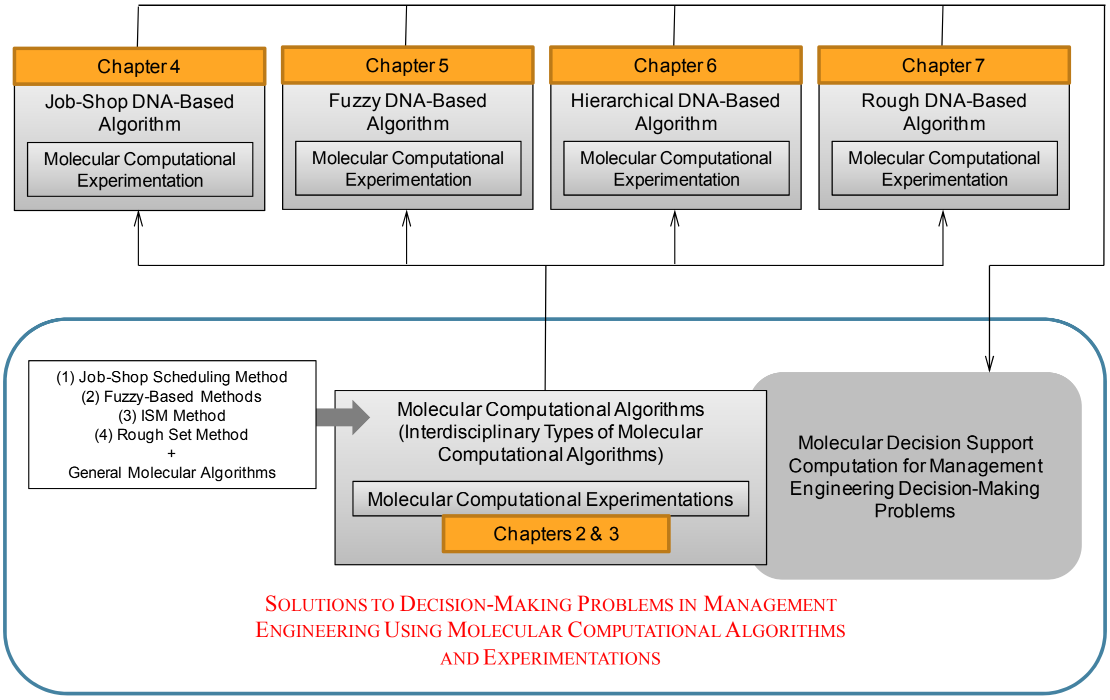

Chapter 4
Job-Shop DNA-Based Algorithm
Molecular Computational Experimentation
Chapter 5
Fuzzy DNA-Based Algorithm
Molecular Computational Experimentation
Chapter 6
Hierarchical DNA-Based Algorithm
Molecular Computational Experimentation
Chapter 7
Rough DNA-Based Algorithm
Molecular Computational Experimentation
(1) Job-Shop Scheduling Method
(2) Fuzzy-Based Methods
(3) ISM Method
(4) Rough Set Method
+
General Molecular Algorithms
Molecular Computational Algorithms (Interdisciplinary Types of Molecular Computational Algorithms)
Molecular Computational Experimentations
Chapters 2 & 3
Molecular Decision Support Computation for Management Engineering Decision-Making Problems
SOLUTIONS TO DECISION-MAKING PROBLEMS IN MANAGEMENT ENGINEERING USING MOLECULAR COMPUTATIONAL ALGORITHMS AND EXPERIMENTATIONS

Figure 1.1. Structure of the proposed molecular computational algorithms and experimentations for management of the engineering decision-making problems.

sible production schedules is often exploited in information, production, and other wide range areas when handling production scheduling data sets. The organisation of the given machines and their job orders and processing times has to be constructed using the job-shop scheduling method. In Chapter 4, we discuss the use of the job-shop DNA-based algorithm as a vehicle for scheduling and elaborate on this molecular computational algorithm in the context of scheduling the given jobs with our encoding-based experimentation. Moreover, by showing the numerical results of computational running time and size comparisons, we demonstrate how well the job-shop DNA-based algorithm works.

Another molecular computational algorithm is described in Chapter 5. This algorithm is here termed a *fuzzy DNA-based algorithm*, which is composed of fuzzy-based methods and a general molecular algorithm. The algorithm also includes our proposed adapted molecular computational experimentation. We focus on two different methods, corresponding to both fuzzy-based and molecular engineering methods. We have determined that these two quite differently characterised methods can be integrated to form an adaptable method and serve as a new molecular computational algorithm. With regard to reasoning in identifying cohesive subsets, we go on to discuss how to combine fuzzy membership grades with the general molecular algorithm to create the fuzzy DNA-based algorithm. The numerical results of computational running time and size comparisons are given to evaluate the use of the fuzzy DNA-based algorithm.

Chapter 6 describes a third molecular computational algorithm, here termed a *hierarchical DNA-based algorithm*; this is composed of an interpretive structural modelling method and a general molecular algorithm. The algorithm involves our proposed adapted molecular computational experimentation. We designed this algorithm for the modelling of interpretive structures. In other words, the algorithm is used to hierarchically structure contextual problems in complex and unpredictable issues or situations. We focus on a directed graph of the complicated contextual relations that can be transformed into DNA molecular sequences to create a novel and biochemically intelligent way of encoding molecular sequences through hierarchical structural modelling. Further, we elaborate on our novel development of the molecular engineering type-setting encoding method that provides different fitting types for each of the elements to create the hierarchical DNA-based algorithm. We also show, with the numerical results of computational running time and size comparisons, how well the hierarchical DNA-based algorithm works.

In Chapter 7, we describe a fourth molecular computational algorithm, here termed a *rough DNA-based algorithm*. This is composed of a rough set method and a general molecular algorithm. We also propose the algorithm's own adapted molecular computational experimentation. The algorithm is used to derive all possible minimal lengths to be used as decision rules, by classifying the given objects into subsets of the most organised and simplified objects to aid decision makers. The rough DNA-based algorithm shows a possible integrated knowledge support system combined with rough set concepts and molecular engineering methods and techniques. Moreover, the numerical results of computational running time and size comparisons are shown to evaluate the use of the rough DNA-based algorithm.

In Chapter 8, we review each of the four molecular computational algorithms (the job-shop DNA-based algorithm, the fuzzy DNA-based algorithm, the hierarchical DNA-based algorithm, and the rough DNA-based algorithm) with their own molecular computational experimentations, which were used to solve the four particular decision-making problems in management engineering. These are presented and explained in Chapters 4, 5, 6, and 7, respectively. Finally, we conclude this dissertation with a summary and suggestions for future molecular decision support computation. This computation makes use of newer and more advanced molecular computational algorithms for the management engineering issues.

Chapter 2

Fundamentals of Molecular Biology for Molecular Computational Algorithms

2.1 Chromosomes

Since the late 19th century, microscopy techniques and applied methods have made great advances; the most significant of these developments has enabled us to see the structure of cells. It is now clearly understood that animal and plant cells are composed of a central body, a cell nucleus, membranes and an amorphous cytoplasm [2]. Although there are many different types of components in the cell, we focus on *chromosomes* (the concomitant elongated bodies contain DNA molecules). One cell is a precise, small and

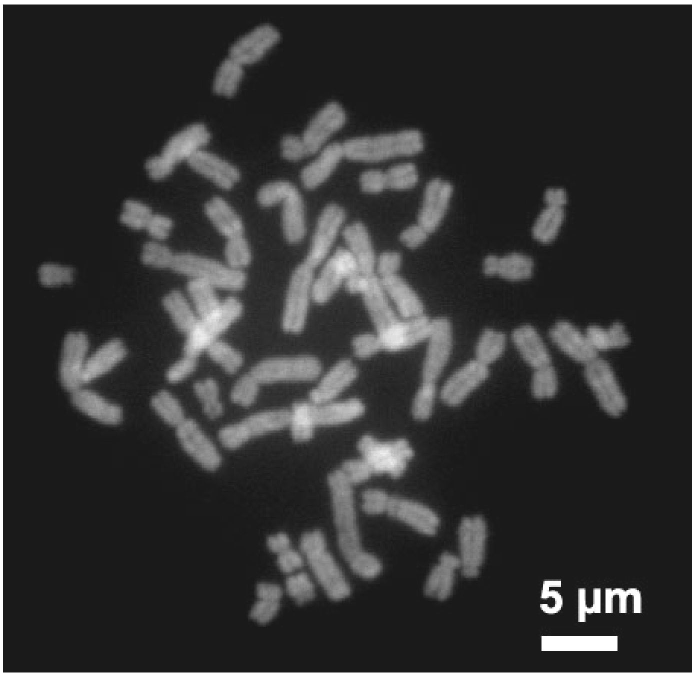


Figure 2.1. Example of human chromosomes in a stage of mitosis during the eukaryotic cell cycle. Creative Commons Attribution-Share Alike 3.0: Steffen Dietzel, 2006.

functional unit of an organism that contains a nucleus. Some or all of the genetic information in the cell nucleus comes from the chromosome structures. Each single chromosome is essentially composed of an organised arrangement of DNA molecules and proteins.

A single chromosome consists of DNA molecules in a folded arrangement. There are several hierarchical levels of organisation in the chromosome, but here we focus on the first level that mainly consists of the DNA molecules. In the first level, the DNA molecules are combined with histones to make a beaded flexible fibre (*histone fibre*). The diameter of the histone fibre is 11nm (110Å); this measurement is roughly derived by considering (1) a sevenfold reduction in the DNA molecule lengths; and (2) five times the free DNA width. The diameter of the histone fibre is only evident when the concentration of salts is low. If the concentration of salts is slightly increased, the linking of the DNA molecules becomes invisible to electron microscopy [3, 4]. Figure 2.1 shows human chromosomes in a stage of mitosis.

2.2 Deoxyribonucleic Acid

Deoxyribonucleic acid, abbreviated as DNA, is polymer biological material within the cells of living matter, which acts as a blueprint for the organism; it performs various functions in order to build the organism and to maintain life. The structure of the human

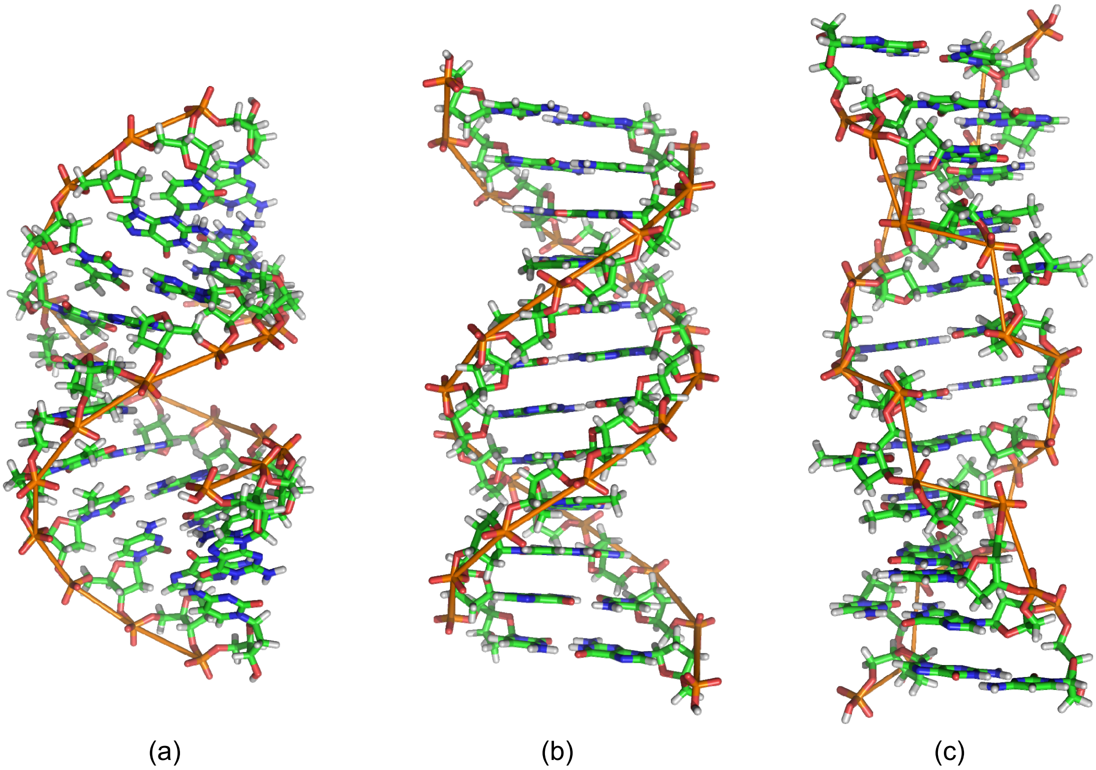
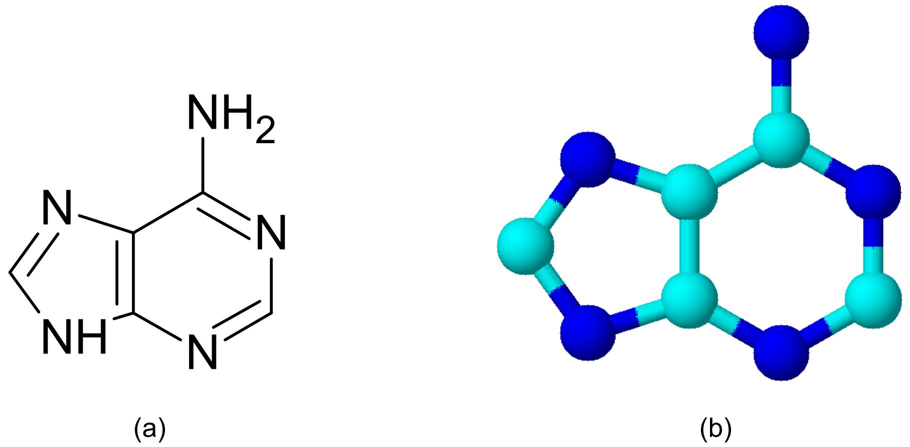
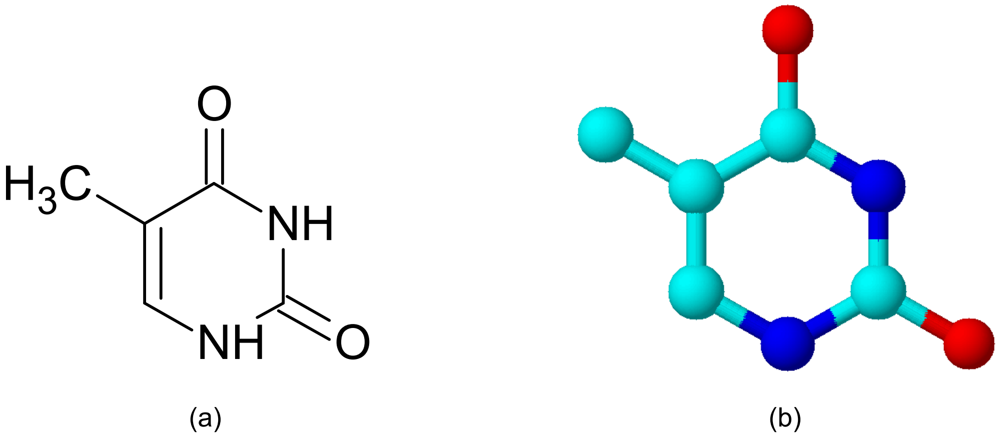


Figure 2.2. Three typical forms of three-dimensional DNA structural views: (a) A-form; (b) B-form; (c) Z-form. Creative Commons Attribution-Share Alike 3.0: Richard Wheeler, 2007.



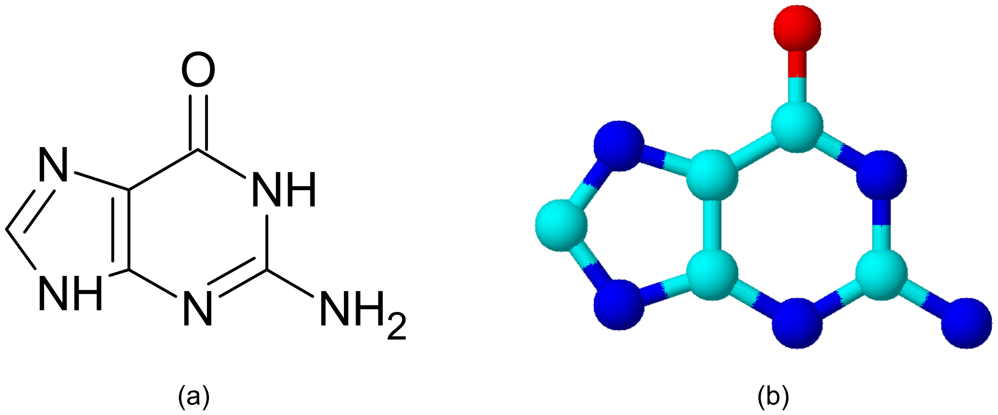
(a) (b)

Figure 2.3. Adenine: (a) structural formula; (b) space-filling model, drawn by Ikno Kim using ACD/ChemSketch 5.12.



(a) (b)

Figure 2.4. Thymine: (a) structural formula; (b) space-filling model, drawn by Ikno Kim using ACD/ChemSketch 5.12.



(a) (b)

Figure 2.5. Guanine: (a) structural formula; (b) space-filling model, drawn by Ikno Kim using ACD/ChemSketch 5.12.

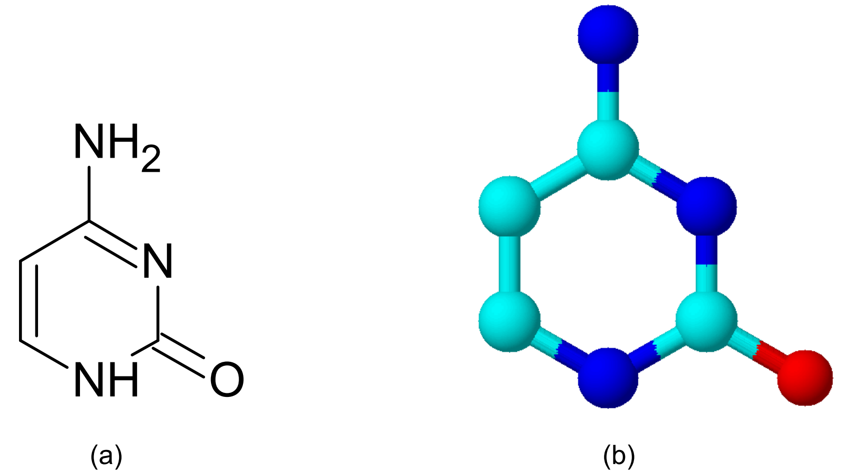


Figure 2.6. Cytosine: (a) structural formula; (b) space-filling model, drawn by Ikno Kim using ACD/ChemSketch 5.12.

genome, corresponding to DNA molecule-based sequences referred as body layout, contains approximately twenty-two thousand genes; 99.9% of them are shared by all humans, and the remaining 0.1% determines the specific characteristics for each individual according to inheritance. This polymer biological material contains genetic information by all living organisms on the earth [5]. As shown in three-dimensional view in Figure 2.2, there are three typical forms of DNA structures [6].

2.2.1 Linear polymer of deoxyribonucleotides

A standard linear polymer of the repeating base units that is constructed by a large DNA molecule is basically composed of five-carbon sugar, phosphoric acid, and the four nitrogen-containing bases. There are four types of nitrogen-containing bases, which correspond to adenine (abbreviated as A), thymine (abbreviated as T), guanine (abbreviated as G), and cytosine (abbreviated as C). Each of these four bases A, T, G, and C is represented as both a structural formula and a space-filling model, shown in Figures 2.3(a) and (b) to 2.6(a) and (b), respectively. As shown in Figures 2.3 and 2.5, the two bases of a double-ring structure correspond to both adenine and guanine, which are referred as purines. As shown in Figures 2.4 and 2.6, the other two bases of a single-ring structure correspond to both thymine and cytosine, which are referred as pyrimidines [3, 7].

2.2.2 Phosphodiester bonds

A nucleoside is formed by a compound of each of the four bases that are linked to the sugar deoxyribose. If a phosphate group is clearly attached to deoxyribose, the nucleoside becomes a nucleotide. In DNA, nucleotides must be joined to a polynucleotide chain. In order to do this, the phosphate is attached to the 5′ carbon of the first sugar linked to the hydroxyl group that is attached to the 3′ carbon of the second sugar. During these chemical attachments, the sugars of adjacent nucleotides are linked in component connections through the phosphates by forming phosphodiester bonds [3]. Figure 2.7

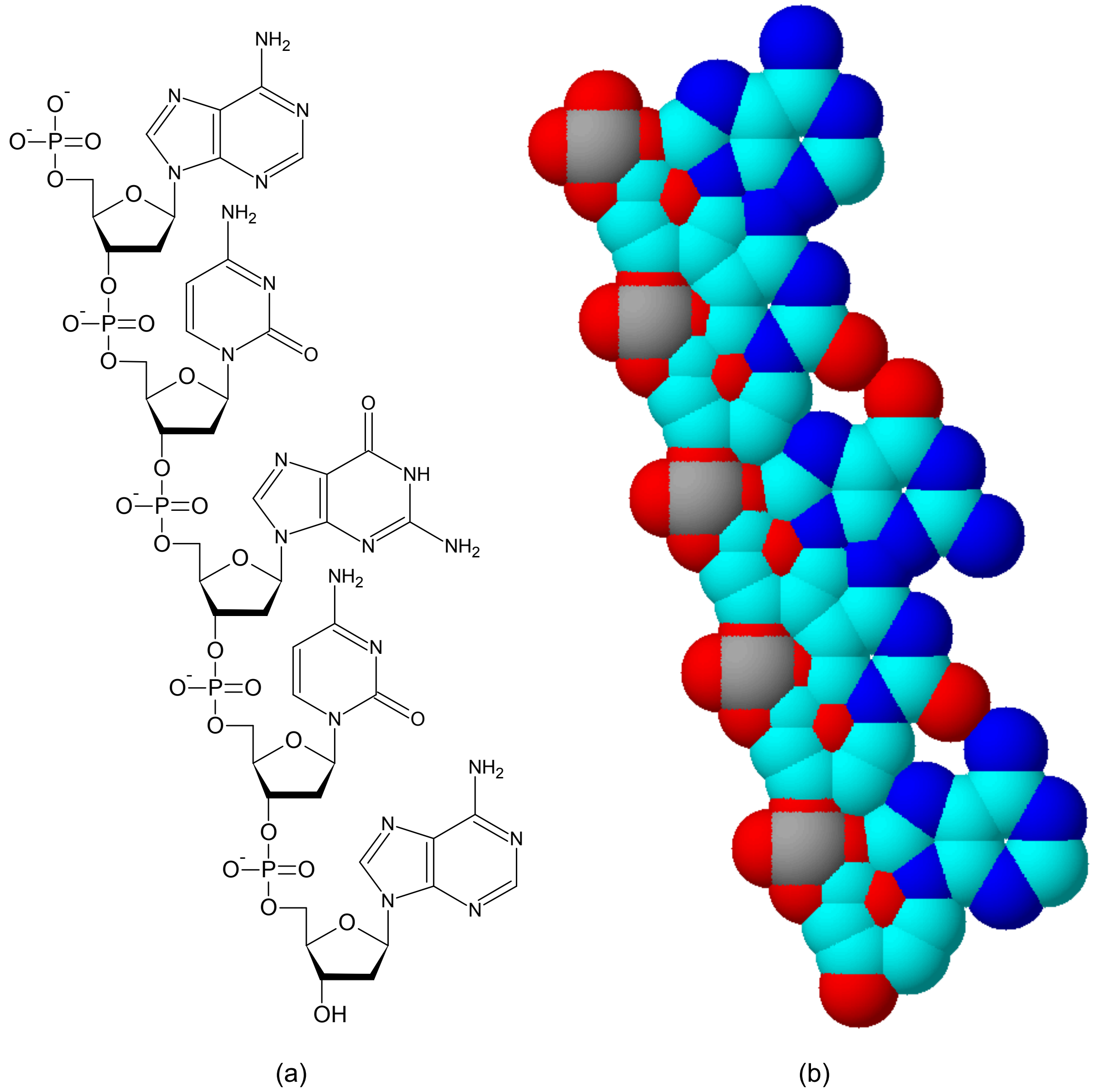


Figure 2.7. Phosphodiester bonds: (a) structural formula; (b) space-filling model, drawn by Ikno Kim using ACD/ChemSketch 5.12.

shows an example of a group of covalent bonds that is also referred to as a *phosphodiester bond*. In addition, chemical compounds with covalent bonds are basically constructed by nonmetal elements, from nanometre-size DNA molecules to macromolecules, such as polythene.

2.2.3 Hydrogen bonds

An important factor in DNA is the binding of two single polynucleotide strands within a pure liquid into a specific form by means of hydrogen atoms, which are shared by two negatively charged atoms. The attachment of two single polynucleotide strands is made by a group of bonds, referred to as a *hydrogen bond*. In particular, a hydrogen bond plays a primary role in the formation of higher-order structure DNA molecules, such as the secondary structure formation of protein by the formation of a right-handed helix. In hydrogen bonds, there are some exact bonding patterns that occur through normal base pairs. Adenine always bonds with only thymine, while guanine always bonds with only cytosine. This phenomenon is referred to as *Watson-Crick complementarity* [8]. A photograph of the DNA reconstruction model of Watson-Crick complementarity is shown in Figure 2.8.

The exact patterns of hydrogen bonds may be seen from building a model of DNA. If a purine of one chain always bonds with a pyrimidine of the other chain, then the dimensions of the bases become the same as the length of the DNA molecule. Hence, one of two purines should selectively bond with one of two pyrimidines. Each of the base pairs is symmetrical in process, and this allows it to be inserted into the double right-handed helix. Again, one complementary base pairing consists of both A and T, and the other complementary base pairing consists of both G and C. From these two complementary base pairings, all of the backbone sugar-phosphate groups clearly have orientations that allow DNA to assume the same structure as any of the DNA sequence bases. Thus, the symmetry of the DNA molecule is associated with the two single polynucleotide strands, each of which runs in opposite directions to become a helix of double DNA strands [2, 9]. Two hydrogen bonds between two different complementary bases, holding A and T together, are shown in Figure 2.9. Three hydrogen bonds between two different complementary bases, holding G and C together, are shown in Figure 2.10. The structure of the DNA molecules is essentially formed by both strong

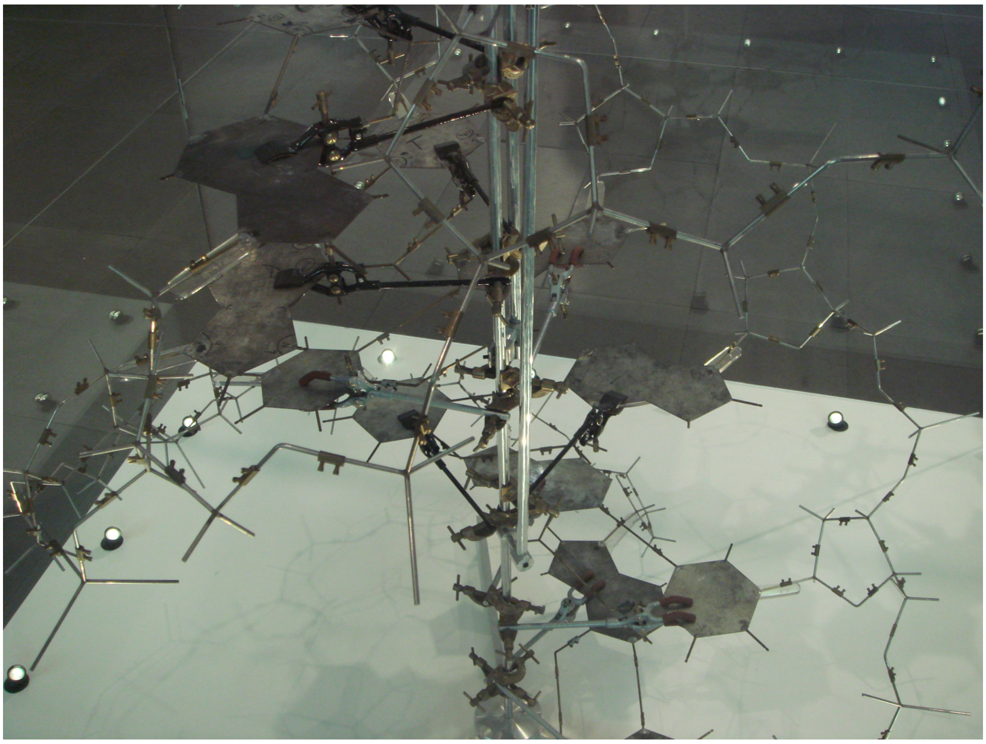


Figure 2.8. Photograph of the DNA reconstruction model of Watson-Crick complementarity, taken by Ikno Kim at Science Museum London.

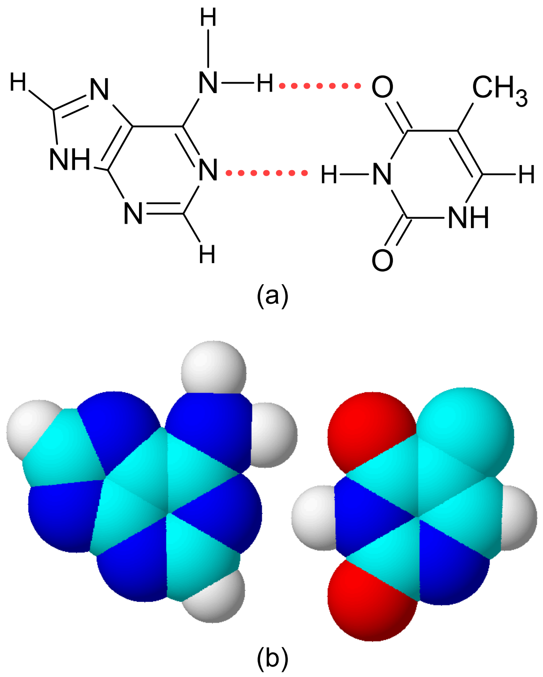


Figure 2.9. Adenine and thymine (two different complementary bases) are being held by two hydrogen bonds: (a) structural formulas; (b) space-filling models, drawn by Ikno Kim using ACD/ChemSketch 5.12.

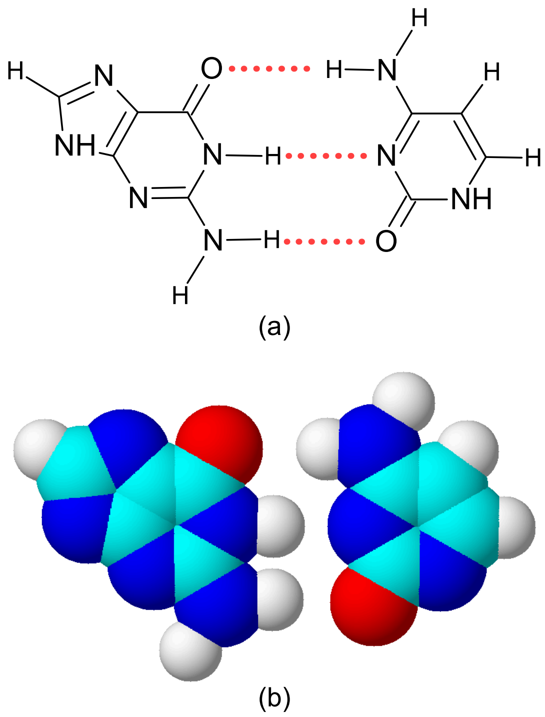


Figure 2.10. Guanine and cytosine (two different complementary bases) are being held by three hydrogen bonds: (a) structural formulas; (b) space-filling models, drawn by Ikno Kim using ACD/ChemSketch 5.12.

phosphodiester bonds and weak hydrogen bonds.

2.3 Ribonucleic Acid

Ribonucleic acid, abbreviated as RNA, exists inside the cytoplasm and is related to the biosynthesis of protein. RNA forms protein in the living organism based on the DNA genetic information. The function of protein in the human body is to create skin, bones, muscles, and blood, all of which require protein.

2.3.1 Chemical synthesis of RNA

In gene expression, the synthesis of the RNA molecules is copied from the DNA segment, meaning each of the RNA molecules derives from a single DNA strand in transcription. For chemical synthesis of RNA, one strand serves as a template in the region of the DNA molecule. The RNA precursors of the four ribonucleoside 5′-triphosphate group are different from the DNA precursors. The significant difference is that for RNA molecules the sugar should be ribose rather than deoxyribose, and thus thymine is replaced by uracil (abbreviated as U). A sugar-phosphate bond is chemically formed between one nucleotide of the 3′-hydroxyl group and the next nucleotide of the 5′-triphosphate to be lined up in the RNA synthesis [3, 10]. Figures 2.11(a) and (b) show both a structural formula and a space-filling model, corresponding to uracil. In the DNA molecule template, the four bases A, T, G, and C are transformed into the four bases A, U, G, and C.

2.3.2 Eukaryotic RNA polymerases

RNA polymerases in eukaryotes have subunits of holoenzyme and are composed of three different types, which are denoted as RNA polymerase I, II, and III. A more detailed explanation follows.

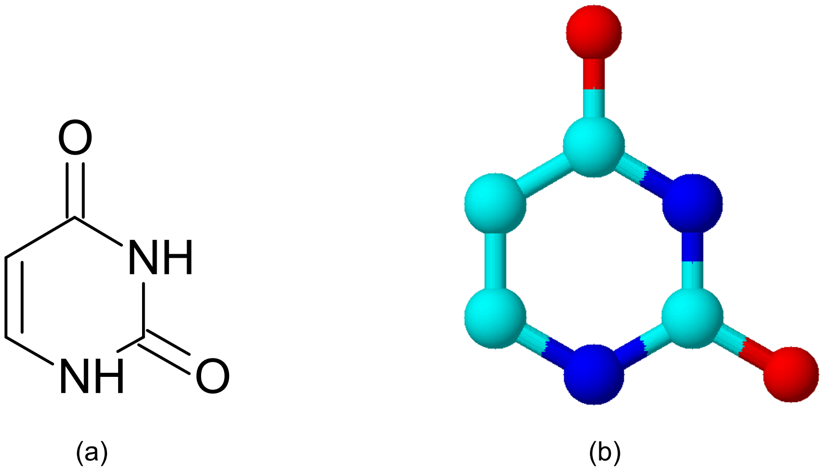


Figure 2.11. Uracil: (a) structural formula; (b) space-filling model, drawn by Ikno Kim using ACD/ChemSketch 5.12.

Three of the RNA polymerase functions are (1) RNA polymerase I, which produces the transcript that starts the processing into ribosomal RNA; (2) RNA polymerase II, which is responsible for transcribing all protein-coded genes and for a large number of small nuclear RNA molecules; and (3) RNA polymerase III, which transcribes all of the transfer RNA genes and a large number of ribosomal subunits in a component [3, 11]. Each of these three different RNA polymerases has a particular RNA transcript.

2.3.3 Translation system

Each of the RNA sequences determines amino acid sequences in all of the information transfer processes, and the amino acid sequences become linked together throughout all of the chemical processes. The entire series of these processes is referred to as a *translation system*. In this system, there are essentially four types of RNA; these are messenger RNA (abbreviated as mRNA), transfer RNA (abbreviated as tRNA), ribosomal RNA (abbreviated as rRNA), and transfer-messenger RNA (abbreviated as tmRNA). A more detailed explanation follows [12-14].

Four of the RNA types are (1) mRNA, which carries a protein sequence information to the specific ribosomes, and is necessary to provide the sequence coding that determines the sequence of amino acids; (2) tRNA, which is attached to a specific amino acid and is a small version of RNA, in which the sequence of amino acids is determined by the sequence base using an adaptor molecule set; (3) rRNA, which is a catalytic component for specific ribosomes, and is regularly divided into four rRNA molecules in eukaryotic ribosomes; and (4) tmRNA, which exists in plastids and bacteria, catches on proteins of the encoded mRNA molecules, and covers the stalling ribosomes.

Chapter 3

Molecular Experiments, Instrumentation, and Manipulations

3.1 Denaturing and Renaturing

Each of the DNA molecules in a double strand is composed of two single DNA strands attached together. One double DNA strand contains a large number of base chain pairs, constructing a double right-handed helix. In normal physiology, none of the double DNA strands ever spontaneously separates to become two single DNA strands. However, if double DNA strands in the helix structure are exposed to close to boiling temperatures, they become denatured or melted into single DNA strands. This occurrence is referred to as *denaturing*, and is explained below.

The high temperatures break the hydrogen bonds of the base chain pairs in a double right-handed helix between complementary DNA strands. In the denaturation of double DNA strands, a pair of guanine and cytosine strands is more stable than a pair of adenine and thymine strands, because guanine attaches to cytosine with three hydrogen bonds and adenine attaches to thymine with only two hydrogen bonds. The temperature required to break the guanine and cytosine pair is therefore slightly higher than that re-

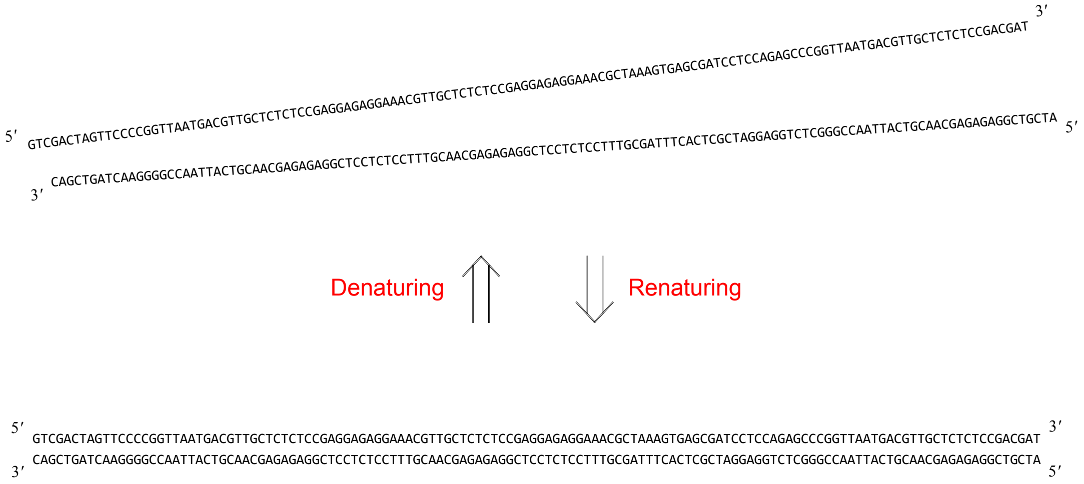

5′ GTCGACTAGTTCCCCGGTTAATGACGTTGCTCTCTCCGAGGAGAGGAAACGTTGCTCTCTCCGAGGAGAGGAAACGCTAAAGTGAGCGATCCTCCAGAGCCCGGTTAATGACGTTGCTCTCTCCGACGAT 3′
3′ CAGCTGATCAAGGGGCCAATTACTGCAACGAGAGAGGCTCCTCTCCTTTGCAACGAGAGAGGCTCCTCTCCTTTGCGATTTCACTCGCTAGGAGGTCTCGGGCCAATTACTGCAACGAGAGAGGCTGCTA 5′
Denaturing
Renaturing
5′ GTCGACTAGTTCCCCGGTTAATGACGTTGCTCTCTCCGAGGAGAGGAAACGTTGCTCTCTCCGAGGAGAGGAAACGCTAAAGTGAGCGATCCTCCAGAGCCCGGTTAATGACGTTGCTCTCTCCGACGAT 3′
3′ CAGCTGATCAAGGGGCCAATTACTGCAACGAGAGAGGCTCCTCTCCTTTGCAACGAGAGAGGCTCCTCTCCTTTGCGATTTCACTCGCTAGGAGGTCTCGGGCCAATTACTGCAACGAGAGAGGCTGCTA 5′

Figure 3.1. Both denaturing and renaturing of a double DNA strand [2].

quired to break the adenine and thymine pair [15].

The reverse of denaturing is referred to as *renaturing* or *annealing*. The denatured or differently created two single DNA strands (both of them containing exactly the same matched bases) are renatured or annealed to each other when the temperature cools. The annealing process or reaction has recently been referred to as *hybridisation*, and this spontaneous reaction is now emerging as a powerful tool in molecular engineering. In particular, the complementary location can be detected in a tissue section in order to label DNA fragments and to observe the existence of annealed DNA strands [2, 16]. Figure 3.1 illustrates a double DNA strand that has been denatured and annealed under its hydrogen bonds.

3.2 Restriction Enzyme Methods

In the exploration of recombinant DNA by molecular computation, breaking the specific sites of DNA molecules and isolating some particular DNA fragments are achieved by molecular engineering methods, which are referred to as *restriction enzyme* methods. Restriction enzymes are also referred to as *nucleases* that break apart DNA molecules, where they match the restriction site of the short sequence nucleotides. The common restriction sites of the enzyme are constructed from four, five or six nucleotides, including their complementary parts on the DNA base sequence. Several hundred restriction enzymes have been isolated from microorganisms, and each of these has been identified as having different restriction sites [17].

DNA fragments with different lengths are obtained by amplifying various DNA base sequences and by breaking them apart using restriction enzyme methods. In other words, the enzyme recognises specific DNA base sequences and breaks at a certain position. For instance, one restriction enzyme is referred to as *Bam*HI (microorganism: *Bacillus amyloliquefaciens* H), which has six bases of the restriction site (base sequences: one strand is represented as 5′-GGATCC-3′, and the other strand is also represented as 5′-GGATCC-3′), broken at the position between one G and the next G [18]. Three examples of the common restriction enzymes (*Bam*HI, *Pst*I, and *Eco*RI) are shown in Figure 3.2.

The restriction enzymes that act as nucleases, breaking at the position of an internal edge in a DNA strand, are known as type II restriction endonucleases, and all have a

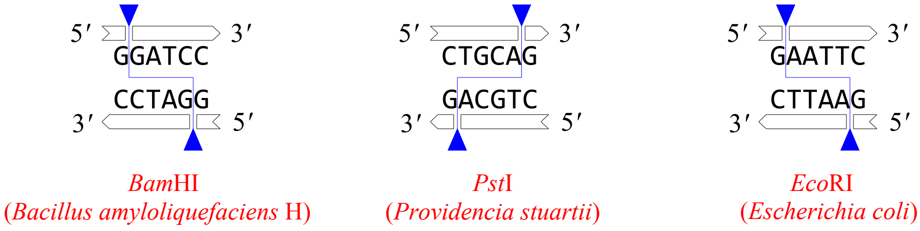


Figure 3.2. Three examples of the restriction enzymes (*Bam*HI, *Pst*I, and *Eco*RI) with DNA base sequences, sources, and breaking sites [3].

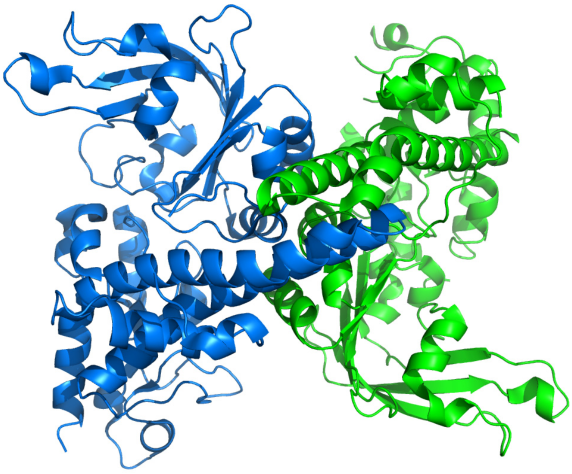


Figure 3.3. Example of the restriction enzyme (*Eco*RII) dimer structure. Creative Commons Attribution-Share Alike 3.0: Thomas Splettstoesser, 2008.

common mode of action. The expected frequency in any of the DNA base sequences is able to be calculated by 4^n, since the possibility of four bases is considered. Here, n is the specific length of the recognised sequence, and the length of base occurrences of three alternative sites can be predicted: (1) every 256-base pair indicates tetranucleotide sites; (2) every 1024-base pair indicates pentanucleotide sites; and (3) every 4096-base pair indicates hexanucleotide sites. The cutting mechanisms in the restriction enzymes are either blunt-ended or sticky-ended [19]. Figure 3.3 shows an example of a restriction enzyme (*Eco*RII) from the protein databank.

Restriction enzymes are used to add the enzyme to the DNA molecules with a buffer. Some restriction enzymes work only in a specific buffer that contains different salt concentrations to ensure a reliable result. It is recommended that most common restriction enzyme reactions be incubated at 37°C. The amount of the restriction endonuclease is required to have a complete digest of 1μg for substrate DNA in a volume reaction of 0.05ml under three optimal solutions of salt concentration, potential hydrogen (pH), and temperature for one hour [20].

3.3 DNA Ligations

DNA ligases repair an unstable double DNA strand, in which one of the single DNA strands or both single DNA strands contain one or more DNA nucleotides that are not bonded together. The attaching formations of phosphodiester bonds are required to be catalysed by DNA ligases for many important purposes in the area of the recombinant DNA experiment using molecular computation, such as for making a sustainable form of a unified double DNA strand complex.

The plasmid vector and DNA fragments are fused to become one DNA molecule by a biochemical process, known as a *DNA ligation* process. T4 DNA ligases [21] are commonly used to seal single DNA strand gaps for catalysing phosphodiester bonds in duplex DNA. T4 DNA ligases clearly work for both blunt-ended and sticky-ended types in the DNA ligation process. A ligation buffer with the enzyme contains adenosine

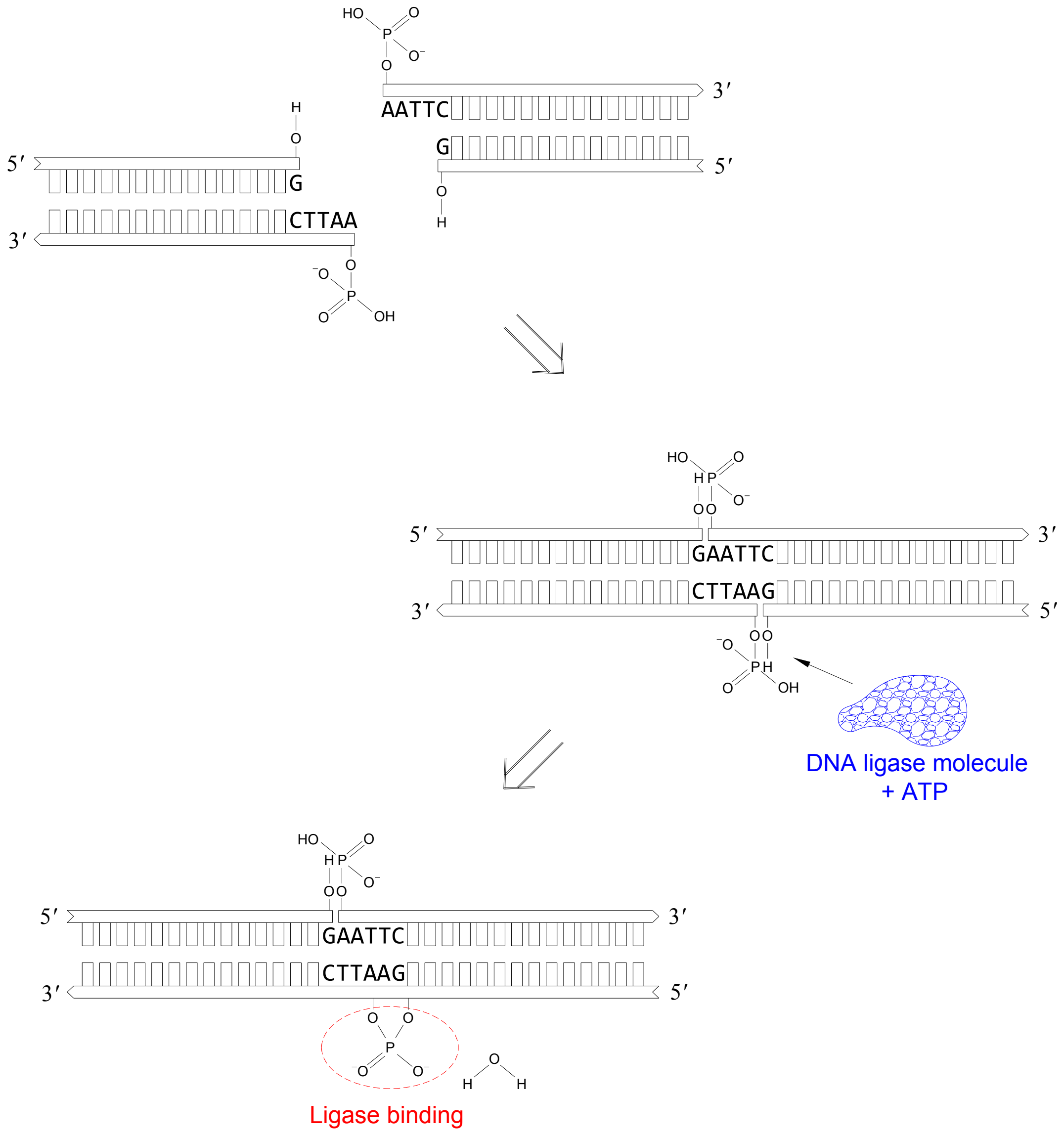


Figure 3.4. Example of the DNA ligation process of DNA strands after being cut by the restriction enzyme (*Eco*RI) [7].

triphosphate (ATP), which is labile at normal laboratory room temperature [22]. All varieties of enzymes and buffers should be kept on ice once they have been removed from the freezer. Figure 3.4 shows an example of the DNA ligation process.

3.4 Cloning Plasmids

In cloning DNA molecules, DNA ligases are associated with either the encoded bacteriophage or the origin bacteria. All eubacteria include a single ligase of the encoded dependent enzyme. In the eukaryote example of the encoded ATP-dependent ligases, the

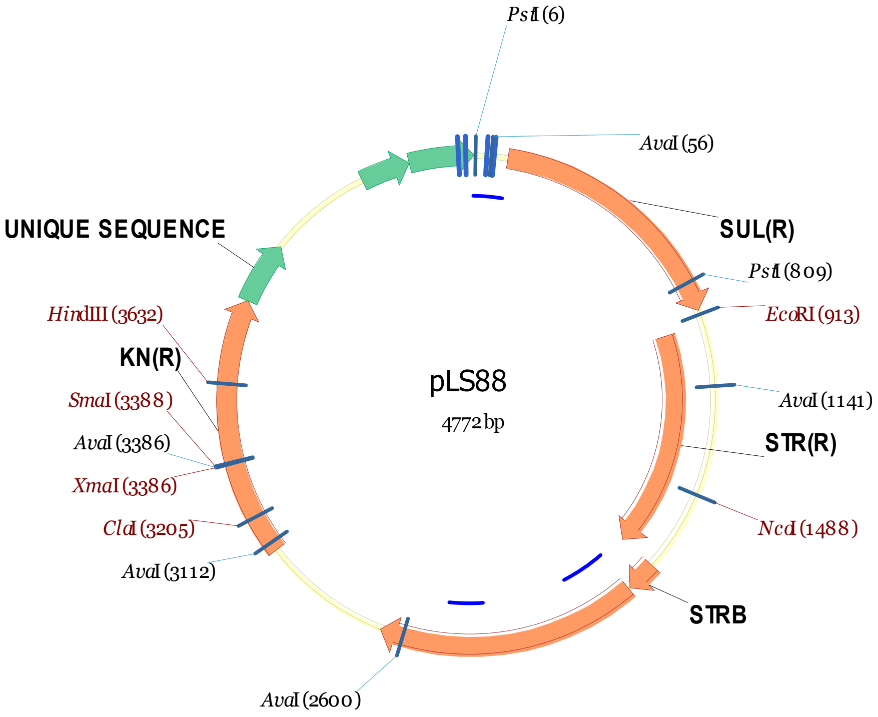


Figure 3.5. Example of the plasmid map (Vector NTI software).

high energy of the intermediate enzyme can be formed by pyrophosphate hydrolysis. The DNA ligases are able to clone various DNA molecules, such as DNA and RNA hybrids, blunt-ended duplexes, and single DNA strands [23].

A *plasmid* is a circular extra-chromosomal genetic element and a double DNA strand. To replicate DNA molecules autonomously in archaeal, bacterial, and eukaryotic cells, the plasmid is associated with an important phenomenon of replication. Figure 3.5 illustrates an example of the plasmid map. A *vector* is another plasmid of a DNA molecule. The term "dealing with vectors" is often used in the genetic engineering field to indicate a carrier of a plasmid or bacteriophage. The term "dealing with clones" is used when a single DNA fragment is linked to the vector [24].

The entire human genomic DNA is estimated to represent approximately 750,000 clones. In fact, about 90% of the genome can only be represented as a clone library, because the probability is associated with any particular fragments of DNA molecules that are ligated into a single plasmid. A restriction enzyme exists in the phage vector and digests the genome, where the DNA molecule is incubated with limited enzymes. Thus, this part of the digest creates a range of DNA fragment sizes. As shown in Figure 3.6, each of the four kinds of vectors indicates the average inserts and the approximate number of clones. In terms of screening the length of DNA fragments, we have to focus on two attributes (a unique DNA sequence and a complementary binding). The unique DNA sequence has an occurrence probability that can be calculated by identifying both the different and the total nucleotide numbers in that DNA sequence [7]. An example of the occurrence probability graph for DNA sequences is shown in Figure 3.7.

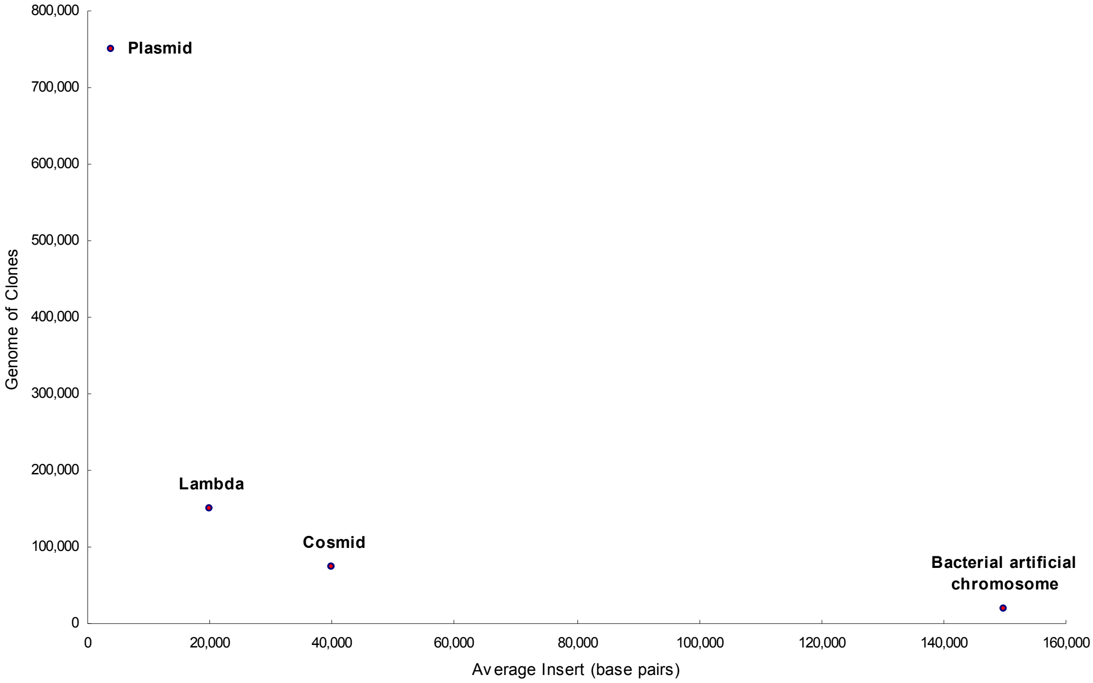


Figure 3.6. Comparison of the given four vectors and the approximate number of clones [7].

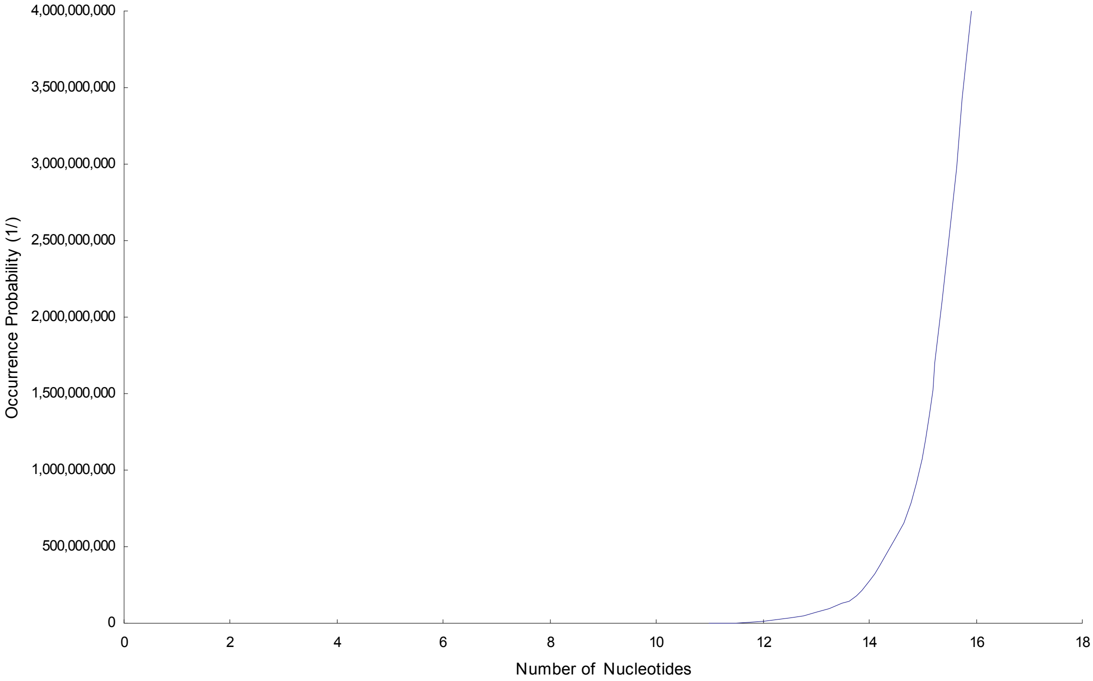


Figure 3.7. Example graph of the occurrence probabilities for DNA sequences, increased by the number of nucleotides [7].

3.5 Polymerase Chain Reaction Technique

To amplify specific DNA sequences into about a half a million by using the simultaneous primers of DNA complementary strands, a molecular engineering technique *in vitro* is used, known as the *polymerase chain reaction* (PCR) technique. PCR works by using two different types of primer extensions that limit the region of DNA for amplifications, and it repeats the template-specific DNA synthesis reaction of specific DNA base sequences by using a heat-resistant DNA polymerase. The PCR technique was specifically developed to analyse the characteristics of DNA and RNA molecules.

For the reaction, the PCR technique requires both nucleosides and the energy to synthesise DNA molecules with DNA polymerases, primers, templates, and buffers. In particular, DNA polymerases are important components, because they are naturally occurring specific enzymes that are biochemical macromolecules, and can be used to catalyse formations of DNA or RNA molecules for repairs [25]. The step of synthesising the DNA molecules is repeated by heating the latest synthesised DNA molecules until they separate, and then cooling them to allow formation of the primers that anneal to the complementary sequences of the DNA molecules. With each of the cycles (heating and cooling), the number of DNA molecules will exponentially increase by conducting each with primer extensions. The predominant reactions create products of DNA molecules, and some of the products are clearly flanked by the primer extensions after several cycles.

Although the enzyme may not synthesise enough DNA molecules or the reaction may decrease in intensity, the DNA molecules continue to increase exponentially during

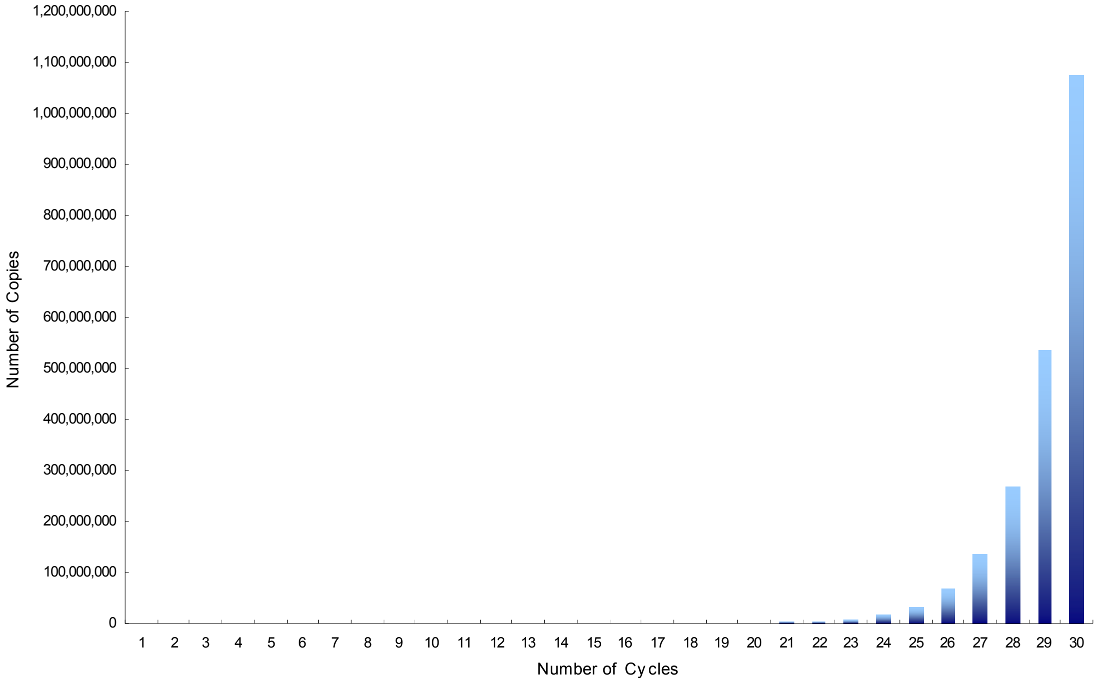


Figure 3.8. Example graph of the number of DNA copies (per cycle) obtained by using the PCR technique [25].

the cycles of heating and cooling. For optimal amplification, the number of cycles is variable depending on each step of amplification or on the amount of initial material. To produce an amount from 100ng to 1μg of DNA molecules, corresponding to a single-copied human sequence coming from 50ng of genomic DNA molecules, it generally requires approximately 25 to 35 cycles of both heating and cooling. The features of PCR remain stable even when the most of the reactions are repeated at high temperatures. The heat-stable enzyme of the *Thermus aquaticus* (*Taq*) DNA polymerase is essentially used to simplify the process, and because this enzyme is rapidly and smoothly active at high temperatures [26].

When conducting the PCR technique, condition setting varies depending on the model of thermal cycler being used. However, any PCR machine can be used to amplify or copy DNA molecules for designing molecular computational algorithms. Figure 3.8 shows the number of DNA copies obtained per cycle by using the PCR technique.

3.6 Affinity Separation Methods

A molecular engineering separation method with affinity chromatography is used to extract one or more specific micromolecules or macromolecules or separate them from others. This method is referred to as an *affinity separation* method. The key role of the affinity separation method scales with a complex mixture of molecules from biochemical extractions, fermentation broths, or cell homogenates [27].

Extraction using magnetic beads has become a useful affinity separation method for collecting and dispersing material within an aqueous solution. This extracting method is also used to separate and extract biochemical matter automatically. In recent years, one use of the technology with magnetic beads is to extract DNA molecules by using fluorescent labels [28]. In this affinity separation method, fluorescently-labelled DNA segments have specific base sequences of ligations, which can bond to magnetic beads. This method involves two techniques: (1) one technique enables the bonding to magnetic beads; and (2) the other technique identifies each magnetic bead with a solidified DNA segment that is differentiated with a particular fluorescent label. The technology can also deal with multiple DNA molecules, which can be extracted and tested simultaneously. The many potential methods and techniques of affinity separation are expected to be automated as a high throughput processor of magnetic beads.

3.7 Gel Electrophoresis Apparatuses

Different sizes of DNA or RNA fragments, specific DNA substrings, DNA strands, and vectors or plasmids commonly result from certain actions (such as digestion) with various types of restriction endonucleases [7]. The different sizes of the DNA molecule structures clearly reveal their various speeds as they move through the electrically charged gel.

DNA molecules (or proteins) are transferred inside the electrically charged gel by generating electric current to separate DNA molecules according to their physical characteristics and then measuring the length of DNA molecules. This molecular engineering mechanism is referred to as a *gel electrophoresis* method, and it is carried out with a gel electrophoresis apparatus. The two most common gel electrophoresis apparatuses are referred to as the *agarose gel electrophoresis* apparatus and the *polyacrylamide gel*

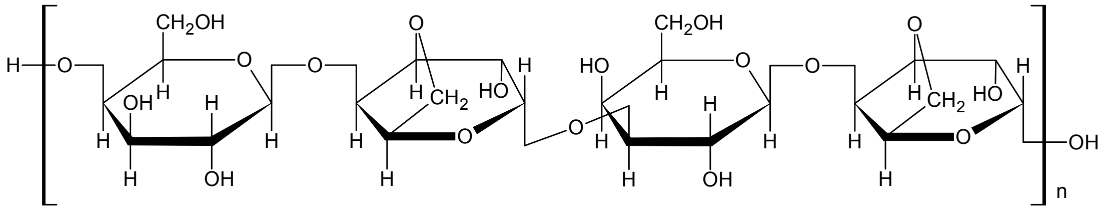


Figure 3.9. Agarose polymer structure, drawn by Ikno Kim using ACD/ChemSketch 5.12. Agarose is used for performing agarose gel electrophoresis.

electrophoresis apparatus. The gel electrophoresis apparatus is useful not only for measuring the length of DNA molecules, but also because it can detach the given DNA fragments from their structures. Figure 3.9 shows an agarose polymer structure that is used for separations in agarose gel electrophoresis.

When clarifying the characteristics of proteins that are included within multiple samples, or when testing multiple proteins that are included within one sample, those proteins can be separated based on their electrophoretic mobility by using *sodium dodecyl sulfate polyacrylamide gel electrophoresis* (SDS-PAGE). The SDS-PAGE apparatus also performs gel electrophoresis using polyacrylamide gel; this method separates both proteins and nucleic acids, and is used to obtain pure protein samples. SDS-PAGE is also used for analysis when studying mass, electrical charge, degree of purity, and testing for existence of proteins [29].

To design molecular computational algorithms and support several DNA experimentations, we used a machine, called *BioCalculator*, which was a fully automatic machine for analysing nucleic acids. The machine automatically measured the length of DNA strands or fragments and provided clear linear band results.

Chapter 4

Minimising the Maximum Production Completion Time Based on a Job-Shop DNA-Based Algorithm

4.1 Overview

In this chapter, a classical job-shop scheduling (we simply call it *job-shop scheduling*) method, which is the general scheduling method used for job-shop scheduling problems, is progressively combined with molecular engineering methods and techniques to create a single DNA-based algorithm. This algorithm is termed a *job-shop DNA-based algorithm*, which has been specifically designed to minimise the maximum production completion time (also known as the *makespan*).

Various types of products or parts that involve their own spending production process times can arise suddenly and should be managed and controlled through production processes in production facilities with different purposes to scheduling. This kind of production scheduling indicates process-focused production scheduling, which deals with low volume and high variety requirements.

The job-shop scheduling method is basically used for solving process-focused production scheduling problems (it is not limited to production scheduling problems). In this production scheduling case, the main purpose of using the job-shop scheduling method is to minimise the maximum production completion time by organising and scheduling the given information about machines, their job orders and each of their own single production process times.

Although the job-shop scheduling method is widely employed for solving process-focused production scheduling problems, computational time consuming of job-shop scheduling still exists in process-focused production management. The problem occurs when handling a large number of machines and their job orders. It is difficult to schedule them to detect an optimal schedule that corresponds to the minimised maximum production completion time. This time offers beneficial and productive ways of scheduling. In job-shop scheduling, minimising the maximum production completion time is an intractable scheduling problem. Further, machine sequences should be considered regarding how all of the given jobs are sequenced by their machine orders.

An example of the problem in job-shop scheduling is shown in this chapter. The example problem involves machines and their job orders with machine sequence orders and process times. The job-shop DNA-based algorithm is used to classify the possible existing schedules into both feasible and unfeasible schedules, and to select the minimised maximum production completion time. This chapter describes several biological and chemical experimental methods and techniques as well as its own molecular computational experimentation, which are all included in parts of the job-shop DNA-based algorithm.

4.2 Background and Motivations

Production companies or organisations continually enhance their productive facilities to offer and distribute a range of new products. At the same time, they should also update their production or other purpose scheduling to last the shortest time, facilitating them to provide their products and satisfy customer or other receiver demands. To fully satisfy these needs, many researchers have proposed solutions for production scheduling problems. Since the beginning of the 1960s, Giffler *et al.* [30] have proposed several solutions for production scheduling problems. As a result, many researchers started to become interested in solving production scheduling problems.

To solve job-shop scheduling problems, many meaningful branch and bound algorithms and methods have been proposed. Carlier *et al.* [31] proposed a branch and bound method focusing on a one-machine scheduling problem. Brucker *et al.* [32] proposed a fast branch and bound algorithm and Streeter *et al.* [33] proposed a new branch and bound algorithm by exploiting local search power. Recently, Artigues *et al.* [34] have proposed a branch and bound method that sets specific times.

A genetic algorithm has become a popular method, and applied methods have been proposed and widely used to provide solutions to job-shop scheduling problems. Yamada *et al.* [35, 36] clearly described the use of genetic algorithms while Lin *et al.* [37] studied parallel genetic algorithms. Cheung *et al.* [38] employed heuristics and genetic algorithms that set specific times and Wu *et al.* recently showed genetic ant colony algorithms.

Several tabu search methods have also been proposed. For example, Hertz *et al.* [39] described a tabu search approach. Thamilselvan *et al.* [40] recently proposed an integrated genetic and tabu search algorithm, and Buscher *et al.* [41] proposed an integrated algorithm to solve the lot streaming problem. Further, a number of meaningful heuristic, meta-heuristic, and other soft computing and evolutionary algorithms and methods have been proposed cf. [42-48].

In this chapter's study, from the viewpoint of molecular computation and engineering, we discuss a molecular computational algorithm (the job-shop DNA-based algorithm), composed of a job-shop scheduling method and a general molecular algorithm that includes its own molecular computational experimentation. The algorithm is used to finally resolve the minimised maximum production completion time.

Regarding common job-shop scheduling problems, we should consider machine sequences, meaning that each of the given jobs has its own machine sequence. Therefore, the job-shop DNA-based algorithm is first associated with consideration of machine sequences. Moreover, the algorithm includes molecular cloning techniques in its new molecular encoding method, which handles the direct relations of machines and their job orders.

Enormous amounts of computation are required, particularly for a large number of machines and job orders with various production process times. Even the job-shop

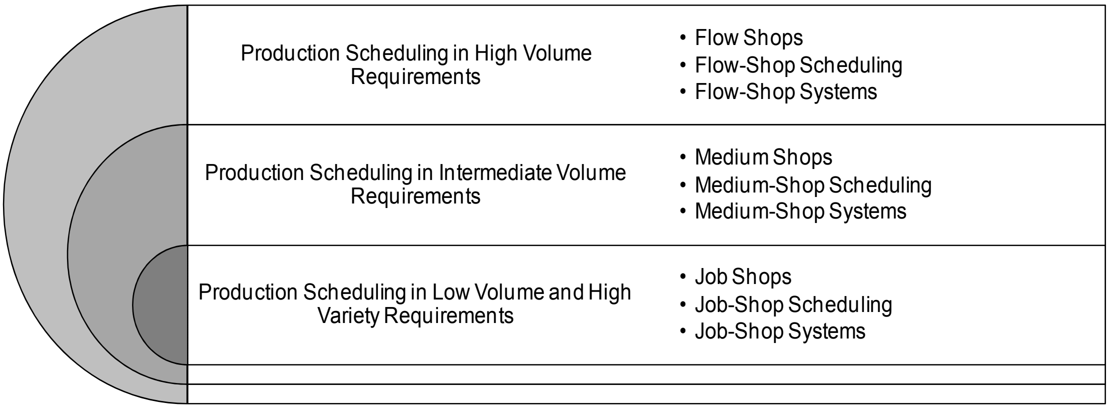

Production Scheduling in High Volume Requirements
• Flow Shops
• Flow-Shop Scheduling
• Flow-Shop Systems
Production Scheduling in Intermediate Volume Requirements
• Medium Shops
• Medium-Shop Scheduling
• Medium-Shop Systems
Production Scheduling in Low Volume and High Variety Requirements
• Job Shops
• Job-Shop Scheduling
• Job-Shop Systems

Figure 4.1. Common production scheduling classified into three different requirements: (1) high volume requirements; (2) intermediate volume requirements; and (3) low volume and high variety requirements.

scheduling of a small number of machines and their job orders brings with it an intractable computational problem [49], meaning that a job-shop scheduling problem is an NP (non-deterministic polynomial-time) -hard problem. Moreover, if some unpredictable products or parts emerge with large numbers with high variety requirements, the solution becomes even more difficult.

4.3 Production Scheduling in Different Volumes

Common scheduling [50] within an organisation determines the point of time when resources from that organisation are closely associated with the utilisation of facilities and human resources. Regardless of the basic nature of corporate activities, common scheduling occurs in most organisations. For instance, to meet the goals for their productions, workforces, facilities, maintenance, etc., most production companies need to build some specific production schedules. As Figure 4.1 shows, common production scheduling can be classified into three (high, intermediate, and low) different volume requirements.

In decision-making processes, production scheduling needs to be complete before the actual products are produced. Many different parts of the decision-making processes on scheduling designs and developments are made long before any of scheduling is planned. These decision-making processes on scheduling include the capacity of productions, product designs, facility selections, machine selections, workforce selections, etc. with their daily, weekly, monthly, overall, and other period scheduling.

In this section, we consider and describe two different types of scheduling that deal with high and intermediate volume requirements. The two types of scheduling are basically focused on production processes, meaning that these scheduling styles are (1) production scheduling in high volume requirements; and (2) production scheduling in intermediate volume requirements.

4.3.1 Production scheduling in high volumes

The characteristics of production scheduling in high volume requirements [50] deal with standardised facilities and activities that enable the same or similar operations for different products. The goal is to facilitate the flow of products through high volume requirements, so as to maximise the operational rate of labour and facilities. The places where high volume products are produced are referred to as *flow shops*. Production scheduling in high volume requirements is referred to as *flow-shop scheduling*. The systems using this flow-shop scheduling are referred to as *flow-shop systems*. Figure 4.2 shows a representation of flow shops in general production machines. In fact, in some cases, such flow-shop scheduling can cover intermediate volume requirements and low volume requirements as well [51].

Due to its repetitive nature, operational loads and orders are significant variables of the flow-shop system in decision making. This system should be designed to facilitate work within the system that deals with specialised materials and equipment, since all the items are produced in each of the same orders.

One important aspect in the design of the flow-shop system should be *line balancing*. A well balanced system enhances not only the productivity, but also the operational rate of facilities and labour. In designing this system process, potential complaints could arise in work relationships in the overall work specialisations, and these problems should be considered. Such problems culminate in exhaustions, long absences, mistakes, etc., due to boredoms and repetition, which might mean less productivity and unexpected quality products. The inherent characteristics of the flow-shop system resolve most of the problems that still exist in production scheduling.

4.3.2 Production scheduling in intermediate volumes

Like flow-shop scheduling, production scheduling in intermediate volume requirements [50] also deals with standardised products. The reason is that its output is not large enough for continuous productions and it is therefore more economical to produce in-

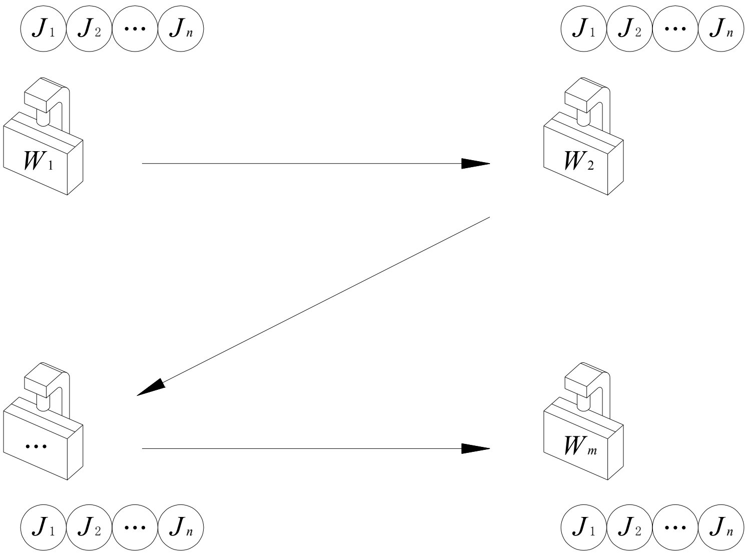


Figure 4.2. Representation of flow shops in general production machines and their job orders.

termediate volumes. In this chapter's study, the places where intermediate volume products are produced are referred to as *medium shops*. Production scheduling in intermediate volume requirements is referred to as *medium-shop scheduling*. The systems using this medium-shop scheduling are referred to as *medium-shop systems*.

Companies focusing on production scheduling in intermediate volume requirements often seek to reduce or eliminate the production time, which consequently leads to down times of the medium-shop system for equipment replacements. Tactically, some preparations for down times might be modular operations and flexible facilities. However, the difficulties come from the fact that the use of product parts is not always the same as it is in theoretical models. Some products can run out faster than expected and need to be filled up more quickly. The main reason for this is that many different types of products are processed simultaneously. This makes it difficult to build such production schedules for optimal running.

The approach used in this medium-shop system is to match the main schedule, which is basically developed based on customer orders and demand forecasting. Such production companies use their own assembly processes based on *material requirements planning* to determine the required amount of parts and the timing. Decision makers compare the expected demand and production volume requirement to build feasible production scheduling.

4.4 Process-Focused Production Scheduling

In the previous section, we described the two different types of scheduling that are controlled by their different (high and intermediate) volume requirements. In this section, we consider not only the volume requirements, but also different variety requirements.

In different kinds of production lines or areas with (1) different volume requirements; and (2) different variety requirements, production scheduling problems often arise. This is because single or group customers or receivers demand faster receipt and more specific products from production companies or organisations acting as their suppliers.

When building production process strategies, production scheduling should be considered and set for each different feature of specific production process strategies. Generally, there are several different types of production process strategies related to scheduling. One production process strategy is mainly associated with *process-focused production scheduling* [52]. This scheduling is devoted to meeting low volume requirements (this is different from the high and intermediate volume requirements described in the previous section) and high variety requirements. The places where low volume and high variety products are produced are referred to as *job shops*. In our production industries, we could say that most production includes process-focused production facilities. Production scheduling in low volume and high variety requirements is referred to as *job-shop scheduling*. The systems using this job-shop scheduling are referred to as *job-shop systems*. As Figure 4.3 illustrates, we can recognise where job-shop scheduling comes from and process-focused production scheduling is positioned between volume and variety. As machine shops, if production facilities focus on process productions, then those facilities have high product flexibilities and highly variable costs with low utilisations.

The recent trend in production would satisfy the needs of individual customers or

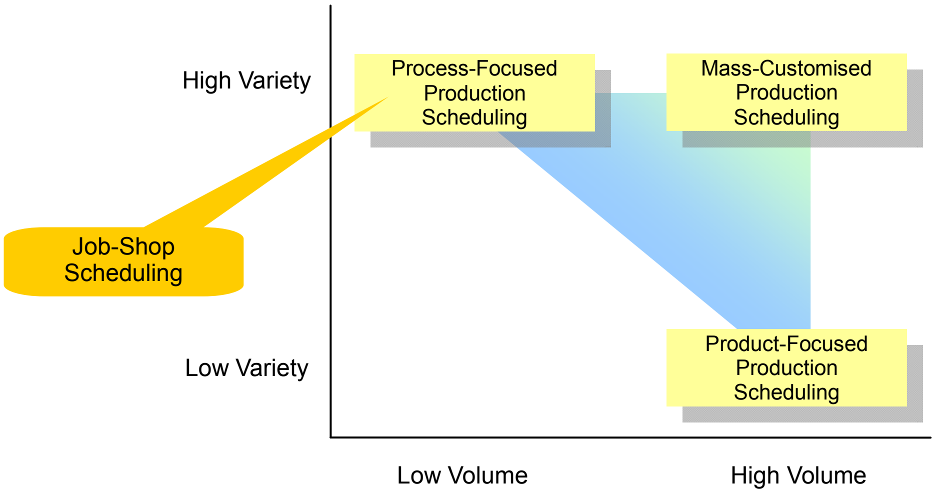

High Variety
Low Variety
Low Volume
High Volume
Process-Focused Production Scheduling
Mass-Customised Production Scheduling
Product-Focused Production Scheduling
Job-Shop Scheduling

Figure 4.3. Representation of the locations of both job-shop scheduling and process-focused production scheduling.

receivers (they can also be groups of customers or receivers), each of which has its own special order, and affect how we provide the ordered products to them on time or as early as possible. In this case, production factories might be required to produce the ordered products in low volumes and high varieties. These factories obviously build and organise process-focused production scheduling for a forward-looking scheme. However, the management of this scheme is complicated, and it would become more complicated if the number of product parts increases. The main problem of complexity is to deal with a large number of machines and their job orders. The most important point is that we have to develop methods or algorithms that can control the system and overcome this computational complexity problem.

4.5 Job-Shop Scheduling

Four different types of production facilities [52], with different approaches to different scheduling styles, can be suggested. These are (1) work cells; (2) repetitive production facilities; (3) product-focused production facilities; and (4) process-focused production facilities. In simple terms, (1) work cells focus on production facilities that deal with group families of similar components; (2) repetitive production facilities focus on such assembly lines or channels; (3) product-focused production facilities deal with high volume and low variety requirements; and (4) process-focused production facilities deal with low volume and high variety requirements. Here, these process-focused production facilities include job-shop scheduling in terms of building production scheduling used for those production facilities. Figure 4.4 shows a representation of job shops in general production machines.

A job-shop scheduling problem is an intractable scheduling problem, because a great number of jobs need to be processed in each of the specific machines within a de-

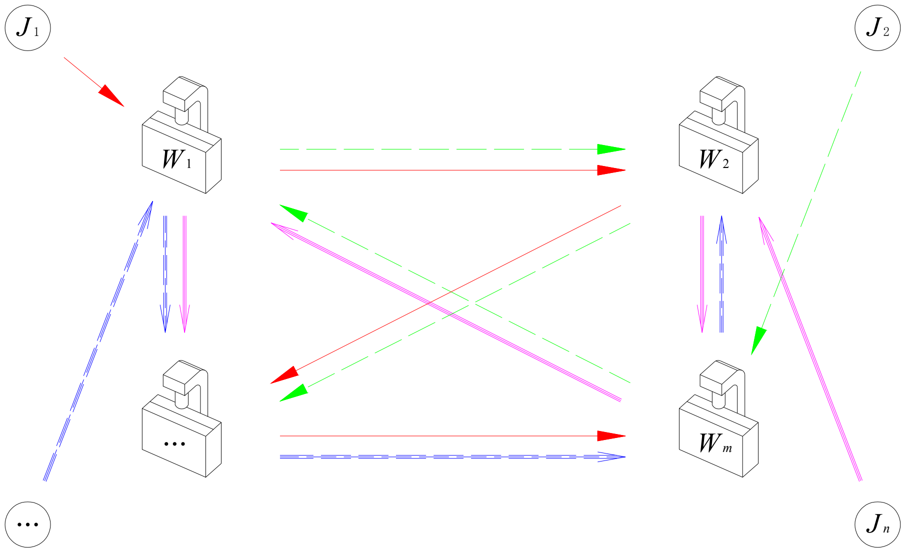


Figure 4.4. Representation of job shops in general production machines and their job orders.

termined time or at the earliest time. The main purpose of using the job-shop DNA-based algorithm is to determine the minimised maximum production completion time in job-shop scheduling. In this section, we describe job-shop scheduling in more detail to identify the main purpose of the algorithm.

4.5.1 Production scheduling in low volumes and high varieties

Recall that job-shop scheduling corresponds to production scheduling in low volume and high variety requirements [50] in terms of productions. The characteristics of production scheduling in low volume and high variety requirements are different from the characteristics of production scheduling in high and intermediate volume requirements. Hence, job-shop scheduling and systems are different from flow-shop and medium-shop scheduling and systems.

In job-shop systems, products are produced according to each of the machines and each of the job orders. In addition, these machines and job orders are comprised of various types of complicated preparation work. Due to these various machines and job orders, it is not easy to plan job-shop scheduling properly. The matter is further complicated by the fact that the companies cannot even plan a schedule before each of the actual job orders is placed. The job-shop system generally causes two basic issues, which are how to (1) allocate job loads to each of the machines; and (2) sequence the given jobs based on their machine orders. Job-load allocations include assigning certain jobs to certain machines. If jobs can be done by one machine, the allocation will not create an issue, but there could be one if there are more than two jobs and multiple machines for them. In this case, a specific rule for allocations is necessary. The job-load allocations aim to minimise the costs required for their preparations, and the main purpose is to minimise the maximum production completion time.

A tool called a *Gantt chart* is often used in association with job-load allocations for job-shop scheduling. The main purpose of the Gantt chart is to visually organise the actual or planned use of resources within a time frame. Time is usually on the horizontal line and resources on the vertical line; this chart reflects the use of resources and idle times. The Gantt chart can be used for job-shop scheduling by adjusting production process times and allocating job loads, and the chart represents all the specific production completion times of either the machines or jobs.

There are two different types of job loading to machines. One is called *infinite loading*, and the other one is called *finite loading*. Under infinite loading, jobs are assigned regardless of the machine's capacity, in which case loading can be excessive or low, depending on their production process times. The common rule is that infinite loading creates waiting lines for some or all machines.

Finite loading scheduling considers required production process times for the machine capacity, so that the machine capacity does not go overboard when planning the actual starting and stopping times. The output from finite loading is the specific forecast of production process times for each machine and any scheduling should be updated frequently.

4.5.2 Production problem of job-shop scheduling

Job-shop scheduling in production is always concerned with machine sequences for each of the given job orders. In the production of job-shop scheduling, we confront one main problem, which is how we can determine an optimal production schedule composed of the most reliable minimised maximum production completion time. This is the main achievement reached by using the job-shop DNA-based algorithm.

To define our focused problem of job-shop scheduling in production, an example of a digraph, representing job flows of job shops, is shown in Figure 4.5. The example digraph is composed of two nodes (the beginning and finishing nodes) and eight nodes

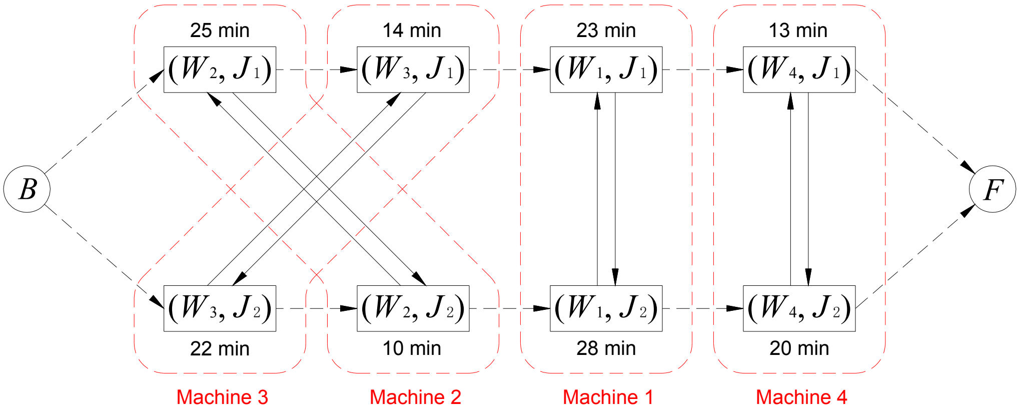


Figure 4.5. Example of a digraph composed of machines and job orders with their production process times for job-shop scheduling. The dashed arcs indicate conjunctive arcs, and the continuous arcs indicate disjunctive arcs.

(the eight operational pair nodes). Here, each of the eight operational pair nodes is composed of one arbitrary machine and one arbitrary job. We generated this simple example of the job-shop scheduling digraph as shown in Figure 4.5. In an attempt to properly describe our algorithm structure, this simple example can exhibit significant potential to cover a large volume of machines and their job orders.

The job-shop scheduling digraph can be described in a mathematical way [53, 54]. The digraph is denoted by *G*. The beginning node and the finishing node are denoted by *B* and *F*, respectively. The arbitrary operational pair is denoted by either (W_i, J_j) or (i, j), where i is the production process step of the machine ($i = 1, 2,\ldots, m$) and j is the production process step of the job ($j = 1, 2,\ldots, n$). When the operational pair (i, j) precedes the operational pair (k, j), this arc of the two operational pairs is represented as $(i, j)\rightarrow(k, j)$. In addition, the digraph consists of two different types of arc subsets. The first arc subset consists of conjunctive arcs (the dashed arcs in the example digraph). The other arc subset consists of disjunctive arcs (the continuous arcs in the example digraph).

The production completion time of job j on machine i is denoted by $T_{i,j}$, and the time job j required for the job-shop system to process, denoted by T_j. Thus, the average of T_j is defined as

$$\overline{T_j} = \frac{T_1 + T_2 + \cdots + T_{n-1} + T_n}{n} = \frac{1}{n}\sum_{j=1}^{n} T_j\,. \tag{4.1}$$

In addition, the maximum production completion time is particularly denoted by $T_{\max}$, and defined as

$$T_{\max} = \max(T_1, T_2, \cdots, T_{n-1}, T_n) = \max_{1\le j\le n} T_j\,. \tag{4.2}$$

Here, the maximum production completion time corresponds to the completion time of the last remaining job that leaves the job-shop system.

In Figure 4.5, a feasible production schedule of the maximum production completion time corresponds to the longest path among the given operational pair nodes (except the beginning and finishing nodes). This longest path starts at one of the operational pair nodes and ends at another of the operational pair nodes. The longest path consists of a subset of operational pair nodes, where the first operational pair node (one or more operational pair nodes) starts at time zero and the last operational pair node (one or more operational pair nodes) completes at a specific time, which corresponds to the maximum production completion time.

The main issue of job-shop scheduling in this chapter's study is how to minimise the length of the production completion time and to select disjunctive arcs, which minimise the length of the longest path. This longest path is referred to as the *critical path*. The program of minimising the maximum production completion time is defined as follows:

$$\text{minimise} \quad T_{\max} \tag{4.3}$$

subject to

$$T_{\max} - u_{i,j} \ge v_{i,j} \quad \text{for all } (i, j) \in N \tag{4.4}$$

$$u_{k,j} - u_{i,j} \geq v_{i,j} \quad \text{for all } (i, j) \rightarrow (k, j) \in C \tag{4.5}$$

$$u_{i,j} - u_{i,l} \geq v_{i,l} \quad \text{or} \quad u_{i,l} - u_{i,j} \geq v_{i,j} \quad \text{for all } (i, l) \text{ and } (i, j) \tag{4.6}$$

$$u_{i,j} \geq 0 \quad \text{for all } (i, j) \in N, \tag{4.7}$$

where (1) $v_{i,j}$, $i = 1, 2,\ldots, m$ and $j = 1, 2,\ldots, n$, is the production process time; (2) $u_{i,j}$, $i = 1, 2,\ldots, m$ and $j = 1, 2,\ldots, n$, is the variable of the starting time for the operational pair (i, j); (3) N is the set of all the operational pairs (i, j); and (4) C is the set of all the routing constraints.

4.6 Encoding Operational Pairs in DNA

A novel molecular encoding method has been designed to be included in the job-shop DNA-based algorithm. In this section, we describe how to encode all the given operational pair nodes in DNA through our purposely designed matrix.

4.6.1 Operational pair matrix

All of the operational pair nodes are connected at their arcs, corresponding to both conjunctive and disjunctive arcs. However, for the molecular encoding method, the conjunctive and disjunctive arcs have exactly the same characteristics. These two different arcs are expressed as continuous arcs only. Both the beginning and finishing nodes have also been removed. This new digraph type for the molecular encoding method is called a *DNA-digraph*, as shown in Figure 4.6. In this chapter's study, the DNA-digraph is transformed into a matrix form, called an *operational pair matrix*.

In the DNA-digraph, assuming that n given operational pair nodes are composed of both m machines $W_1, W_2,\ldots, W_m$ and n jobs $J_1, J_2,\ldots, J_n$. A set of all the operational pair nodes is denoted as Q_s, which can express an operational pair matrix, and this matrix is denoted by Q, representing

$$Q = \begin{bmatrix} Q(W_1,J_1) & Q(W_2,J_1) & \cdots & Q(W_m,J_1) \\ Q(W_1,J_2) & Q(W_2,J_2) & \cdots & Q(W_m,J_2) \\ \vdots & \vdots & \ddots & \vdots \\ Q(W_1,J_n) & Q(W_2,J_n) & \cdots & Q(W_m,J_n) \end{bmatrix}. \tag{4.8}$$

The structure of this operational pair matrix is composed of operational pair entries, and all of the entries consist of both m machines and n jobs. If there are some machines that include fewer job orders than other machines, then the symbol 'ϕ' is used for the empty entries. Further, in the operational pair matrix, each of the given entries should be freely allocated to each of the specific entries based on the machine sequences, depending on their job orders.

In general, we set (1) $W_1, W_2,\ldots, W_m$ that are labelled as machine 1, machine 2,…, machine m, respectively; and (2) $J_1, J_2,\ldots, J_n$ that are labelled as job 1, job 2,…, job n, respectively. Figure 4.7 shows the DNA-digraph of the example job-shop scheduling digraph. A set of the existing operational pair nodes is represented as $Q_s = \{(W_1, J_1),$

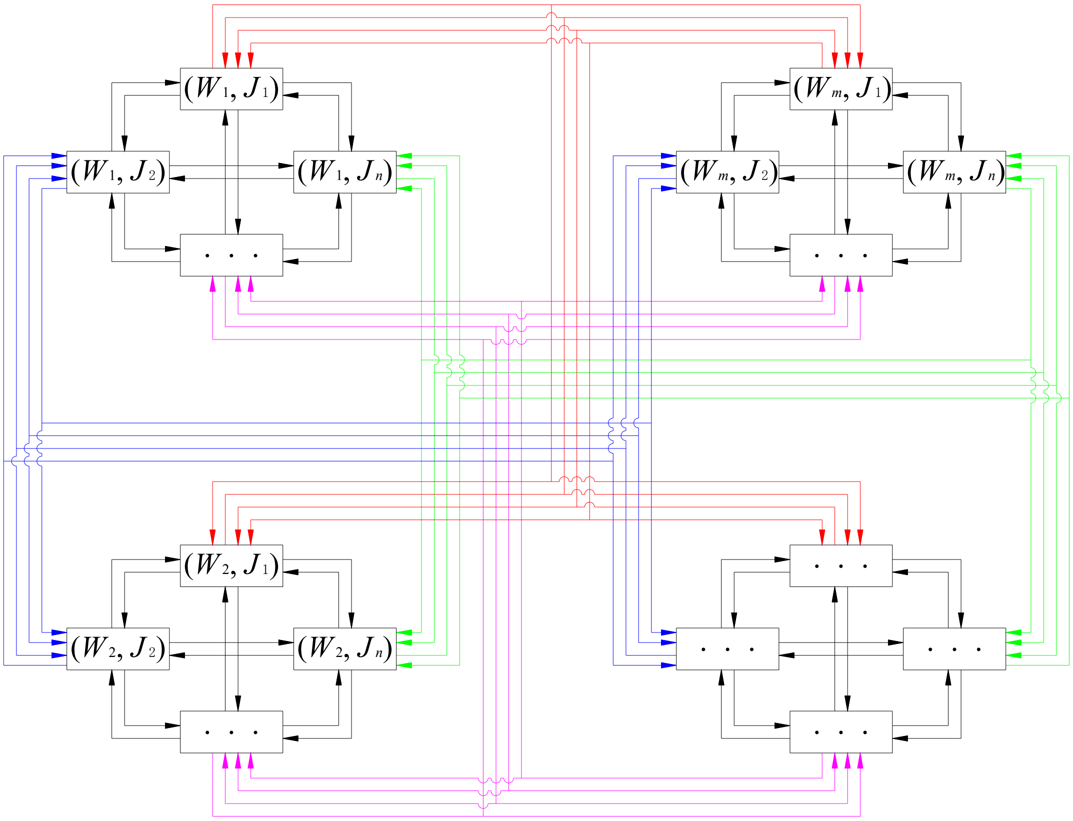


Figure 4.6. DNA-digraph for minimising the maximum production completion time in job-shop scheduling.

(W_1, J_2), (W_2, J_1), (W_2, J_2), (W_3, J_1), (W_3, J_2), (W_4, J_1), $(W_4, J_2)\}$, and an operational pair matrix Q is constructed as

$$Q = \begin{bmatrix} Q(W_1,J_1) & Q(W_2,J_1) & Q(W_3,J_1) & Q(W_4,J_1) \\ Q(W_1,J_2) & Q(W_2,J_2) & Q(W_3,J_2) & Q(W_4,J_2) \end{bmatrix}. \tag{4.9}$$

If this matrix is allocated based on the rules of the given machine sequences, as shown in Figure 4.7, then the machine sequence-based operational pair matrix Q becomes

$$Q = \begin{bmatrix} Q(W_2,J_1) & Q(W_3,J_1) & Q(W_1,J_1) & Q(W_4,J_1) \\ Q(W_3,J_2) & Q(W_2,J_2) & Q(W_1,J_2) & Q(W_4,J_2) \end{bmatrix}. \tag{4.10}$$

This matrix of the example DNA-digraph is mainly used when encoding the DNA sub-strings.

The relation between any of two operational pair nodes in the DNA-digraph is denoted by e, where e is the relation connecting these two operational pair nodes. The relation between two operational pair nodes, if it is an arbitrary operational pair node (W_α,

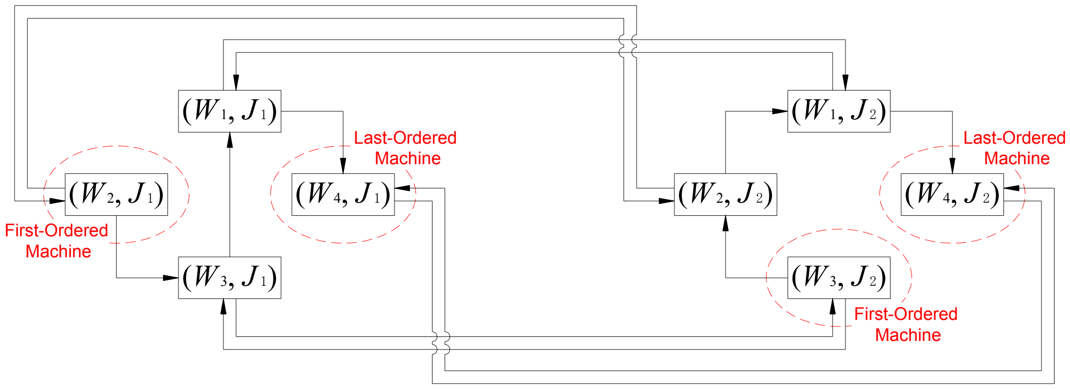


Figure 4.7. Example DNA-digraph composed of eight operational pair nodes, including the first-ordered and last-ordered machines.

J_α), has a relational connection with another arbitrary operational pair node (W_β, J_β), representing (W_α, J_α)e(W_β, J_β), and (W_α, J_α)$\bar{e}$(W_β, J_β) is represented as no relational connection. For direct relations of the DNA-digraph, the representation of the arcs among operational pair nodes, a directional arrow of the operational pair node (W_α, J_α) reaches another operational pair node (W_β, J_β), representing

$$a_l = \overrightarrow{((W_\alpha, J_\alpha),(W_\beta, J_\beta))}, \alpha, \beta, \text{ and } l = 1, 2, \cdots, n \text{ for } \alpha \neq \beta . \tag{4.11}$$

Additionally, a set of all existing arcs from the operational pair node (W_α, J_α) to the operational pair node (W_β, J_β) is denoted as A, where A consists of all the possible arcs,

$$\begin{array}{c|cccccccc|}
 & (W_1, J_1) & (W_1, J_2) & (W_2, J_1) & (W_2, J_2) & (W_3, J_1) & (W_3, J_2) & (W_4, J_1) & (W_4, J_2) \\
(W_1, J_1) & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
(W_1, J_2) & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
(W_2, J_1) & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
(W_2, J_2) & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
(W_3, J_1) & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
(W_3, J_2) & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
(W_4, J_1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
(W_4, J_2) & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}$$

Figure 4.8. Binary adjacency matrix of the example DNA-digraph.

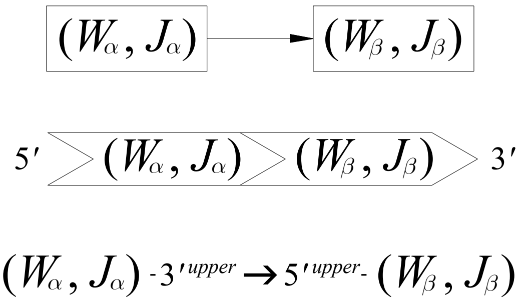


Figure 4.9. Double-encoded substring of type 1.

representing $A = \{a_1, a_2, \ldots, a_n\}$.

The arc set is properly represented as a matrix with binary numbers, called a *binary adjacency matrix*. As Figure 4.8 shows, the binary adjacency matrix of the example DNA-digraph comes with rows and columns labelled as follows:

$$r_{i,j}, i \text{ and } j = 1, 2, \cdots, 8 \text{ for all } (i, j) \in L, \tag{4.12}$$

where i is the row label, j is the column label, and L is the set of all row and column labels. This example matrix has been constructed by setting $r_{i,j} = 1$ wherever there are any arcs directed from an arbitrary operational pair node (W_α, J_α) to another arbitrary operational pair node (W_β, J_β), meaning both $(W_\alpha, J_\alpha) \rightarrow (W_\beta, J_\beta)$ and $(W_\alpha, J_\alpha)e(W_\beta, J_\beta)$, while the example matrix is also constructed by setting $r_{i,j} = 0$ elsewhere, meaning $(W_\alpha, J_\alpha)\bar{e}(W_\beta, J_\beta)$. In a systematic way, ordering rows and columns of the binary adjacency matrix is the best way of transforming them into DNA sequences.

4.6.2 DNA substrings

For the job-shop DNA-based algorithm, two different types of DNA substrings are created to build an initial library of DNA fragments. Each type of the two different DNA substrings has its own row and column labels, which are defined for encoding the DNA substrings of operational pair nodes in single-stranded DNA (ssDNA).

In the DNA-digraph, if any two different single operational pair nodes are connected by an arc, then two different DNA substrings of those two different single operational pair nodes are unified to become one single DNA substring, which is called a *double-encoded substring*.

Type 1 is created from the double-encoded substrings, and type 2 is created from the complementary substrings of two different double-encoded substrings. The two types of DNA sequences (both double-encoded substrings and complementary substrings) are structured in ssDNA.

Firstly, type 1 is indicated as a double-encoded substring that is composed of two different operational pair nodes (W_α, J_α) and (W_β, J_β). For type 1, there are two different

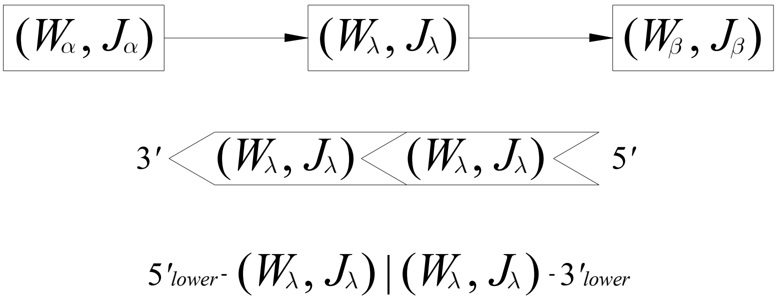


Figure 4.10. Complementary substring of type 2.

operational pair nodes, in which there is an arc that indicates the direction from the operational pair node (W_α, J_α) to the operational pair node (W_β, J_β). A double-encoded substring of this direction is denoted as (W_α, J_α)-$3'^{upper}$→$5'^{upper}$-(W_β, J_β). The two different operational pair nodes are encoded as a single DNA oligonucleotide that consists of two unique sites, and each of the single encoded DNA oligonucleotides has a length of DNA base pairs (bp: after hybridisations and end-filling DNA). Each of the lengths is created based on its production process times. For the example digraph in Figure 4.5, as shown in Tables 4.1 and 4.2, 1 minute was set with a length of 1 bp (for instance, 50 bp equals 50 minutes). As Figure 4.9 shows, (1) one arbitrary operational pair node (W_α, J_α) corresponds to (W_α, J_α)-$3'^{upper}$ with this pair node's own length; and (2) the other arbitrary operational pair node (W_β, J_β) corresponds to $5'^{upper}$-(W_β, J_β) with this pair node's own length.

Secondly, type 2 is indicated as a complementary substring, attaching two different double-encoded substrings. The functions of DNA ligations and hybridisations [55, 56] make connections of all three different operational pair nodes, meaning that these three nodes are sequentially lined up together and become double-stranded DNA (dsDNA). Figure 4.10 shows the attachment of three different linked operational pair nodes, denoted as (W_α, J_α) for the starting node, (W_λ, J_λ) for the middle node, and (W_β, J_β) for the ending node. A complementary substring of the middle operational pair nodes is created to link these three operational pair nodes together. This complementary substring is denoted as a complementary encode $5'_{lower}$-(W_λ, J_λ)|(W_λ, J_λ)-$3'_{lower}$ for type 2.

4.7 Manipulating DNA Plasmids for the Algorithm

Our concept of applying manipulated *DNA plasmids* (we call them 'DNA plasmids' instead of 'plasmids,' because RNA plasmids are extremely rare) to the problem of job-shop scheduling is described in this section. In addition, we describe how this application is related to machine sequences and job orders involving molecular encoding sequences. Moreover, to help understand our approach to manipulating DNA plasmids for the job-shop DNA-based algorithm, we describe techniques for manipulating DNA plasmids with their further characteristics of molecular biology (cloning plasmids was

Table 4.1. Upper DNA sequences in ssDNA used for hybridisations and ligations.

Substring	DNA Sequence (5′ to 3′)	Substring	DNA Sequence (5′ to 3′)
$(W_1, J_1)\text{-}3'^{upper}$	GTAAGAGTAACC	$5'^{upper}\text{-}(W_1, J_1)$	CCGGAGTGAAG
$(W_1, J_2)\text{-}3'^{upper}$	ACCTATGAGGCTGG	$5'^{upper}\text{-}(W_1, J_2)$	ATAAAACTGCCGTG
$(W_2, J_1)\text{-}3'^{upper}$	GCAACCGCCTTCA	$5'^{upper}\text{-}(W_2, J_1)$	ACGACCCGGACG
$(W_2, J_2)\text{-}3'^{upper}$	CGTGG	$5'^{upper}\text{-}(W_2, J_2)$	CACTT
$(W_3, J_1)\text{-}3'^{upper}$	AGTTGCA	$5'^{upper}\text{-}(W_3, J_1)$	TTCCTAG
$(W_3, J_2)\text{-}3'^{upper}$	CCGCTATAATT	$5'^{upper}\text{-}(W_3, J_2)$	GTCTCTTTGCC
$(W_4, J_1)\text{-}3'^{upper}$	AACCACA	$5'^{upper}\text{-}(W_4, J_1)$	CCATAG
$(W_4, J_2)\text{-}3'^{upper}$	GAATACCGTT	$5'^{upper}\text{-}(W_4, J_2)$	TCGGCAGGCG

Table 4.2. Lower DNA sequences in ssDNA used for hybridisations and ligations.

Complementary Substring	DNA Sequence (5′ to 3′)
$5'_{lower}\text{-}(W_1, J_1)\|(W_1, J_1)\text{-}3'_{lower}$	GGTTACTCTTACCTTCACTCCGG
$5'_{lower}\text{-}(W_1, J_2)\|(W_1, J_2)\text{-}3'_{lower}$	CCAGCCTCATAGGTCACGGCAGTTTTAT
$5'_{lower}\text{-}(W_2, J_1)\|(W_2, J_1)\text{-}3'_{lower}$	TGAAGGCGGTTGCCGTCCGGGTCGT
$5'_{lower}\text{-}(W_2, J_2)\|(W_2, J_2)\text{-}3'_{lower}$	CCACGAAGTG
$5'_{lower}\text{-}(W_3, J_1)\|(W_3, J_1)\text{-}3'_{lower}$	TGCAACTCTAGGAA
$5'_{lower}\text{-}(W_3, J_2)\|(W_3, J_2)\text{-}3'_{lower}$	AATTATAGCGGGGCAAAGAGAC
$5'_{lower}\text{-}(W_4, J_1)\|(W_4, J_1)\text{-}3'_{lower}$	TGTGGTTCTATGG
$5'_{lower}\text{-}(W_4, J_2)\|(W_4, J_2)\text{-}3'_{lower}$	AACGGTATTCCGCCTGCCGA

briefly described in Chapter 3).

4.7.1 Application of manipulating DNA plasmids

Our concept of manipulating DNA plasmids, as applied to the job-shop scheduling problem for the job-shop DNA-based algorithm, is to focus on two different features of directed paths (cyclic operational paths and acyclic operational paths) in the DNA-digraph.

The job-shop scheduling digraph consists of two different arc types (conjunctive and disjunctive arcs). A subset of the determined disjunctive arcs is denoted by *D*, and this job-shop scheduling digraph is denoted by *G*(*D*). This digraph is composed of both (1) the subset *D*; and (2) a subset of conjunctive arcs. If and only if *G*(*D*) has no directed cyclic paths, the subset *D* can be a feasible production schedule [53, 54]. In other words, recall that the DNA-digraph is created by the given operational pair nodes and their continuous arcs (there is no distinction between conjunctive and disjunctive arcs). In this DNA-digraph, if there is one cycle of operational pair nodes, then it is impossible to process that production schedule. This schedule is incompatible with common job-shop scheduling.

In the DNA-digraph, cyclic paths of operational pair nodes correspond to unfeasible

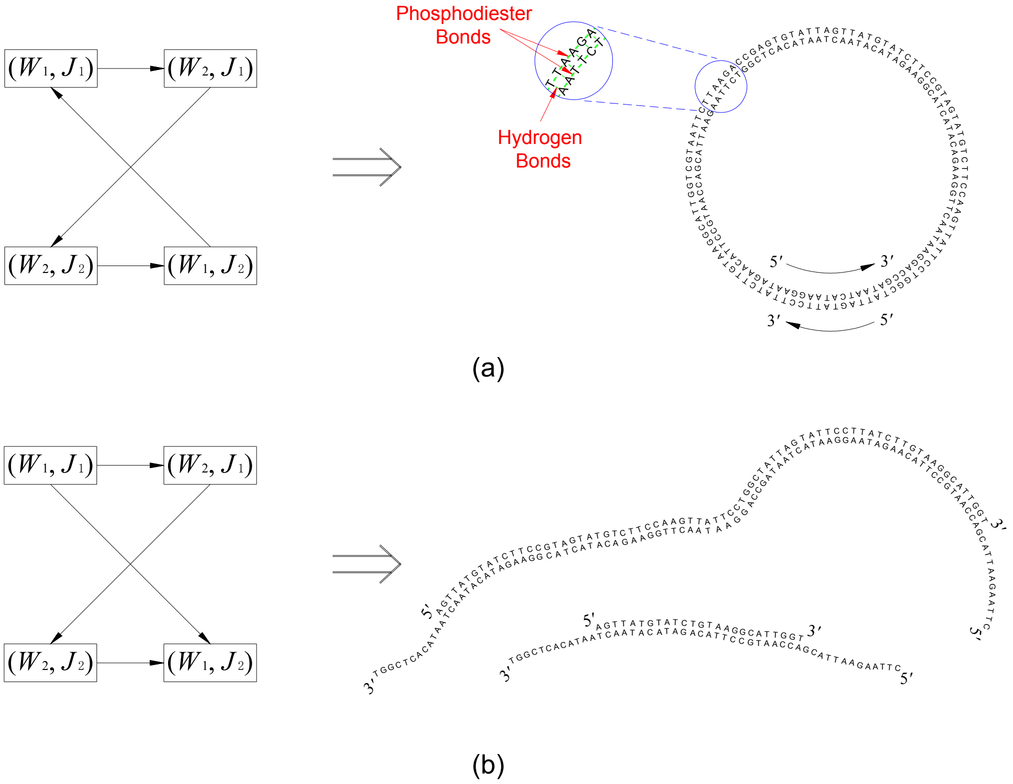


Figure 4.11. Representations of both cyclic and acyclic operational pair nodes and their DNA strands: (a) cyclic path and its one circular DNA fragment; (b) acyclic path and its two linear DNA fragments.

production schedules, whereas acyclic paths of operational pair nodes correspond to feasible production schedules. From this constant scheduling rule, cyclic paths of operational pair nodes can be represented as circular DNA fragments, whereas acyclic paths of operational pair nodes can be represented as linear DNA fragments. Figure 4.11 illustrates how both cyclic and acyclic paths of operational pair nodes are represented as both circular and linear DNA fragments (circular and linear types of DNA strands). Here, circular DNA fragments correspond to DNA plasmids, meaning that techniques for manipulating DNA plasmids enable us to distinguish linear DNA fragments from circular DNA fragments. For example, Figure 4.12 shows a complicated DNA-digraph (twenty-five operational pair nodes), in which one cyclic path of four operational pair nodes is illustrated, and this schedule becomes an unfeasible production schedule.

4.7.2 DNA plasmids and their properties

Self-replicating DNA molecules are independent molecules that correspond to DNA plasmids [57, 58]. The general self-replicating process of a DNA plasmid is shown in Figure 4.13. They are not regarded as a part of chromosomes, but they are the cellular DNA molecules, containing significant genetic information. There are two reasons why they are not considered as parts of the cellular genome: (1) DNA plasmids exist in

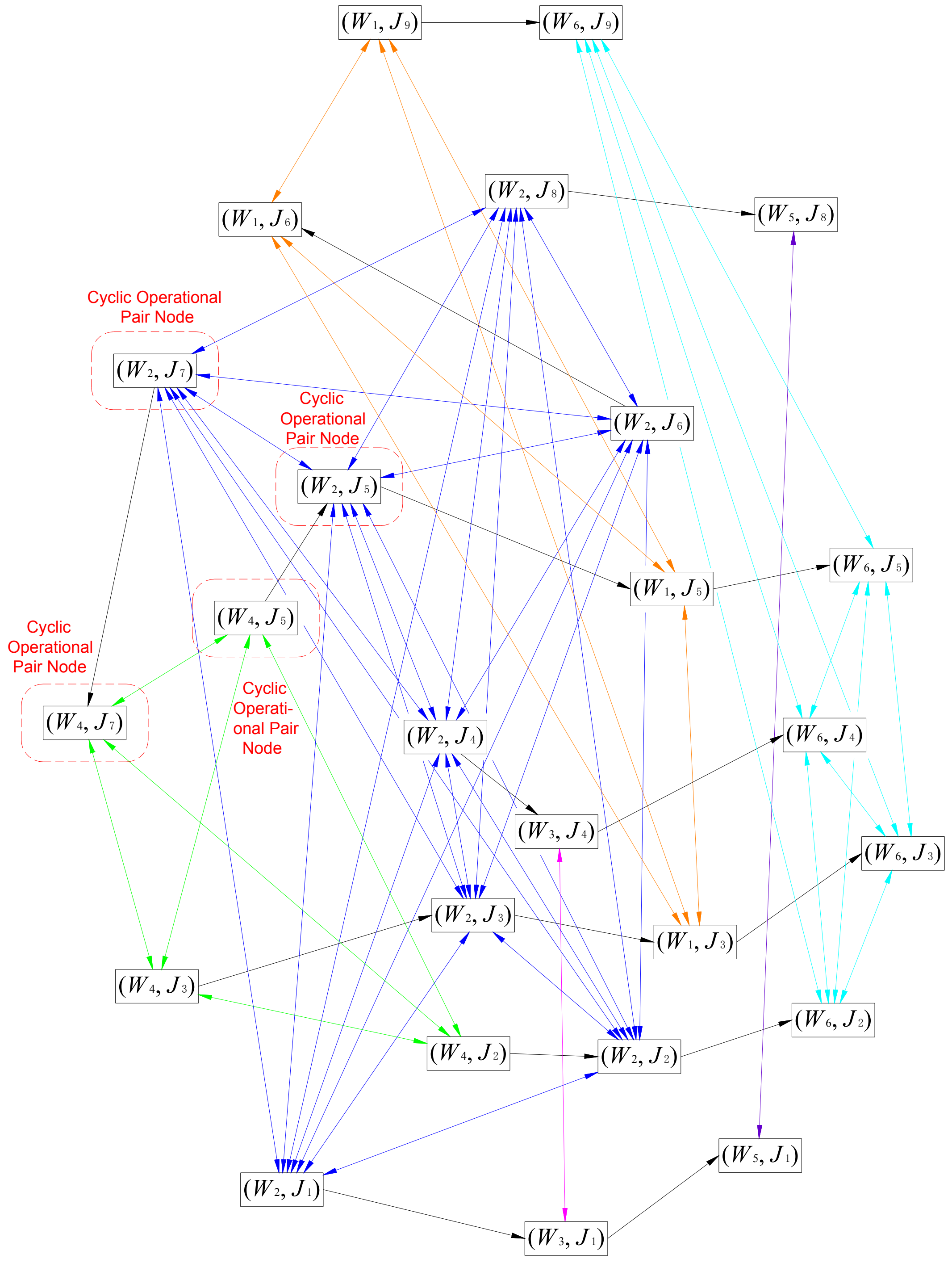


Figure 4.12. Complicated DNA-digraph composed of twenty-five operational pair nodes.

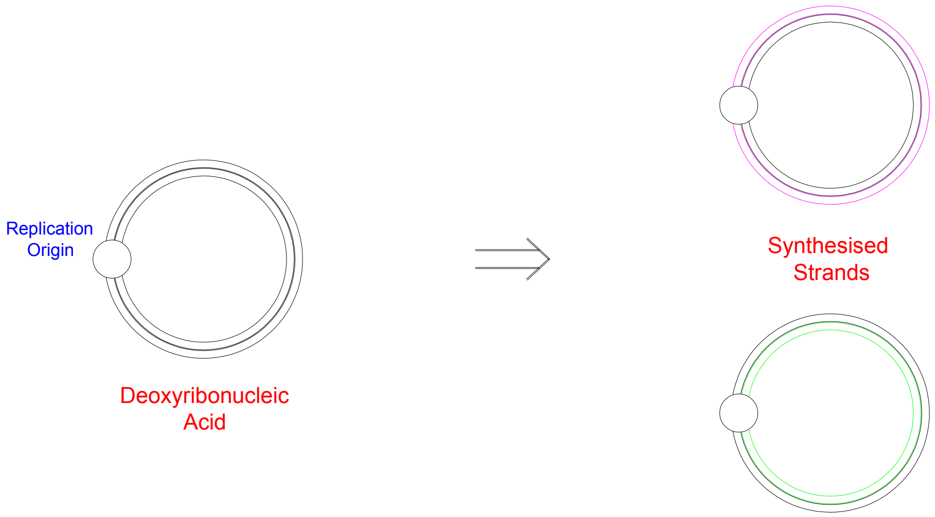


Figure 4.13. General self-replicating process of a DNA plasmid [57].

different cell species, moving from one host species to another; and (2) DNA plasmids either exist or do not exist in a specific host species cell. Although DNA plasmids have expressible genetic information, they are not considered as all-time genetic elements of cells and are not required for cellular growth.

Self-replicating nucleic acids are referred to as *replicons*. The specific replication units do not encode the enzymes needed for their own replications, having their own replication origins for the initiation of DNA synthesis. In addition, they do not produce their own nucleotides. In this way, DNA plasmids can be replication units, which rely on the energy, raw materials, and other enzyme activities of the host cells.

DNA plasmids can be regarded as living organisms growing in the host cells. The reason for this is that DNA plasmids are neither like living cells nor simply parts of cells. In other words, DNA plasmids can be regarded as viruses that have been moving from one cell to another, but have lost their mobility and are now being circulated. DNA plasmids retain the properties of some viruses in that they still need the replication enzymes of the host cells. However, different from viruses, DNA plasmids do not damage the cells, since they neither have a protein coat nor leave the cells. The reason for this is that DNA plasmids are also divided when the cell divides, and all of the daughter cells also have copies of the DNA plasmids.

In general, one characteristic feature in the properties of DNA plasmids is that a DNA plasmid is a circular DNA molecule. They can have one or multiple copies and can contain several to hundreds of genes. DNA plasmids are always replicated within the host cells, and most DNA plasmids generally live in bacteria. In fact, almost half of the cells in nature have one or more DNA plasmids, and the higher organisms also have them.

The copy number is the number of DNA plasmid copies in a bacterial cell. Most DNA plasmids have one or two copies per chromosome. The copy number affects the antimicrobial resistance gene from DNA plasmids. As the copy number per cell increases, the antimicrobial resistance gene becomes greater and the resistance becomes stronger accordingly. The size and copy number of DNA plasmids [59] are important in cloning or for other purpose experiments. The DNA plasmid size generally ranges from 1 kilo base pairs (kbp) to 250 kbp. However, various and either smaller or larger sizes of DNA plasmids might be created in their own purpose experiments from time to time.

4.7.3 Purifications of DNA plasmids

The DNA plasmid-containing cells are cultured in a liquid medium and processed as a cell extract. Proteins and RNA are removed from the cell extract, and DNA molecules are concentrated by the ethanol precipitation. However, DNA plasmid purifications [59] require a procedure for separating the pure plasmid DNA molecules from the many bacteria chromosomal DNA in the cells.

In practice, it is difficult to separate the two different types of DNA. However, in order to use DNA plasmids for cleaning, they must be separated. Here, this separation should be carried out, since it is not good to have even a small amount of polluted bacterial DNA that has been generated during the gene cloning experiment. Fortunately, several useful methods have been suggested for the removal of bacterial DNA in DNA plasmid purifications. Using these techniques, pure plasmid DNA molecules can be separated. Two of the techniques (size-based separations and conformation-based separations) are briefly described as follows:

Firstly, for the technique of size-based separations, size is one physical difference between bacterial DNA and plasmid DNA molecules. Most DNA plasmids are extremely small in size. Hence, plasmid DNA molecules can be effectively purified by separating small DNA molecules from large ones.

When the cells are carefully dissolved in cell extractions, little of the chromosomal DNA is damaged. The obtained DNA fragments are still large (in fact, they are larger than the DNA plasmids). Here, cell disruption should be performed carefully to prevent the overall destruction of the bacterial DNA. The most common method is the centrifugation method. Using this method, the cleared lysate, composed of the plasmid DNA molecules, can be obtained causing almost no damage to the bacterial DNA.

Secondly, for the technique of conformation-based separations, bacterial DNA and DNA plasmids have totally different confirmations. When applied to polymers, such as DNA molecules, conformation means the overall spatial arrangement of DNA molecules, such as either circular or linear DNA molecules. Although a DNA plasmid is a

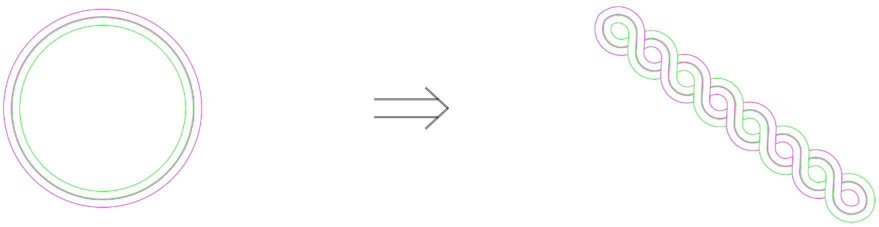


Figure 4.14. Supercoiled DNA molecule of dsDNA [59].

circular type, it can be broken into one or more linear DNA fragments. By separating the linear DNA molecules from the circular DNA molecules, solely pure plasmid DNA molecules can be obtained.

Before considering the method for separating the two different types of DNA molecules (circular and linear DNA molecules) based on the conformational difference of DNA plasmids, the overall conformation of plasmid DNA molecules should be understood. Most DNA plasmids actually exist in cells as a special feature of DNA molecules, called *supercoiled DNA molecules*, as illustrated in Figure 4.14. For example, during the DNA plasmid replication, the double right-handed helix conformation of plasmid DNA molecules is partially unwound to become supercoiled. Supercoiled conformations can be covalently regarded as *closed-circular DNA molecules* when the two polynucleotide strands are complete. If one of the two polynucleotide strands is broken, the plasmid DNA molecules turn into *open-circular DNA molecules* as the double right-handed helix conformation goes back to the general, relieved state. Supercoiling is important in DNA

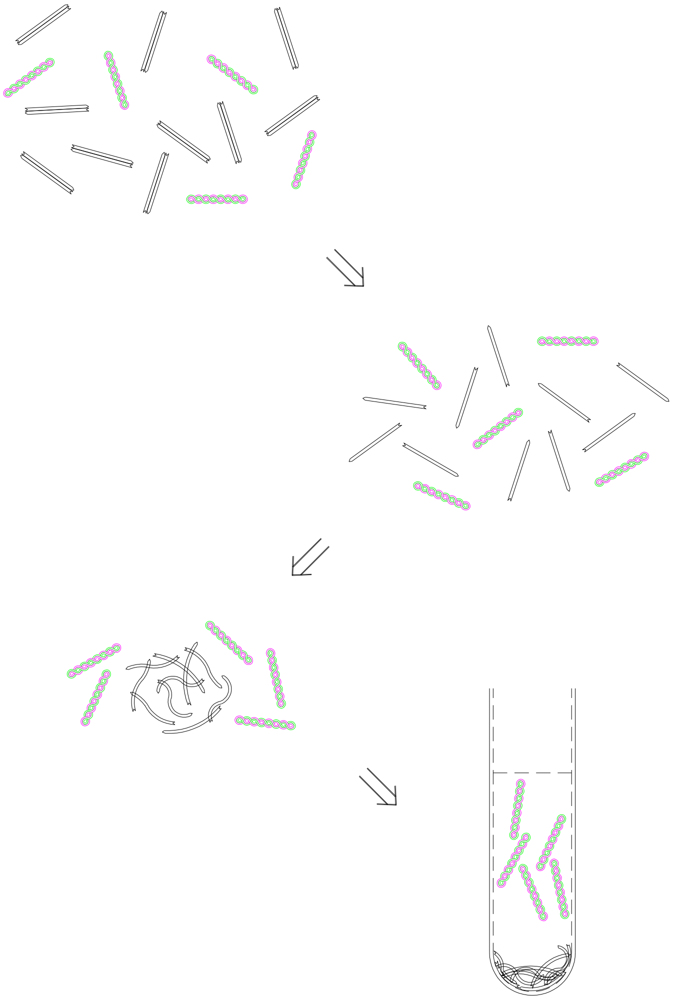


Figure 4.15. General steps of a purification method [59].

plasmid purifications, since supercoiled DNA molecules can be easily separated from *unsupercoiled DNA molecules*. Here, unsupercoiled DNA molecules correspond to linear DNA molecules.

For separation based on the conformation, one method could be alkaline denaturation. While unsupercoiled DNA molecules are denatured within a narrow pH (potential hydrogen) range, supercoiled DNA molecules are not. In the narrow pH range, between 12.0 ~ 12.5, the hydrogen bonds of the unsupercoiled DNA molecules are denatured. Following the addition of acid collect the denatured DNA molecules as they remain entangled. The final step is to extract the supercoiled DNA molecules (circular DNA molecules) in the supernatant after the insoluble materials are gathered together by centrifugation. Figure 4.15 illustrates the steps of this purification method.

4.7.4 Amplifications of DNA plasmids

The basic purpose of DNA plasmid amplifications [59] is to increase the copy number of DNA plasmids. If only the specifically desired plasmid DNA molecules are amplified, the probability of finding them is obviously increased. In addition, the DNA plasmid amplification methods can be usefully applied and practiced in various ways for designing molecular computational algorithms in decision making.

DNA plasmids, in which the copy number is higher than twenty or more, can be replicated without protein synthesis. Taking advantage of this property, bacterial culture can be carried out for plasmid DNA molecule purifications. As the cell density arrives at a satisfactory level, the protein synthesis inhibitor is added and the culture is kept at constant temperature for more than 12 hours. Meanwhile, the plasmid DNA molecules are continuously replicated, but the chromosome replication and cell division are maintained. In this way, a large quantity of plasmid DNA molecule copies can be produced.

4.8 Experimental Studies and Results

In this chapter's study, we applied a splicing operation model [60] to the job-shop DNA-based algorithm in order to minimise the maximum production completion time through the encoding process in DNA with molecular engineering techniques. The method of splicing operations links two different encoded DNA sequences together by a DNA ligation. In this section, we describe our experimental studies and show their results by solving the example job-shop scheduling problem using the job-shop DNA-based algorithm.

4.8.1 Experimental studies

To show the experimental results represented by the length of DNA strands, a program was compiled using Vector NTI software based on its own designed molecular experimentation. The job-shop DNA-based algorithm shows the minimised maximum production completion time based on simulated experimental studies.

The main processes of the molecular experimental part are to (1) detect all the circular DNA fragments; (2) remove all of the circular DNA fragments; (3) resolve all of the linear DNA fragments; and (4) select the shortest length DNA strand in the possible feasible production schedules. Here, the circular and linear DNA fragments correspond to cyclic and acyclic operational paths, respectively. The shortest length DNA strand corresponds to the schedule of the minimised maximum production completion time.

Two different types of DNA substrings correspond to a double-encoded substring and a complementary substring. Moreover, two different operational pair nodes are included in one single double-encoded substring, and each of the complementary substrings is described by each concatenation of two different double-encoded substrings. All of the DNA substrings should be generated to encode the DNA sequences of double-encoded substrings (type 1) and their complementary substrings (type 2). The simulated experimental protocol study is composed of eight steps as follows:

Step 1 (digraph)**:** The DNA-digraph is created by the direct relations among the given operational pair nodes (*m* machines and *n* jobs) based on their machine sequences and job orders.

Step 2 (encoding)**:** For the DNA-digraph, the DNA sequences of double-encoded substrings (type 1) and their complementary substrings (type 2) are encoded on the basis of the molecular encoding method. For the processes of hybridisations and ligations, in the DNA-digraph, the DNA sequences of the existing arcs of the operational pair nodes are generated and encoded in ssDNA.

Step 3 (loading)**:** All the encoded DNA sequences of the double-encoded and complementary substrings should be prepared and placed into a test tube using pipettes.

Step 4 (hybridisation and ligation)**:** All the encoded DNA sequences of the double-encoded and complementary substrings are heated and cooled for hybridisation. Here, the placed oligonucleotides of the DNA mixture are heated to approximately 94˚C and cooled to approximately 20˚C at 1˚C/min. After the hybridisation process, a DNA ligation process should be executed. This process is for catalysing, repairing, and sealing DNA sequences.

Step 5 (separation and removal-1)**:** Recall that circular DNA fragments correspond to cyclic operational paths, and linear DNA fragments correspond to acyclic operational paths. The topological differences between circular and linear DNA fragments are focused on separating circular DNA fragments from linear DNA fragments. A gel electrophoresis method can be used to distinguish circular DNA fragments from linear DNA fragments by using standard DNA markers running in parallel. These distinguished circular DNA fragments are all removed.

Step 6 (affinity separation and removal-2)**:** All the separated DNA strands (the linear DNA fragments) are prepared after the separation process. In all of these linear DNA fragments, we only focus on specific DNA sequences, starting from the first-ordered machines to the last-ordered machines. For the example DNA-digraph, as Figure 4.7 shows, (1) the two first-ordered machines correspond to both (W_2, J_1) and (W_3, J_2); and (2) the two last-ordered machines correspond to both (W_4, J_1) and (W_4, J_2). A technique of affinity separation is used to separate these specific DNA sequences of strands from other DNA stands (these others are all removed) using the complementary sites and magnetic beads.

Step 7 (simulated gel electrophoresis)**:** All the hybridised, ligated, and twice-separated DNA strands are again separated according to their sizes using simulated gel electrophoresis. The lengths of the DNA strands are also measured by simulated gel electrophoresis. The possible number of both feasible and unfeasible production schedules of DNA strands (denoted by X) is obviously limited. The number of circular types (unfeasible production schedules) of DNA strands (denoted by Y) has been removed in Step 5, thus the linear types (feasible production schedules) of DNA strands only appear in the results of the simulated gel electrophoresis.

Step 8 (selection)**:** In order of size, from top to bottom in the simulated gel electrophoresis results, the results clearly show the longest operational paths to the shortest operational paths. However, the schedules from the longest paths to the $(X-Y)th$ length of the operational path are the only possible feasible production schedules, in which we select the $(X-Y)th$ length (the shortest length in the possible feasible production sched-

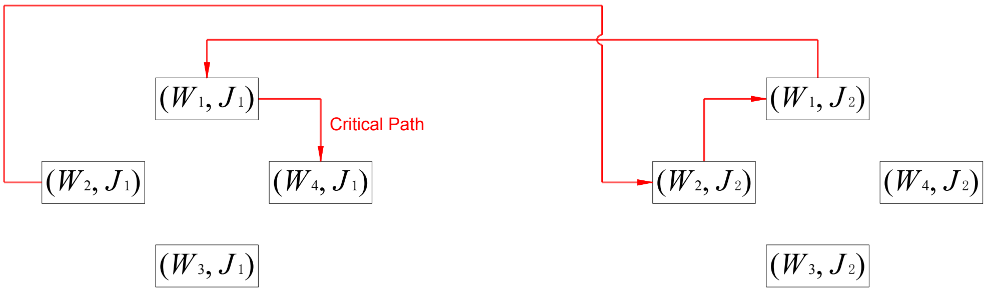


Figure 4.16. Representation of the finally resolved critical path of the example DNA-digraph.

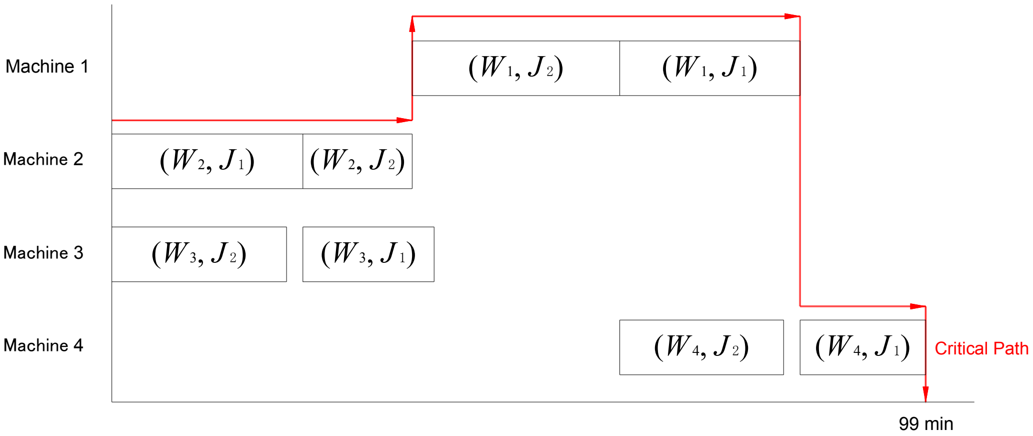


Figure 4.17. Representation of the finally resolved critical path of the Gantt chart of the example digraph.

Table 4.3. Results of the finally determined upper and lower DNA strands and their DNA sequences. After the upper and lower DNA strands were hybridised, the shortest length in the possible feasible production schedules was 99 bp.

Upper and Lower DNA Strands	DNA Sequence (5′ to 3′)	Oligo Length
$(W_2, J_1)\text{-}3'^{upper} \rightarrow 5'^{upper}\text{-}(W_2, J_2)(W_2, J_2)\text{-}3'^{upper} \rightarrow 5'^{upper}\text{-}(W_1, J_2)(W_1, J_2)\text{-}3'^{upper} \rightarrow 5'^{upper}\text{-}(W_1, J_1)(W_1, J_1)\text{-}3'^{upper} \rightarrow 5'^{upper}\text{-}(W_4, J_1)$	GCAACCGCCTTCACACTTCGTGGATAAAACTGCCGTGACC TATGAGGCTGGCCGGAGTGAAGGTAAGAGTAACCCCATAG	80
$5'_{lower}\text{-}(W_2, J_1)\mid(W_2, J_1)\text{-}3'_{lower}5'_{lower}\text{-}(W_2, J_2)\mid(W_2, J_2)\text{-}3'_{lower}5'_{lower}\text{-}(W_1, J_2)\mid(W_1, J_2)\text{-}3'_{lower}5'_{lower}\text{-}(W_1, J_1)\mid(W_1, J_1)\text{-}3'_{lower}5'_{lower}\text{-}(W_4, J_1)\mid(W_4, J_1)\text{-}3'_{lower}$	TGTGGTTCTATGGGGTTACTCTTACCTTCACTCCGGCCAG CCTCATAGGTCACGGCAGTTTTATCCACGAAGTGTGAAGG CGGTTGCCGTCCGGGTCGT	99

ules) of the DNA strand that corresponds to the schedule of the minimised maximum production completion time.

4.8.2 Results of the experimental studies

The splicing operation method with its molecular encoding processes was applied to the job-shop DNA-based algorithm, which determined all of the feasible production schedules, including the schedule of the minimised maximum production completion time.

In the molecular encoding processes, each of the double-encoded substrings was described by each of the operational pair nodes, corresponding to the DNA sequence, and each of the complementary substrings was described by each of the concatenations of the double-encoded substrings. The previous session also described the several biochemical techniques that were used to achieve our purpose of determining the minimised maximum production completion time.

In the example DNA-digraph, the number of either feasible or unfeasible production schedules was sixteen, the number of the detected unfeasible production schedules was four, and the possible feasible production schedules became twelve, thus we obviously selected the $12th$ length long of the DNA strand. The simulated gel electrophoresis results for the example DNA-digraph contain the shortest length DNA strand as well as its complementary DNA strand, represented by upper and lower DNA strands, as shown in Table 4.3. Further, for the example job-shop scheduling digraph, Figures 4.16 and 4.17 illustrate the critical path of the DNA-digraph and the critical path of the Gantt chart, respectively.

At the same time, the job-shop DNA-based algorithm can be expanded to deal with a large number of machines and their job orders by taking the advantage of encoding a variety of combinatorial numbers of DNA sequences to generate DNA substrings.

4.9 Computational Times and Solvable Sizes

In this section, the same number of machines and jobs was set, and we represent the number of either machines or jobs instead of the number of operational pair nodes. The number of either machines or jobs and the number of continuous arcs are compared, and this comparison is shown in Figure 4.18.

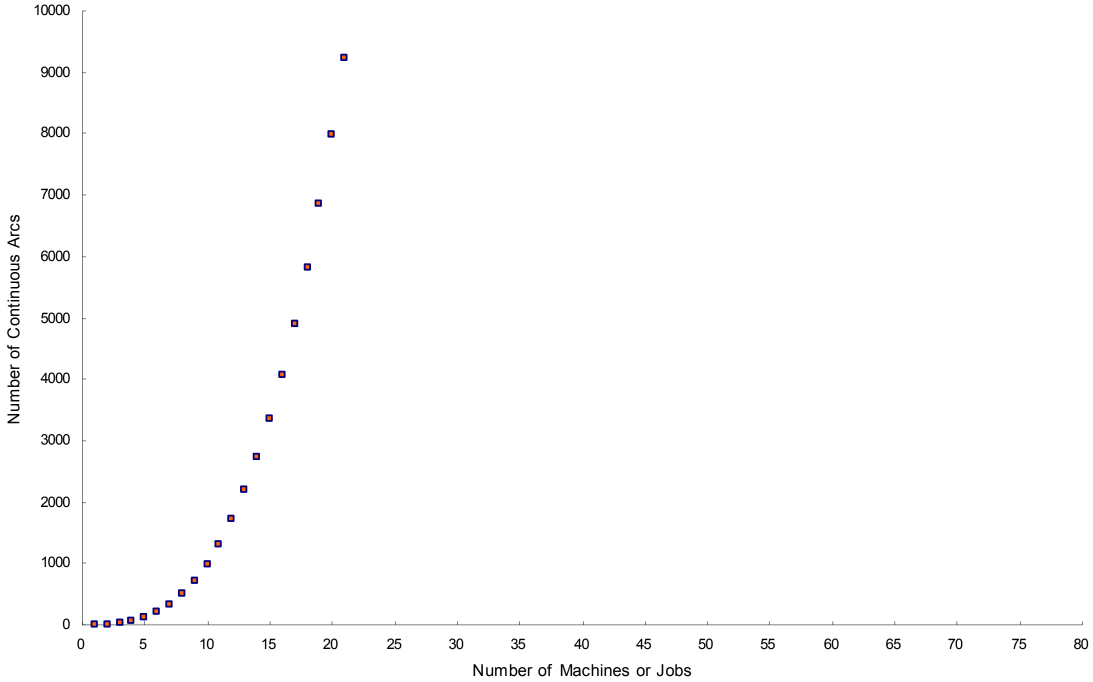


Figure 4.18. Comparison graph of the number of either machines or jobs and the number of continuous arcs.

Although n operational pair nodes are composed of both m machines W_1, W_2,…, W_m and n jobs J_1, J_2,…, J_n, we set the same number of both machines and jobs. Thus, in this section, one machine and one job imply one operational pair node, two machines and two jobs imply four operational pair nodes, three machines and three jobs imply nine operational pair nodes, and so on. In other words, the number of machines was always

Table 4.4. Comparisons of approximated running times for the exponential-time algorithm and the prepared job-shop DNA-based algorithm.

Number of Machines or Jobs	10	30	50	100
The Exponential-Time Algorithm	< 1.00 second	18.00 minutes	36.00 years	100,000,000,000,000,000.00 years
The Prepared DNA-Based Algorithm	1.06 hours	1.20 days	5.58 days	44.64 days

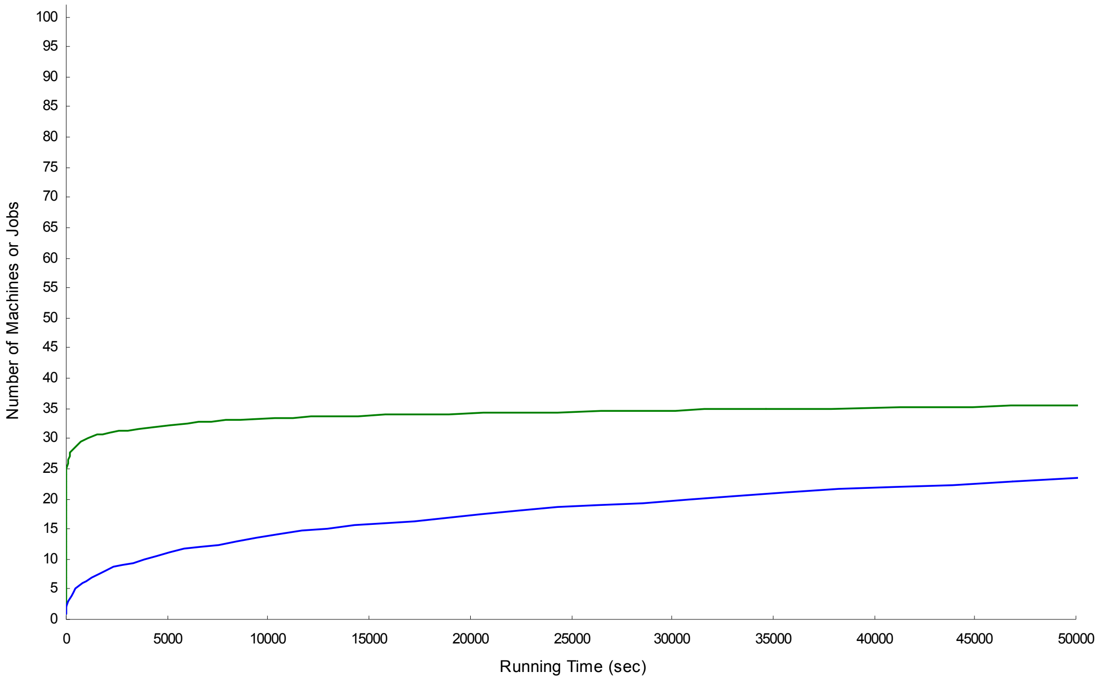


Figure 4.19. Comparison graph of approximated running times in seconds: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared job-shop DNA-based algorithm.

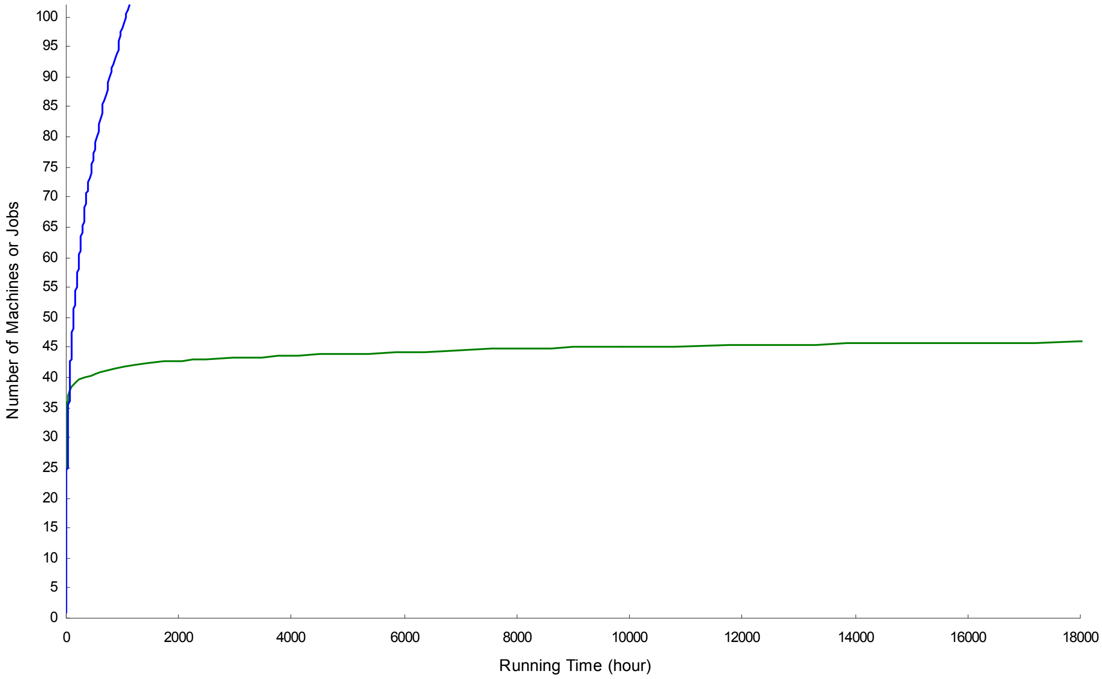


Figure 4.20. Comparison graph of approximated running times in hours: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared job-shop DNA-based algorithm.

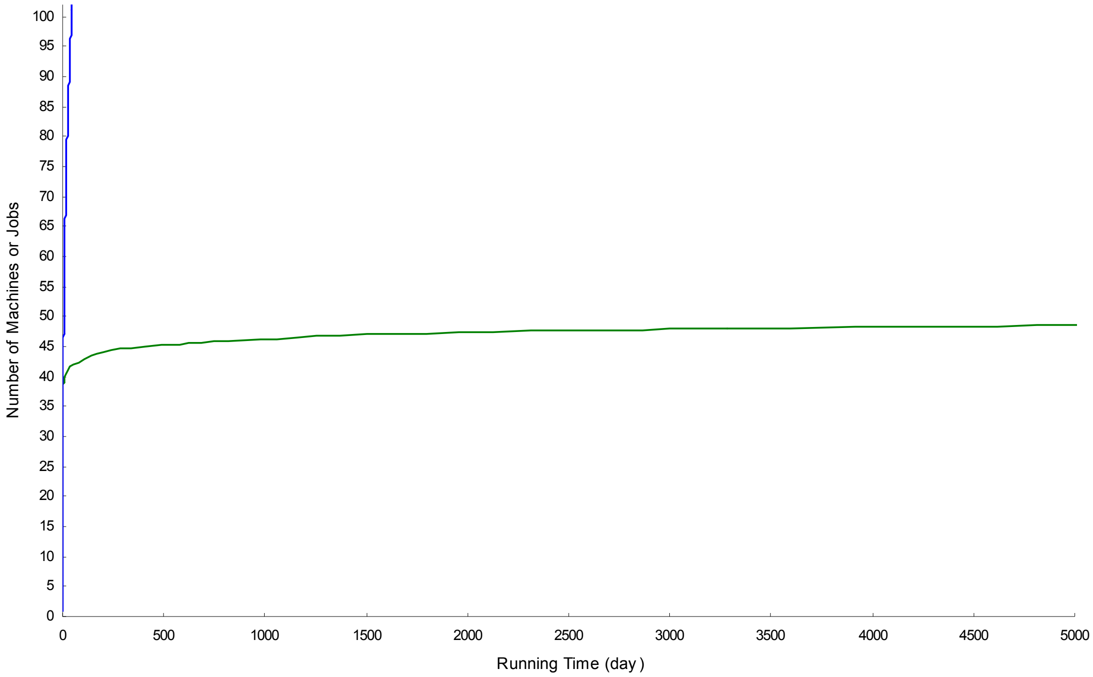


Figure 4.21. Comparison graph of approximated running times in days: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared job-shop DNA-based algorithm.

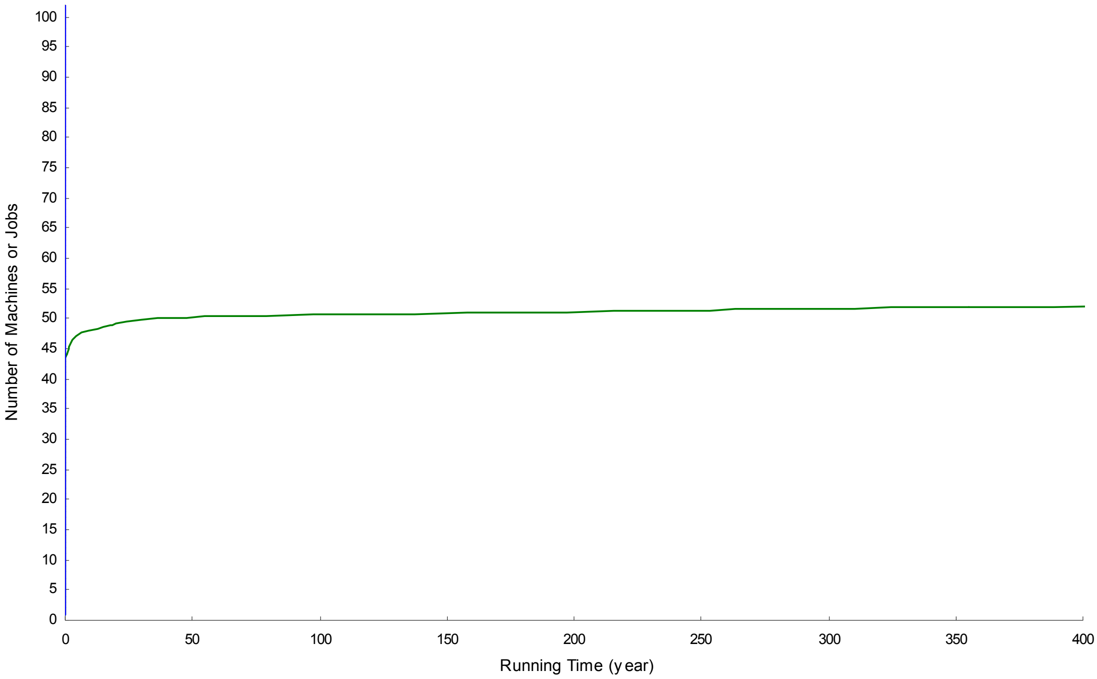


Figure 4.22. Comparison graph of approximated running times in years: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared job-shop DNA-based algorithm.

set to be equal to the number of jobs for this graph comparison, although the number of machines could be different from the number of jobs.

To measure the efficiency of using the job-shop DNA-based algorithm, comparisons of approximated running times were created as shown in Table 4.4. The table deals separately with the exponential-time algorithms and the prepared job-shop DNA-based algorithm (we are ready to detect solutions). The number of operational pair nodes was set to correspond to the inputs of size 10, 30, 50, and 100 separately. Figures 4.19 to 4.22 show four graphic comparison representations of approximated running times for the exponential-time algorithm and the prepared job-shop DNA-based algorithm.

We suppose that a processor executes a million high levels of instructions a second, and the exponential-time algorithm uses an operation of 2^n [61]. The approximated running times of the prepared job-shop DNA-based algorithm were measured and calculated based on previous experimental reports, our experimental experiences, and genetic engineering notes [62-66].

The main problem here is how to clearly and precisely determine the minimised maximum production completion time with a large number of machines and their job orders, which is an NP-hard problem. This means that no polynomial-time algorithm has yet been discovered for job-shop scheduling problems.

4.10 Concluding Remarks

The efficiency of the newly proposed minimisation method in job-shop scheduling using the job-shop DNA-based algorithm has been evaluated and measured by our molecular engineering experimental studies. The results of these studies have also been presented in this chapter. The job-shop DNA-based algorithm was used to minimise the maximum production completion time based on molecular engineering methods and techniques, as shown graphically.

A new way of handling the problem related to the machine sequences in job-shop scheduling was through the job-shop DNA-based algorithm used for the efficient discovery of feasible production schedules, including the most reliable production schedule that provided the minimised maximum production completion time.

The novel concept of developing a DNA-digraph was first proposed in this chapter, and the job-shop DNA-based algorithm came with a new molecular encoding method that encoded the DNA sequences of double-encoded substrings and their complementary substrings. These two different types of DNA substrings were manipulated in the molecular engineering experimental processes to provide the experimental results.

The characteristics of the newly proposed DNA-digraph related to the problem of job-shop scheduling involved two different features of directed paths. These features corresponded to cyclic operational paths and acyclic operational paths represented as circular and linear DNA fragments, respectively. This novel idea could be applied to other process-focused production scheduling problems, particularly regarding adaptable types of products with low volume and high variety requirements.

Chapter 5

Identifying Cohesive Subsets Based on a Fuzzy DNA-Based Algorithm

5.1 Overview

One systematically reinforced and integrated algorithm composed of both fuzzy-based methods and a general molecular algorithm for decision making is a molecular computational algorithm (termed a *fuzzy DNA-based algorithm*). This algorithm is presented in this chapter. Further, we show an applied real-world reorganisational problem in a fuzzy operational network, and describe how to solve it by using the fuzzy DNA-based algorithm.

From the viewpoint of a graph theory application, an undirected graph should be considered, in which the particular nodes and their edges with cohesiveness exist, because the nodes and edges are able to be adapted to become objects and connection chains for real-world applications. Here, a relation of those nodes and edges with cohesiveness in a subset is a clique relation in the undirected graph. If a subset of the clique relation includes the maximum number of nodes, compared to any other clique relations, then this subset is the most cohesive subset in the given clique relations. In addition, an indispensable relation for cohesiveness is a component relation, which makes a connection between two different determined cohesive subsets, or two different nodes that can update the path between two significant subsets, making the path shorter. Moreover, using fuzzy membership grades provides us with a means to casually represent the criterion of relational strengths among nodes in an undirected graph.

When dealing with a large number of nodes in an undirected graph, finding out optimal maximal and maximum clique relations is an intractable problem. To identify cohesive subsets, the fuzzy DNA-based algorithm is used for determining all of the possible clique relations, as well as the component relations in descending order, covering both the maximal and maximum clique relations and their connectors, and also for determining all of the possible fuzzy relations in a fuzzy undirected graph.

For solving the applied reorganisational problem, the chosen problem is one intractable problem that we have taken from various types of intractable operational problems, which is required in order to be able to reorganise the given workforces to create better or optimally operational connections in work-operational situations. This reorganising process can be optimally executed by identifying all of the possible cohesive subgroups (for this case involving the reorganisational problem, it should be subgroups, not subsets) in the fuzzy operational network.

The fuzzy DNA-based algorithm is used firstly to detect all of the possible workforces in clique and component relations, secondly to find workforces in fuzzy relations, and finally to reorganise the determined workforces into fuzzy similarity subgroups. Moreover, the efficiency of performing the fuzzy DNA-based algorithm is proved by measuring the densities of the identified cohesive subgroups in several ways.

5.2 Background and Motivations

Although advanced information sciences and technologies are exploited to install a large amount of useful information data, both creating a new optimal set, and determining several optimal subsets of those bunches of complicated data are still intractable problems. To solve these sorts of problems, various applied models and methods have been proposed in different areas [67, 68]. In particular, numerous concepts are often formulated, which become interesting issues on the integrated application side [69, 70]. One problem in determining optimal subsets is associated with detecting the maximal and maximum clique relations in an undirected graph. Finding these two clique relations are intractable problems among several key problems in graph theory [71].

For the applied reorganisational problem, the problem is basically associated with routine tasks and productive work. Routine tasks involved in regular work affairs, repeat manufacturing, production, and operational systems hold little appeal in modern industrial and organisational societies. These routine tasks rarely provide workforces with opportunities and often cause feelings of lack of achievement, lack of psychological growth, and lack of satisfaction [72]. The productive work among workforces is obviously enhanced by reducing routine tasks. To reduce routine tasks and enhance productive work, the given workforces are required to be reorganised by identifying cohesive subgroups in work-operational situations.

Watada *et al.* [73, 74] have studied the problems of operational tasks in human resources and organisational relations by utilising fuzzy-based methods. To recover and improve work connections in a productive way, operational managers often execute *work rotation*, which is widely practiced, and has been proven to be effective in the manufacturing control fields [75].

Work rotation breaks up the monotony of highly specialised routine tasks by calling on different abilities, experiences, and skills, which can also be expressible as fuzzy membership grades [76], adapted for use in the fuzzy DNA-based algorithm. In executing work rotation, the two important points to take into account are (1) to analyse how all of the given workforces are being built up with cohesiveness in fuzzy operational connection levels; and (2) to determine how accurately cohesive subgroups are being reorganised by an operational manager. However, the main problem in this work rotation process is how to detect the most cohesive subgroups (in which the relations should be the maximal and maximum clique relations) while dealing with a large number of workforces.

Reorganising a huge number of workforces induces an NP-hard problem. In the real world, workforces and their operational connections exponentially rise when dealing with a large number of multiple organisations. Figure 5.1 shows an example of a social information network, representing complicated relationships among actors. For application of the fuzzy operational network, the actors can be converted into the workforces.

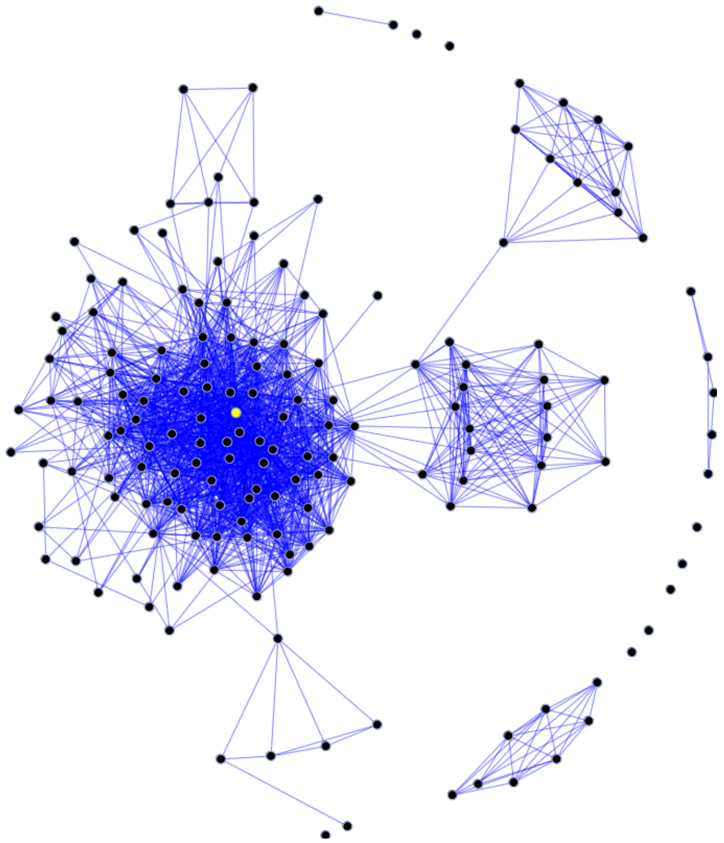


Figure 5.1. Example of a social information network. Creative Commons Attribution-Share Alike 3.0: GUESS, 2009.

5.3 Cohesive Subsets in the Real World

If a graph has one or more connections among nodes, but no direction for each of all the edges, then this graph is called an *undirected graph*. If the undirected graph has some implicit reasons, then this undirected graph is called an *undirected network*. If a particular subset of specific nodes is closely related to each other via strong relations among the nodes of embedded sets, then this subset is called *cohesive*.

In the real world, there are a large number of different and various types of undirected networks, in which cohesive subsets should be distinguished properly from each other in order to make better decisions or derive optimisation solutions for any related processes. Cohesive subsets can be adapted to such areas as subgroups, subparts, subareas, etc., in many real applications or problems.

Cohesive relations in manufacturing control networks are often referred to as the *close interpersonal relationships* in manufacturing control situations or affairs. These are relations that enable nodes, corresponding to operators, as representing manufacturing control networks that exchange or share their related manufacturing information, create solidarity, or act collectively [77, 78]. Numerous direct contacts among all cohesive subgroups, which are combined with few or null ties to outsiders, dispose a group

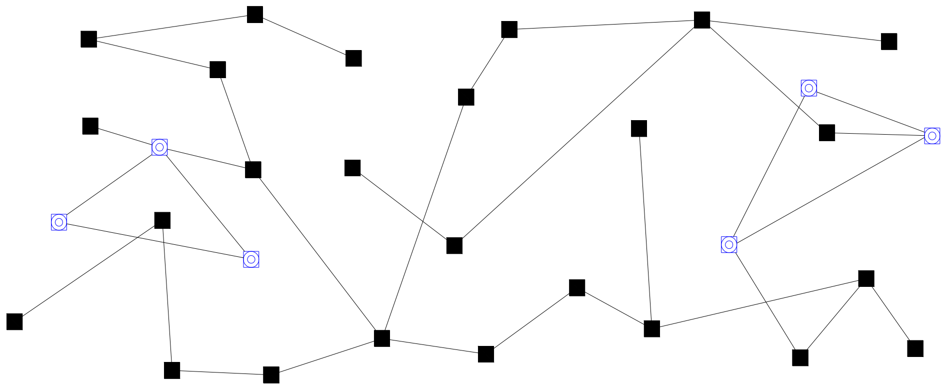


Figure 5.2. Example of the two possible cohesive subgroups (blue colour) in an undirected network.

toward a close interpersonal relationship in manufacturing control networks, but also dispose the relationship to being one of homogeneity of thought, behaviour, and identity. Examples of formal cohesive subgroups often include the personnel department, production department, quality control department, finance department, etc., for the enterprise organisational example, whereas informal cohesive subgroups often include the subgroups among operators who rarely exchange or share their related manufacturing information for any production or organisational example cases [79].

In the case of social information and organisational networks, if a subset of nodes, corresponding to actors in representing social information and organisational networks, is cohesive, then the subgroups of actors are dense, direct, frequent, intense, positive, or have strong ties in comparison to their non-cohesive counterparts [80, 81]. Figure 5.2 shows an example of the two possible cohesive subgroups that are selected in an undirected network.

5.4 Measures of Cohesive Subsets

When measuring structural variables on a distinct set of existing nodes in an undirected graph, this set of nodes is called a *one-mode network* in the case of social information and organisational networks, since all nodes come from one set. While measuring different types of structural variables on two or more subsets of existing nodes, two or more subsets of nodes are referred to as *two-* or *higher-mode networks* [79]. For an example of organisational networks, actors come from different subgroups, one subset consists of profit organisational actors, and the other one consists of non-profit organisational actors. Here, the flows of relationships or supports between profit and non-profit organisational actors can be measured.

Cohesive subsets are related to pairwise relations in one-mode networks. In two- or higher-mode networks, cohesive subsets focus on relations existing among nodes through their joint relations. Thus, cohesive subsets can be represented as any types of subsets, and can be measured by different measurement methods, since the undirected graph exhibit the nodes being connected to each other with relational edges.

5.4.1 Measure of the strength of relations

One way of measuring cohesive subgroups was proposed by Bock *et al.* [82], and this method measures the ratio of the strength of relations for repeatedly constructing subgroups. This measurement basically identifies subsets of the items that are highly correlated by analysing sets of tested items. Assuming that the entire nodes are n nodes in an undirected graph, n_s nodes are in a subset in this undirected graph, and this subset is denoted by P_s, then the ratio of the strength of relations can be measured by the degree to which strong relations are included in (rather than being outside) the subset, and this is defined as

$$\frac{\sum_{i\in P_s}\sum_{j\in P_s} x_{i,j}}{n_s(n_s-1)} \Bigg/ \frac{\sum_{i\in P_s}\sum_{j\notin P_s} x_{i,j}}{n_s(n-n_s)}, \tag{5.1}$$

where $x_{i,j}$, i and j = 1, 2,…, n corresponds to all possible connections among the given nodes in the undirected graph. From this ratio, the denominator is the average strength of the relations coming from subset nodes to outsiders, and the numerator is the average strength of relations that are included in the subsets.

5.4.2 Measure of the density

The density of subsets can also be calculated from a dichotomous relation of the numerator of the ratio of the strength of relations (5.1). For the density calculations in

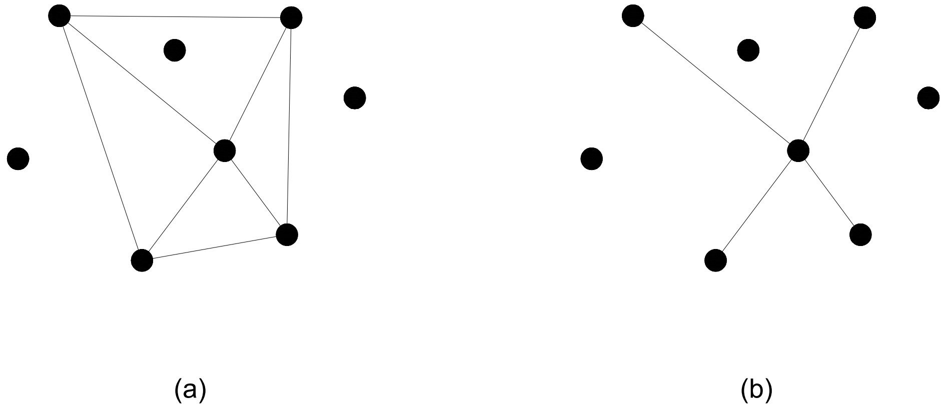


Figure 5.3. Comparison of two undirected graphs. Both (a) and (b) have the same inclusiveness.

evaluating cohesive subsets, there are three common methods used: inclusiveness, density, and the density in a valued undirected graph [79].

First, inclusiveness defines the number of nodes that are completely included in the various connected parts of the undirected graph. The most useful measurement of inclusiveness for comparing various types of undirected graphs is the number of connected nodes expressed as a proportion of the total number of nodes. Thus, the proportion of the inclusiveness is denoted by Π, and can be calculated by

$$\Pi = \frac{CN}{n} \text{ for } 0 \le \Pi \le 1, \tag{5.2}$$

where *CN* is the number of existing connected nodes in the undirected graph. Figure 5.3 shows an example of two undirected graphs that have the same inclusiveness, but the number of edges in each graph is different.

Second, the density defines the number of edges without considering valued relations in an undirected graph. The density is expressed as a proportion of the maximum possible number of edges. The density, without considering valued relations, is denoted by Δ, and the density can be derived from

$$\Delta = \frac{CE}{n(n-1)/2} \text{ for } 0 \le \Delta \le 1, \tag{5.3}$$

where there are $n(n - 1) / 2$ possible unordered pairs of nodes, and thus $n(n - 1) / 2$ possible edges that could be presented in the undirected graph, and *CE* is the number of existing edges in the undirected graph.

Finally, the density in a valued undirected graph defines the number of edges; in this

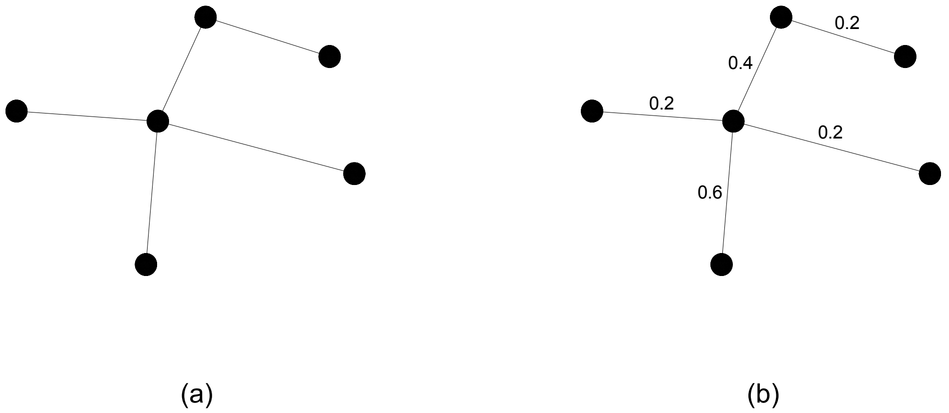


Figure 5.4. Comparison of two undirected graphs. (a) does not include any of the valued relations but (b) includes fuzzy valued relations.

case, valued relations should be considered in the undirected graph. Here, the values can be represented as any possible numerical numbers, such as fuzzy membership grades. One can average the values that are attached to the edges across all edges to generalise the notion of density to a valued undirected graph. The density in a valued undirected graph can also be expressed as a proportion of the maximum possible number of edges. The density considering valued relations is denoted by *δ*, and the density in a valued undirected graph is calculated by

$$\delta = \frac{\sum_{i=1}^{n} w_i}{(\max_i w_i)CE} \text{ for } 0 \le \delta \le 1, \tag{5.4}$$

where w_i is the value of that edge. This measurement derives the average strength of edges in the valued undirected graph. In Figure 5.4, an example of two different undirected graphs is shown, where one is not considered with any of the valued relations, and the other one is considered with fuzzy valued relations.

5.4.3 Measure of the probability

The use of the hypergeometric probability function was proposed by Alba [83]. This measurement calculates the probability of observing CE_s edges in a subset of n_s nodes, which are taken from the total nodes and edges in an undirected graph with *n* nodes and *CE* edges. The observed number of edges in the subset is equal to *OE*, and the hypergeometric probability is defined as

$$\Pr(CE_s = OE) = \frac{\binom{CE}{OE}\binom{\frac{n(n-1)}{2} - CE}{\frac{n_s(n_s-1)}{2} - OE}}{\binom{\frac{n(n-1)}{2}}{\frac{n_s(n_s-1)}{2}}}. \tag{5.5}$$

This hypergeometric probability is able to recognise and draw a random sample without any replacement of $n_s(n_s - 1) / 2$ dyads that are included in the subset, and the observed CE_s edges come from the undirected graph of $n(n - 1) / 2$ dyads and $CE = x_{+,+} / 2$ relations.

For considering *OE* or more edges in the probability, we sum the probabilities from the above hypergeometric probability equation (5.5) for values of *OE* from CE_s to its maximum possible number of edges, which either presents the possible number of existing edges in the subset, corresponding to $n_s(n_s - 1) / 2$, or presents the observed number of edges in the subset, corresponding to $CE = x_{+,+} / 2$. The formula of this type of probability can be calculated by

$$\Pr(CE_s \geq OE) = \sum_{h=OE}^{\min(CE,\frac{n_s(n_s-1)}{2})} \frac{\binom{CE}{h}\binom{\frac{n(n-1)}{2}-CE}{\frac{n_s(n_s-1)}{2}-h}}{\binom{\frac{n(n-1)}{2}}{\frac{n_s(n_s-1)}{2}}}. \tag{5.6}$$

This formula has the probability of the observed *OE* or more edges in a subset of size n_s, which comes from the undirected graph with *CE* edges. From the above probability equation (5.6), we can recognise whether the observed frequency of edges within the subset is greater than those expected or not. Here, this probability can also be interpreted as a specific value for the null hypothesis, in which there is no difference between the density of the subset and the density of the whole set.

5.5 Clique Relations for Cohesive Subsets

The most important concept, and a powerful way of identifying cohesive subsets in an undirected graph, is to detect all possible nodes in clique relations. The most cohesive subset is called a *clique* or *clique relation*, which is a useful and powerful point for the subset that has these formal properties. For example, all of the possible clique relations should be considered in finding out each of all the strong ties among actors in a social information or organisational network.

In an undirected graph, a clique relation is a maximal complete undirected subgraph composed of three or more nodes. All of the nodes in a clique relation are completely adjacent to each other, since any of the other nodes are not adjacent to all of the nodes in this clique relation [84]. Hence, in an undirected graph, if there is a subset of two connected nodes, each of them is not considered to be included in a clique relation.

The clique relation also has specified mathematical properties [85]. In graph theory, an undirected graph where every pair of nodes is clearly connected by an edge, is called a *complete graph*.

Definition 5.1. *An undirected graph $G = (N_s, E_s)$ has a clique relation C, which is a set of nodes $C \subseteq N_s$. The undirected subgraph of G is induced by C and is complete, i.e. $\{N_i, N_j\} \in E_s$ for all distinct nodes $N_i, N_j \in C$.*

The two terms using the maximal clique relation and the maximum clique relation are totally different. If a clique relation of an undirected graph is maximal, called a *maximal clique relation*, then this clique relation should not be included in any of the other cliques in the undirected graph. If a clique relation of an undirected graph is the maximum relation, called a *maximum clique relation*, then this clique relation is a maximal clique, which is composed of the largest number of nodes.

Both the maximal clique relation and the maximum clique relation have strong and complete relationally cohesive subsets, because their inclusiveness and density should be 1 in the measurements of the densities. Moreover, the maximum number of nodes in a clique, corresponding to the maximum clique relation, is the strongest and most cohesive subset in the given undirected graph.

Definition 5.2. *An undirected graph $G = (N_s, E_s)$ has a clique relation C, which is maximal if no clique relation NC exists in G, such that $C \subseteq NC$ and $C \neq NC$.*

Definition 5.3. *An undirected graph $G = (N_s, E_s)$ has a clique relation C, which is a maximum if no clique relation exists in G with more nodes than C. The cardinality of C is the number of nodes that are contained in C.*

When considering both the maximal clique relation and the maximum clique relation, recall that the maximum clique relation of the undirected graph is the undirected subgraph that can also be the maximal clique relation of the undirected graph.

Lemma 5.1. *Each of all possible maximum clique relations $C \subseteq N_s$ of an undirected graph $G = (N_s, E_s)$ is a maximal clique relation.*

Proof. Assume that $C \subseteq N_s$ is a maximum clique relation of an undirected graph $G = (N_s, E_s)$, but it is not a maximal clique relation. When a clique relation is $NC \subseteq N_s$, such that $C \subseteq NC$ and $C \neq NC$, then the clique relation NC has more nodes than the clique relation C, meaning the hypothesis is contradictory that C is both a maximum and maximal clique relation. ■

One hereditary graph property is a clique relation of an undirected graph. The property is shared by all of the undirected subgraphs included in the given undirected graph, and these undirected subgraphs are induced properly by some cohesive subset of the given clique relation.

Lemma 5.2. *Assume that two clique relations are in the same undirected graph, which are given by* (1) *$C \subseteq N_s$ as a clique relation of an undirected graph $G = (N_s, E_s)$; and* (2) *$B \subseteq C$ also as a clique relation of G.*

Proof. Let both $C \subseteq N_s$ be a clique relation of an undirected graph $G = (N_s, E_s)$ and $B \subseteq C$ also be a clique relation of G, then both $\{N_i, N_j\} \in E_s$ for all distinct nodes $N_i, N_j \in C$ and $\{N_i, N_j\} \in E_s$ for all distinct nodes $N_i, N_j \in B$. Thus, $B \subseteq N_s$ is also clearly a clique relation. ■

Figure 5.5 shows an example of two different clique relations of each undirected graph, where one does not include any of the valued relations, and the other one includes fuzzy valued relations.

In order to extract all of the larger possible clique relations in an undirected graph, we briefly explain the proposed extraction process [85]. To extend a clique relation to a large clique relation, assuming that $R \subseteq N_s \setminus C$ is the specific set of candidate nodes, where $R = \{N_j \in N_s \setminus C \mid \{N_i, N_j\} \in E_s \text{ for all } N_i \in C\}$. Each of the candidate nodes in R is

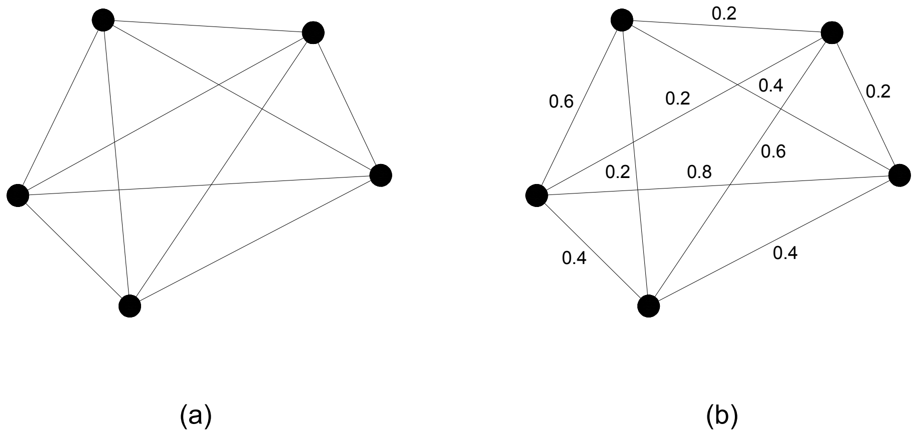


Figure 5.5. Comparison of clique relational undirected graphs. (a) does not include any of the valued relations but (b) includes fuzzy valued relations.

turned and removed to be added to *C*. The set *R* can be updated by removing the nodes that are not completely adjacent to the selected candidate nodes. Thus, the selected candidate nodes in *C* can be removed after the sets *R* and *C* are updated.

Detecting all of the possible maximal clique relations in an undirected graph, and even extracting all of the possible clique relations of cohesive subsets from a small number to a large number of clique relations in the real possible undirected networks, are intractable problems, meaning NP-hard problems [86]. For this chapter's study, when dealing with a large number of nodes and edges (corresponding to workforces and their operational connections in the case of a fuzzy operational network), there is always the restriction of selecting the number of workforces and their operational connections; in detecting their clique relations of cohesive operational connections in work-operational situations.

5.6 Component Relations for Connections

A *component relation* is also an important property for connecting cohesive subsets, because component relations can make some of the determined clique relations of cohesive subsets have connections or can link some important or meaningful nodes to each other. For example, after determining all of the possible clique relations of cohesive subgroups among workforces in a fuzzy operational network, component relations among workforces are needed to be known in order to be able to designate them to possibly being a communicator within two or more cohesive subgroups for better cohesive relations.

Another important reason, when dealing with component relations, is that we are able to clearly recognise an undirected graph that has complicated connections [80]. If a path between every pair of nodes exists in an undirected graph, meaning all of the nodes are reachable, then we say the undirected graph is *connected*. If there are two or more

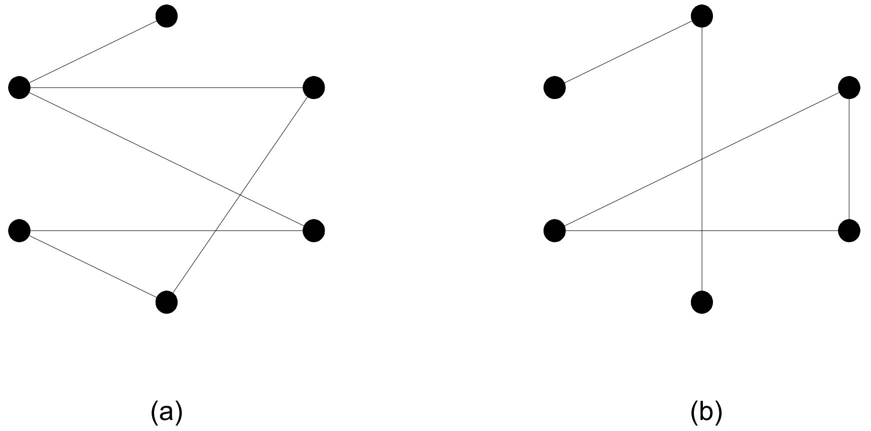


Figure 5.6. Comparison of two undirected graphs. (a) is a connected undirected graph and (b) is a disconnected undirected graph.

node subsets or one or more independent nodes in an undirected graph, meaning the undirected graph either consists of two or more undirected subgraphs or includes one or more isolated nodes, then we say the undirected graph is *disconnected*. In Figure 5.6, an example of two different undirected graphs is shown; one is a connected undirected graph and the other one is a disconnected undirected graph. Figure 5.7 shows an example of two different undirected graphs, including fuzzy valued relations; one is a connected undirected graph and the other one is a disconnected undirected graph.

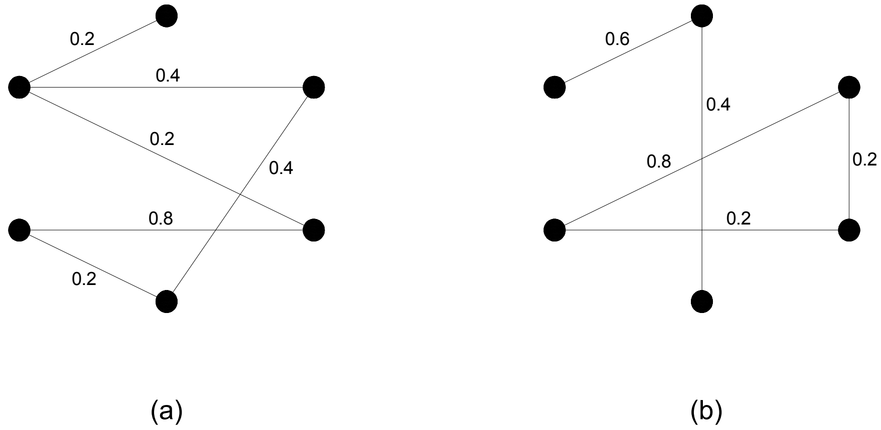


Figure 5.7. Comparison of two undirected graphs. (a) is a connected undirected graph with fuzzy valued relations and (b) is a disconnected undirected graph with fuzzy valued relations.

5.7 Cohesive Subsets with Fuzziness

A pair relation of two nodes in an undirected graph can be expressible as a numerical weight, meaning each of all the edges can have its own weight. The assigned weights are able to represent the criterion of relational strengths. One method to represent and analyse the relational strengths is a fuzzy set method [87, 88]. The fuzzy set permits membership grades in the interval from 0 to 1. In the fuzzy set, fuzzy membership grades can be exploited to casually represent the criterion of relational strengths in an undirected graph. Thus, many possible methods based on the fuzzy set (fuzzy-based methods) [89, 90] can be applied to the identification problem of determining cohesive subsets.

5.7.1 Fuzzy clique relations

An undirected graph that includes fuzzy membership grades is called a *fuzzy undirected graph*. Again, if the fuzzy undirected graph has some implicit reasons, then this fuzzy undirected graph is called a *fuzzy undirected network*. In the fuzzy undirected graph, the mathematical properties of cyclic and tree relations, and of fuzzy cyclic and tree relations should be considered and defined before dealing with a fuzzy clique relation. A mathematical property of the fuzzy clique relation is different from the maximal and maximum clique relations, and is an important relation for identifying cohesive subsets when dealing with fuzzy membership relations. All of the mentioned relations are defined respectively in mathematical ways [91, 92] in the following manner.

The fuzzy undirected graph with a set N_s of nodes and a set E_s of edges is defined to be relation $E_s \subseteq N_s \times N_s$ on a set N_s. A fuzzy relation is represented as μ: $N_s \times N_s \rightarrow [0, 1]$ and an edge is represented as $(N_i, N_j) \in N_s \times N_s$, which has weight, representing $\mu(N_i, N_j) \in [0, 1]$. Fuzzy undirected graphs are considered for simplicity, i.e. the fuzzy relation is symmetric, and each of all the edges should be regarded as each of the unordered pairs of nodes.

Definition 5.4. *A fuzzy undirected graph is represented as* $G = (\sigma, \mu)$, *which is a pair of functions* σ: $N_s \rightarrow [0, 1]$ *and* μ: $N_s \times N_s \rightarrow [0, 1]$, *where for all* N_i, N_j *in* N_s, *then* $\mu(N_i, N_j) \leq \sigma(N_i) \wedge \sigma(N_j)$ *is obtained.*

Definition 5.5. *An undirected subgraph of the fuzzy undirected graph* $G_s = (\tau, v)$ *is called a fuzzy undirected subgraph of G if* $\tau(N_i) \leq \sigma(N_i)$ *for all* $N_i \in N_s$ *and* $v(N_i, N_j) \leq \mu(N_i, N_j)$ *for all* $N_i, N_j \in N_s$.

Definition 5.6. *In a fuzzy undirected graph, a sequence of distinct nodes* $N_0, N_1, N_2, \ldots, N_n$ *is called a path, denoted by* ρ, *such that* $\mu(N_{i\text{-}1}, N_i) > 0$, $1 \leq i \leq n$. *Here,* $n \geq 0$ *is called the length of* ρ, *and the consecutive pairs of nodes* $(N_{i\text{-}1}, N_i)$ *are called the arcs of* ρ. *The weight with the weakest arc of* ρ *is the strength of* ρ.

Definition 5.7. ρ *is also called a cyclic relation if* $N_0 = N_n$, *and* $n \geq 3$. *A fuzzy undirected graph that has no cyclic relations is called an acyclic relation, and a connected acyclic graph is called a tree relation.*

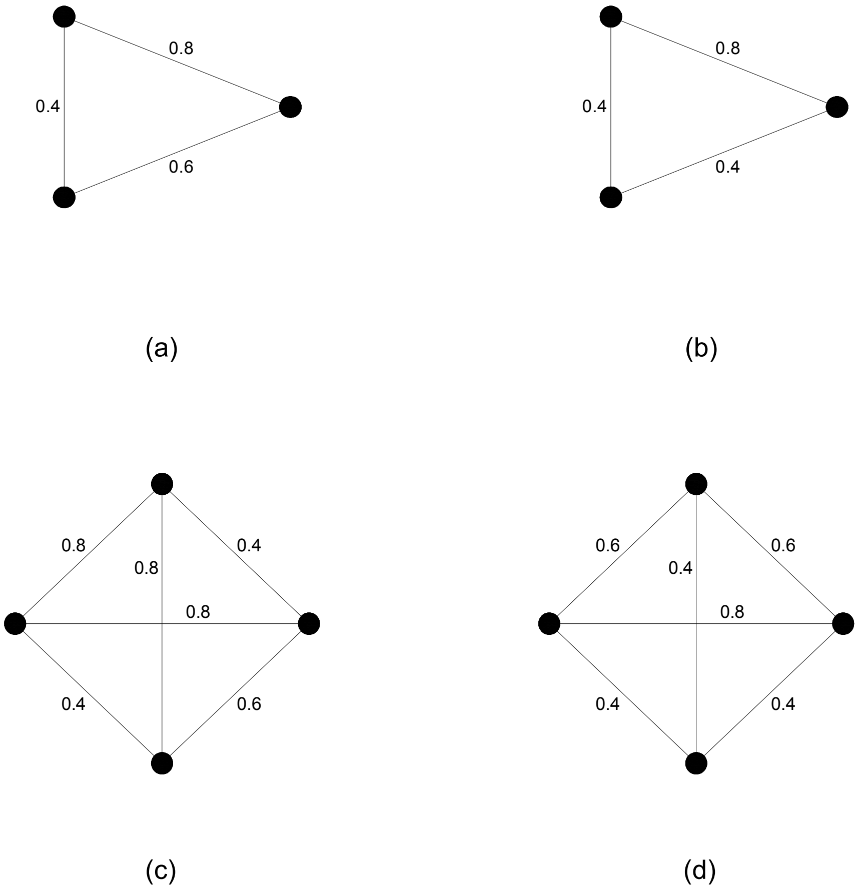


Figure 5.8. Examples of four fuzzy undirected graphs: (a) a cyclic relation; (b) a fuzzy cyclic relation; (c) a clique relation; (d) a fuzzy clique relation.

Definition 5.8. (1) *If and only if* ($supp(\sigma)$, $supp(\mu)$) *becomes a tree relation*, *then* (σ, μ) *is a tree relation*; *and* (2) (σ, μ) *is also called a fuzzy tree relation if and only if* (σ, μ) *includes a fuzzy spanning subgraph* (σ, v) *that is a tree relation*, *such that* $\forall(u, t) \in supp(\mu) \setminus supp(v)$, $\mu(u, t) < v^{\infty}(u, t)$, *meaning there is a path in* (σ, v) *between u and t*, *and this strength is greater than* $\mu(u, t)$. *Here*, *the support of* σ *represents* $supp(\sigma)$.

Definition 5.9. (1) *If and only if* ($supp(\sigma)$, $supp(\mu)$) *becomes a cyclic relation*, *then* (σ, μ) *is a cyclic relation*; *and* (2) (σ, μ) *is also called a fuzzy cyclic relation if and only if* ($supp(\sigma)$, $supp(\mu)$) *is a cyclic relation and* $\nexists$ *unique* $(N_i, N_j) \in supp(\mu)$, *such that* $\mu(N_i, N_j) = \wedge\{\mu(u, t) \mid (u, t) \in supp(\mu)\}$.

Definition 5.10. *If a fuzzy undirected subgraph of G is* $G_s = (\tau, v)$, *which is induced by*

$SN_s \subseteq N_s$, where SN_s is a subset of N_s, then G_s becomes a clique relation if $(supp(\tau)$, $supp(v))$ *is a clique relation, and G_s is called a fuzzy clique relation if G_s is a clique relation, and every cyclic relation is a fuzzy cyclic relation in* (τ, v).

For a numerical example of the above definitions, in Figure 5.8, an example of four fuzzy undirected graphs depicts the differences (1) between a cyclic relation and a fuzzy cyclic relation; and (2) between a clique relation and a fuzzy clique relation.

5.7.2 Fuzzy similarity relations

To identify cohesive subsets in a fuzzy undirected graph, fuzzy similarity relations among nodes should be considered, and these are represented in mathematical ways [93, 94]. The concept of this fuzzy similarity relation is to consider all of the possible complete subsets of nodes, which can become cohesive subsets in the fuzzy undirected graph.

Definition 5.11. *Let ε be a fuzzy relation among nodes on a set N_s, and the following notions are defined as* (1) *ε is κ-reflexive if $\forall N_i \in N_s$, $\varepsilon(N_i, N_i) \geq \kappa$, where $\kappa \in$* $[0, 1]$; (2) *ε is irreflexive if $\forall N_i \in N_s$, $\varepsilon(N_i, N_i) = 0$; and* (3) *$\varepsilon$ is weakly reflexive if all N_i, N_j are in N_s and for all $\kappa \in$* $[0, 1]$, *$\varepsilon(N_i, N_j) = \kappa \Rightarrow \varepsilon(N_i, N_i) \geq \kappa$.*

Lemma 5.3. *The fuzzy relations among nodes $\varepsilon \circ \varepsilon^{-1}$ should be weakly reflexive and symmetric if ε is a fuzzy relation from N_s into Q.*

Proof. (1) $\varepsilon \circ \varepsilon^{-1}$ is weakly reflexive, which is proved by $(\varepsilon \circ \varepsilon^{-1})(N_i, N_i') = \vee\{\varepsilon(N_i, N_j) \wedge \varepsilon^{-1}(N_j, N_i') \mid N_j \in Q\} \leq \vee\{\varepsilon(N_i, N_j) \wedge \varepsilon(N_i, N_j) \mid N_j \in Q\} = \vee\{\varepsilon(N_i, N_j) \wedge \varepsilon^{-1}(N_j, N_i) \mid N_j \in Q\} = (\varepsilon \circ \varepsilon^{-1})(N_i, N_i)$; and (2) $\varepsilon \circ \varepsilon^{-1}$ is symmetric, which is proved by $(\varepsilon \circ \varepsilon^{-1})(N_i, N_i') = \vee\{\varepsilon(N_i, N_j) \wedge \varepsilon^{-1}(N_j, N_i') \mid N_j \in Q\} = \vee\{\varepsilon^{-1}(N_j, N_i) \wedge \varepsilon(N_i', N_j) \mid N_j \in Q\} = \vee\{\varepsilon(N_i', N_j) \wedge \varepsilon^{-1}(N_j, N_i) \mid N_j \in Q\} = (\varepsilon \circ \varepsilon^{-1})(N_i', N_i)$. ■

A family composed of non-fuzzy subsets of nodes is denoted by D^{ε}, which is defined as $D^{\varepsilon} = \{M \subseteq N_s \mid (\exists 0 < \kappa \leq 1)(\forall N_i \in N_s)[N_i \in M \Leftrightarrow (\forall N_i' \in M)[\varepsilon(N_i, N_i') \geq \kappa]]\}$, where ε is a weakly reflexive fuzzy relation and a symmetric fuzzy relation among nodes on N_s, and M is a set of maximal nodes. Accordingly, if $D^{\varepsilon}{}_{\kappa} = \{M \subseteq N_s \mid (\forall N_i \in N_s)[N_i \in M \Leftrightarrow (\forall N_i' \in M)[\varepsilon(N_i, N_i') \geq \kappa]]\}$ is given, then $\kappa 1 \leq \kappa 2 \Rightarrow D^{\varepsilon}{}_{\kappa 2} \preccurlyeq D^{\varepsilon}{}_{\kappa 1}$ can be shown, where '$\preccurlyeq$' is denoted as a covering fuzzy relation, meaning every node in $D^{\varepsilon}{}_{\kappa 2}$ is a subset of a node in $D^{\varepsilon}{}_{\kappa 1}$.

If $\forall N_i, N_i' \in H$ and $\varepsilon(N_i, N_i') \geq \kappa$ where H is a complete subset of nodes of N_s, then H is called *κ-complete* with respect to ε. Here, a κ-complete subset that is not contained in any other κ-complete subset of nodes is called a *maximal κ-complete subset*.

Lemma 5.4. *D^{ε} is the family composed of all of maximal κ-complete subsets of nodes in a fuzzy undirected graph with respect to ε for* $0 \leq \kappa \leq 1$.

Proof. There exists $0 < \kappa \leq 1$ such that $\forall N_i' \in M$, $\varepsilon(N_i, N_i') \geq \kappa$ if $M \in D^{\varepsilon}$ and $N_i, N_i'' \in M$, thus $\varepsilon(N_i, N_i'') \geq \kappa$ and M is κ-complete. $M \subseteq H$ and H is κ-complete if H is a subset of X where $N_i \in X$. Let $N_i \in H$, since H is κ-complete, $\forall N_i' \in M$, $\varepsilon(N_i, N_i') \geq \kappa$, and since $M \in D^\varepsilon$, $N_i \in M$, thus $H \subseteq M$. Hence, M is maximal. Then let M be a maximal κ-complete subset, and let $N_i \in X$, then $N_i \in M \Leftrightarrow \forall N_i' \in M$, $\varepsilon(N_i, N_i') \geq \kappa$ and $M \in D^\varepsilon$. ■

Lemma 5.5. *There exists one or more κ-complete subsets of nodes* $M \in D^\varepsilon$ *such that* $\{N_i, N_i'\} \subseteq M$, *whenever* $\varepsilon(N_i, N_i') > 0$ *is fulfilled.*

Proof. If $N_i = N_i'$ is fulfilled, $\{N_i\}$ is certainly κ-complete for $\kappa = \mu_R(N_i, N_i)$ where μ_R is a fuzzy set relation R. Let us suppose $N_i \neq N_i'$, then $\varepsilon(N_i, N_i') = \varepsilon(N_i', N_i)$ by symmetry, $\varepsilon(N_i, N_i) \geq \varepsilon(N_i, N_i')$ and $\varepsilon(N_i', N_i') \geq \varepsilon(N_i, N_i')$ by weak reflexivity, and also $\{N_i, N_i'\}$ is shown as κ-complete where $\kappa = \varepsilon(N_i, N_i')$. The family composed of all of κ-complete subsets is denoted by F_κ, which has a maximal node set M. In addition, this node set is maximal in the family composed of all of κ-complete subsets of nodes, since any of the sets include M, which includes $\{N_i, N_i'\}$ as well, thus $M \in D^\varepsilon$ by Lemma 5.4. ■

A subclass of D^ε is noted to fulfil the condition of Lemma 5.5 that covers the node set N_s. For the fuzzy undirected graph, let ε be the fuzzy similarity relations of nodes on $N_s = \{N_1, N_2, \ldots, N_n\}$. In this case of the fuzzy undirected graph, all of the n nodes can be partitioned into families, each of which is composed of the maximal complete subsets on the basis of Lemma 5.5, but if the clique relations of nodes are automatically selected from the previous maximal complete subsets, then one or more clique relations are not included in the family.

An example is given here in order to better understand fuzzy similarity relations. Assume that κ is the fuzzy similarity relations of six nodes, representing $N_s = \{N_1, N_2, \ldots, N_6\}$, as shown in Figure 5.9. In this example of the fuzzy undirected graph, the family consists of the three maximal complete subsets of nodes, which are $\{N_1, N_2, N_6\}$, $\{N_1, N_3, N_5\}$, and $\{N_2, N_3, N_4\}$. These three maximal complete subsets fulfil the condition of Lemma 5.5. Here, the family does not contain the subset $\{N_1, N_2, N_3\}$. The set $D^\varepsilon{}_\kappa$ of this example becomes

$$D^{\varepsilon}{}_{\kappa} = \begin{cases} \{\{N_1\},\{N_2\},\{N_3\},\{N_4\},\{N_5\},\{N_6\}\} \\ \text{if } 0.8 < \kappa \leq 1.0, \\ \{\{N_3, N_5\},\{N_1\},\{N_2\},\{N_4\},\{N_6\}\} \\ \text{if } 0.6 < \kappa \leq 0.8, \\ \{\{N_1, N_3, N_5\},\{N_2, N_6\},\{N_4\}\} \\ \text{if } 0.4 < \kappa \leq 0.6, \\ \{\{N_1, N_2, N_6\},\{N_1, N_3, N_5\},\{N_2, N_4\},\{N_3, N_4\}\} \\ \text{if } 0.2 < \kappa \leq 0.4, \\ \{\{N_1, N_2, N_3\},\{N_1, N_2, N_6\},\{N_1, N_3, N_5\},\{N_2, N_3, N_4\}\} \\ \text{if } 0 < \kappa \leq 0.2, \end{cases} , \qquad (5.7)$$

where κ is not transitive. Here, as an example, we note the subset $\{N_1, N_2, N_6\}$ that is the

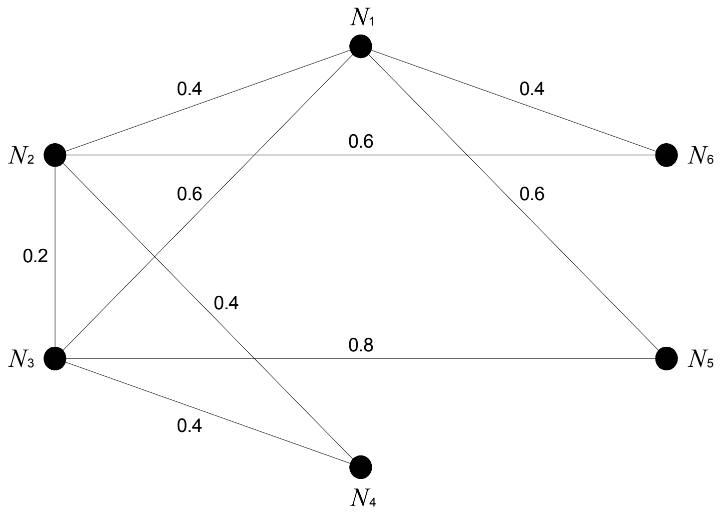


Figure 5.9. Example of the fuzzy undirected graph representing the fuzzy similarity relations.

maximal κ'-complete subset $\forall 0 < \kappa' \leq \kappa$, where $\kappa = 0.4$, since $\kappa(N_1, N_4) = \kappa(N_2, N_5) = \kappa(N_6, N_3) = 0$, and this subset is one of the existing subsets in the example fuzzy undirected graph, as shown in Figure 5.9.

5.7.3 Fuzzy hierarchical subsets

A classification of the given nodes in a fuzzy undirected graph is an important process for resolving sufficient nodes, which can connect the determined cohesive subsets or other meaningful nodes. Here, those resolved nodes play roles of component relations to figure out how each of the determined cohesive subsets makes links to each other through the node connectors. Particularly, we deal with the fuzzy undirected graph that is able to be classified into hierarchically different α-cut levels in fuzzy mathematical ways [95-97].

The concept of the α-cut levels with the aforementioned equations described in the fuzzy clique relations and similarity relations can be applied to real classification problems. We seek the specific and characterised hierarchical blocks of fuzzy sets in different levels. The classified fuzzy undirected subsets of nodes are used to better identify cohesive subsets in the fuzzy undirected graph. For example, the extension process of classifying the whole group of the real adapted fuzzy undirected network into hierarchical similarity subgroups, each of which has its own implication of one or more characteristics, is executed again in order to provide better information.

Definition 5.12. *An α-cut of a fuzzy undirected graph that is included in both σ and μ for* $\alpha \in [0, 1]$, *which are the fuzzy set relations*, *is represented as*

$$\sigma_\alpha = \{N_i \in N_s \mid \sigma(N_i) \geq \alpha\} \tag{5.8}$$

and

$$\mu_\alpha = \{(N_i, N_j) \in N_s \times N_s \mid \mu(N_i, N_j) \geq \alpha\}, \tag{5.9}$$

where $\mu_\alpha \subseteq \sigma_\alpha \times \sigma_\alpha$, then $(\sigma_\alpha, \mu_\alpha)$ is a fuzzy undirected graph with the fuzzy-based node set σ_α and fuzzy-based edge set μ_α.

Now, any member of the fuzzy node set σ_α is denoted by S_σ, which is defined in N_s by a union of α-cuts, representing

$$S_\sigma = \bigcup_{\alpha \in [0,1]} \alpha \sigma_\alpha, \tag{5.10}$$

and any member of the fuzzy relation set μ_α is denoted by S_μ, which is defined in $N_s \times N_s$ by a union of α-cuts, representing

$$S_\mu = \bigcup_{\alpha \in [0,1]} \alpha \mu_\alpha. \tag{5.11}$$

Further, the similarity relation based on the α-cut is denoted by L, the α-cut of L is denoted by L_α, a universe denoted by U, and the concept of L_α has been basically established [98]. If L is a similarity relation in U, then $\forall \alpha \in]0, 1]$, and each L_α should be an equivalent relation in U. In reverse, if there are L_α, where $0 < \alpha \leq 1$, which is a nested sequence that has distinct equivalent relations in U, if and only if $L_{\alpha 1} \subset L_{\alpha 2}$ with $\alpha 1 > \alpha 2$, L_1 is non-empty, and $\mathrm{dom}(L_\alpha) = L_1 \forall \alpha$, then a similarity relation in U is

$$L = \bigcup_\alpha \alpha L_\alpha, \tag{5.12}$$

where $\alpha = 1$ is included in any of the choices of α's in $]0, 1]$. The equivalence relation concept of this similarity relation based on the α-cut is able to be used for hierarchically organising a fuzzy undirected graph to deal with the fuzzy hierarchical subset purposes.

5.8 Fuzzy Operational Network

The applied reorganisational problem in a fuzzy operational network is given for an empirical study, and is solved by the fuzzy DNA-based algorithm. The fuzzy operational network mainly has two different properties, compared to normal operational networks. One different property is that of the fuzzy operational network that corresponds to a fuzzy undirected graph that has implicit operational values, which are represented as fuzzy membership grades to each connected pair of all workforces. The other different property is that the fuzzy operational network is composed of fuzzy operational connection levels among workforces in either clique relations or component relations with fuzzy membership grades, representing more specific operational connection levels in operations.

5.8.1 Model of the network

In this chapter's study, possibly existing workforces of the organisation are selected to artificially generate an operationally relative network associated with fuzzy membership grades, which are based on workforces' records, descriptions, and other important criteria of operational content. This connection level can be defined mathematically [99] in the following manner.

Let us suppose *n* workforces are given in a fuzzy operational network with operational connections denoted by $\mu(N_i, N_j)$, which measure a fuzzy operational connection level between workforces N_i and N_j. Here, $\mu(N_i, N_j)\in[0, 1]$, $N_i \neq N_j$, and $\mu(N_i, N_j) = \mu(N_j, N_i)$. The fuzzy operational connection level is denoted by $\sim \zeta$, and is in the *n* workforces, defined as

$$N_i \sim \zeta\ N_j :\Leftrightarrow \mu(N_i, N_j) \leq \zeta, \zeta \in [0,1]. \tag{5.13}$$

The multiple fuzzy operational levels among workforces are illustrated with a fuzzy undirected network. In the same manner as a fuzzy undirected graph, the fuzzy operational network is also denoted by $G = (N_s, E_s)$, where N_s is a workforce set of G and E_s is a set of operational connections. In the fuzzy operational network, whenever $N_i \sim \zeta\ N_j$ with respect to a given $\zeta\in[0, 1]$, there is a fuzzy operational connection level $(N_i, N_j)\in E_s$ that connects workforce N_i with workforce N_j.

The operational network with fuzzy operational connection levels is parameterised by ζ, which has a set of the constant number of workforces, and a varying set of fuzzy operational connection levels, and thus the fuzzy operational network is defined as follows:

$$G_\zeta = (N_s, E_{s\zeta}), \zeta \in [0,1], \tag{5.14}$$

where

$$E_{s\zeta'} \subseteq E_{s\zeta}, \zeta' \leq \zeta; E_{s0} = \phi, \tag{5.15}$$

which represents the more similar operational connection level of ζ, referring to the higher connections among the workforces.

Figure 5.10 shows the fuzzy operational network for this model. In the model fuzzy operational network, there are 40 workforces and operational connections with fuzzy membership grades. The 40 workforces are labelled N_i, i = 1, 2,…, 40. Although this fuzzy operational network involves only 40 workforces, it seems unorganised and complicated. The unorganised network can be reorganised, or optimally organised, by executing work rotation. The terms of work rotation are important for the operational manager to be able to enhance productive work. Hence, the operational manager should discover all of the workforces in each of all the clique component relations for work rotation.

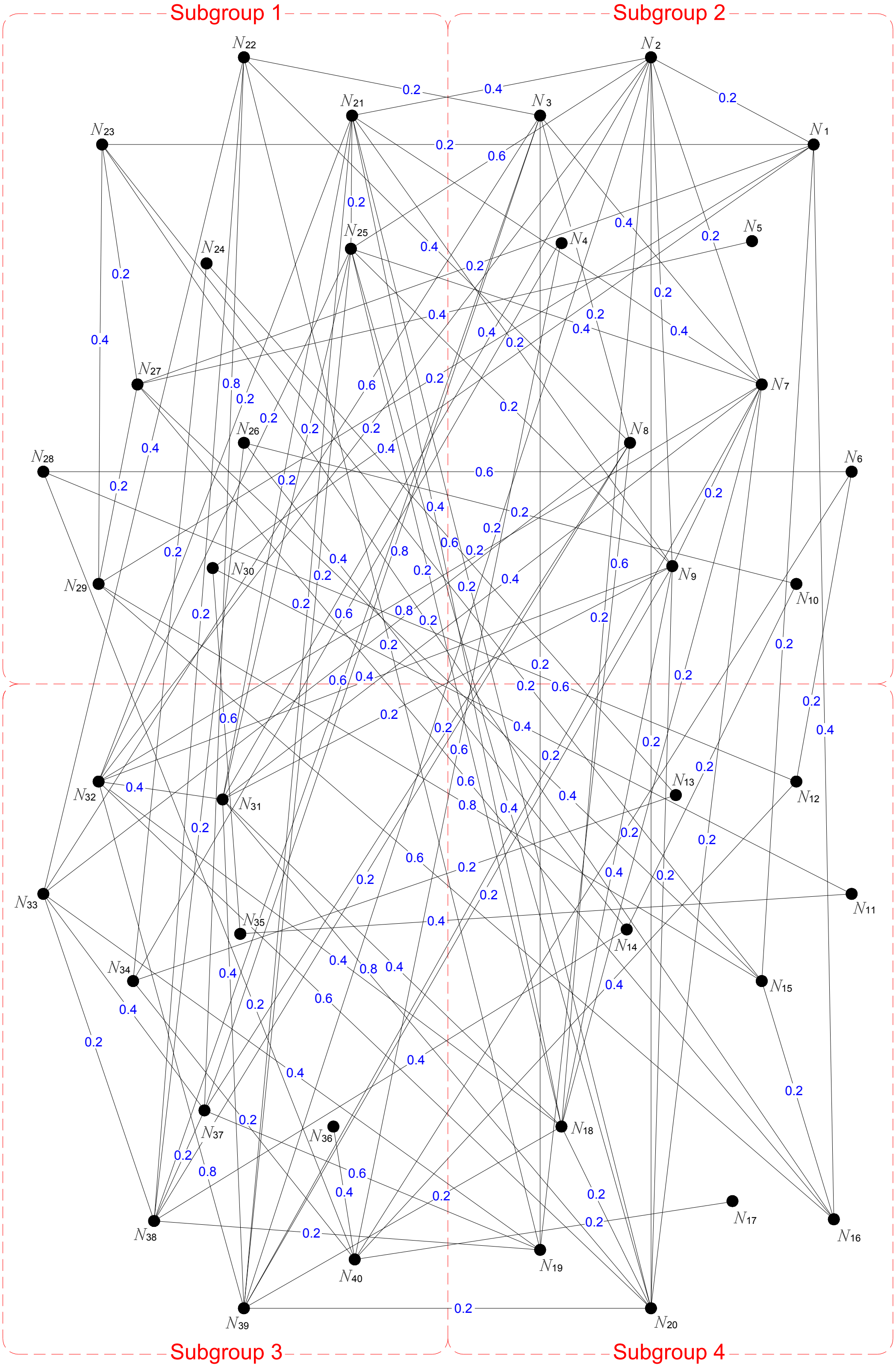


Figure 5.10. Model of a fuzzy operational network consisting of workforces.

5.8.2 Workforces in a clique relation

A fuzzy operational network, without considering fuzzy membership grades, can be composed of specific workforces in one or more clique relations. Recall that the clique relation is composed of at least three workforces. If there is a subset that is composed of only two workforces with an operational connection, then we call this subset as an *independent relation* in this chapter's study. For work rotation, the clique relation is a useful starting point for specifying formal properties, and has well-specified mathematical properties, and captures much of the intuitive notion of the cohesive subsets.

All of the workforces of the clique relation should be adjacent to each other and to the fuzzy operational network. For this chapter's study, the term "clique relation" is used for those workforces (three or more) who are adjacent to one another in a subgroup, and the term "independent relation" is used for a set of two connected workforces who are adjacent to each other. Workforces in a clique relation are represented mathematically [100, 101].

The complement fuzzy operational network of the fuzzy operational network G can be created and is denoted by CG, which becomes $CG = (N_s, CE_s)$, where

$$CE_s = \{(N_i, N_j) \mid N_i, N_j \in N_s, N_i \neq N_j \text{ and } (N_i, N_j) \notin E_s\}. \tag{5.16}$$

The subset of the workforce set can be represented as $SN_s \subseteq N_s$, and the subset of the workforce set in the fuzzy operational network is defined as

$$G(SN_s) = (SN_s, E_s \bigcap SN_s \times SN_s), \tag{5.17}$$

where SN_s induces the subgroup of workforces. If $\forall N_i$, $N_j \in N_s$ and $(N_i, N_j) \in E_s$, then a fuzzy operational network $G = (N_s, E_s)$, without considering fuzzy membership grades, can be *complete*, and is called a *complete operational network*. The workforces in a clique relation are denoted by C, meaning a subset of the workforces is composed of completely connected workforces with their operational connections.

5.8.3 Maximum number of workforces in a clique relation

In the fuzzy operational network, the maximum number of workforces in a clique relation can be represented by a subgroup of the workforces who are all completely adjacent to each other, showing the most cohesive subgroup and their strong connections in work-operational situations.

A complete operational network contains operational connections, each of which connects every pair of workforces. A fuzzy operational network $G = (N_s, E_s)$ is said to be complete if all its workforces are pairwise adjacent. The clique relation number of G is denoted by $\eta(G)$, and this number is the size of the maximum number of workforces in a clique relation. The number of clique relations is defined as follows:

$$\eta(G) = \max\{| SN_s |: SN_s \text{ is a clique relation of workforces in } G\}, \tag{5.18}$$

where the cardinality of the subset SN_s is the number of workforces, and $|SN_s|$ is the number of its workforces, meaning the objective of the maximum number of workforces in a clique relation is to find a clique relation of the maximum cardinality in G [102]. In other words, the maximum number of workforces in a clique relation is used to determine the maximum number of workforces in a subgroup, in which each workforce is connected with all other workforces.

5.9 Detection of Cohesive Subgroups with DNA

A set of possible fuzzy operational data is examined to aid in finding subgroups of workforces who have relatively strong connections in operational situations. The data set can become visible by displaying functions of a network, which can also be transformed to a matrix. We note one detection approach to represent the matrix of the fuzzy operational network as a *socio-matrix* [103]. If we deal with fuzzy membership grades, the socio-matrix is called a *fuzzy socio-matrix*, which should initially be created in the process of the fuzzy DNA-based algorithm. Creating the socio-matrix is the most important analytical procedure to analyse and recognise connection strengths of each pair of all the given workforces for detecting all of the possible workforces in each of all the clique relations and each of all the component relations.

Figure 5.11 shows a fuzzy socio-matrix of the model fuzzy operational network. The matrix consists of 40 rows and 40 columns labelled $x_{i,j}$, i and j = 1, 2,…, 40. Each of all

$$\begin{array}{c|cccccccccccc|}
 & N_1 & N_2 & N_3 & N_4 & N_5 & N_6 & N_7 & N_8 & N_9 & N_{10} & \cdots & N_{40} \\
\hline
N_1 & 1.0 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
N_2 & 0.2 & 1.0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0.2 & 0 & \cdots & 0 \\
N_3 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0.4 & 0.2 & 0 & 0 & \cdots & 0 \\
N_4 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0.6 \\
N_5 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
N_6 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & \cdots & 0.2 \\
N_7 & 0 & 0.2 & 0.4 & 0 & 0 & 0 & 1.0 & 0 & 0.2 & 0 & \cdots & 0 \\
N_8 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & \cdots & 0 \\
N_9 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0.2 & 0 & 1.0 & 0 & \cdots & 0 \\
N_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
N_{40} & 0 & 0 & 0 & 0.6 & 0 & 0.2 & 0 & 0 & 0 & 0 & \cdots & 1.0 \\
\hline
\end{array}$$

Figure 5.11. Fuzzy socio-matrix of the model fuzzy operational network.

the operational connection values comes from a workforce N_i to a workforce N_j, and $x_{i,j}$ records which pairs of workforces are adjacent to each other. If workforces N_i and N_j are adjacent, then $0 < x_{i,j} \leq 1.0$, and if workforces N_i and N_j are not adjacent, then $x_{i,j} = 0$. Moreover, an operational connection between two workforces can be either present or absent. If an operational connection is present, then it goes both from N_i to N_j and from N_j to N_i, thus, $x_{i,j} = x_{j,i}$ in the fuzzy socio-matrix. If an operational connection is absent, then it is $x_{i,j} = 0$, meaning that two workforces are not adjacent.

The fuzzy DNA-based algorithm is proposed for this chapter's study based on the algorithm of the maximal clique problem solution that was proposed by Ouyang *et al.* [104], and we extend the algorithm to determine all the workforces in clique and component relations from small to large values of cohesiveness, and the maximum number of workforces in a clique relation in the fuzzy case of the reorganisational problem. The proposed algorithm proceeds as follows:

Step 1: Workforces in either a clique relation or an independent relation are represented by either "present" or "absent" in *n* workforces. If the workforce is included in the clique relation or the independent relation, then we set the value to 1 meaning present, otherwise it is 0, meaning absent.

Step 2: All the possible combinations among the encoded workforce DNA sequences are created from 2^n, basically by using two techniques, which are parallel overlap assembly (POA) and polymerase chain reaction (PCR). The POA technique is used to amplify the groups of the initially encoded workforce DNA sequences into all of the possible combinatorial connections among the given workforces. The PCR technique is used to amplify and to detect only specific DNA strands, each of which has the path from the starting labelled workforce to the ending labelled workforce.

Step 3: An existing operational connection between a pair of two workforces in the fuzzy operational network, we call a *valid connection*, whereas a non-existent operational connection is called an *invalid connection*. In other words, all of the given workforces are connected to each other by their operational connections, which are valid connections. Any of two workforces that do not have any operational connections are invalid connections, corresponding to the operational connections in the complement fuzzy operational network. In this step, all of the operational connections should be distinguished into either valid connections or invalid connections by checking all of the possible connections in both the fuzzy operational network and the complement fuzzy operational network.

Step 4: Each of the given workforces has been encoded by its own DNA sequences, each of which also includes its own available restriction enzyme site. By using the available restriction enzymes, all of the invalid connections can be removed in the fuzzy operational network. All of the valid connections can be sorted by removing the invalid connections based on cutting at the restriction enzyme sites. After removing all the invalid connections among workforces, the remaining data pool is sorted in order to select the existing DNA sequences from 2 bits to *n* bits of value 1.

Step 5: All the subgroups of workforces in each clique relation are found, which correspond to all the lengths of DNA strands using a simulated gel electrophoresis apparatus. In this step, the shortest DNA strand corresponds to the maximum number of workforces in a clique relation, and the second shortest DNA strand corresponds to the second largest maximum number of workforces in a clique relation. Each subgroup of workforces in component relations are distinguished and marked by checking all of the resolved clique relations.

Step 6: Step 4 is repeated until all the workforces in a clique relation are obtained. This step is completed when there is only one independent relation remaining, representing a set of two connected workforces in the fuzzy operational network.

5.10 Experimental Studies and Results

To detect all of the possible workforces in each of both clique relations and independent relations, the proposed algorithm uses novel simulated experimental studies and results. In this section, we describe simulated experimental studies and results to resolve specific workforces who are included in each clique and component relation using the algorithm.

5.10.1 Experimental studies

For the simulated experimental studies, we basically design all of the DNA sequences that are formed in double-stranded DNA (dsDNA). The 40 workforces are given for the model network, and each workforce represents the DNA sequence in a binary number that is either 1 or 0, meaning either present or absent, as explained above.

Two kinds of DNA sequences are given by a *binding sequence* and a *value sequence* [104]. 80 DNA oligonucleotides are created for the simulated experimental studies. The binding sequence is denoted by Eu_i for the upper DNA strand and El_i for the lower DNA strand, while the value sequence is denoted by N_i, which also corresponds to each of all the workforces. The binding sequences are used for connecting each workforce of the DNA sequence, and the value sequences are used for distinguishing whether or not the binding sequences contain a particular workforce. Both Eu_i and El_i are set to have a length of 10 base pairs (bp: after hybridisations and replications). N_i has a length of 0 bp if the value is 1 (present), and 6 bp if the value is 0 (absent). In addition, each DNA oligonucleotide consists of two different binding motifs. It is important when encoding DNA sequences, each of which has a different pattern for each workforce and contains its own available restriction enzyme site. One is denoted by $5'\text{-}Eu_iN_iEu_{i+1}\text{-}3'$, where i = 1, 3, 5,…, 39 represents odd numbers. The other one is denoted by $5'\text{-}El_{i+1}N_iEl_i\text{-}3'$, where i = 2, 4, 6,…, 40 represents even numbers. Based on the sticking operation [105], these two different DNA strands (upper and lower DNA strands) of single-stranded DNA (ssDNA) stick to each other by the annealing process, and they become a dsDNA sequence.

5.10.2 Results of the experimental studies

For the reorganisational problem, three main biochemical techniques were used in the simulated experimental studies. First, both POA and PCR were used to amplify DNA strands. Second, 40 available restriction enzymes were applied to remove all complementary sequences, corresponding to all of the invalid connections in the complement fuzzy operational network. Finally, the simulated gel electrophoresis was used for repeatedly selecting all of the final products of the shortest DNA strands that corresponded to all of the possible workforces in each of the clique relations and independent relations.

Each of the determined clique relations and component relations was obtained using the simulated experimental studies. A program compiled using Vector NTI software was used to represent the length of DNA strands. Further, a DNA sequence table was also created based on the expected results of all the determined possible workforces in each clique and component relation. The final results of the DNA sequence table were created by monitoring the length of the DNA strands in the simulated gel electrophoresis. Each clique and independent relation included one or more lengths of 0 bp (empty spaces) in the DNA sequence, meaning that the value was 1 and was present in each subgroup. In each subgroup, one or more empty spaces represented any possible component relations among the workforces. Those workforces linked two different subgroups and were found by noting all of the empty spaces, each of which was vertically located in the same columns.

All of the possible workforces in each clique and independent relation were determined based on the above results. In more detail, in the fuzzy operational network, there were fifteen subsets in clique and independent relations, denoted by P_i, $i = 1, 2,\ldots, 15$, which were also composed of two connected subgroups by operational connections. One large connected subgroup is denoted by LP_1, and the other connected subgroup is denoted by LP_2. The workforce numbers and the DNA strand length of each subgroup for the two connected subgroups are described as follows:

Firstly, in LP_1, (1) the maximum clique relation of workforces was detected as $P_1 = \{N_2, N_7, N_9, N_{18}, N_{20}, N_{21}, N_{25}, N_{31}, N_{32}, N_{39}\}$ that was connected at a distance of 590 bp by ten workforces; (2) the clique relation of seven workforces was found as $P_2 = \{N_3, N_8, N_{19}, N_{22}, N_{33}, N_{37}, N_{38}\}$ that was connected at a distance of 608 bp; (3) the clique relation of six workforces was found as $P_3 = \{N_1, N_{15}, N_{16}, N_{23}, N_{27}, N_{29}\}$ that was connected at a distance of 614 bp; (4) three subsets (P_5, P_6, and P_7) of three workforces were found at a distance of 632 bp; and (5) four subsets (P_{10}, P_{11}, P_{12}, and P_{13}) of two connected workforces were found at a distance of 638 bp. Secondly, in LP_2, (1) the maximum clique relation of workforces was detected as $P_4 = \{N_6, N_{12}, N_{28}, N_{40}\}$ that was connected at a distance of 626 bp by four workforces; (2) two subsets (P_8 and P_9) of three workforces were found at a distance of 632 bp; and (3) two subsets (P_{14} and P_{15}) of two connected workforces were found at a distance of 638 bp.

5.11 Identification of Cohesive Subgroups

All of the specific workforces in each clique and component relation are determined based on the simulated experimental studies and results. Moreover, we extend this to detect all of those workforces in fuzzy relations in order to identify cohesive subgroups for the final purpose. Obviously, the results, containing both (1) clique and component relations; and (2) fuzzy relations, make work rotation execution more efficient, and provide better and much more useful information to operational managers.

The specific workforces in a similarity subgroup certainly have a similar operational

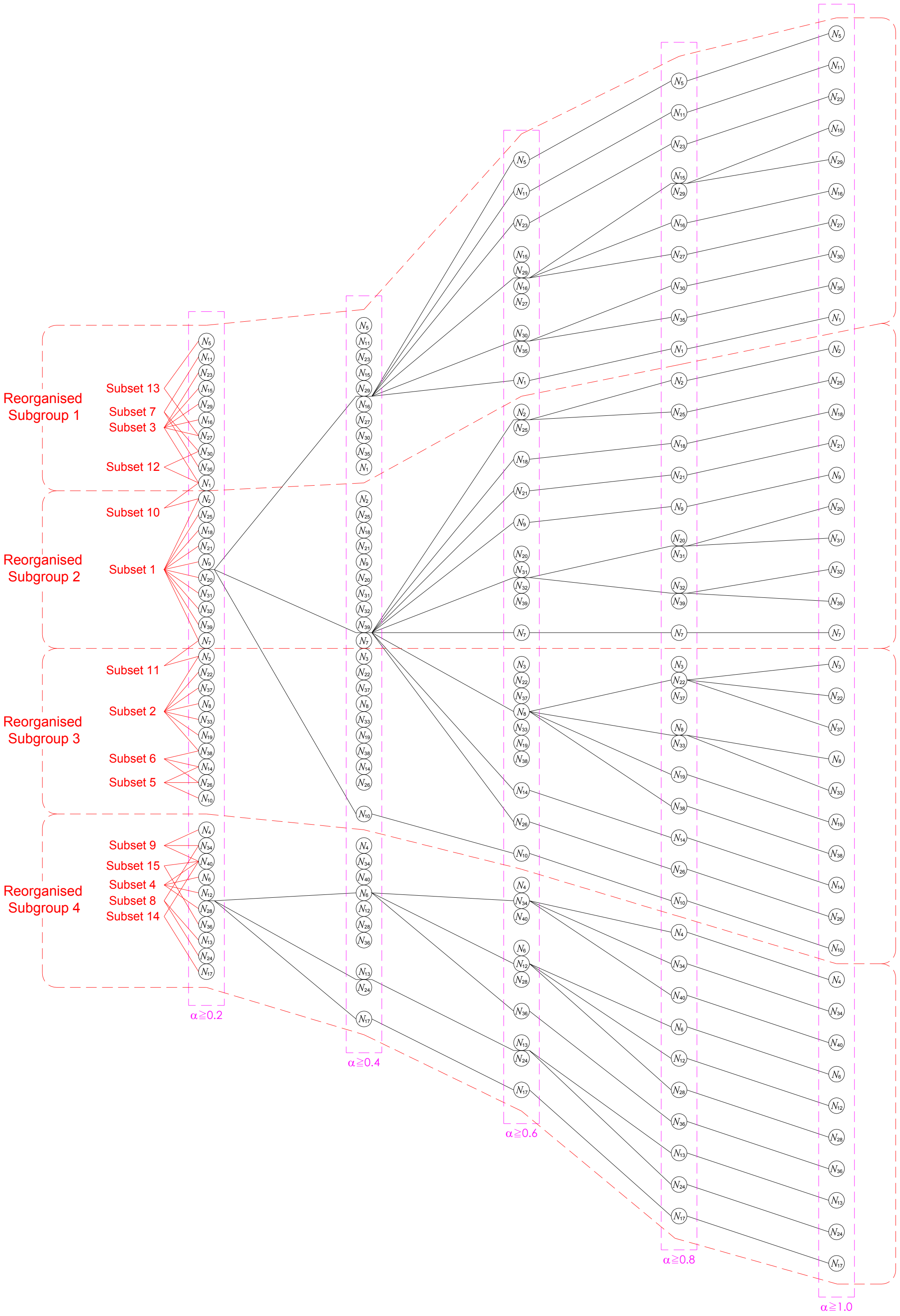


Figure 5.12. Similarity hierarchical structure of the reorganised subgroups composed of the identified cohesive workforce subsets in different fuzzy hierarchical levels.

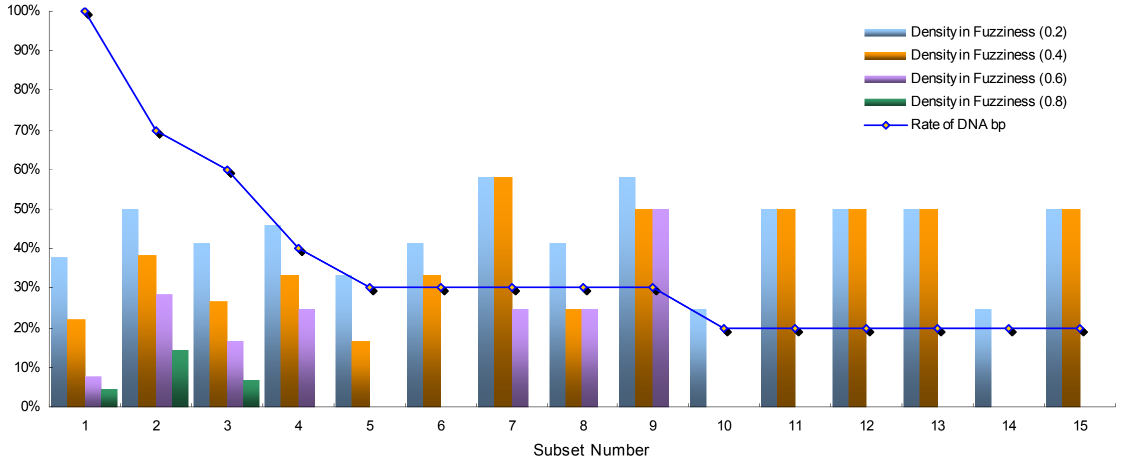


Figure 5.13. Comparison graph of the densities in fuzziness and the rate of DNA bp in each identified cohesive workforce subset.

connection. In a fuzzy operational network, this connection is represented by a fuzzy connection among workforces. Exploiting the α-cuts of the aforementioned equations, fuzzy sets in the mathematical processes described earlier will reorganise all of the workforces into fuzzy similarity relations, which are all divided into fuzzy hierarchical levels. All of the workforces in fuzzy relations can be determined and can be reorganised into similarity subgroups to identify cohesive subgroups according to the definitions and lemmas explained in the previous sections.

To detect all the possible workforces in fuzzy relations, let $\sigma(N_i)$ = 1.0 for all $N_i \in N_s$, and let each μ also be the fuzzy subset of each of all of the two-workforce combinations. All of the workforces in similarity subgroups, corresponding to fuzzy similarity relations, are determined by calculating the given fuzzy membership grades (strengths of the operational connections). An α-cut of a fuzzy set can be exploited by dividing those workforces into hierarchical similarity levels.

Figure 5.12 illustrates the similarity hierarchical structure that is created by all of the obtained and calculated results. In the structure, all of the possible workforces have been reorganised into five different fuzzy hierarchical levels, representing five different α-cuts, which have been taken at $\alpha \geq 0.2$, $\alpha \geq 0.4$, $\alpha \geq 0.6$, $\alpha \geq 0.8$, and $\alpha \geq 1.0$. In addition, Figure 5.13 shows the comparison analysis between the densities in fuzzy membership grades and the rates of DNA bp. In Figure 5.12, we can recognise that the hierarchically reorganised subgroups 1 and 2 are clearly connected by $\alpha \geq 0.2$ between the two workforces (N_1 and N_2) that are particularly recognised to be operational connectors between the reorganised subgroups 1 and 2. In addition, the hierarchically reorganised subgroups 2 and 3 are clearly connected by $\alpha \geq$ from 0.2 to 0.4 between the two workforces (N_3 and N_7) that are particularly recognised to be operational connectors between the reorganised subgroups 2 and 3.

5.12 Density Analysis

The density analysis of the finally identified cohesive subgroups is focused on measuring the efficiency of using the fuzzy DNA-based algorithm, which consists of fuzzy-based methods and a general molecular algorithm. The previous fuzzy operational network, shown in Figure 5.10, is the network prior to being reorganised based on the results of the fuzzy DNA-based algorithm. We compare the previous fuzzy operational network with the reorganised fuzzy operational network, shown in Figure 5.12, by analysing the densities of these two networks.

5.12.1 Comparison of the two networks

The reorganised fuzzy operational network was compared to the previous fuzzy operational network to prove the efficiency of the reorganised network. For this process, we calculated the inclusiveness and the densities of the two networks. The calculating method was explained in the previous section, but we briefly describe the measure of the densities for the case of the applied reorganisational problem in the fuzzy operational network.

Inclusiveness in the fuzzy operational network was basically calculated by examining the total number of workforces minus the number of isolated workforces. The measurement of inclusiveness is conducted by looking at the number of connected workforces who are represented as a proportion of the total number of workforces in the fuzzy operational network. In the fuzzy operational network, the given operational connections among the workforces are considered without fuzzy membership grades, meaning each of all the operational connections does not involve any values. Another proportion of the maximum possible number of operational connections among the workforces can be expressed as the density. To generalise the notion of the density to the fuzzy operational network, one can average the fuzzy membership grades attached to the operational connections across all of the operational connections. The other proportion of the maximum possible number of operational connections among the workforces can be graphically expressed and represented as the density with fuzzy membership grades.

5.12.2 Density analysis results

Table 5.1 shows the results of inclusiveness (proportion), density, density in fuzzy membership grades, and other comparators that were used to compare the two different networks.

Each of the four identified cohesive subgroups had a comparison of the previous fuzzy operational network and the reorganised fuzzy operational network, each of which was also divided into the different α-cut levels. Figure 5.14 shows the comparisons of the previous subgroups and the reorganised subgroups in level 0.2. Figure 5.15 shows the comparisons of the previous subgroups and the reorganised subgroups in level 0.4. Figure 5.16 shows the comparisons of the previous subgroups and the reorganised subgroups in level 0.6. Figure 5.17 shows the comparisons of the previous subgroups and the reorganised subgroups in level 0.8. From all of the density results, we measured the efficiency of identifying cohesive subsets for the reorganisational problem using the

Table 5.1. Results of inclusiveness, density, and density in fuzzy membership grades for both previous and new subgroups in each level.

Level	Subgroup No.	Number of Operational Connections	Inclusiveness	Density	Density in Fuzzy Membership Grades
0.2	Subgroup 1	4	0.500	0.089	0.313
	New Subgroup 1	20	1.000	0.444	0.450
	Subgroup 2	6	0.600	0.133	0.292
	New Subgroup 2	45	1.000	1.000	0.378
	Subgroup 3	8	0.900	0.178	0.469
	New Subgroup 3	26	1.000	0.578	0.471
	Subgroup 4	2	0.400	0.044	0.250
	New Subgroup 4	14	1.000	0.311	0.464
0.4	Subgroup 1	1	0.200	0.022	0.500
	New Subgroup 1	11	1.000	0.244	0.614
	Subgroup 2	1	0.200	0.022	0.500
	New Subgroup 2	17	1.000	0.378	0.588
	Subgroup 3	5	0.700	0.111	0.600
	New Subgroup 3	13	0.900	0.289	0.692
	Subgroup 4	0	0.000	0.000	0.000
	New Subgroup 4	7	0.900	0.156	0.679
0.6	Subgroup 1	0	0.000	0.000	0.000
	New Subgroup 1	4	0.600	0.089	0.813
	Subgroup 2	0	0.000	0.000	0.000
	New Subgroup 2	4	0.600	0.089	0.875
	Subgroup 3	1	0.200	0.022	1.000
	New Subgroup 3	7	0.700	0.156	0.857
	Subgroup 4	0	0.000	0.000	0.000
	New Subgroup 4	5	0.800	0.111	0.750
0.8	Subgroup 1	0	0.000	0.000	0.000
	New Subgroup 1	1	0.200	0.022	1.000
	Subgroup 2	0	0.000	0.000	0.000
	New Subgroup 2	2	0.400	0.044	1.000
	Subgroup 3	1	0.200	0.022	1.000
	New Subgroup 3	3	0.500	0.067	1.000
	Subgroup 4	0	0.000	0.000	0.000
	New Subgroup 4	0	0.000	0.000	0.000
1.0	Subgroup 1	0	0.000	0.000	0.000
	New Subgroup 1	0	0.000	0.000	0.000
	Subgroup 2	0	0.000	0.000	0.000
	New Subgroup 2	0	0.000	0.000	0.000
	Subgroup 3	0	0.000	0.000	0.000
	New Subgroup 3	0	0.000	0.000	0.000
	Subgroup 4	0	0.000	0.000	0.000
	New Subgroup 4	0	0.000	0.000	0.000

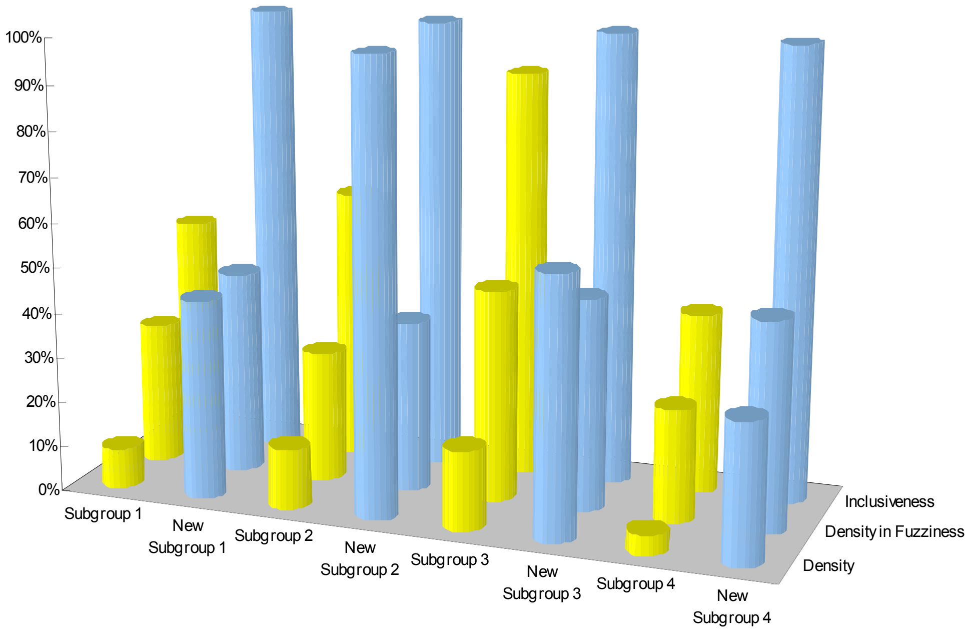


Figure 5.14. Comparisons of inclusiveness, density in fuzziness, and density between previous and newly reorganised subgroups in *α*-cut level 0.2.

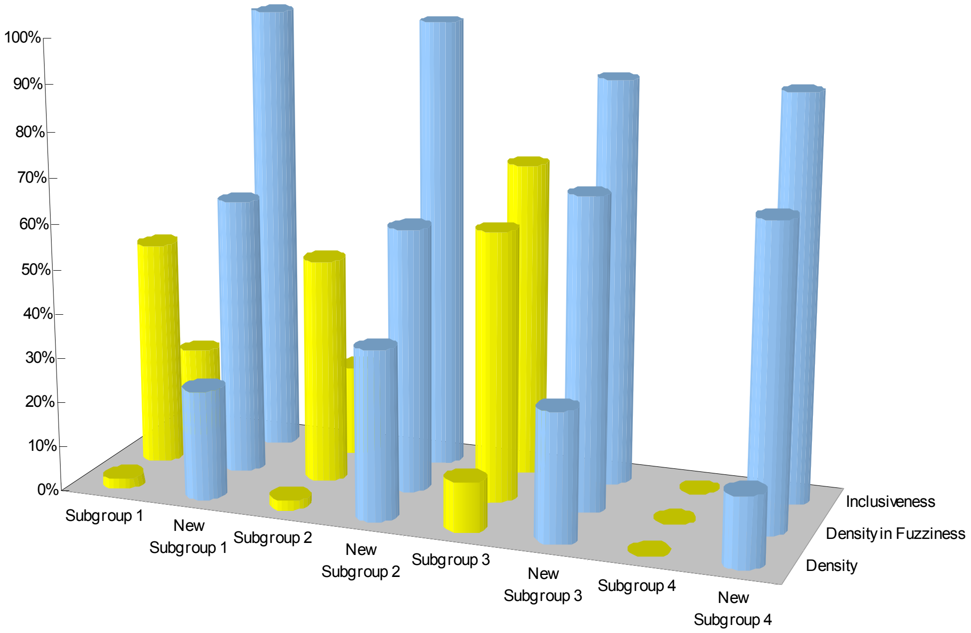


Figure 5.15. Comparisons of inclusiveness, density in fuzziness, and density between previous and newly reorganised subgroups in *α*-cut level 0.4.

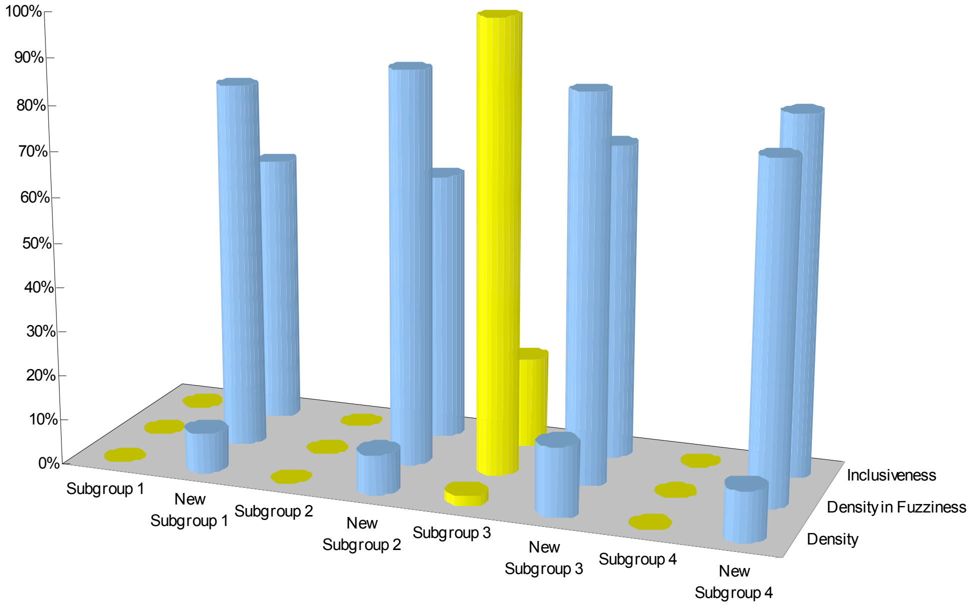


Figure 5.16. Comparisons of inclusiveness, density in fuzziness, and density between previous and newly reorganised subgroups in *α*-cut level 0.6.

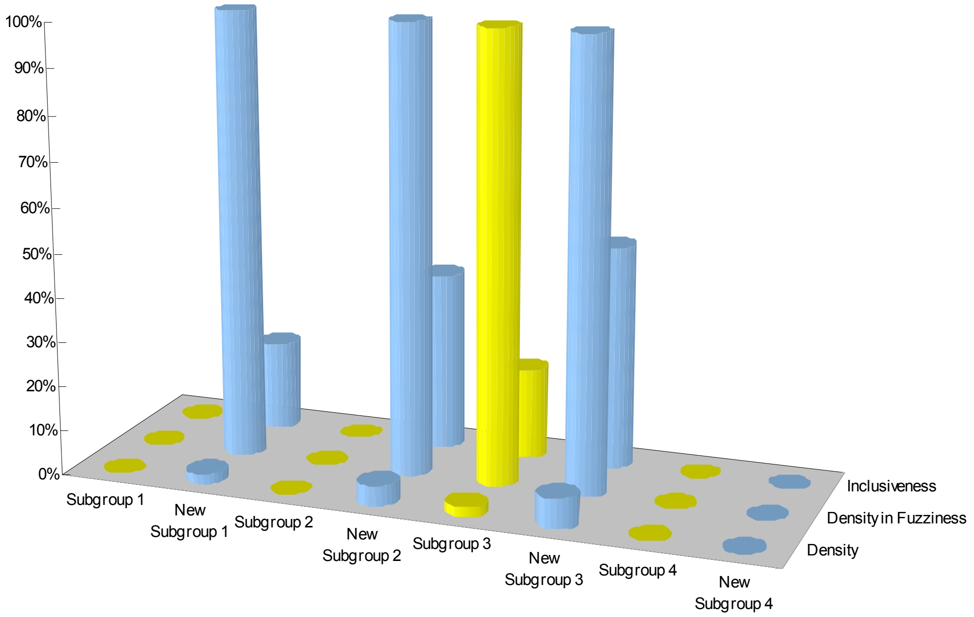


Figure 5.17. Comparisons of inclusiveness, density in fuzziness, and density between previous and newly reorganised subgroups in *α*-cut level 0.8.

fuzzy DNA-based algorithm.

5.13 Classification Method Applied to Clustering

The fuzzy DNA-based algorithm was designed for classifying the specific labelled nodes (workforces) into subsets (subgroups), and reorganising them into fuzzy similarity relations (fuzzy hierarchical levels) based on their fuzzy relations (fuzzy operational connection levels). These processes enabled us to identify their cohesiveness. Moreover, we would like to mention that, from this chapter's study, the classification method using the fuzzy DNA-based algorithm can be applied to clustering coordinated patterns. In other words, without considering specific labels, we can switch (1) workforces to patterns; (2) subgroups to clusters; and (3) fuzzy operational connection levels to distance values.

A large number of heterogeneous patterns often emerge in different kinds of situations. For instance, various types of Web documents keep increasing in various divisions or departments of each information database through the rapid expansion of the Internet. Further, in bioinformatics, the genetic data render huge and unpredictable heterogeneous patterns. To analyse, design, and improve a given system composed of large amounts of granular data, an appropriate technique is required to extract the necessary information and knowledge for better decision making. The widely practiced techniques include clustering analysis.

Clustering techniques often play profound roles in information and communication technology as well as other management engineering applications and techniques. They serve to cluster valuable information and knowledge objects in various ways in order to analyse, control, design, improve, organise, and visualise complicated and heterogeneous pattern data.

For a stable system, the objective of clustering heterogeneous patterns is to sort the data into clusters based on their high degrees of resemblance among the data of the same cluster and low resemblance among the data of different clusters. However, the main issue in clustering heterogeneous patterns includes the uncertainty data, which may be clustered. Thus, the quality of the system can be properly evaluated by clustering the uncertainty data.

A general molecular algorithm can be integrated with mathematical methods to become a new molecular computational algorithm (including its own adapted molecular computational experimentation) for the clustering of heterogeneous patterns and other complex coordinated patterns. The new molecular computational algorithm for clustering heterogeneous patterns provides a novel method of cluster analysis. In addition, the main reason for combining mathematical methods is to reduce the number of DNA fragments in encoding DNA sequences. Switching clustering to classification enables us to approach adaptations and applications for fuzzy coordinated values of heterogeneous patterns in real-world applications.

5.14 Computational Times and Solvable Sizes

When mathematically dealing with discrete algorithms, people may require a reasonable running time that should be clearly quantified. For natural combinatorial problems, search spaces exponentially grow depending on the input size, where the size increases

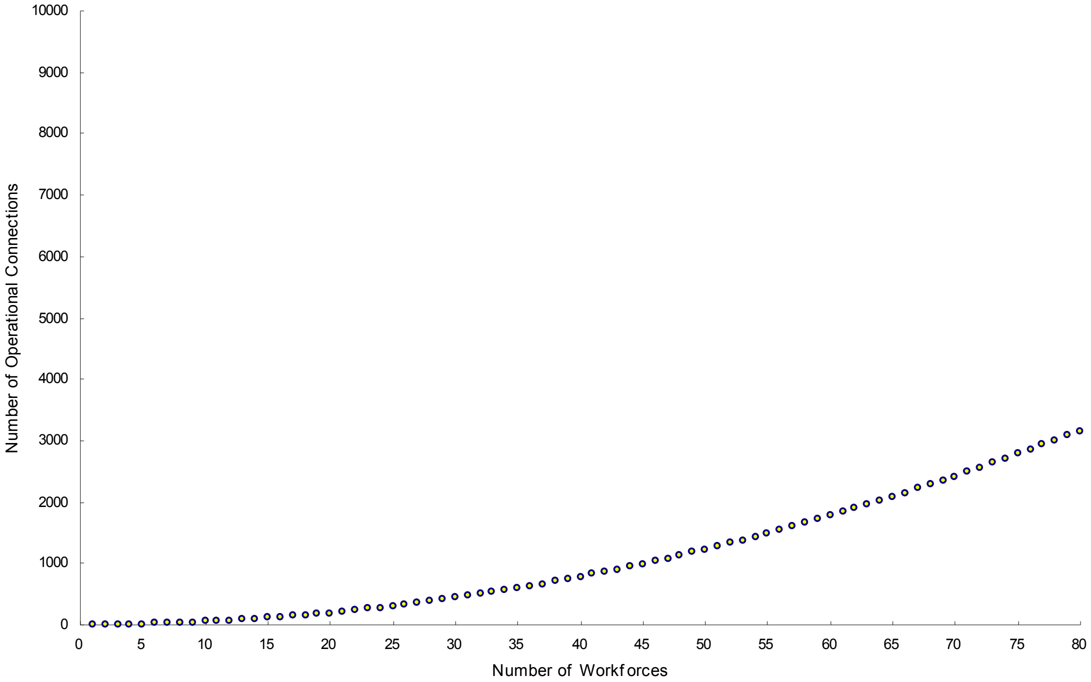


Figure 5.18. Comparison graph of the number of workforces and the number of operational connections.

by one and the number of its possibilities multiplicatively increases [106, 107]. A comparison of the number of workforces and the number of operational connections is shown in Figure 5.18.

Table 5.2 shows comparisons of approximated running times to organise workforces

Table 5.2. Comparisons of approximated running times for the exponential-time algorithm and the prepared fuzzy DNA-based algorithm.

Number of Workforces	10	30	50	100
The Exponential-Time Algorithm	< 1.00 second	18.00 minutes	36.00 years	100,000,000,000,000,000.00 years
The Prepared DNA-Based Algorithm	2.89 minutes	27.96 minutes	1.31 hours	5.30 hours

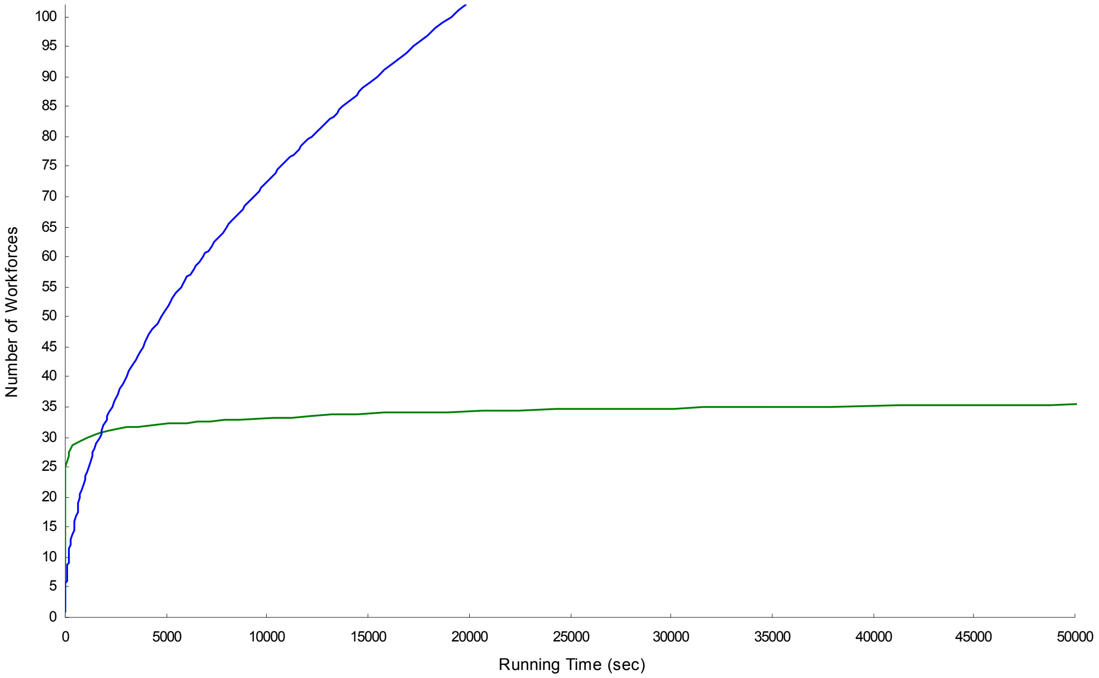


Figure 5.19. Comparison graph of approximated running times in seconds: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared fuzzy DNA-based algorithm.

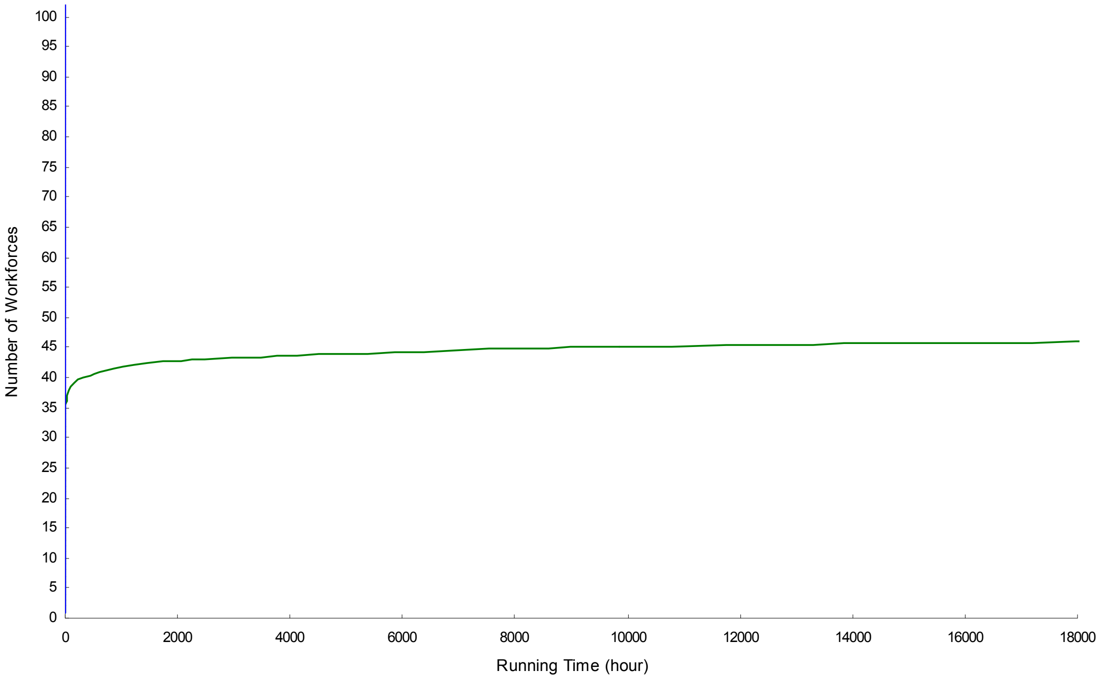


Figure 5.20. Comparison graph of approximated running times in hours: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared fuzzy DNA-based algorithm.

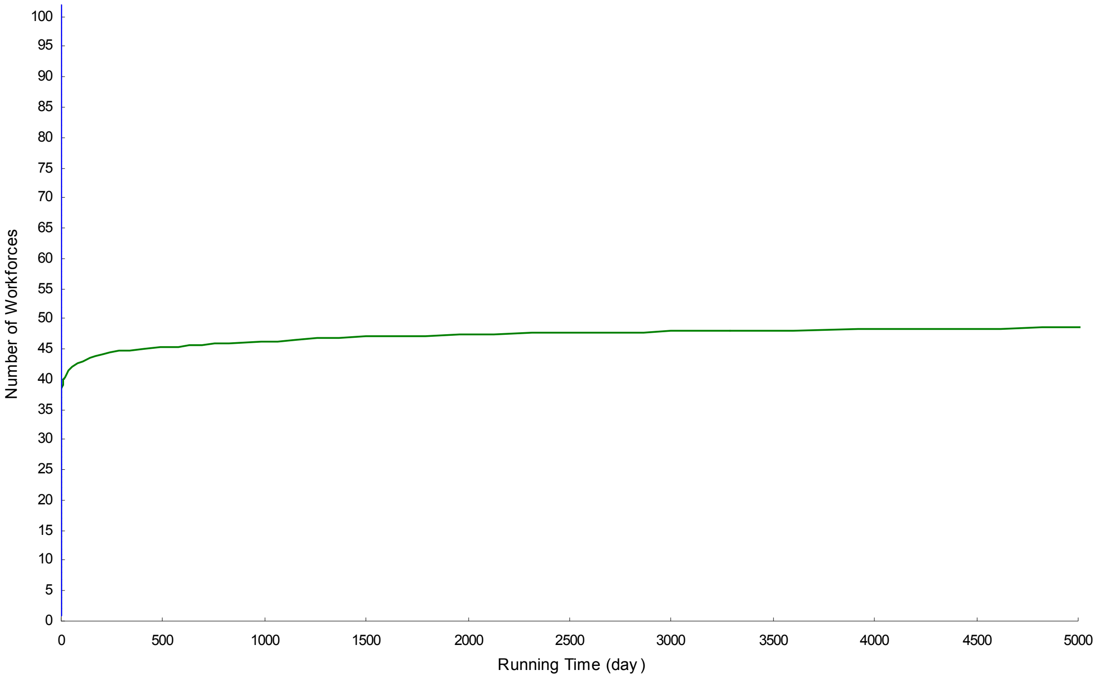


Figure 5.21. Comparison graph of approximated running times in days: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared fuzzy DNA-based algorithm.

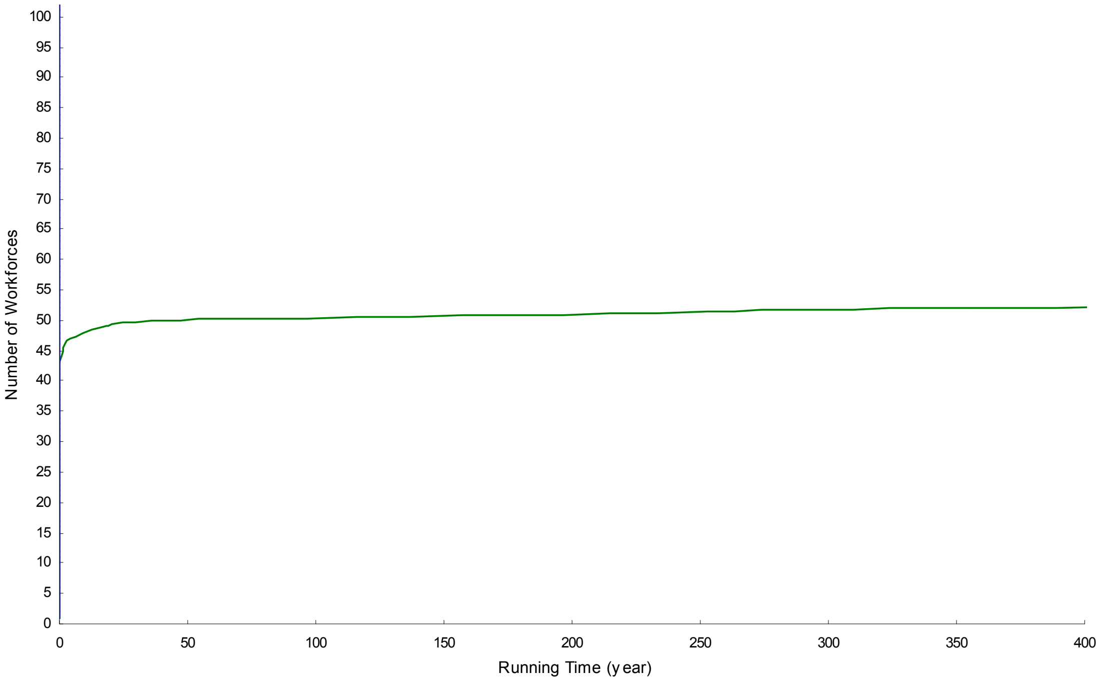


Figure 5.22. Comparison graph of approximated running times in years: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared fuzzy DNA-based algorithm.

with operational connections for cohesiveness, in which the number of workforces corresponds to the inputs of size 10, 30, 50, and 100. The table deals separately with the exponential-time algorithm and the prepared fuzzy DNA-based algorithm (we are ready to detect solutions). For the graphic representations, Figures 5.19 to 5.22 show a graphic comparison of approximated running times for the exponential-time algorithm and the prepared fuzzy DNA-based algorithm.

For the exponential-time algorithm, we suppose that a processor executes a million high levels of instructions a second, and the exponential-time algorithm uses an operation of 2^n [106]. For the prepared fuzzy DNA-based algorithm, we measured and calculated the approximated running times of the algorithms on the basis of previous experimental reports, our experimental experiences, and genetic engineering notes [62-66].

If we suppose that there are two different typical algorithms that are an exponential-time algorithm and a polynomial-time algorithm, obviously the polynomial-time algorithm is efficient. However, in this chapter's study, reorganising a large number of workforces induces an NP-hard problem, which means that a polynomial-time algorithm has not been properly and truly discovered yet for this reorganisational problem.

5.15 Concluding Remarks

In this chapter, a new molecular computational algorithm was proposed, and was combined with several fuzzy-based methods to create a fuzzy DNA-based algorithm. This algorithm was used to identify cohesive subsets and was applied to the reorganisational problem to identify cohesive subgroups in a fuzzy operational network for work rotation.

The fuzzy DNA-based algorithm was exploited to identify various types of cohesive subgroups. In particular, the algorithm was used to efficiently detect all of the workforces in each clique relation and component relation, including the maximal and maximum numbers of workforces in clique relations and independent relations. The molecular engineering detection was able to reveal four cohesive subgroups. Further, the fuzzy-based molecular engineering detection was able to finally identify cohesive subgroups by calculating fuzzy relations of the fuzzy operational network. The fuzzy DNA-based algorithm reduced routine tasks and enhanced productive work for operational processes. The density analysis of the reorganised subgroups in the two different networks also produced reliable results to prove the efficiency of the fuzzy DNA-based algorithm.

This chapter implies that an unconventional potential approach to creating a novel integrated algorithm that can identify cohesive subgroups in executing work rotation for real-world applications is possible. For the manufacturing control field or other operational management fields, the fuzzy DNA-based algorithm has the possibility of being used for any applied reorganisational problems. In addition, in this chapter's study, we showed the possibilities of using the fuzzy DNA-based algorithm in various ways to approach applications of relationally intractable problems.

Chapter 6

Modelling Interpretive Structures Based on a Hierarchical DNA-Based Algorithm

6.1 Overview

This chapter introduces a novel molecular computational algorithm that is used for building a new interpretive structural modelling (ISM) method (a new decision-making method), which models interpretive structures. In other words, the algorithm is used for structuring a model of interpretively problematic situations into hierarchical levels using DNA molecules with molecular engineering techniques. This algorithm is termed a *hierarchical DNA-based algorithm*. Further, we show an example of a contextual problem that will be solved by using the hierarchical DNA-based algorithm.

For decision making, an ISM method is often associated with classifying complicated contexts, composed of contextual content, into subgroups to construct a hierarchically restructured digraph, which properly provides comprehensible information and results. Those complicated contexts may be represented as a set of relational elements, and this relational set is represented as a digraph. The ISM method may be the key tool (a classification support tool) of the problem-solving process when the same or similar problems have conceptualising interactions in different departments, systems or other various kinds of organisational groups.

The issue of computational complexity is of profound relevance when dealing with complex relations or problems involving a large number of ideas or problems, corresponding to elements or element nodes in a digraph. Structuring the problem with a large number of elements in an ISM process, where minimising the crossings among those elements, is an intractable problem. To address this problem, the hierarchical DNA-based algorithm is used for minimising all crossings among the given elements to appropriately construct a hierarchically restructured digraph. The restructured digraph provides an easier and more reliable way to build an efficient and reliable decision support system.

In this chapter, we show an example of a contextual problem that is solved by using the hierarchical DNA-based algorithm. This chapter also presents a new approach for applying a computational molecular method to ISM to measure the efficiency of the algorithm, exploited by advanced bioscience technologies, in calculating a large number of elements to be an innovative ISM method.

6.2 Background and Motivations

In many different areas, such as science, engineering, organisational and social sciences, etc., complex and unpredictable issues or situations, composed of a set of given problem contexts, often emerge and should be simplified to be understandable information. ISM has been used as one method that simplifies such complicated contextual issues by constructing a structural digraph.

Since Warfield [108-110] first developed ISM, also referred to as *structural modelling of problematic situations* [111], ISM has become a methodology and a useful decision support tool. ISM assists in the consideration of a set of relational elements that is composed of individual and group perceptions, ordinarily through idea-generation support tools, such as affinity diagrams [112], the KJ method [113, 114], brainstorming methods [115], and persuasive writing methods [116]. The relational elements of the set are connected by contextual relations.

In executive decision making, the different contexts of organisational problems often come from different areas of the organisations, and the given organisational problems can be a set of elements. The element set is selected as a possible statement of a relationship, represented as a relational digraph. This relational digraph can be simplified and restructured for constructing a hierarchically restructured digraph based on the mathematical process underlying the ISM concepts [117, 118]. Hence, ISM can be an

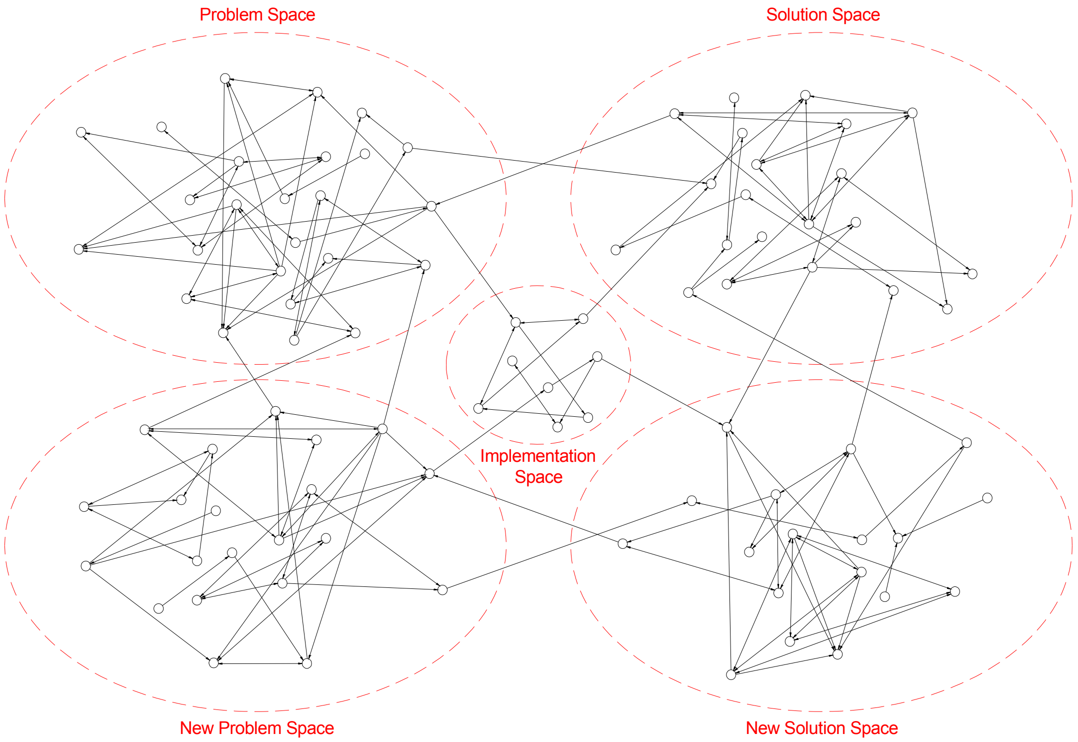


Figure 6.1. Example of an idea or problem digraph in different spaces.

important part of the problem-solving process when the same roles or problems have conceptualising interactions among the contextual elements in different organisations [119-121].

The complex issues of the contextual relations among the element set in the ISM process are represented as a digraph in order to construct a hierarchically restructured digraph to develop an understanding of complex situations. However, with a large number of elements with relational directions, which are complicated and interconnected according to the elements' contextual relations, properly constructing a hierarchically restructured digraph in a computational complexity becomes extremely difficult. Moreover, while collecting contextual issues in the real world, the number of elements and their directions increase significantly. Figure 6.1 shows an example of a digraph that is composed of different spaces.

Constructing a hierarchically restructured digraph with many relational elements is problematic, because the calculation time and the number of relational elements increase at the same time. In the first digraph shown, if there are a number of elements that connect to each other with many complicated relations, corresponding to arcs for a digraph representation, then the number of arcs is obviously huge.

The main concept of the ISM process is how to reduce as many connected arc crossings possible, meaning the crossing minimisation problem should be considered to construct a properly simplified hierarchical digraph. This requires placing the elements to minimise the crossings. Thus, the main problem of minimising arc crossings in a digraph is the combinatorial problem of selecting appropriate element ordering for each level. However, arc crossing minimisation with a large number of elements in a digraph and computational complexity is an NP-complete problem [122-124], which still remains difficult in constructing a hierarchically restructured digraph.

6.3 Interpretive Structural Modelling

In the organisational or administrative world, decision makers may have unresolved problems and always confront various types of unpredictable new issues. As a matter of fact, most decision makers are afraid of making major role decisions, and even minor irritant issues do not make decision makers feel completely comfortable. A small issue sometimes or often becomes a big issue, and decision makers must take responsibility for their decisions and should be afraid of making wrong decisions, which obviously harm the decision makers' enterprises and careers.

In the real world, there is no perfect decision support tool, and it is impossible to build a perfect optimal decision support tool when dealing with complex issues and many collected ideas, because no decision support tool can perfectly dispose of catastrophic failures and can tie up unexpected assignments even from stable operations. However, we use true decision support tools to minimise these problems. One true decision support tool is an ISM method that organises the given ideas, synthesises many ideas, and provides a visual digraph to understand complex situations better. ISM connects the ideas to create a situation model without breaking any relations. Thus, ISM is exploited to deal with a higher level of the problem-solving process [125].

The ISM method is useful as an aid not only to a single decision maker but also to groups or subgroups whose perceptions regard mutual complex issues, because ideas often come from different departments or organisations, and decision makers attempt to

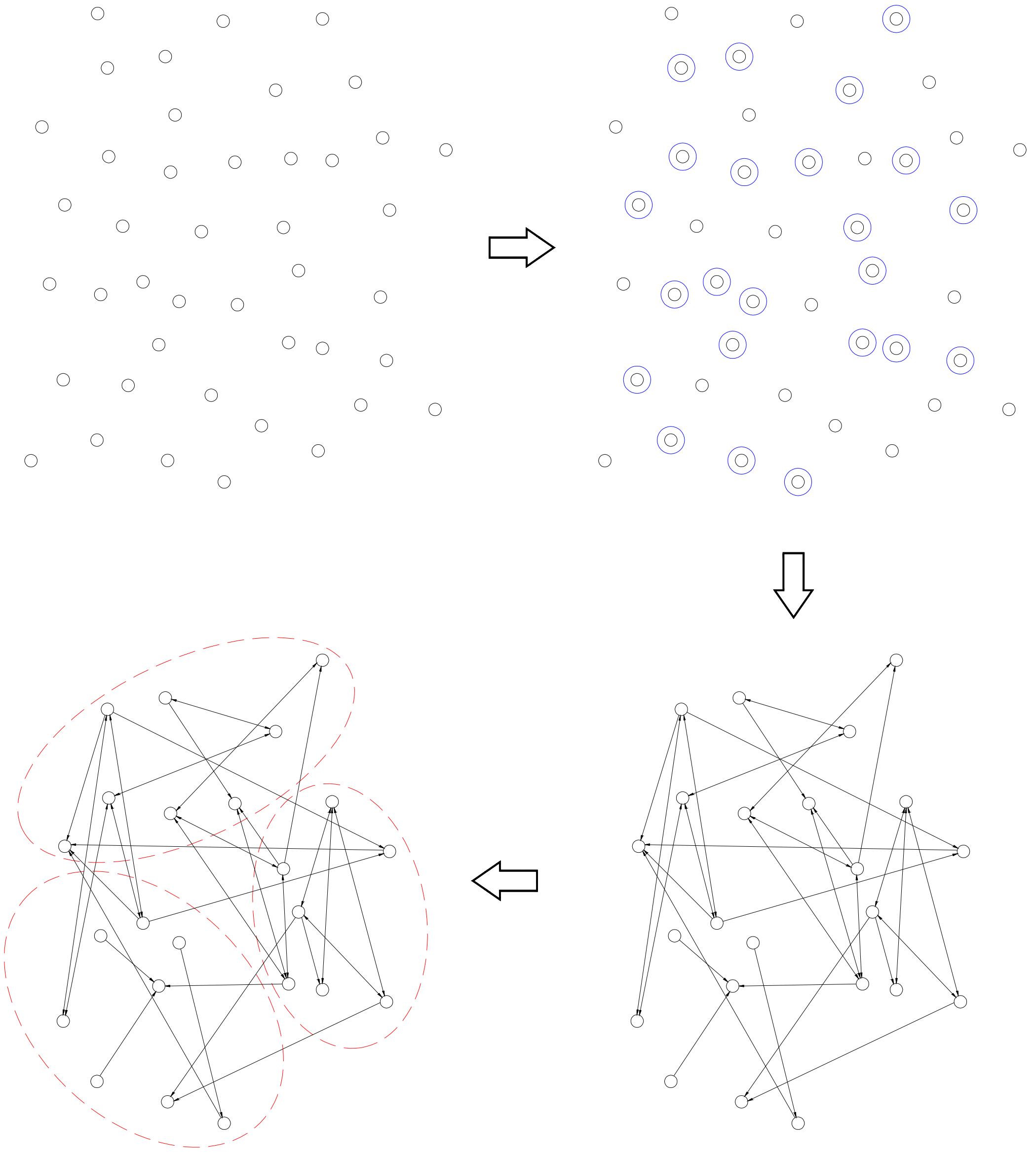


Figure 6.2. Brief representation of how to create an idea or problem digraph in a decision-making process of the ISM method: (1) generate ideas or problematic points; (2) extract and enumerate the generated ideas or problematic points; (3) make direct relations among ideas or problematic points; and (4) classify them into the similar or exactly same groups.

collect as many as ideas possible from many groups to create many samples and resolve ideas [126]. Individuals and groups use ISM to develop an understanding of complex issues. Particularly, individual or group members use ISM to add benefits to discussions and controversies based on exploiting the final result of a visually restructured digraph in different levels, called a *hierarchically restructured digraph*. All members in a group may also synthesise from the clarified and restructured result.

A comprehensive method of visually understanding a complex issue or situation is to transform the given ideas into a *pairwise-based digraph*. Each pair of the given ideas can be represented as a binary direction. If two ideas have either one direct relation or two direct relations (two parallel arrows indicate opposite directions) to each other, then those two ideas have one or two contextual relations. In contrast, two ideas have no direct relations, and then those two ideas have no contextual relations. ISM basically includes all of the given ideas without eliminating ideas, meaning any one of the ideas could conflict with each other, and those ideas can all be analysed in the spaces. Here, we can determine any possible spaces, including related ideas. For problem-solving, ordinarily two large spaces are generated: the problem space and the solution space. Sometimes three spaces are added to include the implementation space. For relational subpart analysis, any number of ideas can be classified into any number of subparts, which can also be spaces.

The number of ideas in either generating or prioritising ideas is no restriction for any subject type, since the matter can be analysed. Figure 6.2 illustrates the ordinary steps of the ISM method before starting to construct a hierarchically restructured digraph. Since an objective digraph is determined for the ISM process, the terms for using ideas and direct relations in this chapter's study are called an *element* or *element node* and an *arc*, respectively, for the digraph representation.

ISM is basically used (1) to utilise the proposed cogitations or conceptions in a systematic way to solve a complex issue or situation; and (2) to communicate the finalised cogitations or conceptions with others. The main reason for implementing a digraph is to conceptually represent a structural model of the proposed and finalised cogitations or conceptions. A digraph is composed of element nodes and arcs, and the digraph can be easily converted into a structural model and readily revised and inspected to capture the perceptions of the issue or situation. Before a hierarchically restructured digraph is determined, the entire process of constructing a digraph of complex issues or situations is often performed based on a human-machine interaction. This interaction is implemented to subjectively judge (1) whether two element nodes (two ideas) are connected by one or two parallel arcs (one or two direct relations) or not connected; and (2) which of two element nodes should be directed to the other element node by one or two parallel arcs.

6.4 ISM Representations

To graphically represent the complex issues or situations of the given ideas and direct relations, we can transform the ideas into element nodes and transform direct relations into arcs to efficiently construct a digraph based on graph theory. This digraph is a pairwise-based digraph, which is concerned with the existence of each of the direct relations between two of the element nodes in the digraph. Further, the finalised digraph representation can be transformed into a matrix representation based on the existing ideas and their direct relations in the digraph.

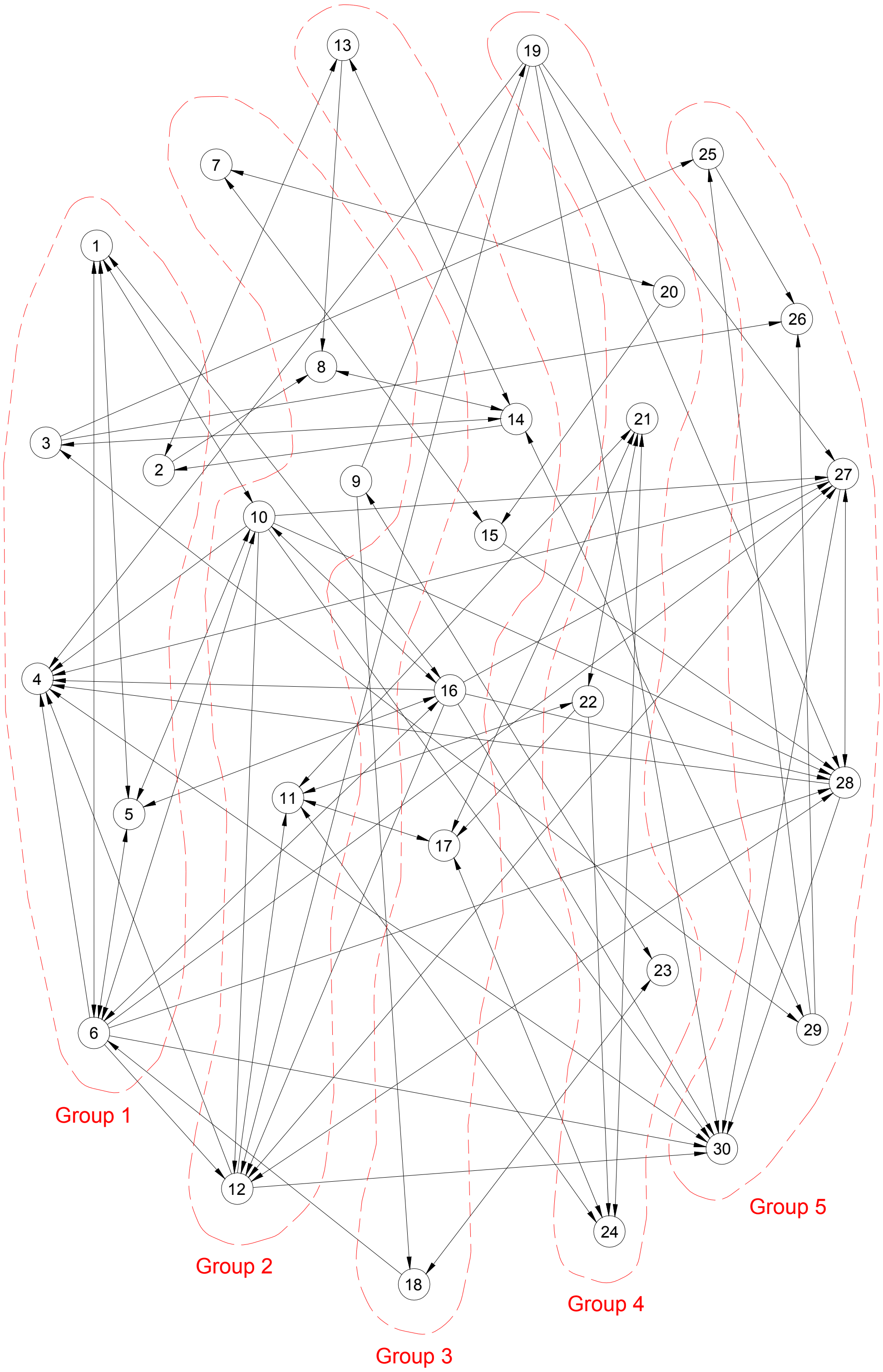


Figure 6.3. Example of a contextual digraph composed of the finalised ideas and their perceived direct relations in different groups.

$$
\begin{array}{c|cccccccccccccccccccccccccccccc|}
 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} & v_{16} & v_{17} & v_{18} & v_{19} & v_{20} & v_{21} & v_{22} & v_{23} & v_{24} & v_{25} & v_{26} & v_{27} & v_{28} & v_{29} & v_{30} \\
v_1 & 0&0&0&0&1&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
v_2 & 0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
v_3 & 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0 \\
v_4 & 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \\
v_5 & 1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
v_6 & 1&0&0&1&1&0&0&0&0&1&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0&1 \\
v_7 & 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\
v_8 & 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
v_9 & 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&0&0&0&0&0&0&0 \\
v_{10} & 1&0&0&1&1&1&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0&1 \\
v_{11} & 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&0&1&0&0&0&0&0&0 \\
v_{12} & 0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1 \\
v_{13} & 0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
v_{14} & 0&1&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0 \\
v_{15} & 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\
v_{16} & 1&0&0&1&1&1&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1 \\
v_{17} & 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0 \\
v_{18} & 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\
v_{19} & 0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1 \\
v_{20} & 0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
v_{21} & 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&0 \\
v_{22} & 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&0&0&0 \\
v_{23} & 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\
v_{24} & 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0 \\
v_{25} & 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\
v_{26} & 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
v_{27} & 0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1 \\
v_{28} & 0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1 \\
v_{29} & 0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0 \\
v_{30} & 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}
$$

Figure 6.4. Binary adjacency matrix of the example digraph.

6.4.1 Digraph representation

For the digraph representation, let us assume that n given element nodes are denoted by v_1, v_2,…, v_n, and the set of these element nodes is denoted by $V = \{v_1, v_2,…, v_n\}$. The current direct relation between any two element nodes is denoted by e, a directed line connecting those two element nodes. A direct relation between an element node v_i and an element node v_j is represented by $v_i e v_j$. In contrast, $v_i \bar{e} v_j$ represents no direct relation between two element nodes. In addition, when an element node v_i has a directional arrow (an arc) that reaches element node v_j, each of the existing arcs is represented as

$$a_l = \overrightarrow{(v_i, v_j)},\ l = 1, 2, \cdots, n \text{ for } i \neq j\ . \tag{6.1}$$

A set of these arcs from the element node v_i to the element node v_j is denoted by A, where A is composed of all the possible arcs that represent $A = \{a_1, a_2,…, a_n\}$. For an example of a contextual problem, an example digraph in Figure 6.3 has been constructed with 30 specific related ideas and the direct relations in five different groups.

6.4.2 Matrix representation

As shown in Figure 6.3, in the example digraph, the 30 specific ideas are represented by 30 element nodes that are labelled v_i, $i = 1, 2,\ldots, 30$, and the set of the 30 element nodes is represented as $V = \{v_1, v_2,\ldots, v_{30}\}$. In addition, in this example digraph, a set of numerous arcs among the given element nodes can be denoted by A. Further, the two sets in the example digraph, A and V, can be properly expressed as a matrix. For the case of the example digraph, a binary adjacency matrix (the meaning of the binary adjacency matrix was described in Chapter 4) is shown in Figure 6.4. The binary adjacency matrix of the example digraph is denoted here by **B**. In addition, **B** has rows and columns labelled as

$$t_{i,j},\, i \text{ and } j = 1, 2, \cdots, 30 \text{ for } i \neq j\,. \tag{6.2}$$

This matrix is constructed by setting $t_{i,j} = 1$ wherever there is an arc in the example digraph from an element node v_i to an element node v_j, meaning $v_i e v_j$, and by setting $t_{i,j} = 0$ elsewhere, meaning $v_i \bar{e} v_j$. This example digraph of the contextual problem will be hierarchically restructured by using the hierarchical DNA-based algorithm.

6.5 Crossing Minimisation Methods

Several heuristic methods have been proposed to reduce and minimise crossings among element nodes in an undirected graph and a digraph. In this section, we generally define and explain crossing minimisation problems and methods. The different types of methods [124] are explained below and are helpful in showing the hierarchical DNA-based algorithm for ISM.

6.5.1 Level-by-level sweep method

To reduce crossings among element nodes between two different levels, the level-by-level sweep method can be used as one method [127, 128]. To describe the method, an element node should be first ordered at level 1, denoted by T_1. Next, the element node ordering of level $T_{i\text{-}1}$ should be held and fixed while the element nodes are recorded in level T_i for $i = 2, 3,\ldots, h$. Here, the crossings between $T_{i\text{-}1}$ and T_i are reduced to deal with the two-level crossing minimisation.

Let us consider a bipartite graph as an undirected graph $G = (T_1, T_2, E_s)$, where T_1 and T_2 are two different levels, each of which includes element nodes, and E_s is a set of edges connected between T_1 and T_2. Each specific element node is denoted by v_α and represented as $v_\alpha \in T_i$, $i = 1, 2$, and is given by a unique ξ-coordinate $\xi_i(v_\alpha)$ in order to specify element node orderings for T_1 and T_2. Here, both ξ_1 and ξ_2 specify the number of crossings among element nodes in G, denoted by *crossing*(G, ξ_1, ξ_2). We obtain the minimum number of crossings in ordering the element nodes in T_1, denoted by *optimum*(G, ξ_1), representing

$$optimum(G, \xi_1) = \min_{\xi_2} crossing(G, \xi_1, \xi_2)\,. \tag{6.3}$$

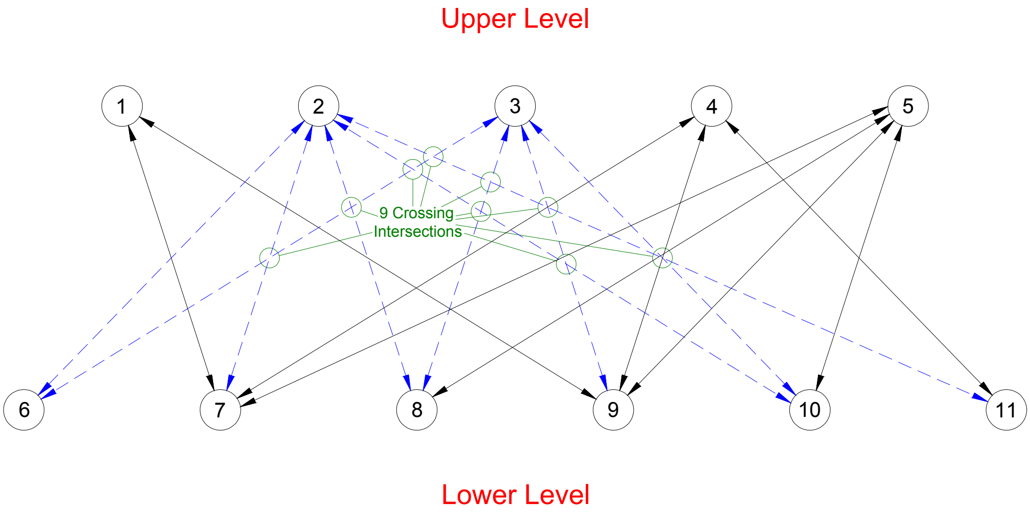


Figure 6.5. Example of the bipartite digraph composed of both upper level and lower level, in which nine crossing intersections between the element node 2 and the element node 3 with their arcs are shown.

The undirected bipartite graph $G = (T_1, T_2, E_s)$ and the permutation ξ_1 of T_1 are given to find the permutation ξ_2 of T_2, minimising the given crossings among the given element nodes, such that *crossing*(G, ξ_1, ξ_2) is equal to *optimum*(G, ξ_1).

For each pair of element nodes, represented as v_α, $v_\beta \in T_2$, the crossing number is denoted by $c_{\alpha,\beta}$ and defined as the number of edge crossings between v_α and v_β, when $\xi_2(v_\alpha) < \xi_2(v_\beta)$. Further, $c_{\alpha,\alpha} = 0$ is also defined for all $v_\alpha \in T_2$. Figure 6.5 shows an example of the bipartite digraph, composed of two levels. To compute *crossing*(G, ξ_1, ξ_2) and give a simple lower bound for *optimum*(G, ξ_1), we note the following lemma.

Lemma 6.1. *If an undirected bipartite graph $G = (T_1, T_2, E_s)$ is composed of two levels, ξ_1 is an ordering of T_1, and ξ_2 is an ordering of T_2, then*

$$crossing(G, \xi_1, \xi_2) = \sum_{\xi_2(v_\alpha) < \xi_2(v_\beta)} c_{\alpha,\beta} = \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} c_{i,j} \tag{6.4}$$

and

$$optimum(G, \xi_1) \geq \sum_{v_\alpha, v_\beta} \min(c_{\alpha,\beta}, c_{\beta,\alpha}), \tag{6.5}$$

where all unordered pairs of the element nodes correspond to the sum in the top level.

Proof. Each one of all the orderings of T_2 includes an optimal ordering, has either $\xi_2(v_\alpha)$

$< \xi_2(v_\beta)$ or $\xi_2(v_\beta) < \xi_2(v_\alpha)$, and is noted for equality (6.4) and inequality (6.5). ■

To solve the two-level crossing problem, we discussed one heuristic method above, which was the level-by-level sweep method used to reduce crossings among element nodes.

6.5.2 Integer programming method

Similar to the level-by-level sweep method, an integer programming method is also used for the two-level crossing problem. The integer programming method reduces crossings among element nodes between two different levels [129, 130]. To describe this method, we recall that an undirected bipartite graph has been denoted as $G = (T_1, T_2, E_s)$ above, and a binary vector is newly defined with ξ that can be an element of either 0 or 1.

The binary vector contains each possible entry, which is denoted by $\xi_{\alpha,\beta}$ for $v_\alpha < v_\beta$ and $\xi_{\alpha,\beta} = 1$; otherwise, $\xi_{\alpha,\beta} = 0$ if v_α is located on the left side of v_β. The number of crossings among the element nodes can be deduced based on Lemma 6.1 as follows:

$$\begin{aligned} crossing(G, \xi_1, \xi_2) &= \sum_{v_\alpha < v_\beta \in T_2} (c_{\alpha,\beta}\xi_{\alpha,\beta} + c_{\beta,\alpha}(1-\xi_{\alpha,\beta})) \\ &= \sum_{v_\alpha < v_\beta \in T_2} (c_{\alpha,\beta}\xi_{\alpha,\beta} + c_{\beta,\alpha} - c_{\beta,\alpha}\xi_{\alpha,\beta}) \\ &= \sum_{v_\alpha < v_\beta \in T_2} c_{\alpha,\beta}\xi_{\alpha,\beta} - \sum_{v_\alpha < v_\beta \in T_2} c_{\beta,\alpha}\xi_{\alpha,\beta} + \sum_{v_\alpha < v_\beta \in T_2} c_{\beta,\alpha} \\ &= \sum_{v_\alpha < v_\beta \in T_2} (c_{\alpha,\beta} - c_{\beta,\alpha})\xi_{\alpha,\beta} + \sum_{v_\alpha < v_\beta \in T_2} c_{\beta,\alpha}, \end{aligned} \tag{6.6}$$

in which the constant exists; thus, the two-level crossing problem is stated again as follows:

$$\text{minimise } \rho = \sum_{v_\alpha < v_\beta \in T_2} (c_{\alpha,\beta} - c_{\beta,\alpha})\xi_{\alpha,\beta} \tag{6.7}$$

$$\begin{aligned} &\text{subject to (1) } 0 \le \xi_{\alpha,\beta} + \xi_{\beta,\gamma} - \xi_{\alpha,\gamma} \le 1 \text{ for all } v_\alpha < v_\beta < v_\gamma \text{ of distinct} \\ &\quad \text{element nodes in } T_2\text{; and (2) } \xi_{\alpha,\beta} \in \{0, 1\} \text{ for all pairs } v_\alpha < \\ &\quad v_\beta \text{ of distinct element nodes in } T_2. \end{aligned} \tag{6.8}$$

The optimal value of ρ is represented as ρ^*, and the cost function of this optimal value is not the same as the number of crossings that should be minimised.

The integer programming method could be successfully used for small undirected bipartite graphs, but there is still no guarantee they will be terminated in polynomial time.

6.5.3 Crossing minimisation problem in dense two-level graphs

For dense two-level graphs, a crossing problem should also be considered to minimise crossings among element nodes [131]. Again, both *crossing*(G, ξ_1, ξ_2) and *optimum*(G, ξ_1) are given to describe the problem and show that *crossing*(G, ξ_1, ξ_2) should be close to *optimum*(G, ξ_1) for any orderings of T_2.

If both v_α and v_β include many common neighbours, then both $c_{\alpha,\beta}$ and $c_{\beta,\alpha}$ are obviously large numbers. Hence, the degree of an element node v_γ is denoted by *degree*(v_γ), and the number of the existing common neighbours of both v_α and v_β is denoted by $\varsigma_{\alpha,\beta}$, which is related to each pair of element nodes.

Lemma 6.2. *If an element node v_α and an element node v_β are included in T_2, then we have* (1) $c_{\alpha,\beta} + c_{\beta,\alpha} + \varsigma_{\alpha,\beta} = degree(v_\alpha) \times degree(v_\beta)$; (2) $c_{\alpha,\beta} \geq \binom{\varsigma_{\alpha,\beta}}{2}$; *and* (3) $c_{\alpha,\beta} \leq degree(v_\alpha) \times degree(v_\beta) - \binom{\varsigma_{\alpha,\beta}+1}{2}$.

For dense two-level graphs, we derive the maximum number of crossings among the element nodes, and this maximum number is basically close to the minimum number of crossings.

Theorem 6.1. *We suppose* (1) *an undirected bipartite graph G = (T_1, T_2, E_s) is a two-level graph*; (2) $|T_1| = |T_2| = n$; *and* (3) $|E_s| = \psi n^2$. *Then, we represent*

$$\lim_{\psi \to 1} \frac{\max\limits_{\xi_2} crossing(G,\, \xi_1,\, \xi_2)}{optimum(G,\, \xi_1)} = 1. \tag{6.9}$$

Proof. The notation $c_{\alpha,\beta} \geq \binom{\varsigma_{\alpha,\beta}}{2}$ is considered for the above theorem. Both $c_{\alpha,\beta}$ and $c_{\beta,\alpha}$ are obviously large numbers, since $\varsigma_{\alpha,\beta}$ is equal to $\varsigma_{\beta,\alpha}$. From Lemma 6.1, both *crossing*(G, ξ_1, ξ_2) and *optimum*(G, ξ_1) are used to represent another notation as follows:

$$crossing(G,\, \xi_1,\, \xi_2) - optimum(G,\, \xi_1) \leq \sum_{\xi_2(v_\alpha) < \xi_2(v_\beta)} |\, c_{\alpha,\beta} - c_{\beta,\alpha} \,|. \tag{6.10}$$

From Lemma 6.2, a new notation is also represented as

$$|\, c_{\alpha,\beta} - c_{\beta,\alpha} \,| \leq degree(v_\alpha) \times degree(v_\beta) - \binom{\varsigma_{\alpha,\beta}+1}{2} - \binom{\varsigma_{\alpha,\beta}}{2}. \tag{6.11}$$

Thus, we obtain

$$\sum_{\xi_2(v_\alpha) < \xi_2(v_\beta)} (degree(v_\alpha) \times degree(v_\beta) - \varsigma_{\alpha,\beta}^2)$$
$$\geq crossing(G, \xi_1, \xi_2) - optimum(G, \xi_1), \tag{6.12}$$

since

$$\binom{\varsigma_{\alpha,\beta} + 1}{2} + \binom{\varsigma_{\alpha,\beta}}{2} = \varsigma_{\alpha,\beta}^2 . \tag{6.13}$$

In addition, we obtain both an upper bound that is represented as

$$\sum_{\xi_2(v_\alpha) < \xi_2(v_\beta)} degree(v_\alpha) \times degree(v_\beta) \leq \frac{|E_s|^2}{2} = \frac{\psi^2 n^4}{2}, \tag{6.14}$$

and a lower bound that is represented as two equations

$$\sum_{\xi_2(v_\alpha) < \xi_2(v_\beta)} \varsigma_{\alpha,\beta} = \sum_{v_\gamma \in T_1} \binom{degree(v_\gamma)}{2} \tag{6.15}$$

and

$$\sum_{v_\gamma \in T_1} degree(v_\gamma) = |E_s| = \psi n^2 . \tag{6.16}$$

These two equations derive the notation, representing

$$\sum_{v_\gamma \in T_1} \binom{degree(v_\gamma)}{2} \geq n \binom{\psi n}{2}. \tag{6.17}$$

Both (6.15) and (6.17) are deduced as

$$\sum_{\xi_2(v_\alpha) < \xi_2(v_\beta)} \varsigma_{\alpha,\beta}^2 \geq \binom{n}{2} \left(\frac{n\binom{\psi n}{2}}{\binom{n}{2}} \right)^2 = \frac{\psi^2 n^3 (\psi n - 1)^2}{2(n-1)} . \tag{6.18}$$

From (6.12), (6.14), and (6.18), we have

$$crossing(G, \xi_1, \xi_2) - optimum(G, \xi_1) \le \frac{\psi^2 n^4}{2} - \frac{\psi^2 n^3 (\psi n - 1)^2}{2(n-1)}, \tag{6.19}$$

which becomes

$$crossing(G, \xi_1, \xi_2) - optimum(G, \xi_1) = \psi^2 n^2 \left(\psi n - \frac{1}{2} \right), \tag{6.20}$$

and similarly shows

$$optimum(G, \xi_1) \ge \frac{\psi^2 n^4}{2} - \mathrm{O}(n^3). \tag{6.21}$$

Finally, from both (6.20) and (6.21), the theorem follows the deduced O(ψ / n), which is ($crossing(G, \xi_1, \xi_2) - optimum(G, \xi_1)$) / $optimum(G, \xi_1)$. ■

The above described problems and methods deal only with two levels, meaning more levels become more intractable in an undirected graph and a dense two-level graph, but the two-level crossing problem, corresponding to the crossing minimisation problem, is an NP-complete problem [132].

6.6 Mathematical ISM Method

A mathematical ISM method has been used in various significant works [133-135] as an approach for constructing a hierarchically restructured digraph. In a digraph, an element node v_j is said to be reachable from an element node v_i if a direct path from v_i to v_j is an arc. The length of the path corresponds to the number of such arcs. Therefore, the binary adjacency matrix **B** shows a reachability example in a path of length 1. To obtain a reachability matrix, the identity matrix **I** is added to **B**, expressed as **R** = **B** + **I**, where **R** is the starting point to reach the reachability matrix, and the resulting matrix is raised to successive powers based on *Boolean algebra* [136] until no new entries are obtained. Hence, each successive multiplication preserves the entries of the previous power, and matrix equality or inequality can be determined based on an entry-by-entry comparison, which can be expressed as follows:

$$\mathbf{B} \ne \mathbf{B} + \mathbf{I} \ne (\mathbf{B} + \mathbf{I})^2 \ne \cdots \ne (\mathbf{B} + \mathbf{I})^{m-1} = (\mathbf{B} + \mathbf{I})^m = \mathbf{R}. \tag{6.22}$$

When (6.22) is satisfied, **R** is called the *reachability matrix*. In the reachability matrix, the path length becomes n-1 when at most n element nodes exist, because all of the ISM mathematical operations are Boolean. Further, the finalised reachability matrix is used to construct a particular digraph, called a *reachability digraph*.

The utility of the reachability matrix should be used to clearly develop a hierarchical restructuring of the digraph. In the reachability matrix, the entries labelled by rows and columns are denoted by $r_{i,j}$. Hence, for each element node v_i in V, two sets can be defined from the reachability matrix **R** as follows:

$$R_s(i) = \{v_j \in V \mid r_{i,j} = 1\}\,, \tag{6.23}$$

$$A_s(i) = \{v_j \in V \mid r_{j,i} = 1\}\,, \tag{6.24}$$

where $R_s(i)$ is a reachability number set of element nodes that should be reachable from v_i, and $A_s(i)$ is an antecedent number set of element nodes that should be able to reach v_i. In other words, in the reachability matrix **R**, there is a set of element node numbers $R_s(i)$ whose columns have an entry of 1 in row i, and a set of element node numbers $A_s(i)$ whose rows have an entry of 1 in column j. Here, a set of the intersections of $R_s(i)$ and $A_s(i)$ is defined as follows:

$$Z = \{v_i \in V \mid R_s(i) \cap A_s(i) = R_s(i)\}\,. \tag{6.25}$$

Two different arbitrary element nodes v_α and v_β are included in the same subdigraph if

$$R_s(\alpha) \cap R_s(\beta) \neq \phi\,; \tag{6.26}$$

otherwise, they are partitioned into different subdigraphs

$$R_s(\alpha) \cap R_s(\beta) = \phi\,. \tag{6.27}$$

In a mathematical ISM method, it is necessary to examine all given element nodes to determine whether they are partitioned or not based on (6.26) and (6.27). Figure 6.6 illustrates a mathematical ISM method flowchart.

For a mathematical ISM procedure, the steps of constructing one or more hierarchically restructured digraphs are as follows:

Step 1: Collect, classify, resolve, and enumerate the finalised ideas, representing element nodes using idea-generation support tools.

Step 2: A digraph can be created based on the given ideas and their direct relations, created by the contextual binary directions of individual or group members. Here, the binary directions correspond to binary numbers.

Step 3: A binary adjacency matrix can be created by transforming the digraph that is composed of element nodes and arcs. Then the reachability matrix can be calculated with (6.22), and the intersection set Z with (6.23), (6.24), and (6.25).

A particular set of element nodes that belongs to this intersection set Z is shown in (6.25). This particular set is obtained for division into hierarchical levels, and obtained for constructing (1) a particular matrix, called a *structural matrix*; and (2) a particular digraph, called a *structural digraph*, in the following steps:

Step 4: Examine all the element nodes in the set Z to determine whether they are composed of a single digraph or partitioned in different digraphs based on both (6.26) and

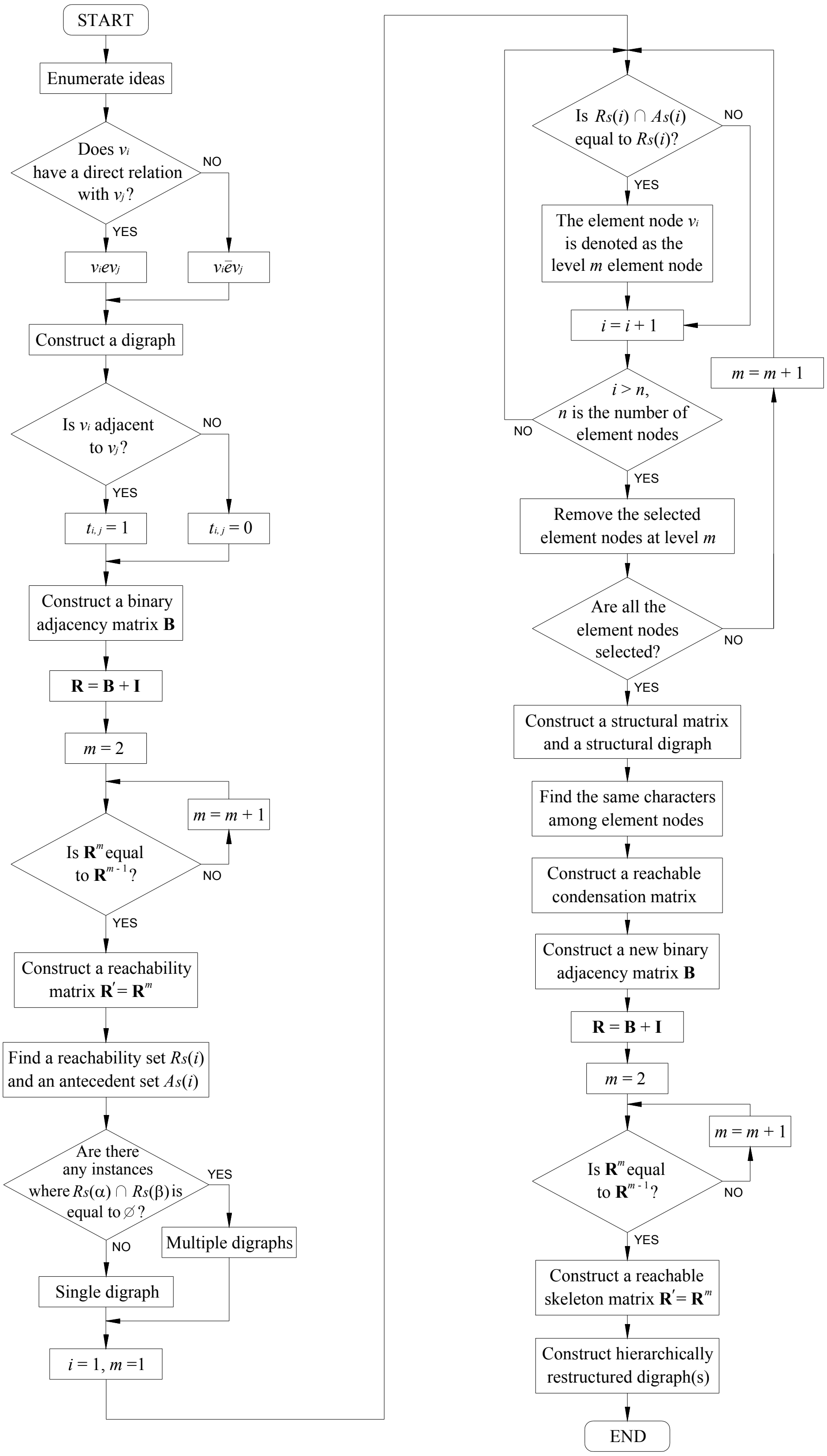

START
Enumerate ideas
Does v_i have a direct relation with v_j?
NO
YES
$v_i e v_j$
$v_i \bar{e} v_j$
Construct a digraph
Is v_i adjacent to v_j?
NO
YES
$t_{i,j} = 1$
$t_{i,j} = 0$
Construct a binary adjacency matrix **B**
R = **B** + **I**
$m = 2$
$m = m + 1$
Is $\mathbf{R}^m$ equal to $\mathbf{R}^{m-1}$?
NO
YES
Construct a reachability matrix $\mathbf{R}' = \mathbf{R}^m$
Find a reachability set $Rs(i)$ and an antecedent set $As(i)$
Are there any instances where $Rs(\alpha) \cap Rs(\beta)$ is equal to $\varnothing$?
YES
NO
Multiple digraphs
Single digraph
$i = 1, m = 1$
Is $Rs(i) \cap As(i)$ equal to $Rs(i)$?
NO
YES
The element node v_i is denoted as the level m element node
$i = i + 1$
$m = m + 1$
$i > n$, n is the number of element nodes
NO
YES
Remove the selected element nodes at level m
Are all the element nodes selected?
NO
YES
Construct a structural matrix and a structural digraph
Find the same characters among element nodes
Construct a reachable condensation matrix
Construct a new binary adjacency matrix **B**
R = **B** + **I**
$m = 2$
$m = m + 1$
Is $\mathbf{R}^m$ equal to $\mathbf{R}^{m-1}$?
NO
YES
Construct a reachable skeleton matrix $\mathbf{R}' = \mathbf{R}^m$
Construct hierarchically restructured digraph(s)
END

Figure 6.6. Flowchart of a mathematical ISM method.

(6.27).

Step 5: All the element nodes that satisfy (6.25) are selected, and all of these selected nodes are denoted as level 1 element nodes.

Step 6: The selected element nodes at level 1 should be removed, and the selection of element nodes that satisfy (6.25) should be repeated; these selected element nodes are denoted as level 2 element nodes.

Step 7: Similarly, the selected element nodes should be removed, and the element nodes should be selected again, until all the element nodes have been removed. Additionally, when all the element nodes disappear in n repetitions, those element nodes are observed to be divided into an n-level hierarchy.

Step 8: Both a structural matrix and a structural digraph can be constructed by representing each level of all divided element nodes.

Several more steps should be processed using complicated mathematical operations to finally construct a hierarchically restructured digraph using both the structural matrix and the structural digraph obtained from the above steps, because the structural digraph with the n-level of hierarchy is difficult to understand, even if the digraph contains a small number of element nodes or arcs. The remaining steps for constructing a hierarchically restructured digraph are the following:

Step 9: Transform the structural matrix and the structural digraph to construct a particular matrix, called a *reachable condensation matrix*, which can be determined by reducing specific element nodes that contain the exact same characters and are adjacent to each other.

Step 10: Transform the reachable condensation matrix to again construct a particular matrix, called a *reachable skeleton matrix*, which can be determined by calculating the reachable condensation matrix with Boolean algebra until no new entries are obtained.

Step 11: Construct one or more hierarchically restructured digraphs, which are represented by the finalised reachable skeleton matrix.

6.7 DNA-Based ISM Method

In this chapter's study, the hierarchical DNA-based algorithm is used to construct a hierarchically restructured digraph, which is the main purpose of the calculation in the ISM process. The hierarchical DNA-based algorithm is basically executed by using DNA oligonucleotides to achieve DNA computation. An ISM method using the hierarchical DNA-based algorithm is abbreviated as a *DNA-based ISM method*.

The first step in designing the hierarchical DNA-based algorithm is to encode the DNA sequences based on the binary adjacency matrix that is obtained from the digraph. After that, several kinds of molecular engineering techniques that use the encoded DNA sequences can be used to implement the hierarchical DNA-based algorithm to construct

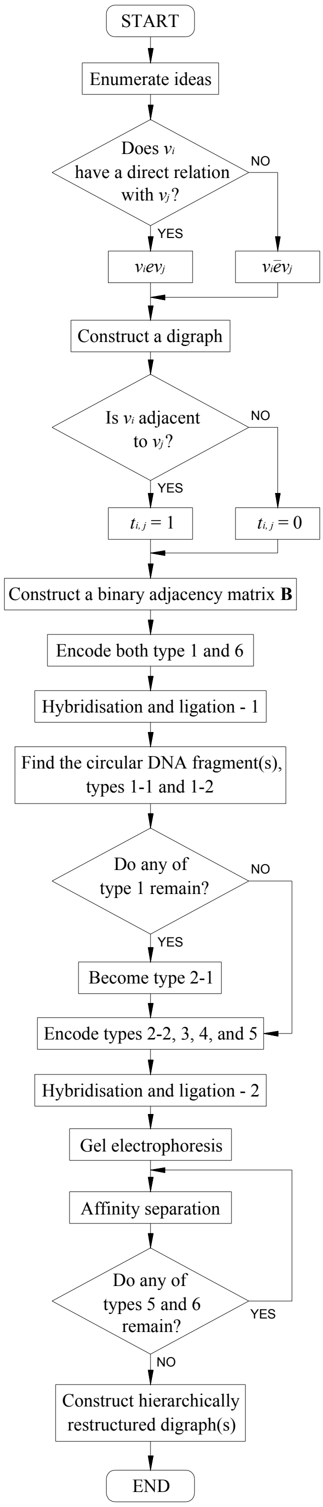

START
Enumerate ideas
Does v_i have a direct relation with v_j?
NO
YES
$v_i e v_j$
$v_i \bar{e} v_j$
Construct a digraph
Is v_i adjacent to v_j?
NO
YES
$t_{i,j} = 1$
$t_{i,j} = 0$
Construct a binary adjacency matrix **B**
Encode both type 1 and 6
Hybridisation and ligation - 1
Find the circular DNA fragment(s), types 1-1 and 1-2
Do any of type 1 remain?
NO
YES
Become type 2-1
Encode types 2-2, 3, 4, and 5
Hybridisation and ligation - 2
Gel electrophoresis
Affinity separation
Do any of types 5 and 6 remain?
YES
NO
Construct hierarchically restructured digraph(s)
END

Figure 6.7. Flowchart of a DNA-based ISM method.

one or more hierarchically restructured digraphs.

A DNA-based ISM method flowchart is shown in Figure 6.7. For the DNA-based ISM procedure, the method for constructing one or more hierarchically restructured digraphs is shown in the following steps:

Step 1: The finalised ideas, representing element nodes, are collected, classified, resolved, and enumerated by idea-generation support tools.

Step 2: A pairwise comparison evaluates the direct relations among the given ideas. In that case, the contextual binary directions of individual or group members are shown as binary numbers, and a digraph can be created.

Step 3: A binary adjacency matrix can be created by transforming the digraph, and the DNA sequences for the hierarchical DNA algorithm can be encoded based on the binary adjacency matrix.

Step 4: Several molecular engineering techniques, including hybridisation and ligation, a simulated gel electrophoresis apparatus, polymerase chain reaction (PCR), and affinity separation, are used to distinguish the encoded DNA sequences, measure the length of the DNA strands, amplify the DNA strands, and separate the DNA strands into each hierarchical level in each hierarchically restructured digraph.

Step 5: One or more hierarchically restructured digraphs can be constructed based on the selected DNA strands that have been classified into each level by the simulated experimental studies and results.

6.8 Encoding Element Nodes in DNA

In the finalised digraph, each of the existing element nodes and each of the existing arcs are encoded in the best way for ISM to exploit the binary adjacency matrix **B**, as shown in Figure 6.4, which can also be called a *directional matrix* in the term of the DNA-based ISM method. In a DNA-based process, unlike the mathematical ISM process, it is not necessary to raise the directional matrix to successive powers, based on Boolean algebra, to become a reachability matrix, a structural matrix, a reachable condensation matrix, and a reachable skeleton matrix. Instead, a systematic way of ordering rows and columns of the directional matrix can be transformed into DNA sequences in contextual relations.

The directional matrix of the example digraph has been created and is shown in Figure 6.4. This matrix of $n \times n$ has 30 rows and 30 columns, and it has a set of element nodes $V = \{v_1, v_2, \ldots, v_{30}\}$. Each directional order of two element nodes has its own row and column, and the rows and columns are labelled $t_{i,j}$, i and $j = 1, 2, \ldots, 30$ in the directional matrix. Recall that if there is an arc from an element node v_i to an element node v_j, then $t_{i,j} = 1$; otherwise, $t_{i,j} = 0$. To construct an initial library of DNA fragments for the ISM process, eight different types are created. Each of the eight different types has its own row and column labels, which are defined for encoding element nodes in single-stranded DNA (ssDNA).

6.8.1 Strong components

Type 1 represents two element nodes, in which the direction of the arrow indicates the direction from the element node v_i to the element node v_j. For the example digraph, type 1 can be encoded from the given DNA substrings, shown in Table 6.1. Type 1 represents a double-encoded substring (the meaning of the double-encoded substring was described in Chapter 4), which is each of all two different single element nodes in the digraph. Type 1 is basically encoded for detecting types 1-1 and 1-2. Here, type 1-1 is called *strong chain components*, and type 1-2 is called *strong cyclic components*. In addition, type 1 is encoded for adapting the non-components to type 2-1 after detecting types 1-1 and 1-2. This means that the remaining encoded DNA substrings of type 1 will be used for type 2-1. Between the element node v_i and the element node v_j, there could be either a single direction or a set of two parallel arrows that indicate opposite directions. For type 1, all of the row and column labels in a directional matrix are denoted by i and j for detecting types 1-1 and 1-2, and the entries are defined as

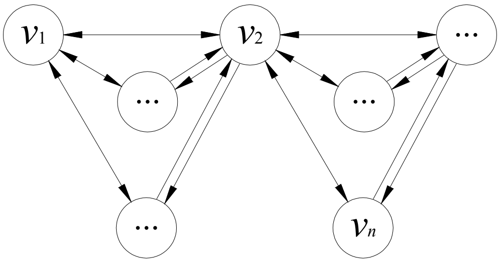
$$t_{i,j} = 1 \text{ for } i \text{ and } j = 1, 2, \cdots, n, i \neq j, \text{ and all } (i, j) \in E\,, \tag{6.28}$$

where E is the set of all possible row and column labels in direct relations. Each of the different element nodes is encoded as an oligonucleotide consisting of two unique sites. Thus, v_i was set with the length of 25 base pairs (bp: after hybridisations and end-filling DNA), and v_j with the length of 25 bp. After the hybridisation and ligation process of all the double-encoded substrings and their complementary substrings, if one or more circular DNA fragments are found, the circular DNA fragments, corresponding to either type 1-1 or 1-2, will be encoded again.

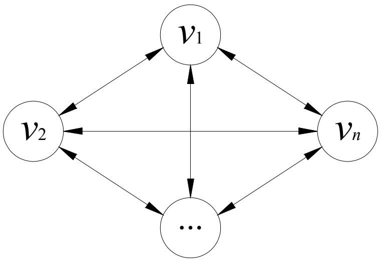
As shown in Figure 6.8, type 1-1 refers to strong chain components that were created for encoding types 2-2 or 3 or 4. Types 2-2, 3, and 4 are either double (type 2-2) or half (types 3 and 4) of a double-encoded substring. In a digraph, type 1-1 represents a mutually connected subset of two or more element nodes, in which any of its element nodes is mutually connected to any other element nodes within a set of two parallel arrows that indicate opposite directions. For type 1-1, the row labels are denoted by both p and y, and the column labels are denoted by both x and q for encoding element nodes in DNA. The entries are defined as

$$\begin{aligned} &t_{p,x} = t_{y,q} = 1 \text{ for } \exists p \in N_s, x \in X, y \in Y, q \in Q, \\ &p \neq x, y \neq q, p = q, x = y, \text{ and all } (y, q) \in Eb\,, \end{aligned} \tag{6.29}$$

where (1) N_s is a subset of a set N that includes all the given element node numbers in a digraph, representing $N_s \subseteq N$. Here, if $N_s = N$, then all element nodes are mutually connected to the others in the digraph; (2) Y is equal to X, X is a proper subset of N_s, and Q is a subset of N_s, meaning $Y = X$, $X \subset N_s$, and $Q \subseteq N_s$; and (3) Eb is a mutually connected subset of all row and column labels in the set E. In addition, Eb includes some of the row labels and column labels that should be included in this mutually connected subset. In addition, a subset of arcs for type 1-1 in A is denoted by $A_{1\text{-}1}$, which is defined as



(a)



(b)

Figure 6.8. DNA encoding types (a) and (b) of strong chain components.

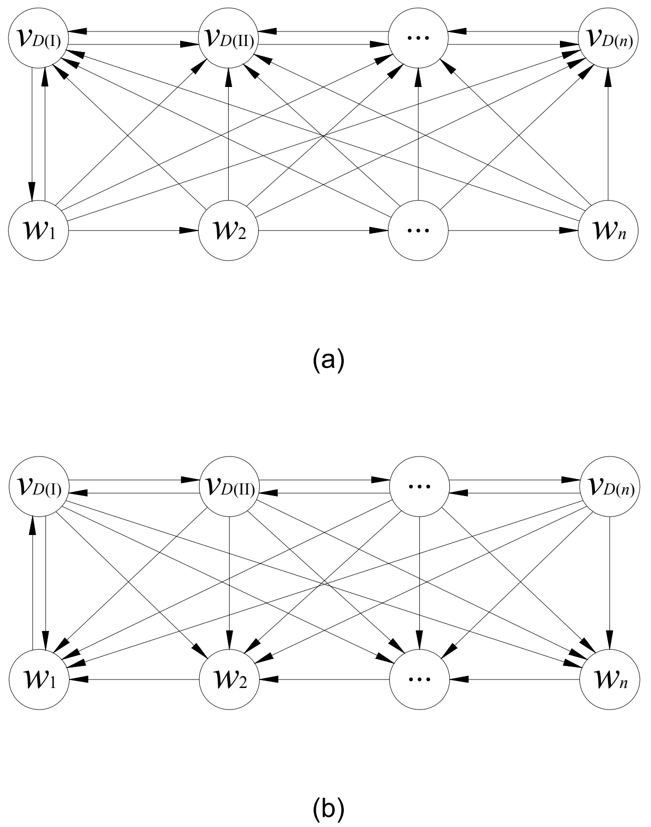
$$A_{1-1} = \left\{ \overrightarrow{(v_y, v_q)} \in A \mid y \in Y,\ q \in Q,\ y \neq q, \text{ and } v_y e v_q \right\}. \tag{6.30}$$

Hence, for type 1-1, a set of each element node v_y in this subset, where the element node number y is transformed into the subset of element node numbers $D(y)$, which is defined as

$$D(y) = \{v_q \in V \mid t_{p,x} = t_{y,q} = 1\}\,. \tag{6.31}$$

Thus, for type 1-1, the element node subset is denoted by $v_{D(y)}$, which was set with the length of 25 bp, where y = 1, 2,…, n and each new sequentially obtained subset of the row labels y in the directional matrix will be independently re-denoted by I, II,…, n, each of which represents a subset of element node numbers. These element nodes are encoded as an oligonucleotide consisting of one unique site. Here, type 1-1 should be encoded with type 2-2 or 3 or 4.

At the same time, if there are any element node numbers that are not included in the subset of element node numbers $D(y)$, y = I, II,…, n and those element nodes consistently have directed cycles either from or to the element node subset $v_{D(y)}$, then the element nodes should be included in the element node subset $v_{D(y)}$ and become type 1-2. As



(a)

(b)

Figure 6.9. DNA encoding types (a) and (b) of strong cyclic components.

shown in Figure 6.9, the paths between the element nodes and the element node subset $v_{D(y)}$ are strong cyclic components. For these directed cycles, by excluding the element nodes in the element node subset $v_{D(y)}$, let us denote Ec, which is a cyclic directed subset of all row and column labels in the set E, and A_c, which is a subset of arcs in a set A. Finally, we denote a subset of element node numbers $C(y)$ that is transformed by y, where y is the number of each element node w_y, y = 1, 2,…, n, each of which has directed cycles either from or to the element node subset $v_{D(y)}$. Here, a subset of all of the row and column labels for type 1-2 is denoted by Es, which contains all elements of the subsets Eb and Ec, defined as

$$Es = Eb \bigcup Ec\,. \tag{6.32}$$

A subset of all the arcs for type 1-2 is denoted by $A_{1\text{-}2}$, which contains all elements of the subsets $A_{1\text{-}1}$ and A_c, defined as

$$A_{1-2} = A_{1-1} \bigcup A_c\,. \tag{6.33}$$

In addition, a subset of all the element node numbers for type 1-2 is denoted by $G(y)$,

Table 6.1. DNA sequences in ssDNA for hybridisation and ligation-1. The underlined letters indicate the available restriction enzymes (AatII, AclI, AfeI, AflII, AgeI, ApaI, ApaLI, AseI, AvrII, BamHI, BglII, BsrBI, BseYI, BsiWI, BspHI, ClaI, EcoRI, HindIII, MluI, NcoI, PsiI, PstI, PvuI, SalI, SnaBI, SpeI, SphI, SspI, StuI, and XhoI from ©New England BioLabs, Inc.), and the small letters indicate the complementary sites of the available restriction enzymes. CS stands for complementary substring.

Substring	DNA Sequence (5' to 3')	CS	DNA Sequence (5' to 3')
v_1	GTCGACTAGTTCCCCGGTTAATGAC	L_1	GTCATTAACCGGGGAACTAGTCgacgtcATTAACCGGGGAACTAGTCGAC
v_2	GTTGCTCTCTCCGAGGAGAGGAAAC	L_2	GTTTCCTCTCCTCGGAGAGAGCaacgttTCCTCTCCTCGGAGAGAGCAAC
v_3	GCTAAAGTGAGCGATCCTCCAGAGC	L_3	GCTCTGGAGGATCGCTCACTTTagcgctCTGGAGGATCGCTCACTTTAGC
v_4	AAGTCGTTAACATATTGTTACCCTT	L_4	AAGGGTAACAATATGTTAACGActtaagGGTAACAATATGTTAACGACTT
v_5	GGTGCCCAATGACGTAGCTATAACC	L_5	GGTTATAGCTACGTCATTGGGCaccggtTATAGCTACGTCATTGGGCACC
v_6	CCCATTGTATTTTGCACAGGTGGGG	L_6	CCCCACCTGTGCAAAATACAATgggcccCACCTGTGCAAAATACAATGGG
v_7	CACTTTAGCCAACGGGTTTCAGGTG	L_7	CACCTGAAACCCGTTGGCTAAAgtgcacCTGAAACCCGTTGGCTAAAGTG
v_8	AATTCACATTTCACAGATAGGTATT	L_8	AATACCTATCTGTGAAATGTGAattaatACCTATCTGTGAAATGTGAATT
v_9	AGGTCTGGGGATCCCGGCAAGACCT	L_9	AGGTCTTGCCGGGATCCCCAGAcctaggTCTTGCCGGGATCCCCAGACCT
v_{10}	TCCTGAGGGCGTATATTTGCAGGGA	L_{10}	TCCCTGCAAATATACGCCCTCAggatccCTGCAAATATACGCCCTCAGGA
v_{11}	TCTAGGGTCCAACATAGGCGGCAGA	L_{11}	TCTGCCGCCTATGTTGGACCCTagatctGCCGCCTATGTTGGACCCTAGA
v_{12}	CTCCATAAACTACGATGGCATTCCG	L_{12}	CGGAATGCCATCGTAGTTTATGgagcggAATGCCATCGTAGTTTATGGAG
v_{13}	AGCTCCCTACTCAGACGCAGTCCCC	L_{13}	GGGGACTGCGTCTGAGTAGGGAgctgggGACTGCGTCTGAGTAGGGAGCT
v_{14}	ACGTGTAACGTACTCATCCCGGCGT	L_{14}	ACGCCGGGATGAGTACGTTACAcgtacgCCGGGATGAGTACGTTACACGT
v_{15}	TGACCAACTGATTCTCGGCAAATCA	L_{15}	TGATTTGCCGAGAATCAGTTGGtcatgaTTTGCCGAGAATCAGTTGGTCA
v_{16}	GATCAATCTACGGAGCGACAGTATC	L_{16}	GATACTGTCGCTCCGTAGATTGatcgatACTGTCGCTCCGTAGATTGATC
v_{17}	TTCCGTGATTATCAACAGCTGTGAA	L_{17}	TTCACAGCTGTTGATAATCACGgaattcACAGCTGTTGATAATCACGGAA
v_{18}	CTTTTGTCTAGCAGTTCTAAGTAAG	L_{18}	CTTACTTAGAACTGCTAGACAAaagcttACTTAGAACTGCTAGACAAAAG
v_{19}	CGTTGTCTTTTGCCATGGTCCCACG	L_{19}	CGTGGGACCATGGCAAAAGACAacgcgtGGGACCATGGCAAAAGACAACG
v_{20}	TGGTGGTAAAAGCCTCCAAGTCCCA	L_{20}	TGGGACTTGGAGGCTTTTACCAccatggGACTTGGAGGCTTTTACCACCA
v_{21}	TAATTAGATTGATCATACCTACTTA	L_{21}	TAAGTAGGTATGATCAATCTAAttataaGTAGGTATGATCAATCTAATTA
v_{22}	CAGGCGTAGCGGACTTTAGGCCCTG	L_{22}	CAGGGCCTAAAGTCCGCTACGCctgcagGGCCTAAAGTCCGCTACGCCTG
v_{23}	TCGTGTCAACCACAGTTCGGATCGA	L_{23}	TCGATCCGAACTGTGGTTGACAcgatcgATCCGAACTGTGGTTGACACGA
v_{24}	GACAAGCGGCGTACATCACTGAGTC	L_{24}	GACTCAGTGATGTACGCCGCTTgtcgacTCAGTGATGTACGCCGCTTGTC
v_{25}	GTATATTGTATGTGCAACGTCCTAC	L_{25}	GTAGGACGTTGCACATACAATAtacgtaGGACGTTGCACATACAATATAC
v_{26}	AGTTCGCCCAAGTGGCGCCATCACT	L_{26}	AGTGATGGCGCCACTTGGGCGAactagtGATGGCGCCACTTGGGCGAACT
v_{27}	TGCTAGTTCCTGTGTTAGCTCTGCA	L_{27}	TGCAGAGCTAACACAGGAACTAgcatgcAGAGCTAACACAGGAACTAGCA
v_{28}	ATTGGCAGCTCTTTGAACATGCAAT	L_{28}	ATTGCATGTTCAAAGAGCTGCCaatattGCATGTTCAAAGAGCTGCCAAT
v_{29}	CCTTCAACGGTCGAGAAGCCTCAGG	L_{29}	CCTGAGGCTTCTCGACCGTTGAaggcctGAGGCTTCTCGACCGTTGAAGG
v_{30}	GAGTCCATAGTACCTCGGATGACTC	L_{30}	GAGTCATCCGAGGTACTATGGActcgagTCATCCGAGGTACTATGGACTC

Table 6.2. DNA sequences in ssDNA for hybridisation and ligation-2. The underlined letters indicate the available restriction enzymes (AatII, AclI, AfeI, AflII, AgeI, ApaI, and ApaLI from ©New England BioLabs, Inc.), and the small letters indicate the complementary sites of the available restriction enzymes. CS stands for complementary substring.

Substring	DNA Sequence (5' to 3')	CS	DNA Sequence (5' to 3')
$v_{D\,(\mathrm{I})}$	GTCCGACCAAGTTGCGCAGGTGGAC	$L_{D\,(\mathrm{I})}$	GTCCACCTGCGCAACTTGGTCGgacgtcCACCTGCGCAACTTGGTCGGAC
$v_{D\,(\mathrm{II})}$	GTTAACTATAGCTTGAAGCTTCAAC	$L_{D\,(\mathrm{II})}$	GTTGAAGCTTCAAGCTATAGTTaacgttGAAGCTTCAAGCTATAGTTAAC
$v_{D\,(\mathrm{III})}$	GCTTGCTCTTCTTCTGATGAATAGC	$L_{D\,(\mathrm{III})}$	GCTATTCATCAGAAGAAGAGCAagcgctATTCATCAGAAGAAGAGCAAGC
$v_{D\,(\mathrm{IV})}$	AAGAGTCGTGAAGTCGATTCTTCTT	$L_{D\,(\mathrm{IV})}$	AAGAAGAATCGACTTCACGACTcttaagAAGAATCGACTTCACGACTCTT
$v_{D\,(\mathrm{V})}$	GGTAACTTCGATGGTTAAAATAACC	$L_{D\,(\mathrm{V})}$	GGTTATTTTAACCATCGAAGTTaccggtTATTTTAACCATCGAAGTTACC
$v_{D\,(\mathrm{VI})}$	CCCACATCAAAGGCTCAGAGGCGGG	$L_{D\,(\mathrm{VI})}$	CCCGCCTCTGAGCCTTTGATGTgggcccGCCTCTGAGCCTTTGATGTGGG
$v_{D\,(\mathrm{VII})}$	CACAAGTCGCCTTTCACATTCGGTG	$L_{D\,(\mathrm{VII})}$	CACCGAATGTGAAAGGCGACTTgtgcacCGAATGTGAAAGGCGACTTGTG

which contains all elements of the element node number subsets *D*(*y*) and *C*(*y*), and is defined as

$$G(y) = D(y) \bigcup C(y)\,. \tag{6.34}$$

Thus, the element node subset is denoted by $v_{G(y)}$, which was set with the length of 25 bp for type 1-2, where y = 1, 2,…, n and each new sequentially obtained subset of the row labels y are also independently re-denoted by I, II,…, n, each of which represents a subset of element node numbers. These element nodes are also encoded as an oligonucleotide consisting of one unique site. Here, type 1-2 should also be encoded with type 2-2 or 3 or 4.

6.8.2 Double-encoded substrings

First, type 2-1 comes from the remaining encoded DNA substrings of type 1, which excluded the encoded DNA substrings of both type 1-1 and 1-2, meaning that type 2-1 is a double-encoded substring of two element nodes in one direction, as shown in Figure 6.10. In other words, there are two specific element nodes for type 2-1, in which the direction of the arrow indicates the direction from the element node v_i to the element node v_j. For type 2-1, the row and column labels are denoted by i and j for encoding two element nodes in DNA, a subset of all the row and column labels is denoted by *Ed*, and the entries are defined as

$$\begin{aligned} &t_{i,j} = 1 \text{ for } i \text{ and } j = 1, 2, \cdots, n, i \neq j, \text{ and} \\ &\text{all } (i,\, j) \in E, \text{since all } (i,\, j) \notin Es \bigcup Ef \bigcup Eg \bigcup Eh\,. \end{aligned} \tag{6.35}$$

In addition, for type 2-1, a subset of arcs in A is denoted by $A_{2\text{-}1}$, which is defined as

$$\begin{aligned} A_{2-1} = \Big\{ &\overrightarrow{(v_i, v_j)} \in A \mid i \text{ and } j = 1, 2, \cdots, n, i \neq j, \\ &v_i e v_j, \text{ and all } \overrightarrow{(v_i, v_j)} \notin A_{1-2} \bigcup A_{2-2} \bigcup A_3 \bigcup A_4 \Big\}. \end{aligned} \tag{6.36}$$

For type 2-1, there are two specific element nodes that are encoded as an oligonucleo-

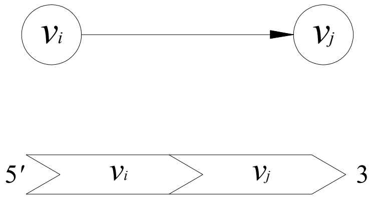


Figure 6.10. Double-encoded substring of type 2-1.

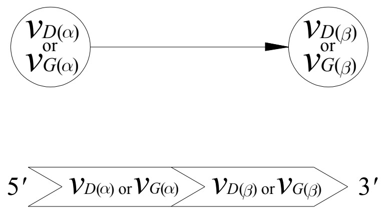

$v_{D(\alpha)}$ or $v_{G(\alpha)}$
$v_{D(\beta)}$ or $v_{G(\beta)}$
5′
$v_{D(\alpha)}$ or $v_{G(\alpha)}$
$v_{D(\beta)}$ or $v_{G(\beta)}$
3′

Figure 6.11. Double-encoded substring of type 2-2.

tide consisting of two unique sites. Thus, v_i was set with the length of 25 bp, and v_j with the length of 25 bp.

At the same time, type 2-2 is a double-encoded substring of two subsets of element nodes in one direction. In other words, as shown in Figure 6.11, there are two specific subsets of element nodes for type 2-2, in which the direction of the arrow indicates the direction from the element node subset ($v_{D(\alpha)}$ or $v_{G(\alpha)}$) to the other element node subset ($v_{D(\beta)}$ or $v_{G(\beta)}$). For type 2-2, the row and column labels are denoted by s_α and s_β for encoding two subsets of element nodes in DNA, both s_α and s_β are the subset members of either $D(y)$ or $G(y)$, a subset of all the row and column labels is denoted by Ef, and the entries are defined as

$$t_{s_\alpha, s_\beta} = 1 \text{ for } s_\alpha \text{ and } s_\beta \in D(y) \text{ or } G(y),\, s_\alpha \neq s_\beta, \text{ and}$$
$$\text{all } (s_\alpha, s_\beta) \in E, \text{since all } (s_\alpha, s_\beta) \notin Es \bigcup Ed \bigcup Eg \bigcup Eh\,. \tag{6.37}$$

In addition, for type 2-2, a subset of arcs in A is denoted by $A_{2\text{-}2}$, which is defined as

$$A_{2-2} = \Big\{ \overrightarrow{(v_{s_\alpha}, v_{s_\beta})} \in A \mid s_\alpha \text{ and } s_\beta \in D(y) \text{ or } G(y),\, s_\alpha \neq s_\beta,$$
$$v_{s_\alpha} e v_{s_\beta}, \text{ and all } \overrightarrow{(v_{s_\alpha}, v_{s_\beta})} \notin A_{1-2} \bigcup A_{2-1} \bigcup A_3 \bigcup A_4 \Big\}. \tag{6.38}$$

For type 2-2, there are two specific subsets of element nodes that are also encoded as an oligonucleotide consisting of two unique sites. Thus, both $v_{D(\alpha)}$ and $v_{G(\alpha)}$ were set with the length of 25 bp, and both $v_{D(\beta)}$ and $v_{G(\beta)}$ with the length of 25 bp.

Second, type 3 is basically encoded with either type 1-1 or 1-2. As shown in Figure 6.12, type 3 represents a double-encoded substring of the element nodes, in which the element node v_i is directed to the two or more element nodes included in either the element node subset $v_{D(y)}$ or $v_{G(y)}$ and is satisfied in either type 1-1 or 1-2. In other words, for type 3, the element node v_i has one or more directional arrows that indicate the one or more directions from the element node v_i to the two or more element nodes included in either the element node subset $v_{D(y)}$ or $v_{G(y)}$. For type 3, the row and column labels are denoted by i and s for encoding an element node and a subset of element nodes in DNA. s is a subset member of either $D(y)$ or $G(y)$, a subset of all the row and column labels is

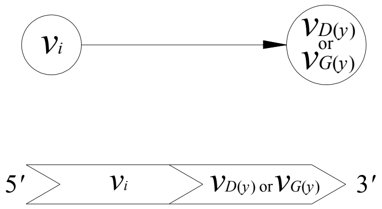


Figure 6.12. Double-encoded substring of type 3.

denoted by *Eg*, and the entries are defined as

$$t_{i,s} = 1 \text{ for } i = 1, 2, \cdots, n,\ s \in D(y) \text{ or } G(y),\ i \neq s, \text{ and}$$
$$\text{all } (i, s) \in E, \text{since all } (i, s) \notin Es \cup Ed \cup Ef \cup Eh\,. \qquad (6.39)$$

In addition, for type 3, a subset of arcs in *A* is denoted by A_3, which is defined as

$$A_3 = \Big\{ \overrightarrow{(v_i, v_s)} \in A \mid i = 1, 2, \cdots, n,\ s \in D(y) \text{ or } G(y),\ i \neq s,$$
$$v_i e v_s, \text{ and all } \overrightarrow{(v_i, v_s)} \notin A_{1-2} \cup A_{2-1} \cup A_{2-2} \cup A_4 \Big\}. \qquad (6.40)$$

For type 3, the single element node and the subset of the element nodes are encoded together as an oligonucleotide consisting of two unique sites. Thus, v_i was set with the length of 25 bp and both $v_{D(y)}$ and $v_{G(y)}$ were set with the length of 25 bp.

Finally, type 4 is also basically encoded with either type 1-1 or 1-2. As shown in Figure 6.13, type 4 also represents a double-encoded substring of the element nodes in the same manner as type 3, but the element node v_j is directed from the two or more element nodes included in either the element node subset $v_{D(y)}$ or $v_{G(y)}$. In other words, for type 4, the element node v_j has one or more directional arrows that indicate the one or more directions from the two or more element nodes included in either the element node subset $v_{D(y)}$ or $v_{G(y)}$ to the element node v_j. For type 4, to encode a subset of element nodes and an element node in DNA, the row and column labels are denoted by *s* and *j*, a subset of all of the row and column labels is denoted by *Eh*, and the entries are defined as

$$t_{s,j} = 1 \text{ for } s \in D(y) \text{ or } G(y),\ j = 1, 2, \cdots, n,\ s \neq j, \text{ and}$$
$$\text{all } (s, j) \in E, \text{since all } (s, j) \notin Es \cup Ed \cup Ef \cup Eg\,. \qquad (6.41)$$

In addition, for type 4, a subset of arcs in *A* is denoted by A_4, which is defined as

$$A_4 = \Big\{ \overrightarrow{(v_s, v_j)} \in A \mid s \in D(y) \text{ or } G(y),\ j = 1, 2, \cdots, n,\ s \neq j,$$

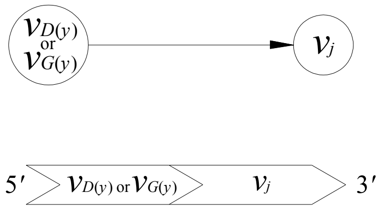


Figure 6.13. Double-encoded substring of type 4.

$$v_s e v_j \text{, and all } \overrightarrow{(v_s, v_j)} \notin A_{1-2} \cup A_{2-1} \cup A_{2-2} \cup A_3 \Big\}. \tag{6.42}$$

For type 4, the subset of the element nodes and the single element node are encoded together as an oligonucleotide consisting of two unique sites. Thus, both $v_{D(y)}$ and $v_{G(y)}$ were set with the length of 25 bp in the same manner as type 3, and v_j was set with the length of 25 bp.

Based on the above defined subsets of the row and column labels, the set of all possible row and column labels in direct relations is E, which can be expressed as

$$E = Es \cup Ed \cup Ef \cup Eg \cup Eh\,, \tag{6.43}$$

and the set of all possible arcs in the digraph based on the above defined subsets of arcs, which can be expressed as

$$A = A_{1-2} \cup A_{2-1} \cup A_{2-2} \cup A_3 \cup A_4\,. \tag{6.44}$$

The two same or different types 2-1, 2-2, 3, and 4 of the double-encoded substring are attached by types 5 and 6.

6.8.3 Complementary substrings

First, type 5 is created for attaching two different double-encoded substrings that are (1) types 2-2 and 2-2; (2) types 2-2 and 4; (3) types 3 and 2-2; and (4) types 3 and 4 lined up sequentially. As shown in Figure 6.14, type 5 represents a complementary substring between (1) types 2-2 and 2-2; (2) types 2-2 and 4; (3) types 3 and 2-2; and (4) types 3 and 4, which are all sequenced in a 5′ to 3′ direction. Each subset of these complementary substrings is encoded. Here, a subset of the element nodes is encoded as an oligonucleotide, consisting of two complementary substrings sequentially. Thus, both $L_{D(y)}$ and $L_{G(y)}$ were set with the length of 50 bp for type 5, where y is independently denoted by I, II,…, n, based on both $v_{D(y)}$ and $v_{G(y)}$.

Second, type 6 is temporarily created for attaching each of the different double-encoded substrings to detect one or more circular DNA fragments, corresponding to type 1-1 or 1-2. Further, type 6 is re-created for attaching specific double-encoded

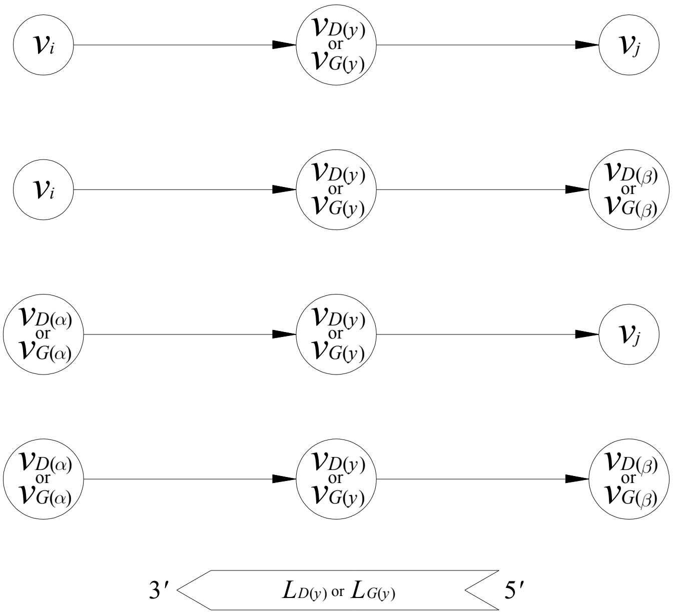


Figure 6.14. Complementary substring of type 5.

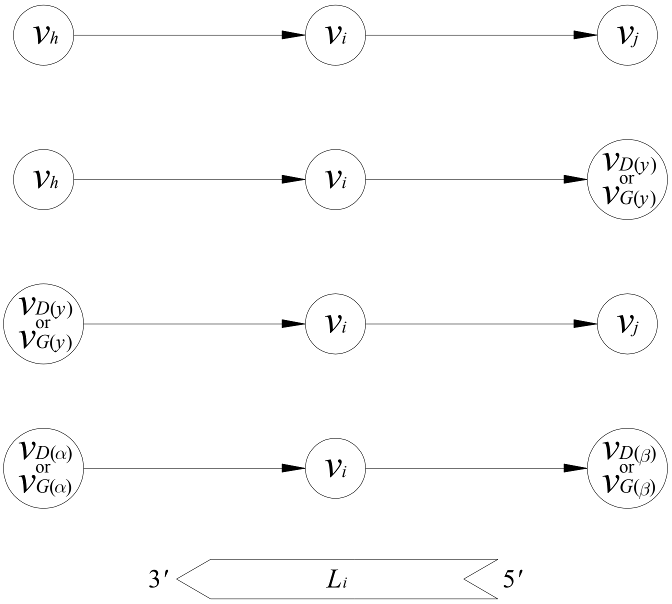


Figure 6.15. Complementary substring of type 6.

substrings that are (1) types 2-1 and 2-1; (2) types 2-1 and 3; (3) types 4 and 2-1; and (4) types 4 and 3 lined up sequentially. As shown in Figure 6.15, type 6 also represents a complementary substring between (1) types 2-1 and 2-1; (2) types 2-1 and 3; (3) types 4 and 2-1; and (4) types 4 and 3, which are all sequenced in a 5′ to 3′ direction. Each element node of these complementary substrings is encoded. Thus, each element node except the element nodes included in either the element node subset $v_{D(y)}$ or $v_{G(y)}$ is encoded as an oligonucleotide consisting of two complementary sequential sites. Therefore, L_i, $i = 1, 2,\ldots, n$ was set to a length of 50 bp.

6.9 Experimental Studies and Results

In this chapter's study, a splicing operation model [60] is also applied to the DNA-based ISM method based on the DNA encoding process. The splicing operation method is used for a formal model of the recombinant behaviour of DNA molecules that act under the influence of restriction enzymes and ligases [137].

To concatenate the crosswise DNA fragments, two different kinds of encoded DNA sequences should be spliced. As shown in Tables 6.1 and 6.2, for the example digraph, the pattern of each DNA substring is described by each element node and subset of element nodes that correspond to DNA sequences, and the pattern of each complementary substring is described by each concatenation of DNA fragments.

6.9.1 Experimental studies

To execute the simulated experimental studies, the hierarchical DNA-based algorithm was proposed for ISM using the splicing operation method. A program compiled using Vector NTI software was used to represent the length of DNA strands. The hierarchical DNA-based algorithm performs to construct one or more hierarchically restructured digraphs based on simulated experimental studies.

To distinguish the features of the circular DNA fragments from other DNA fragments, all of the double-encoded substrings, corresponding to type 1, from the given DNA substrings in Table 6.1 and their complementary substrings, corresponding to type 6, should be first generated to determine types 1-1 and 1-2, corresponding to one or more circular DNA fragments, from the hybridisation and ligation-1 process.

Two methods can be used to properly determine one or more circular DNA fragments. For one method, the different topologies of circular DNA fragments and linear DNA fragments are focused to distinguish circular DNA fragments from linear DNA fragments (described in Chapter 4). For the other method, restriction enzyme sites are added to each DNA substring and complementary substring, so that restriction sites are available on the circular DNA fragments. After this additional process, we can do a restriction digest and run the digested and undigested DNA in parallel on an agarose gel. Thereby, the two different types of DNA should have different mobility.

In this chapter's study, the function of restriction enzymes is to cleave and denature one or more circular DNA fragments to measure the total length of circular and linear DNA fragments. To measure them together, the detected circular DNA fragments should be cleaved, denatured, and become linear DNA fragments. Figure 6.16 illustrates the molecular encoding method by which the positioning element node is associated with the DNA substring. The PCR technique is used to generate all the double-encoded sub-

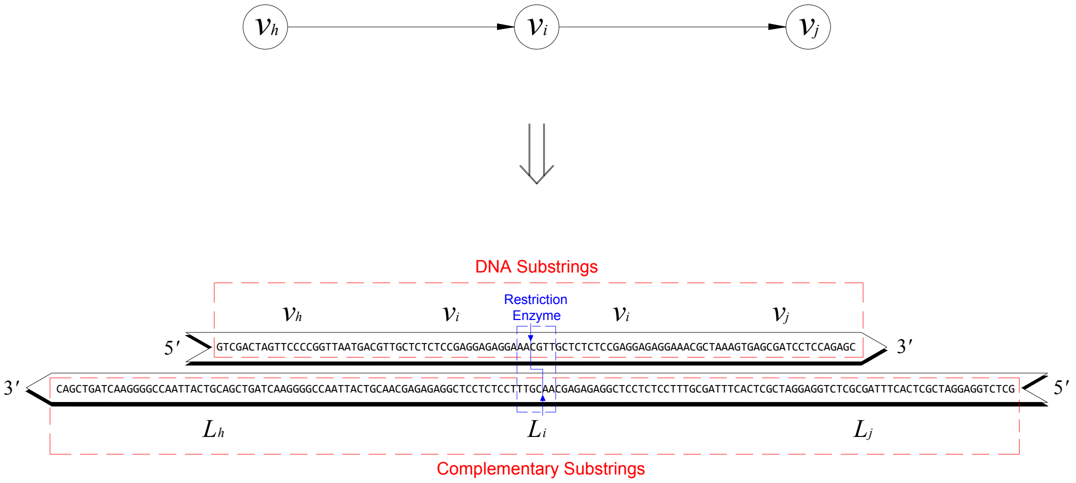

v_h
v_i
v_j
DNA Substrings
Restriction Enzyme
v_h
v_i
v_i
v_j
5′
GTCGACTAGTTCCCCGGTTAATGACGTTGCTCTCTCCGAGGAGAGGAAACGTTGCTCTCTCCGAGGAGAGGAAACGCTAAAGTGAGCGATCCTCCAGAGC
3′
3′
CAGCTGATCAAGGGGCCAATTACTGCAGCTGATCAAGGGGCCAATTACTGCAACGAGAGAGGCTCCTCTCCTTTGCAACGAGAGAGGCTCCTCTCCTTTGCGATTTCACTCGCTAGGAGGTCTCGCGATTTCACTCGCTAGGAGGTCTCG
5′
L_h
L_i
L_j
Complementary Substrings

Figure 6.16. Example of the DNA encoding scheme; how DNA substrings and their complementary substrings are bound for three direct element nodes.

strings and their complementary substrings.

A nucleotide precursor is referred to a single deoxyadenosine triphosphate (dATP) that can be used for DNA synthesis. As shown in Figure 6.17, in particular phosphorus, denoted as ^{32}P, has a radioactive isotope, which is clearly carried by incorporated nucleotides. Here, those incorporated nucleotides label a single DNA molecule [138]. A method of nick translations is used for labelling regions of the hybridised DNA molecules (see Figure 6.18), and a method of end filling is also used for the hybridised DNA molecules (see Figure 6.19). These two methods are references only.

A simulated experimental protocol study was designed and corresponds to Step 4 in the process of the DNA-based ISM method above, which is based on the hierarchical DNA-based algorithm for ISM using the splicing operation method, and the simulated

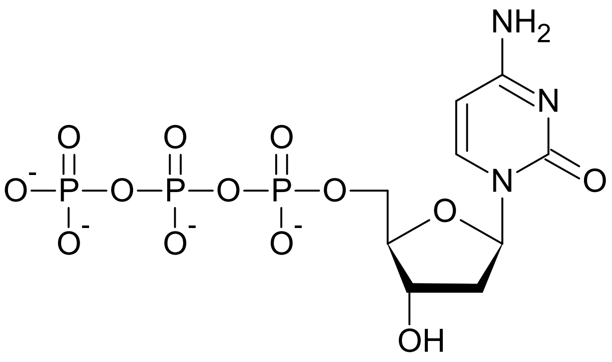


Figure 6.17. Structural formula of deoxyadenosine triphosphate.

experimental protocol study of Step 4 is composed of six substeps as follows:

Substep 1 (hybridisation and ligation-1)**:** For the given digraph, all of the encoded element nodes of DNA sequences (type 1) and their complementary substrings (type 6) are artificially synthesised and placed in a test tube. For this hybridisation, the DNA sequences and their complementary substrings are heated to approximately 94˚C and cooled to approximately 20˚C at 1˚C/min. In addition, DNA ligases must be added for ligation among DNA strands.

Substep 2 (detection and distinction)**:** One or more circular DNA fragments that correspond either to type 1-1 or 1-2, which are marked respectively, as either *D*(*y*) or *G*(*y*), *y* = I, II,…, *n* to distinguish them, are detected and encoded again for types 2-2, 3, 4, and 5. In this process, if either type 1-1 or 1-2 is isolated, meaning not connected to any other element nodes, the DNA fragment should be located in level 1. After circular DNA fragments are detected, linear DNA fragments are distinguished from the detected circular DNA fragments. The non-components of element nodes in the digraph correspond to the linear DNA fragments (the remaining encoded DNA substrings of type 1), which are clearly used for type 2-1.

Substep 3 (hybridisation and ligation-2)**:** All of the encoded element nodes of DNA sequences, corresponding to types 2-1, 2-2, 3, and 4, and their complementary substrings, corresponding to types 5 and 6, are hybridised to each other and ligated in the same manner as Substep 1.

Substep 4 (simulated gel electrophoresis)**:** DNA strands can be separated according to their sizes using the simulated gel electrophoresis apparatus. Two parts of the simulated gel electrophoresis are prepared. For the first part of the simulated gel electrophoresis, the detected and marked circular DNA fragments (Substep 2) are transferred to wells. For the second part of the simulated gel electrophoresis, the hybridised DNA strands (Substep 3) are transferred to wells. For the first part of the wells, although the length of circular DNA fragments can be separately measured, we cleave and denature circular DNA fragments using restriction enzymes to measure both circular and linear DNA fragments at the same time. For the second part of the wells, we focus on the longest DNA strand that obviously contains as many different element nodes as possible. Thus, all of each group of the same length as the longest DNA strands should be selected.

Substep 5 (affinity separation)**:** In this chapter's study, affinity separation is employed to distinguish among the given element nodes. The longest DNA strands should be heated to approximately 94˚C to become ssDNA before the method of affinity separation is applied. The complementary substrings of all of both the subsets of element nodes (type 5) and the single element nodes (type 6) are prepared and attached to magnetic beads. The complementary substrings of two kinds of subsets correspond to all the possible beginning and ending element nodes, represented as both v_{start} and v_{end}, which are used to verify whether or not each group of the same length as the longest DNA strand is composed of all possible beginning and ending levels. After this verification, the complementary substrings are used to distinguish among the given element nodes in the longest DNA strands. These element nodes should be included in each of the middle

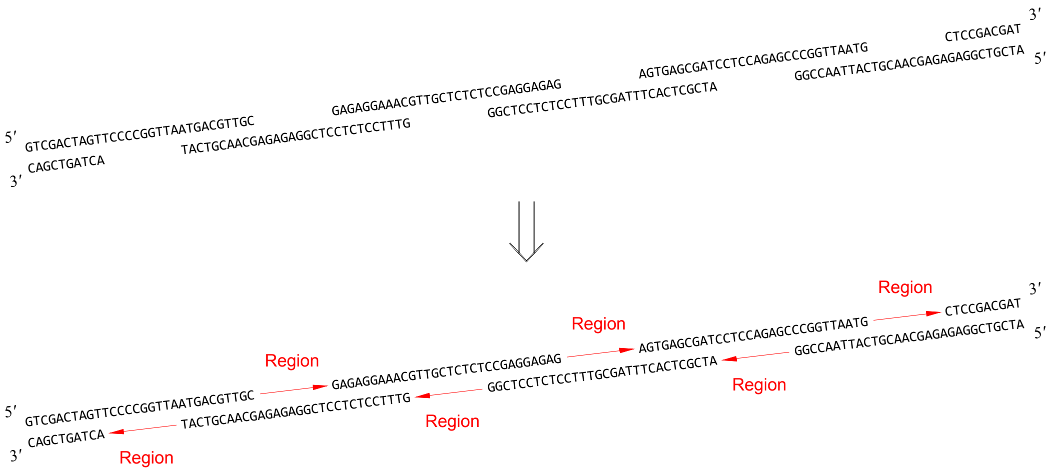

3′
CTCCGACGAT
5′
AGTGAGCGATCCTCCAGAGCCCGGTTAATG
GGCCAATTACTGCAACGAGAGAGGCTGCTA
GAGAGGAAACGTTGCTCTCTCCGAGGAGAG
GGCTCCTCTCCTTTGCGATTTCACTCGCTA
5′
GTCGACTAGTTCCCCGGTTAATGACGTTGC
TACTGCAACGAGAGAGGCTCCTCTCCTTTG
3′
CAGCTGATCA
Region
Region
Region
Region
Region
Region
Region

Figure 6.18. Method of nick translations used for labelling DNA molecules [59].

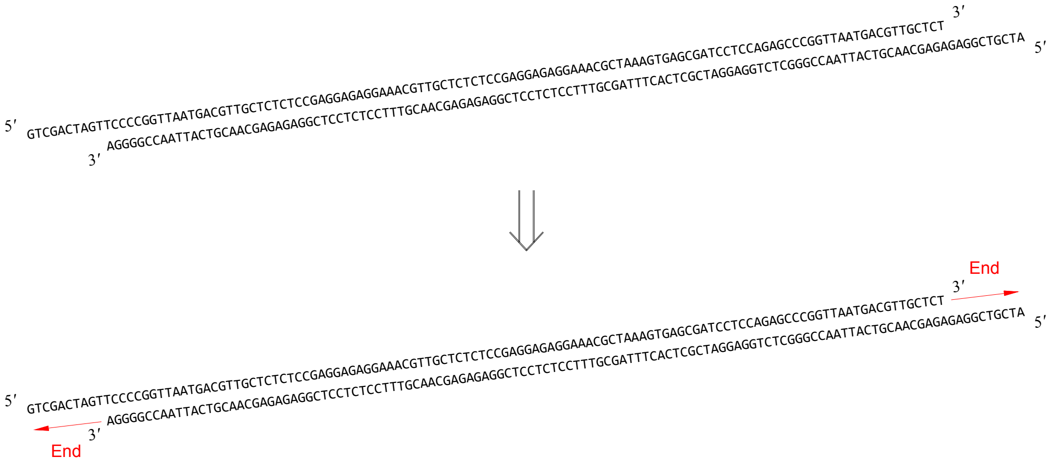

3′
GTCGACTAGTTCCCCGGTTAATGACGTTGCTCTCTCCGAGGAGAGGAAACGTTGCTCTCTCCGAGGAGAGGAAACGCTAAAGTGAGCGATCCTCCAGAGCCCGGTTAATGACGTTGCTCT
5′
AGGGGCCAATTACTGCAACGAGAGAGGCTCCTCTCCTTTGCAACGAGAGAGGCTCCTCTCCTTTGCGATTTCACTCGCTAGGAGGTCTCGGGCCAATTACTGCAACGAGAGAGGCTGCTA
5′
3′
End
End

Figure 6.19. Method of end fillings used for labelling DNA molecules [59].

levels.

Substep 6 (check)**:** If there are remaining complementary substrings, corresponding to both types 5 and 6, those were not annealed to any of the longest DNA strands in the first group, meaning that two or more hierarchically restructured digraphs are sure to exist, then we should select another group of the same length as the second longest DNA strands from the simulated gel electrophoresis, and perform the same affinity separation process. The groups should be selected until all remaining complementary substrings are annealed to some of the longest DNA strands.

6.9.2 Results of the experimental studies

For the example digraph, the expected final results of the simulated gel electrophoresis are shown in Figure 6.20, which illustrates all of the selected DNA strands. In Figure 6.20, each lane represents its own length of DNA strand in a different number of element nodes. A larger number among the element nodes corresponded to a longer DNA strand, whereas a smaller number among the element nodes corresponded to a shorter DNA strand. More detailed results in Figure 6.20 are explained below.

First, all of the detected, cleaved, and denatured circular DNA fragments were represented in lanes 1 to 7. Lane 1 was for the subset $D(\mathrm{I}) = \{1, 5, 6, 10, 16\}$ at a distance of 250 bp, lane 2 was for the subset $D(\mathrm{II}) = \{2, 3, 8, 13, 14, 29\}$ at a distance of 300 bp, lane 3 was for the subset $D(\mathrm{III}) = \{4, 30\}$ at a distance of 100 bp, lane 4 was for the subset $D(\mathrm{IV}) = \{7, 15, 20\}$ at a distance of 150 bp, lane 5 was for the subset $D(\mathrm{V}) = \{9, 18, 23\}$ at a distance of 150 bp, lane 6 was for the subset $D(\mathrm{VI}) = \{11, 17, 21, 22, 24\}$ at a distance of 250 bp, and lane 7 was for the subset $D(\mathrm{VII}) = \{12, 27, 28\}$ at a distance of 150 bp.

Second, the first group of the same length as the longest DNA strand was represented in all possible routing directions. The first group was composed of lanes 8 to 11. Lane 8 was for the route $v_{D(\mathrm{V})} \rightarrow v_{D(\mathrm{I})} \rightarrow v_{D(\mathrm{VII})} \rightarrow v_{D(\mathrm{III})}$ at a distance of 200 bp, lane 9 was for the route $v_{D(\mathrm{V})} \rightarrow v_{D(\mathrm{I})} \rightarrow v_{D(\mathrm{VII})} \rightarrow v_{D(\mathrm{VI})}$ at a distance of 200 bp, lane 10 was for the route $v_{D(\mathrm{V})} \rightarrow v_{19} \rightarrow v_{D(\mathrm{VII})} \rightarrow v_{D(\mathrm{III})}$ at a distance of 200 bp, and lane 11 was for the route $v_{D(\mathrm{V})} \rightarrow v_{19} \rightarrow v_{D(\mathrm{VII})} \rightarrow v_{D(\mathrm{VI})}$ at a distance of 200 bp.

Finally, the second group of the same length as the second longest DNA strand was

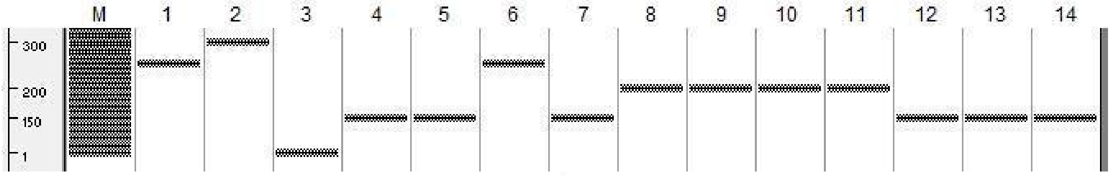


Figure 6.20. Representation results of the simulated gel electrophoresis. Lane M is for marker. Lanes 1 to 7 are for the cleaved and denatured circular DNA fragments. Lanes 8 to 11 indicate one group of the same length as the longest DNA strand in all possible routing directions. Lanes 12 to 14 indicate the other group of the same length as the longest DNA strand in all possible routing directions.

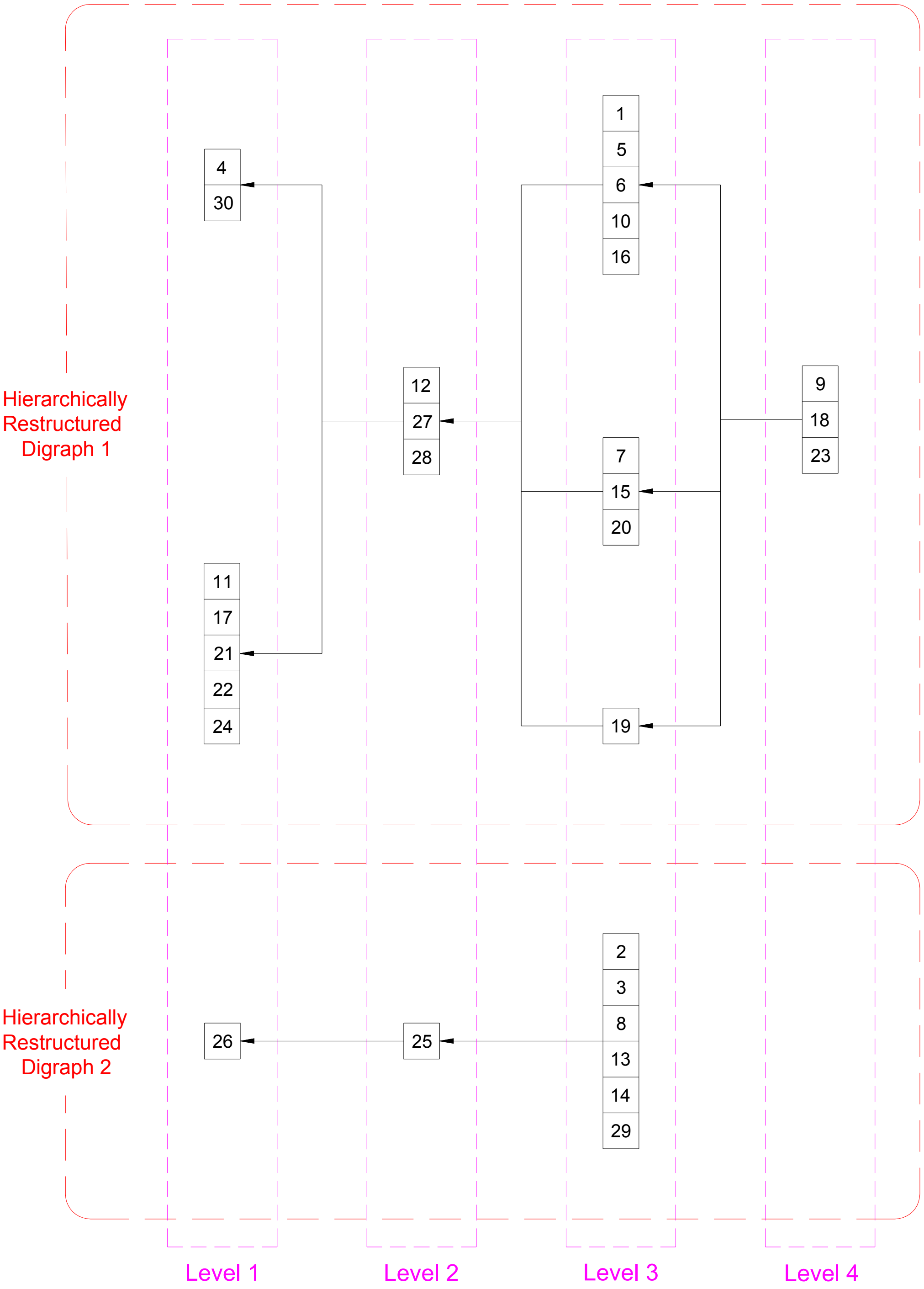


Figure 6.21. Result of the two hierarchically restructured digraphs lined up from the beginning level to the ending level.

also detected and comprised lanes 12 to 14. Lane 12 was for the route $v_{D(\mathrm{IV})} \rightarrow v_{D(\mathrm{VII})} \rightarrow v_{D(\mathrm{III})}$ at a distance of 150 bp, lane 13 was for the route $v_{D(\mathrm{IV})} \rightarrow v_{D(\mathrm{VII})} \rightarrow v_{D(\mathrm{VI})}$ at a distance of 150 bp, and lane 14 was for the route $v_{D(\mathrm{II})} \rightarrow v_{25} \rightarrow v_{26}$ at a distance of 150 bp. The above two groups included all of the element nodes from v_1 to v_{30} that clearly existed in the example digraph.

Based on the studies and results of the simulated experimentations, the hierarchically restructured digraphs have been constructed and are shown in Figure 6.21. The new hierarchical digraphs were constructed from the selected groups of the same length of the longest DNA strands, identified by their beginning and ending elements, and from distinguishing the complementary substrings of the element nodes in Substep 5. We identified which element was included in which level and how they were sequentially lined up from the beginning level to the ending level by checking which complementary substring of either element node subset or element node was annealed to which one in the next place.

In the simulated experimental studies and results, two different types of the hierarchically restructured digraphs were determined, which meant that it was necessary to execute Substep 6. The hierarchically restructured digraph 1 is mainly composed of both the 4-level digraph and the 3-level digraph, and the hierarchically restructured digraph 2 is composed only of the 3-level digraph, as shown in Figure 6.21.

6.10 Approach to Solving a Communication Problem

For an empirical study, we approach a communication problem that is introduced in this section, and describe the possibility that this problem can be solved by the hierarchical DNA-based algorithm. The communication problem is particularly hard to solve when dealing with a large number of contextual data from the compounded communication channels or sources. In this case, we are able to use the hierarchical DNA-based algorithm to solve this problem.

In our advanced communication and information societies, single individuals or groups often or continue to receive, provide, and exchange their contextual information with each other in their communications. If a large number of communicators are included in some communication channels, their communication contexts are obviously complex to be handled properly.

In complex communications, when groups or subgroups propose different ideas and opinions, misunderstandings easily arise among communicator interchanges. Representing the given complex communications among communicators with their contextual relations makes it easier to properly handle all of their different contextual contents of ideas and opinions. After the given complex communications are represented with their contextual relations, we can extract, segment, and organise the complex communications (already represented with their contextual relations). Based on the representation with their contextual relations, the intended communicators are transformed into an element node set, and their directions of communication codifications and decodifications can also be transformed into an arc set. Thus, the transformed element node and arc sets are finally represented as a digraph. Figure 6.22 illustrates the transforming process to construct a digraph.

The main roles of using the hierarchical DNA-based algorithm for this approach are (1) to deal with the first constructed digraph from the transforming process; (2) to con-

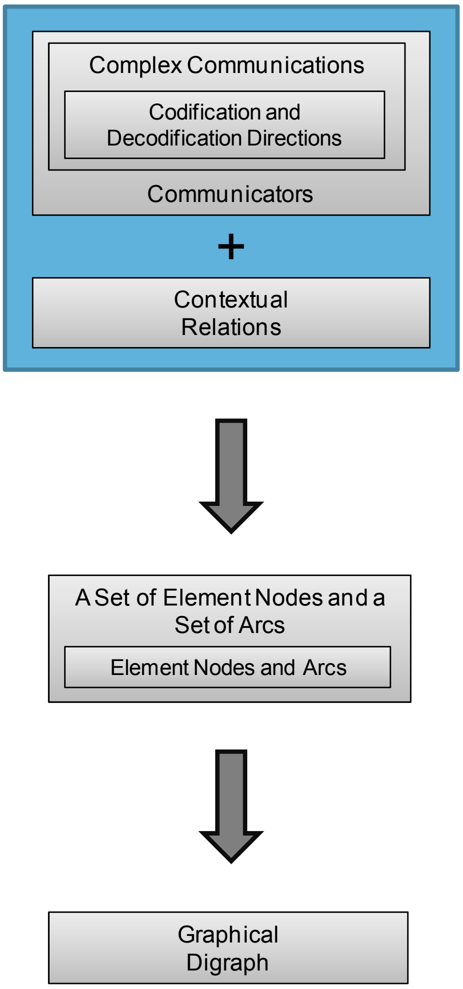


Figure 6.22. Transforming process for constructing a graphical digraph.

struct one or more hierarchically restructured digraphs; and (3) to provide comprehensive and understandable visible information (coming from the constructed hierarchically restructured digraphs) to decision makers who manage or organise complex communications with contextual relations.

6.11 Computational Times and Solvable Sizes

A comparison of the number of element nodes and the number of arcs is shown in Figure 6.23. In addition, the comparisons of approximated running times are shown in Table 6.3. The number of ideas corresponds to the inputs of size 10, 30, 50, and 100 separately for the exponential-time algorithms and the prepared hierarchical DNA-based algorithm (we are ready to detect solutions). Four graphic comparison representations of approximated running times are properly shown in Figures 6.24 to 6.27 for the expo-

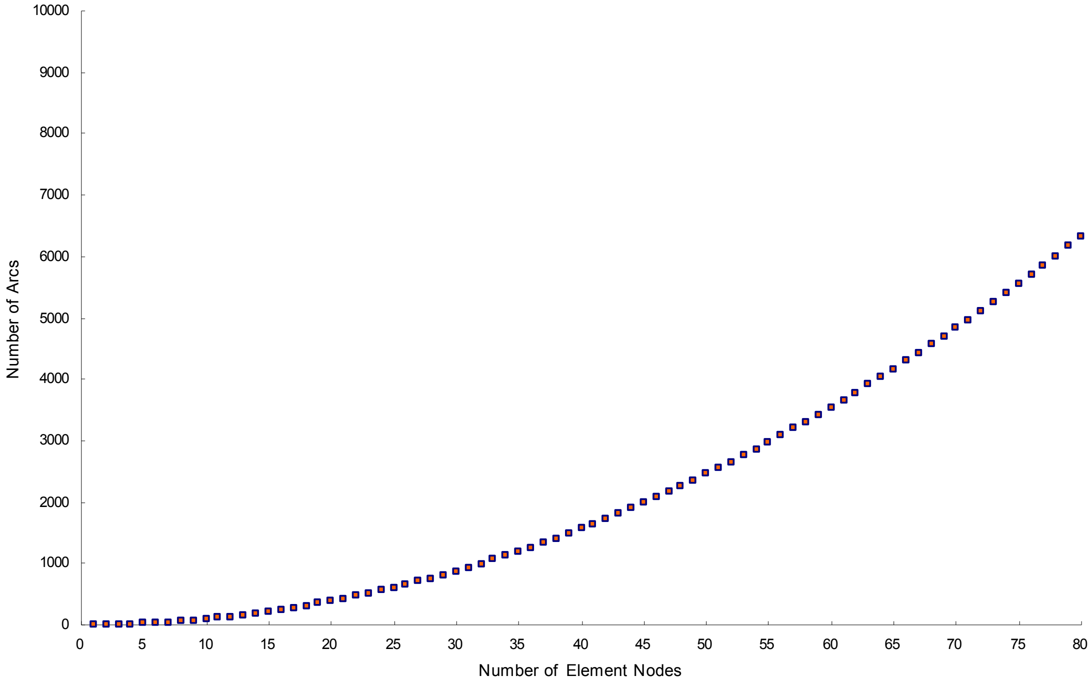


Figure 6.23. Comparison graph of the number of element nodes and the number of arcs.

nential-time algorithm and prepared hierarchical DNA-based algorithm.

For the exponential-time algorithm, we suppose that a processor executes a million high levels of instructions a second, and the exponential-time algorithm uses an operation of 2^n [106]. Based on previous experimental reports, our experimental experiences, and genetic engineering notes [62-66], we measured and calculated the approximated

Table 6.3. Comparisons of approximated running times for the exponential-time algorithm and the prepared hierarchical DNA-based algorithm.

Number of Element Nodes	10	30	50	100
The Exponential-Time Algorithm	< 1.00 second	18.00 minutes	36.00 years	100,000,000,000,000,000.00 years
The Prepared DNA-Based Algorithm	5.79 minutes	55.93 minutes	2.63 hours	10.61 hours

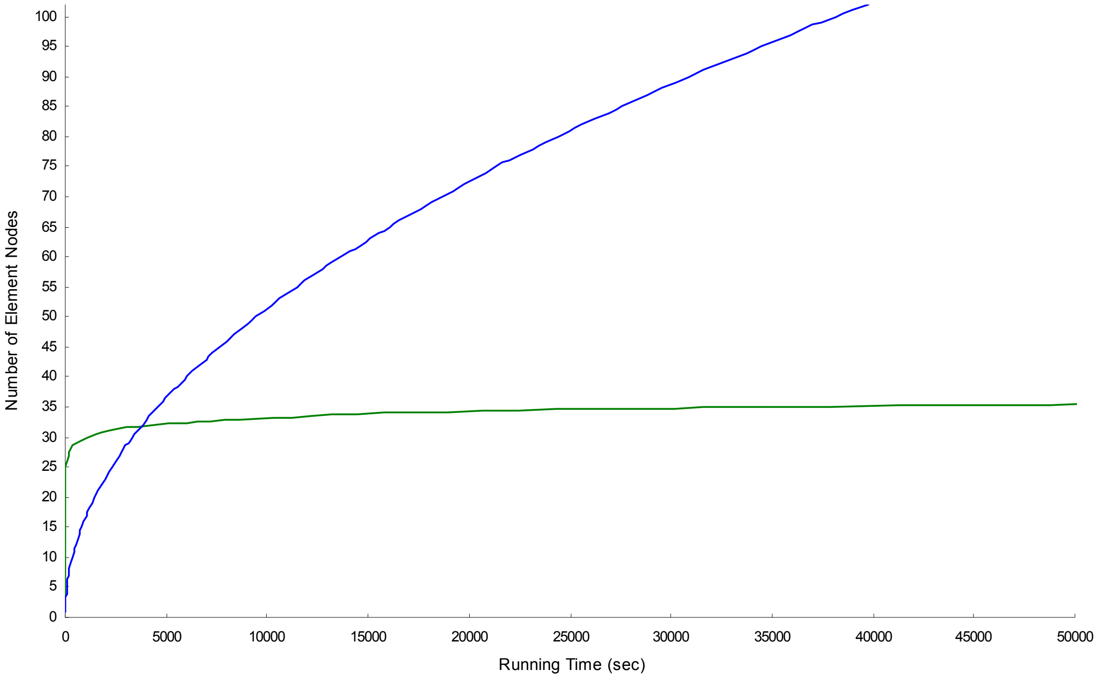


Figure 6.24. Comparison graph of approximated running times in seconds: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared hierarchical DNA-based algorithm.

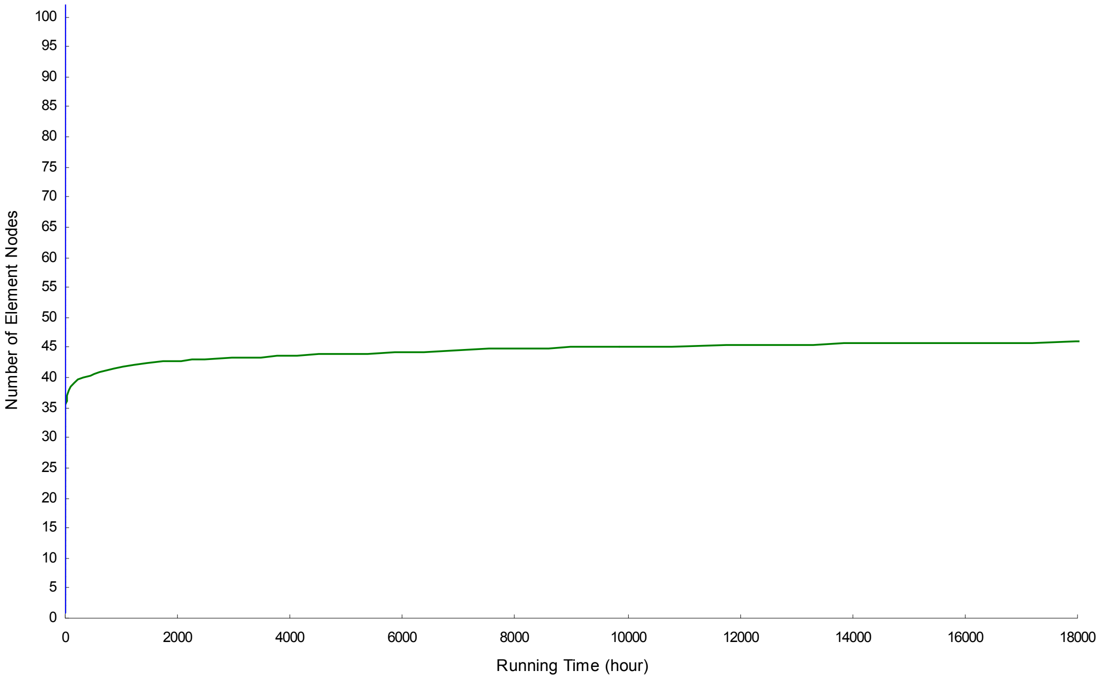


Figure 6.25. Comparison graph of approximated running times in hours: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared hierarchical DNA-based algorithm.

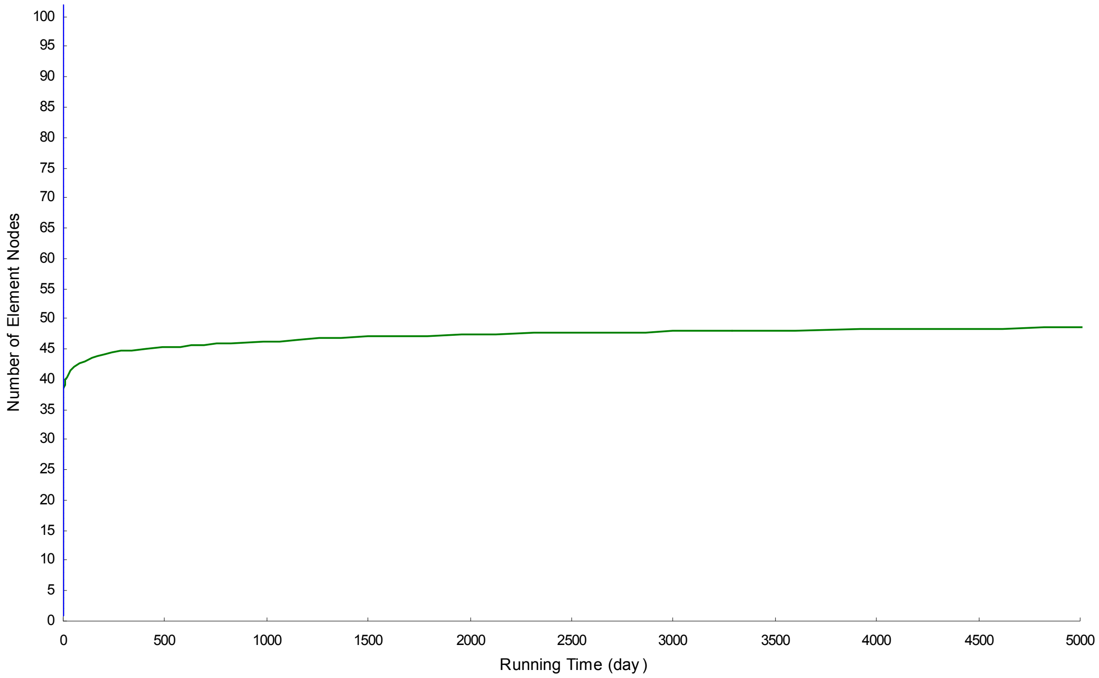


Figure 6.26. Comparison graph of approximated running times in days: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared hierarchical DNA-based algorithm.

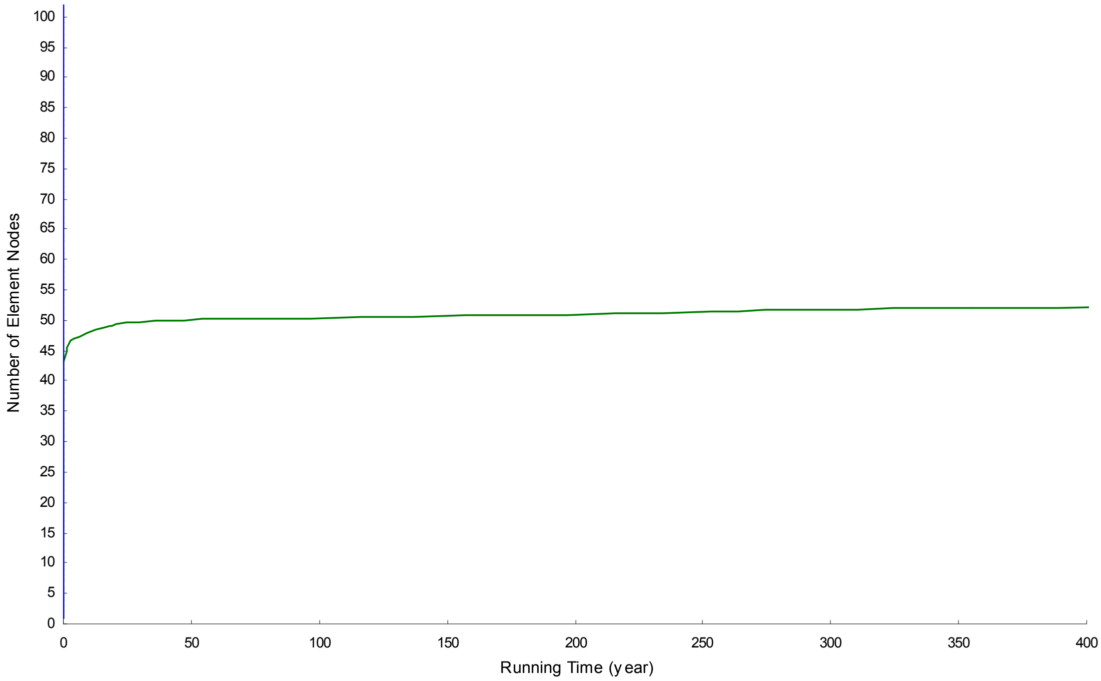


Figure 6.27. Comparison graph of approximated running times in years: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared hierarchical DNA-based algorithm.

running times of the prepared hierarchical DNA-based algorithm.

Generally speaking, a polynomial-time algorithm is efficient, compared to an exponential-time algorithm when dealing with running complexities. In this chapter's study, the main problem is how to clearly and precisely minimise arc crossings among a huge number of element nodes to construct one or more hierarchically restructured digraphs. This problem is an NP-complete problem, meaning a polynomial-time algorithm has not been truly discovered yet for minimising all crossings among element nodes.

6.12 Concluding Remarks

In this chapter, a new ISM method was efficiently measured using the proposed algorithm to create a hierarchical DNA-based algorithm. The hierarchical DNA-based algorithm was used for ISM to construct one or more hierarchically restructured digraphs based on bioscience techniques and simulated experimental studies.

In the mathematical algorithm of the ISM method, we needed to raise the binary adjacency matrix to successive powers based on Boolean algebra for the matrix to become a reachability matrix. This must be repeated to calculate a structural matrix, also repeated to calculate a reachable condensation matrix, and again be repeated to calculate a reachable skeleton matrix. Moreover, for the mathematical algorithm, we must examine all the given elements to discover whether they were partitioned or not, based on calculating each of the given pair elements, including either the same or different subparts. In contrast, the hierarchical DNA-based algorithm of the ISM method, based on encoding each of the specific elements, allowed us to directly obtain a reachable skeleton matrix, and awaked the partition status from the final results. This meant we did not need to examine all of the elements, whether they were partitioned or not. The hierarchical DNA-based algorithm was an efficient and improved algorithm to determine the key results for decision makers.

This chapter has shown a novel algorithm that was easily adapted to the splicing operation method in encoding element nodes in DNA for ISM, thereby constructing two hierarchically restructured digraphs for the example digraph. Beyond that, this algorithm can be applied to expand with effort to other interpretive or relationally connected models that are composed of directed graphs.

Chapter 7

Minimising Decision Rules Based on a Rough DNA-Based Algorithm

7.1 Overview

A new knowledge support system is needed for decision making. A new cutting-edge approach to enhancing cooperative data processing using both computer-based and molecular engineering technologies is demonstrated in this chapter. This new molecular computational algorithm is created to derive decision rules of minimal length in rough sets and is termed a *rough DNA-based algorithm*. This algorithm is designed particularly for handling a large number of objects and their attributes in data processing.

With the development of information technology, proposers or decision makers in each branch can smoothly exchange and distribute their decision lists or other information in the form of data. Moreover, in various operations of collaborative decision making, conflicts exist between individual proposers that may result in large amounts of uncertain data. In this case, different types of objects and their data attributes can be reduced and classified to provide comprehensible information. The best method might be a rough set method, in which large data sets are handled, because it is particularly applicable to cooperative data processing. The rough set method is a new knowledge discovery method for a data classification system. This method has emerged with the need to distinguish the distinctive features of given objects and their data attributes.

The large numbers of objects that often emerge from many different proposers in database handling pose an intractable problem for computing all minimal length decision rules of given objects. To address this dilemma, a new computational method using the rough DNA-based algorithm is discussed and demonstrated. This algorithm performs by deriving decision rules of minimal length, thereby identifying a new potential algorithm. A new knowledge support system approach based on this algorithm also shows the potential for an integrated knowledge support system, including both computer-based and molecular engineering technologies, and offers a better method for data processing.

The rough DNA-based algorithm is used (1) to detect all subsets of low approximations classified into each decision class; and (2) to determine all minimal length decision rules by handling given objects and their related attributes based on the detected decision classes. In this chapter, the rough DNA-based algorithm, including mathematical rough set concepts and molecular engineering techniques, can be measured for the efficiency of minimising decision rules, therefore demonstrating its possibility as a new integrated algorithm for building a new knowledge support system.

7.2 Background and Motivations

In dynamic and flexible environments of uncertain data interactions, achieving a resolution to certain problems or issues in data mining may arise as a result of dispute resolution. Many applied information and database technology systems [139-141] make it possible to solve this problem and promptly provide data, such as ideas, features, decisions, information, and knowledge. The powerful ability of these new systems can be used to classify specific objects and their specific attributes in data between the proposers in knowledge collaboration.

A rough set method is a new methodology that is used for grasping characteristics of the classified objects and object attributes in the case of data processing. The rough set method is based on rough set theory, which was first described by Pawlak [142]. It offers a new way of machine learning, discovering knowledge in data, and pioneering a new type of knowledge support system. Rough set theory also became an important base of reasoning, providing understandable information derived from different data, inductive classification, and knowledge reduction.

In rough sets, a decision table is often used for representing given objects, attributes, and decision attributes. The decision table must be interpreted in terms of a set of *if-then* decision rules that are used to improve or support decision making. The main purpose of modifying decision rules is to reduce and simplify them efficiently.

Several algorithms and applications were edited by Slowinski [143]. These algorithms and applications used rough sets to minimise subsets of decision rules and approaches to making decisions under uncertainties. Grzymala-Busse [144, 145] proposed a computational method of all minimal length decision rules used for managing uncertainty in expert systems. Subsequently, Ziarko [146] studied a decision matrix in a decision system. An incremental identification for minimising decision rules was proposed by Skowron *et al*. [147] who exemplified the discernibility functions and matrix. A new incremental algorithm for minimising all minimal length decision rules was proposed by Shan *et al*. [148] who tried to identify decision rules in a different way.

In their data processes, a number of different or even the same proposers exchange, provide, and receive their objects, including attributes, in different types of settings. Here, many objects are evident in real situations, and these objects are difficult to handle for the easy application of rule reductions, which would provide better and simpler information to users or providers. In rough sets, the most essential and difficult problem is how to minimise decision rules and compute all minimal length decision rules for each entry of both a condition attribute and a condition attribute value, because this intractable problem is an NP-hard problem [148, 149] that still exists in dealing with mathematical concepts in rough set theory.

7.3 Computational Method with Rough Sets

In this section, the rough set method (the rough set theory-based method) is generally defined with the example of a model decision table, which is helpful in showing the rough DNA-based algorithm for minimising decision rules.

7.3.1 Rough set theory

Based on a concept of mathematics underlying set theory, the rough set has been developed as rough set theory. It was proposed by Pawlak [142] who first introduced many innovations, such as a new formal computational machine, foundations of granular computation, and perceptional knowledge discovery, among others [150]. His studies of several rough set theory researches [151, 152] also identified several advantages for (1) evaluating significant data, which analysed hidden data; (2) minimising sets of decision rules; and (3) discovering novel knowledge in knowledge support system areas.

Various pieces of information and knowledge are roundly used in executing knowledge cooperative data processing between proposers, in which knowledge can be represented in a mathematical way [149, 153]; the knowledge can be interpreted for its ability to classify specific objects. Let us assume that a set of these kinds of objects is a universe denoted by U. A constraint set on objects is referred to as a *disjoint relation* of classifying objects. The constraint set is denoted by T in each of objects x and y, and each is an element of a universe U, representing $(x, y) \in T$, but (x, y) is not equal to (y, x) and both (x, x) and (y, y) can each be elements of T. If there are three different elements x, y, and z, the constraint set T satisfies the following three properties:

(1) $(x, x) \in T$ for all $x \in U$, meaning reflexivity;
(2) if $(x, y) \in T$ then $(y, x) \in T$, meaning symmetry; and
(3) if $(x, y) \in T$ and $(y, z) \in T$ then $(x, z) \in T$, meaning transitivity.

If a relation satisfies these three properties, then the relation is called an *equivalence relation*. Here, an equivalence relation on a universe is considered, and this relation is denoted by Ξ. In rough set theory, the form of a pair (U, Ξ) is given by elementary knowledge. If x is an element of U, expressing $x \in U$, then the equivalence class of x is defined as

$$[x]_\Xi = \{y \in U \mid (x, y) \in \Xi\}, \tag{7.1}$$

where all of the objects in $[x]_\Xi$ are collected to be a part of the same category for the initial partition, and x is the element of the restored category. Elementary granules of knowledge are the categories of this form $[x]_\Xi$, known as equivalence classes of the equivalence Ξ.

The equivalence relation Ξ is often called a *discernibility relation* and when it is used, all the objects of universe U can be divided into three disjoint sets. These are called the *lower approximation*, the *upper approximation*, and the *boundary region*. For any subset $X \subseteq U$, the three disjoint sets represent the plural objects that (1) are precisely in X (class 1); (2) are not precisely in X (class 2); and (3) are possibly in X (class 3), where X is a subset of U. The objects in class 1 form the lower approximation of X, the objects in both classes 1 and 3 form the upper approximation of X, and the objects in class 3 form the boundary region of X. These three sets are defined in the following manner. First, the lower approximation of X (class 1) is denoted by Ξ_A and defined by the union of all the elementary sets as follows:

$$\Xi_{\mathrm{A}}(X) = \{x_i \in U \mid [x_i]_{\Xi} \subseteq X\}, \tag{7.2}$$

where x_i, $i = 1, 2, \ldots, n$, is the union of all the elementary sets and each of them is classified as precisely belonging to the subset X. Second, the upper approximation of X (classes 1 and 3) is denoted by Ξ^{A} and defined by the union of all the elementary sets as follows:

$$\Xi^{\mathrm{A}}(X) = \{x_i \in U \mid [x_i]_{\Xi} \cap X \neq \phi\}, \tag{7.3}$$

where x_i, $i = 1, 2, \ldots, n$, is the union of all the elementary sets and each of them is classified as possibly belonging to the subset X. Finally, the boundary region of X (class 3) is denoted by Ξ^{B} and defined by the union of all the elementary sets as follows:

$$\Xi^{\mathrm{B}}(X) = \Xi^{\mathrm{A}}(X) - \Xi_{\mathrm{A}}(X), \tag{7.4}$$

where the elementary sets were not found to be classified for certain. If Ξ^{A} is equal to Ξ_{A}, then (1) this set is not the rough set, and it becomes the standard set; and (2) the boundary region of X is also non-existent [142, 149, 152, 153].

The primary focus of this chapter's study is on subsets of the lower approximation, which should be determined for each of all the subsets of the decision table. In the present computational algorithm, a discernibility matrix is often used to determine subsets of the lower approximation for each decision value of the decision attribute. The rough DNA-based algorithm will first determine all of the subsets of the lower approximation before all minimal length decision rules are computed.

7.3.2 Information system

A formal model of an attribute-value system is called an *information system*, which is known as a synergetic representation of knowledge in a table that refers to the information system through rough sets. Occasionally, an information system is also termed an *information table*. The information system, which is represented in the information table, shows what kinds of attributes belong to which proposed object.

An information system is denoted by *IS*, defined as *IS* = (U, Γ_t, ω), and consists mainly of two sets, corresponding to both U and Γ_t, where U is a finite set of objects, Γ_t is a finite set of attributes, and ω is a component that is used for making value assignments distinct. An attribute element is denoted as ζ_t. Γ_t is composed of n attribute elements $\zeta_{t1}, \zeta_{t2}, \ldots, \zeta_{tn}$, corresponding to $\Gamma_t = \{\zeta_{t1}, \zeta_{t2}, \ldots, \zeta_{tn}\}$. Assuming that a pair ($\zeta_t$, u) occurs in the information system, then ζ_t represents an attribute, u is meant to be an object, and a value $\zeta_t(u)$ is also assumed. If $\omega(u, \zeta_t)$ is equal to $\zeta_t(u)$, then a mapping that is of the form ω: $U \times \Gamma_t \rightarrow I^v$, where I^v is a set of attribute values in the information system, is defined as follows:

$$I^v = \bigcup \{I^v{}_{\zeta t} \mid \zeta_t \in \Gamma_t\}, \tag{7.5}$$

where $I^v{}_{\zeta t}$ is a domain with at least two existing attributes. In this case, an attribute ζ_t is an element of the set Γ_t and x is an object, and a value $\zeta_t(x) \in \Gamma^{v}_{\zeta t}$ for each object $x \in U$ is assigned as a function for understanding each $\zeta_t \in \Gamma_t$. Thus, assuming that there are m attribute values for each attribute, they are denoted by $\psi^{\zeta t1}_{1}$, $\psi^{\zeta t1}_{2}, \ldots, \psi^{\zeta t1}_{m}$ for $\Gamma^{v}_{\zeta t1}$, $\psi^{\zeta t2}_{1}$, $\psi^{\zeta t2}_{2}, \ldots, \psi^{\zeta t2}_{m}$ for $\Gamma^{v}_{\zeta t2}$, $\cdot \cdot \cdot$, and $\psi^{\zeta tn}_{1}$, $\psi^{\zeta tn}_{2}, \ldots, \psi^{\zeta tn}_{m}$ for $\Gamma^{v}_{\zeta tn}$, respectively.

In addition, if there is another subset of attributes in an object y, representing $\Delta \subseteq \Gamma_t$ in the information system, it is defined as follows:

$$\Xi(\Delta) = \{(x, y) \mid \forall \zeta_t \in \Delta \text{ and } \zeta_t(x) = \zeta_t(y)\}, \tag{7.6}$$

and is also called an *equivalence relation* if and only if both $x \in U$ and $y \in U$. This relation is also called an *indiscernibility relation*, because x and y are practically indiscernible in the information system.

7.3.3 Model of a decision table

The information system is different from a *decision system*; hence, the information table is also different from a *decision table*. However, every so often a decision system is called a *decision table*. The main difference between the two is that the information table consists of both a finite set of objects and attributes, whereas the decision table consists of three sets, in which two of the sets are the same as the information table and an additional set is a set of decision attributes. Moreover, the decision table represents all the objects, attributes, and decision attributes, which are completely determined and resolved by proposers. In the rough DNA-based algorithm, the computation of all minimal length decision rules starts with the decision table.

A decision system is denoted by *DS* and defined as $DS = (U, \Gamma, \varepsilon, \omega)$, where Γ is a finite set of condition attributes, ε is a decision attribute, and ω is a value assignment. U is composed of objects $x_1, x_2, \ldots, x_n$, represented as $U = \{x_1, x_2, \ldots, x_n\}$. First, let us denote n condition attributes by $\zeta_1, \zeta_2, \ldots, \zeta_n$, and Γ is composed of n condition attributes, represented as $\Gamma = \{\zeta_1, \zeta_2, \ldots, \zeta_n\}$. Second, when we deal with condition attribute values for each of n condition attributes $\zeta_1, \zeta_2, \ldots, \zeta_n$, m condition attribute values are denoted by $\psi^{\zeta 1}_{1}, \psi^{\zeta 1}_{2}, \ldots, \psi^{\zeta 1}_{m}$, $\psi^{\zeta 2}_{1}, \psi^{\zeta 2}_{2}, \ldots, \psi^{\zeta 2}_{m}$, $\cdot \cdot \cdot$, and $\psi^{\zeta n}_{1}, \psi^{\zeta n}_{2}, \ldots, \psi^{\zeta n}_{m}$. Finally, n pair sets of the above both n condition attributes and m condition attribute values are denoted by $\Gamma^{c}_{\zeta 1}, \Gamma^{c}_{\zeta 2}, \ldots, \Gamma^{c}_{\zeta n}$.

The form ε: $U \rightarrow D^{v}$ is taken by the mapping case, where D^{v} is a set of decision values in the decision system. The value assignment ω encompasses the decision attribute ε, expressed as ω: $U \times (\Gamma \cup \{\varepsilon\}) \rightarrow \Gamma^{v} \cup D^{v}$, where $\{\varepsilon\}$ is a set of decision attributes denoted by E, and represented as $E = \{\varepsilon\}$, which is handled by a single decision attribute. Thus, a pair (ε, u) belongs to D^{v}, which is meant to fulfil all of the value assignments. Since a set of all the attributes actually represents $\Gamma_t = \Gamma \cup \{\varepsilon\}$ in the case where the decision part has been added. We assume n decision values of the decision attribute that are denoted by $\tau_1, \tau_2, \ldots, \tau_n$ and a set of decision values that corresponds to $D^{v} = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Thus, in the given n-decision value, n decision classes can be denoted by $D^{v}_{\tau 1}$, $D^{v}_{\tau 2}, \ldots, D^{v}_{\tau n}$.

A model of a decision table is shown in Table 7.1, which was used in this research to minimise decision rules in rough sets. For example, this decision table is composed of (1) eight different proposed objects; (2) four different types of condition attributes with

Table 7.1. Model decision table composed of 8 objects and 4 types of condition attributes with attribute values.

Object	Colour	Design	Function	Price	Decision Class
Object-1	(C, 4)	(D, 3)	(F, 2)	(P, 1)	Product-2
Object-2	(C, 3)	(D, 4)	(F, 4)	(P, 2)	Product-2
Object-3	(C, 1)	(D, 1)	(F, 1)	(P, 3)	Product-1
Object-4	(C, 4)	(D, 3)	(F, 3)	(P, 2)	Product-1
Object-5	(C, 2)	(D, 2)	(F, 1)	(P, 1)	Product-2
Object-6	(C, 3)	(D, 4)	(F, 4)	(P, 2)	Product-1
Object-7	(C, 4)	(D, 1)	(F, 2)	(P, 3)	Product-2
Object-8	(C, 2)	(D, 2)	(F, 1)	(P, 1)	Product-1

attribute values; and (3) one decision attribute with decision values. Each set of elements in Table 7.1 is represented as follows:

(1) U = {Object-1, Object-2,…, Object-8};
(2) Γ = {Colour (C), Design (D), Function (F), Price (P)};
(3) I^{c}_{Colour} = {(C, (blue = 1)), (C, (purple = 2)), (C, (red = 3)),
(C, (yellow = 4))};
(4) I^{c}_{Design} = {(D, (ancient = 1)), (D, (environmental = 2)), (D, (industrial = 3)),
(D, (modern = 4))};
(5) I^{c}_{Function} = {(F, (simple = 1)), (F, (a bit simpler = 2)),
(F, (a bit more complex = 3)), (F, (complex = 4))};
(6) I^{c}_{Price} = {(P, (inexpensive = 1)), (P, (medium = 2)),
(P, (expensive = 3))}; and
(7) D^{v} = {Product-1, Product-2} (in this case, Product-1 corresponds to
Decision Class-1 and Product-2 corresponds to Decision Class-2).

7.4 Present Computational Algorithm

When classifying the decision data from a given decision system, the existing rules are hard to treat properly under the established rules and hard to reduce for the simplification of decision making. For that reason, resolving specific rules and reducing those rules are two important processes required before concluding and estimating some characteristics of the decision table in the decision-making process. In this section, the discernibility matrix and decision matrix of the present computational algorithm [146-149] for reductions are described by using the decision table model.

7.4.1 Discernibility matrix

In a decision table, a discernibility matrix is first used to decide the simplified decision rules, which will permit an expression of each subset for the lower approximations. The discernibility matrix consists mainly of sets of the condition attributes between two different objects. A decision system *DS* defines a discernibility matrix, denoted by *S*, which is a symmetric $n \times n$ matrix composed of n rows and n columns. Each entry of two objects has its own row and column. For the model decision table, a set of eight objects represents U = {Object-1, Object-2,…, Object-8} and is composed of eight rows and eight columns. The discernibility matrix *S* with entries $s_{i,j}$, i and $j = 1, 2, \ldots, n$ is defined as follows:

$$s_{i,j} = \{\zeta \in \Gamma \mid \zeta(x_i) \neq \zeta(x_j)\}, \tag{7.7}$$

where between the objects x_i and x_j, i and $j = 1, 2, \ldots, n$ are discerned by sets of condition attributes underlying two such objects that are $x_i \in U$ and $x_j \in U$.

The discernibility matrix *S*, with all of each entry between arbitrary objects x_i and x_j, is denoted by $S(x_i, x_j)$, where each consists of the set of condition attributes for $x_i \in U$ and $x_j \in U$. A single object or a set of objects dealing with a distinct subset of the objects in a universe is discerned by a function, which is called a *discernibility function*, denoted by $\kappa_\Gamma(x_i)$. The object x_i is an element of a finite set U from the above discernibility matrix, and the discernibility function is defined as follows:

$$\kappa_\Gamma(x_i) = \prod_{x_j \in U} \left\{ \sum_{h=1}^{n} \zeta_h^* \mid \zeta \in S(x_i, x_j) \text{ and } S(x_i, x_j) \neq \phi \right\}, \tag{7.8}$$

where $\zeta^*_1, \zeta^*_2, \ldots, \zeta^*_n$ represent the Boolean variables that correspond to condition attributes $\zeta_1, \zeta_2, \ldots, \zeta_n$, respectively. A specific condition attribute discriminates between two different objects, and each binding of $\kappa_\Gamma(x_i)$ comes from the object x_j that is also an element of a finite set U. In addition, the entire discernibility function with all the objects is also defined as follows:

$$\lambda_\Gamma(U) = \prod_{x_i \in U} \kappa_\Gamma(x_i), \tag{7.9}$$

where this entire function consists of all of the given objects x_i in a finite set U, each of which needs to be discerned distinctly from the other.

7.4.2 Decision matrix

Both the discernibility and decision matrix have the same purpose. They are structured and employed to provide the reduced sets of decision rules while carrying on all principal information. Although the two matrices have the same purpose, each matrix has different properties. First, only one discernibility matrix exists if a single set of condition attributes is given, whereas multiple decision matrices exist in a decision system. Second, the discernibility matrix is created based on computed subsets of the lower approximation from the decision table, whereas the decision matrices are created in each decision value of the decision attribute. Third, the discernibility matrix consists of sets of the condition attributes between two different objects, whereas each decision matrix comprises an object subset of the lower approximation in rows and its complementary object subset of the present decision class in columns.

Recall that n multiple decision classes are $D^{v}_{\tau k}$, $k = 1, 2,\ldots, n$, and a set of decision attributes is $E = \{\varepsilon\}$ based on the decision system $DS = (U, \Gamma, \varepsilon, \omega)$ used to describe the multiple decision matrices. The decision matrix is denoted as C with each entry between objects x_i and x_j, where all objects of each x_i correspond to the objects from the subsets of the lower approximations, and all objects of each x_j correspond to the complementary objects of the present decision class. Thus, the decision matrix C with entries $c_{i,j}$, i and $j = 1, 2,\ldots, n$ and a set of entries is defined as follows:

$$c_{i,j} = \{(\zeta, \zeta(x_i)) \mid \zeta(x_i) \neq \zeta(x_j)\}, \tag{7.10}$$

where $\zeta(x_i)$ in this case is a condition attribute value of the condition attribute ζ for object x_i (viz. each entry corresponds to a pair of both a condition attribute and a condition attribute value, and $\zeta(x)$ corresponds to ψ^{ζ}) and also where x_i and x_j are defined for all $x_i \in U$ and $x_j \in U$, i and $j = 1, 2,\ldots, n$, in which the subscripts i and j are defined for $i \in P^{i}_{\tau k}$ and $j \in P^{j}_{\tau k}$. The subset of subscript $P^{i}_{\tau k}$ contains an object subset of the lower approximation, and $P^{j}_{\tau k}$ contains its complementary objects of the current decision class. Thus, a set of all the pairs in the decision matrix $c_{i,j}$ is expressed as

$$c_{i,j} = \{(\zeta_1, \zeta_1(x_i)), (\zeta_2, \zeta_2(x_i)), \cdots, (\zeta_n, \zeta_n(x_i))\}. \tag{7.11}$$

If (7.11) is truly made up of a Boolean expression, denoted by $\Psi(c_{i,j})$ for object x_i, then any condition attribute values of object x_j are not consonant with any condition attribute values of object x_i, represented as

$$CB_j(c_{i,j}) = \bigwedge_{j \in P^{j}_{\tau k}} \Psi(c_{i,j}), \tag{7.12}$$

which signifies all the pair sets of the Boolean expression for $j \in P^{j}_{\tau k}$. In this instance, assuming a Boolean expression of both objects x_i and x_j in the decision table can also be represented as

$$CB^{i}{}_{j}(c_{i,j}) = \bigvee_{i \in P^{i}_{\tau k}} \bigwedge_{j \in P^{j}_{\tau k}} \Psi(c_{i,j}). \tag{7.13}$$

If this expression is determined to become a simplified disjunctive normal form, then all minimal lengths of each decision class, including condition attributes and value pairs, can be found in order to decide on the minimal size of decision rules.

7.5 New Computational Algorithm

In designing the rough DNA-based algorithm, constructing a binary adjacency matrix (the meaning of the binary adjacency matrix was described in Chapter 4) is the main concept through encoding DNA sequences. The binary adjacency matrix can be constructed from the digraph network. Subsequently, several kinds of molecular engineering techniques are needed to use the encoded DNA sequences and employed to implement the rough DNA-based algorithm, which is used for (1) finding subsets of the lower approximations in each decision class; and (2) determining all minimal length decision rules based on results from (1). The rough DNA-based algorithm for minimising decision rules is described in this section, and we elaborate on how to execute its own molecular experimentation.

7.5.1 Digraph setting

A digraph includes n nodes, which are classified into two distinct categories of element nodes, corresponding to (1) an object element node; and (2) a node of a pair element for both a condition attribute and a condition attribute value (abbreviated as a pair element node). In the same way as the decision table, again we assume (1) n given object elements that are $x_1, x_2,\ldots, x_n$, and a set of these elements that is U, represented as $U = \{x_1, x_2,\ldots, x_n\}$ for the object element node; and (2) n given pair elements that consist of both n condition attributes $\zeta_1, \zeta_2,\ldots, \zeta_n$ and m condition attribute values $\psi^{\zeta 1}{}_1, \psi^{\zeta 1}{}_2,\ldots, \psi^{\zeta 1}{}_m, \psi^{\zeta 2}{}_1, \psi^{\zeta 2}{}_2,\ldots, \psi^{\zeta 2}{}_m, \cdots$, and $\psi^{\zeta n}{}_1, \psi^{\zeta n}{}_2,\ldots, \psi^{\zeta n}{}_m$, and a set of these pair elements that is denoted as P_s, represented as a pair matrix and denoted by P as follows:

$$P = \begin{bmatrix} P(\zeta_1,\psi^{\zeta_1}{}_1) & P(\zeta_1,\psi^{\zeta_1}{}_2) & P(\zeta_1,\psi^{\zeta_1}{}_3) & \cdots & P(\zeta_1,\psi^{\zeta_1}{}_m) \\ P(\zeta_2,\psi^{\zeta_2}{}_1) & P(\zeta_2,\psi^{\zeta_2}{}_2) & P(\zeta_2,\psi^{\zeta_2}{}_3) & \cdots & P(\zeta_2,\psi^{\zeta_2}{}_m) \\ P(\zeta_3,\psi^{\zeta_3}{}_1) & P(\zeta_3,\psi^{\zeta_3}{}_2) & P(\zeta_3,\psi^{\zeta_3}{}_3) & \cdots & P(\zeta_3,\psi^{\zeta_3}{}_m) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P(\zeta_n,\psi^{\zeta_n}{}_1) & P(\zeta_n,\psi^{\zeta_n}{}_2) & P(\zeta_n,\psi^{\zeta_n}{}_3) & \cdots & P(\zeta_n,\psi^{\zeta_n}{}_m) \end{bmatrix}. \tag{7.14}$$

A structure of the digraph is composed of both n object elements and n pair elements, and this structure is used for a generation of the minimal decision rules. If some condition attributes have less condition attribute values than other condition attributes, then we use the symbol 'ϕ' for any of the empty entries. Figure 7.1 shows a DNA-digraph (the meaning of the DNA-digraph was described in Chapter 4), created for the rough DNA-based algorithm.

For the new computational calculation, an example (see Figure 7.11(a)) shows a model digraph that has been constructed based on the model decision table. The model digraph consists of eight objects, four condition attributes, and each condition attribute has at least two or more different condition attribute values. The eight object elements are represented as $U = \{x_1, x_2,\ldots, x_8\}$, and the four condition attributes are represented as $\zeta_1, \zeta_2,\ldots, \zeta_4$. Each of the condition attributes, corresponding to ζ_1, ζ_2, and ζ_3, has four condition attribute values, and the remaining condition attribute, corresponding to ζ_4, has three condition attribute values. A pair matrix of these pair element nodes can be denoted by P, expressed as follows:

$$P = \begin{bmatrix} P(\zeta_1,\psi^{\zeta_1}{}_1) & P(\zeta_1,\psi^{\zeta_1}{}_2) & P(\zeta_1,\psi^{\zeta_1}{}_3) & P(\zeta_1,\psi^{\zeta_1}{}_4) \\ P(\zeta_2,\psi^{\zeta_2}{}_1) & P(\zeta_2,\psi^{\zeta_2}{}_2) & P(\zeta_2,\psi^{\zeta_2}{}_3) & P(\zeta_2,\psi^{\zeta_2}{}_4) \\ P(\zeta_3,\psi^{\zeta_3}{}_1) & P(\zeta_3,\psi^{\zeta_3}{}_2) & P(\zeta_3,\psi^{\zeta_3}{}_3) & P(\zeta_3,\psi^{\zeta_3}{}_4) \\ P(\zeta_4,\psi^{\zeta_4}{}_1) & P(\zeta_4,\psi^{\zeta_4}{}_2) & P(\zeta_4,\psi^{\zeta_4}{}_3) & \phi \end{bmatrix}. \tag{7.15}$$

The above pair matrix will be used mainly when encoding double-encoded substrings (the meaning of double-encoded substrings was described in Chapter 4) for type 3 and complementary substrings for type 6.

We have described above the two preceding distinctively different types of element nodes, which correspond to the same type of nodes in the DNA-digraph, both of which are simply expressed in this chapter's study as elements. In the DNA-digraph, the current relation between any of two elements is denoted by e, where e is indicated to the relation of a direct line that connects those two nodes. We denote three different types of relational directions between two elements: (1) if an object element node x_i has a direct relation with a pair element node $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$, represented as $x_ie(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$, and $x_i\bar{e}(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$ represents no direct relation; (2) if a pair element node $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$ has a direct relation with an object element node x_j, represented as $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)ex_j$, and $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)\bar{e}x_j$ represents no direct relation; and (3) if a pair element node $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$ has a direct relation with a pair element node $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$, represented as $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)e(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$, and $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)\bar{e}(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$ represents no direct relation.

Three types of relational arcs are sought to represent in the DNA-digraph the arcs among element nodes: (1) when an object element node x_i contains a directional arrow that reaches a pair element node $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$, expressing

$$b^a{}_l = \overrightarrow{(x_i,(\zeta_\alpha,\psi^{\zeta_\alpha}{}_\alpha))},\ i,\ \alpha, \text{ and } l = 1, 2, \cdots, n\,; \tag{7.16}$$

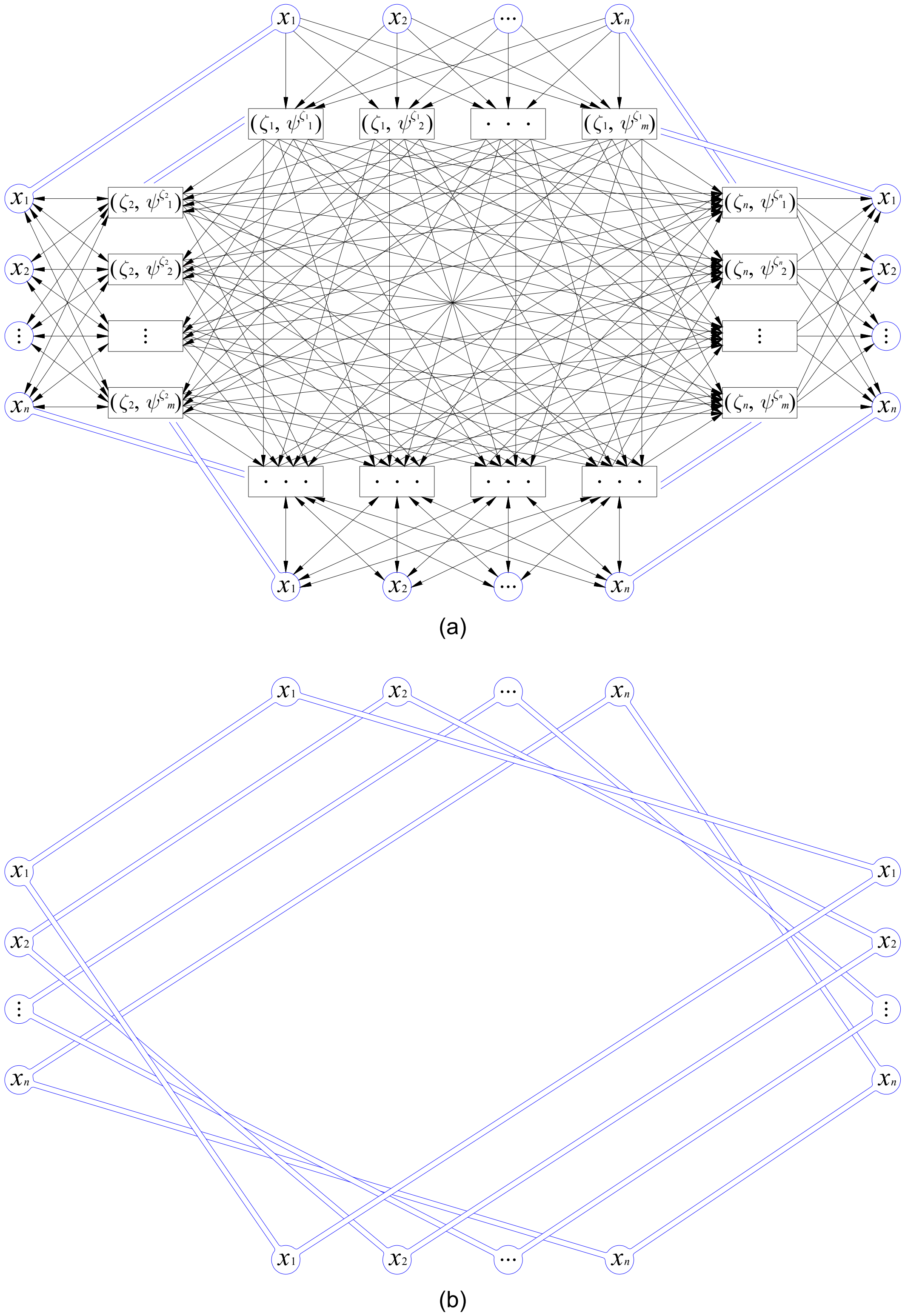
(2) when a pair element node $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$ contains a directional arrow that reaches an object element node x_j, expressing

$$b^b{}_l = \overrightarrow{((\zeta_\beta,\psi^{\zeta_\beta}{}_\beta),x_j)},\ \beta,\ j, \text{ and } l = 1, 2, \cdots, n\,; \tag{7.17}$$

and (3) when a pair element node $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$ contains a directional arrow that reaches a pair element node $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$, expressing

$$b^c{}_l = \overrightarrow{((\zeta_\alpha,\psi^{\zeta_\alpha}{}_\alpha),(\zeta_\beta,\psi^{\zeta_\beta}{}_\beta))},\ \alpha,\ \beta, \text{ and } l = 1, 2, \cdots, n \text{ for } \alpha \neq \beta\,. \tag{7.18}$$

Additionally, (1) a subset of the arcs from the object element node x_i to the pair element node $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$ is denoted by B^a, where B^a consists of all the possible arcs, represented as $B^a = \{b^a{}_1, b^a{}_2,\ldots, b^a{}_n\}$; (2) a subset of the arcs from the pair element node $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$ to the object element node x_j is denoted as B^b, where B^b consists of all the possible arcs, represented as $B^b = \{b^b{}_1, b^b{}_2,\ldots, b^b{}_n\}$; and (3) a subset of the arcs from the pair element node $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$ to the pair element node $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$ is denoted as B^c, where B^c consists of all the possible arcs, represented as $B^c = \{b^c{}_1, b^c{}_2,\ldots, b^c{}_n\}$. A set of the subsets B^a, B^b,


x_1
x_2
$\cdots$
x_n
$(\zeta_1, \psi^{\zeta_1}_1)$
$(\zeta_1, \psi^{\zeta_1}_2)$
$\cdot\ \cdot\ \cdot$
$(\zeta_1, \psi^{\zeta_1}_m)$
$(\zeta_2, \psi^{\zeta_2}_1)$
$(\zeta_2, \psi^{\zeta_2}_2)$
$\vdots$
$(\zeta_2, \psi^{\zeta_2}_m)$
$(\zeta_n, \psi^{\zeta_n}_1)$
$(\zeta_n, \psi^{\zeta_n}_2)$
$(\zeta_n, \psi^{\zeta_n}_m)$

(a)

(b)

Figure 7.1. DNA-digraph for minimising decision rules: (a) relations among *n* object element nodes and *n* pair element nodes; (b) the representation of connections in each *n* object element node, one by one.

Type 1

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	(C, 1)	(C, 2)	(C, 3)	(C, 4)	(D, 1)	(D, 2)	(D, 3)	(D, 4)	(F, 1)	(F, 2)	(F, 3)	(F, 4)	(P, 1)	(P, 2)	(P, 3)
x_1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0
x_2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
x_3	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0
x_4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0
x_5	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0
x_6	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
x_7	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
x_8	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0
(C, 1)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1
(C, 2)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0
(C, 3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0
(C, 4)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	1
(D, 1)	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
(D, 2)	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
(D, 3)	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
(D, 4)	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
(F, 1)	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
(F, 2)	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
(F, 3)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
(F, 4)	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
(P, 1)	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(P, 2)	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(P, 3)	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Type 3

Type 2

Figure 7.2. Binary adjacency matrix of the model DNA-digraph.

and B^c is denoted by B, where B consists of the entire existing arcs in the DNA-digraph and is represented as $B = B^a \cup B^b \cup B^c$.

A binary adjacency matrix can be represented by using (1) the two sets of object element nodes U and the pair element nodes P_s; and (2) the three different types of the arc subsets B^a, B^b, and B^c. Figure 7.2 illustrates the binary adjacency matrix of the model DNA-digraph. The model binary adjacency matrix is composed of rows and columns labelled as follows:

$$r_{i,j}, i \text{ and } j = 1, 2, \cdots, 23 \text{ for all } (i, j) \in L, \tag{7.19}$$

where L is the set of all possible row and column labels in the model binary adjacency matrix. The matrix can be constructed by setting $r_{i,j} = 1$ wherever there are any three different types of arcs in the example DNA-digraph. For $r_{i,j} = 1$, the first arc is directed from an object element node x_i to a pair element node (ζ_α, $\psi^{\zeta\alpha}{}_\alpha$), the second arc is directed from a pair element node (ζ_β, $\psi^{\zeta\beta}{}_\beta$) to an object element node x_j, and the third arc from a pair element node $(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$ to a pair element node $(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$, meaning $x_i e(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$, $(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)ex_j$, and $(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)e(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$, respectively, while the matrix is also constructed by setting $r_{i,j} = 0$ elsewhere, meaning $x_i\bar{e}(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$, $(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)\bar{e}x_j$, and $(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)\bar{e}(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$, respectively.

7.5.2 DNA encoding process

A directional matrix (a binary adjacency matrix becomes the directional matrix in the term of the rough DNA-based algorithm) is used for encoding DNA sequences. In a systematic way, ordering rows and columns of the directional matrix, which includes the object and pair elements, can be transformed to DNA sequences.

The model DNA-digraph of the directional matrix of size $n \times n$ becomes 23 rows and 23 columns (8 object element nodes and 15 pair element nodes), as shown in Figure 7.2. Each directional order of two element nodes in the model digraph implies its own row and column. The rows and columns in the model directional matrix are labelled as $r_{i,j}$, i and $j = 1, 2, \ldots, 23$. We recall that if there is a single arc from either an object element node or a pair element node to the other object element node or the other pair element node, then entries become that $r_{i,j} = 1$; in contrast, if there is no direction associated with each element node or $r_{i,j}$, $i = j$, then entries become that $r_{i,j} = 0$. As shown in Figures 7.3 to 7.9, seven different types are generated to construct an initial library of DNA fragments for the process of the rough DNA-based algorithm. Each of the seven different types involves its own row and column labels, which are defined for encoding both object and pair element nodes in forms of single-stranded DNA (ssDNA). Types 1, 2, and 3 are generated from the given DNA substrings, and types 4, 5, 6, and 7 are generated from their complementary substrings of two different given DNA substrings. All the types of DNA sequences of both double-encoded substrings and complementary substrings are basically structured in ssDNA.

First, type 1 expresses a double-encoded substring that is indicated to each of all the possible two different single element nodes x_i and $(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$ in the digraph. In other words, for type 1, the two different types of element nodes, in which a direction of the arrow indicates the direction from the object element node x_i to the pair element node

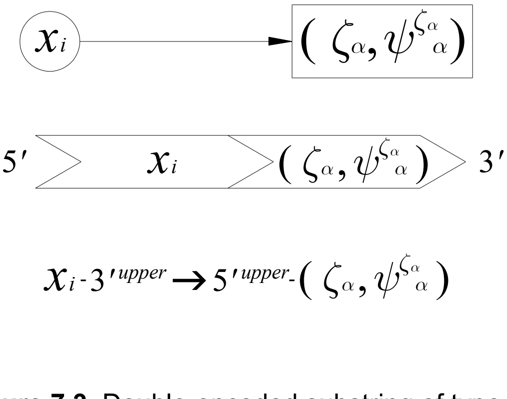

x_i
$(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$
5′
x_i
$(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$
3′
x_i-3′upper → 5′upper-$(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$

Figure 7.3. Double-encoded substring of type 1.

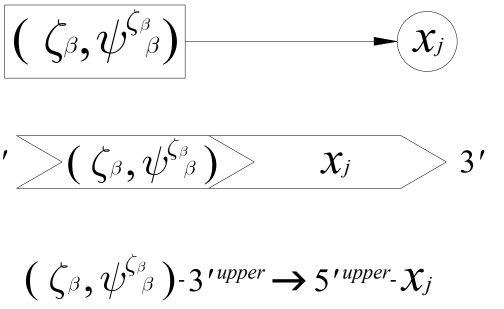

$(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$
x_j
$5'$
$(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$
x_j
$3'$
$(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$-$3'^{upper} \rightarrow 5'^{upper}$-$x_j$

Figure 7.4. Double-encoded substring of type 2.

$(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$, are denoted as a double-encoded substring x_i-$3'^{upper}$→$5'^{upper}$-$(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$, as shown in Figure 7.3. In the directional matrix of the model DNA-digraph (see Figure 7.2), for type 1, all of the row and column labels are denoted by i and j, and the entries are defined as follows:

$$r_{i,j} = 1 \text{ for } i = 1, 2, \cdots, 8,\ j = 9, 10, \cdots, 20, \text{ and all } (i, j) \in L\,. \tag{7.20}$$

The two different element nodes are encoded as a single oligonucleotide that is composed of two unique sites, and each single encoded oligonucleotide has its own length of DNA base pairs (bp: after hybridisations and end-filling DNA). Thus, we set each object element node x_i, i = 1, 2,…, 8 with all the same length of 18 bp, and each pair element node $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$ corresponds to $5'^{upper}$-(C$\vee$D$\vee$F, 1$\vee$2$\vee$3$\vee$4) with the different length bp in each of the different pair element nodes.

Second, similar to type 1, type 2 expresses a double-encoded substring that is indicated to each of all two different single element nodes $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$ and x_j in the digraph. For type 2, the two different types of element nodes, in which a direction of the arrow also indicates the direction from the pair element node $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$ to the object element node x_j, are denoted as a double-encoded substring $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$-$3'^{upper}$→$5'^{upper}$-$x_j$, as shown in Figure 7.4. In the directional matrix of the model DNA-digraph, for type 2, all of the row and column labels are denoted by i and j, and the entries are defined as follows:

$$r_{i,j} = 1 \text{ for } i = 13, 14, \cdots, 23,\ j = 1, 2, \cdots, 8, \text{ and all } (i, j) \in L\,. \tag{7.21}$$

The two different element nodes are also encoded as a single oligonucleotide that is composed of two unique sites and has its own length of DNA bp as type 1. Thus, each pair element node $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$ corresponds to (D$\vee$F$\vee$P, 1$\vee$2$\vee$3$\vee$4)-$3'^{upper}$ with the different length bp in each of the different pair element nodes, and we set each object element node x_j, j = 1, 2,…, 8 with all the same length of 18 bp.

Third, type 3 also expresses a double-encoded substring that is indicated to each of all two different single pair element nodes $(\zeta_\alpha, \psi^{\zeta\alpha}{}_\alpha)$ and $(\zeta_\beta, \psi^{\zeta\beta}{}_\beta)$ in the digraph. The

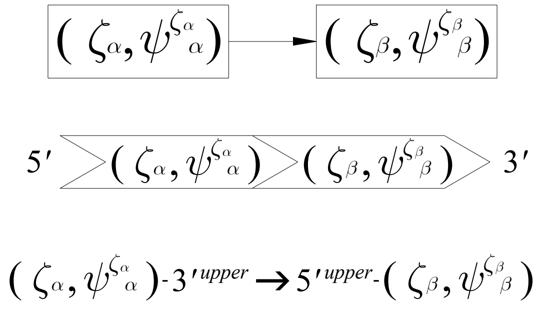

$(\zeta_\alpha, \psi^{\zeta_\alpha}_{\alpha})$
$(\zeta_\beta, \psi^{\zeta_\beta}_{\beta})$
5′
$(\zeta_\alpha, \psi^{\zeta_\alpha}_{\alpha})$
$(\zeta_\beta, \psi^{\zeta_\beta}_{\beta})$
3′
$(\zeta_\alpha, \psi^{\zeta_\alpha}_{\alpha})\text{-}3'^{upper} \rightarrow 5'^{upper}\text{-}(\zeta_\beta, \psi^{\zeta_\beta}_{\beta})$

Figure 7.5. Double-encoded substring of type 3.

two different types of pair element nodes for type 3, in which there is a direction of the arrow between only pair element nodes and this arrow indicates the direction from the pair element node (ζ_α, $\psi^{\zeta\alpha}{}_{\alpha}$) to the other pair element node (ζ_β, $\psi^{\zeta\beta}{}_{\beta}$), are denoted as a double-encoded substring (ζ_α, $\psi^{\zeta\alpha}{}_{\alpha}$)-$3'^{upper}$→$5'^{upper}$-($\zeta_\beta$, $\psi^{\zeta\beta}{}_{\beta}$), as shown in Figure 7.5. In the directional matrix of the model DNA-digraph, for type 3, all of the row and column labels are denoted by i and j, and the entries are defined as follows:

$$r_{i,j} = 1 \text{ for } i = 9, 10, \cdots, 20,\ j = 13, 14, \cdots, 23, \text{ and all } (i, j) \in L\,. \qquad (7.22)$$

The two different pair element nodes are encoded as a single oligonucleotide that is composed of two unique sites, and each single encoded oligonucleotide has its own length of DNA bp as types 1 and 2. Thus, each pair element node (ζ_α, $\psi^{\zeta\alpha}{}_{\alpha}$) corresponds to (C$\vee$D$\vee$F, 1$\vee$2$\vee$3$\vee$4)-$3'^{upper}$ with the different length bp in each of the different pair element nodes, and each element node (ζ_β, $\psi^{\zeta\beta}{}_{\beta}$) corresponds to $5'^{upper}$-(D$\vee$F$\vee$P, 1$\vee$2$\vee$3$\vee$4) with the different length bp in each of the different pair element nodes.

Types 4, 5, 6, and 7 express complementary substrings between two of the double-encoded substrings (types 1 and 3, types 3 and 2, types 3 and 3, and types 2 and 1), since the left double-encoded substring in a 5′ to 3′ direction has the ending DNA sequence that indicates the element node, and the right double-encoded substring in a 5′ to 3′ direction has the starting DNA sequence that indicates the same element node that the ending DNA sequences of the left one indicates. The forces of both hybridisations and DNA ligations [154-157] make all three different DNA substrings (these three DNA substrings are indicated to three different element nodes, which are linked by directions) connect to each other so that they are sequentially aligned and become forms of a double-stranded DNA (dsDNA). For attaching three different element nodes containing a pair element node located in the middle of the three nodes, types 4, 5, and 6 have the same purposes as their complementary substrings. Particularly, for attaching three different element nodes containing an object element node located in the middle of the three nodes, type 7 has its own purpose as a complementary substring.

Fourth, in the digraph, type 4 was generated for the attachment of the three different

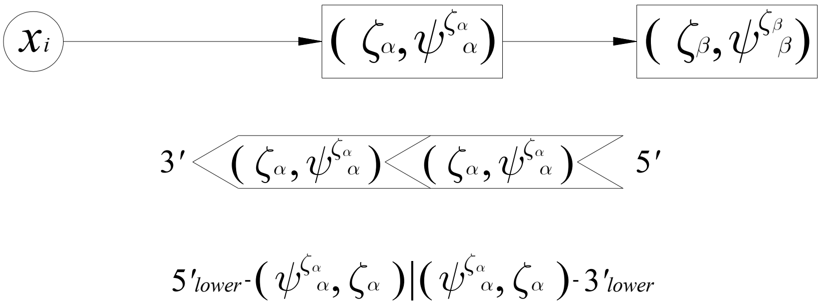

x_i
$(\zeta_\alpha, \psi^{\zeta_\alpha}_{\alpha})$
$(\zeta_\beta, \psi^{\zeta_\beta}_{\beta})$
3′
$(\zeta_\alpha, \psi^{\zeta_\alpha}_{\alpha})$
$(\zeta_\alpha, \psi^{\zeta_\alpha}_{\alpha})$
5′
$5'_{lower}$-$(\psi^{\zeta_\alpha}_{\alpha}, \zeta_\alpha)|(\psi^{\zeta_\alpha}_{\alpha}, \zeta_\alpha)$-$3'_{lower}$

Figure 7.6. Complementary substring of type 4.

linked element nodes, which are (1) an object element node (starting element node) x_i; and (2) two different pair element nodes (middle and ending element nodes) $(\zeta_\alpha, \psi^{\zeta_\alpha}_{\alpha})$ and $(\zeta_\beta, \psi^{\zeta_\beta}_{\beta})$. For type 4, a complementary substring of the middle pair element node was generated to link these three element nodes together. Type 4 is denoted as a complementary encode $5'_{lower}$-$(\psi^{\zeta_\alpha}_{\alpha}, \zeta_\alpha)|(\psi^{\zeta_\alpha}_{\alpha}, \zeta_\alpha)$-$3'_{lower}$, as shown in Figure 7.6. For type 4, $5'_{lower}$-$(1\vee 2\vee 3\vee 4, \mathrm{C}\vee \mathrm{D}\vee \mathrm{F})|(1\vee 2\vee 3\vee 4, \mathrm{C}\vee \mathrm{D}\vee \mathrm{F})$-$3'_{lower}$ corresponds to the complementary substring of the middle pair element node.

Fifth, in the digraph, type 5 was generated for the attachment of the three different linked element nodes, which are (1) two different pair element nodes (starting and middle element nodes) $(\zeta_\alpha, \psi^{\zeta_\alpha}_{\alpha})$ and $(\zeta_\beta, \psi^{\zeta_\beta}_{\beta})$; and (2) an object element node (ending element node) x_j. A complementary substring of the middle pair element node for type 5 was generated to link these three element nodes together. Type 5 is denoted as a complementary encode $5'_{lower}$-$(\psi^{\zeta_\beta}_{\beta}, \zeta_\beta)|(\psi^{\zeta_\beta}_{\beta}, \zeta_\beta)$-$3'_{lower}$, as shown in Figure 7.7. For type 5, $5'_{lower}$-$(1\vee 2\vee 3\vee 4, \mathrm{D}\vee \mathrm{F}\vee \mathrm{P})|(1\vee 2\vee 3\vee 4, \mathrm{D}\vee \mathrm{F}\vee \mathrm{P})$-$3'_{lower}$ corresponds to the complementary substring of the middle pair element node.

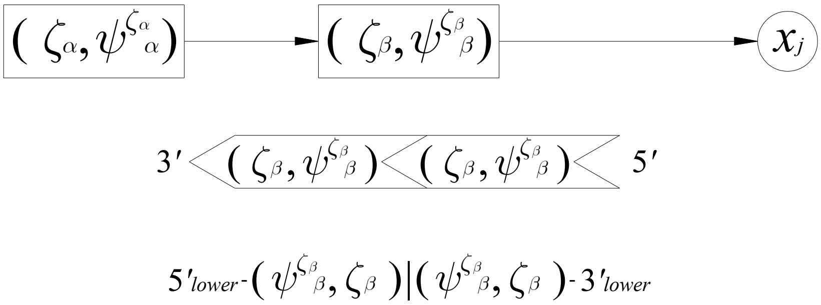

$(\zeta_\alpha, \psi^{\zeta_\alpha}_{\alpha})$
$(\zeta_\beta, \psi^{\zeta_\beta}_{\beta})$
x_j
3′
$(\zeta_\beta, \psi^{\zeta_\beta}_{\beta})$
$(\zeta_\beta, \psi^{\zeta_\beta}_{\beta})$
5′
$5'_{lower}$-$(\psi^{\zeta_\beta}_{\beta}, \zeta_\beta)|(\psi^{\zeta_\beta}_{\beta}, \zeta_\beta)$-$3'_{lower}$

Figure 7.7. Complementary substring of type 5.

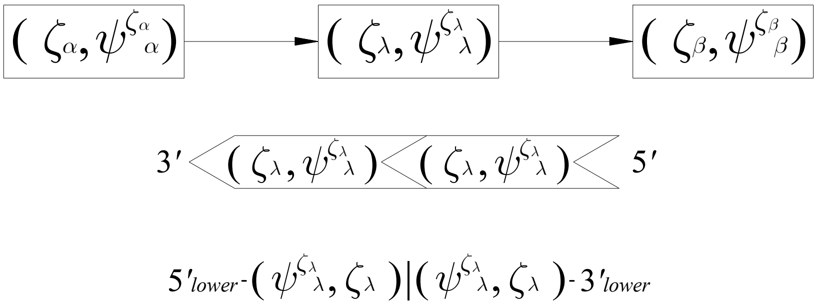

$(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$
$(\zeta_\lambda, \psi^{\zeta_\lambda}{}_\lambda)$
$(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$
3′
$(\zeta_\lambda, \psi^{\zeta_\lambda}{}_\lambda)$
$(\zeta_\lambda, \psi^{\zeta_\lambda}{}_\lambda)$
5′
$5'_{lower}$-$(\psi^{\zeta_\lambda}{}_\lambda, \zeta_\lambda)|(\psi^{\zeta_\lambda}{}_\lambda, \zeta_\lambda)$-$3'_{lower}$

Figure 7.8. Complementary substring of type 6.

Sixth, in the digraph, type 6 was generated for the attachment of the three different linked pair element nodes (starting, middle, and ending nodes), which are denoted by (ζ_α, $\psi^{\zeta\alpha}{}_\alpha$), ($\zeta_\lambda$, $\psi^{\zeta\lambda}{}_\lambda$), and ($\zeta_\beta$, $\psi^{\zeta\beta}{}_\beta$), respectively. For type 6, a complementary substring of the middle pair element node was generated to link these three pair element nodes together. Type 6 is denoted as a complementary encode $5'_{lower}$-($\psi^{\zeta\lambda}{}_\lambda$, ζ_λ)|($\psi^{\zeta\lambda}{}_\lambda$, ζ_λ)-$3'_{lower}$, as shown in Figure 7.8. For type 6, $5'_{lower}$-(1∨2∨3∨4, D∨F)|(1∨2∨3∨4, D∨F)-$3'_{lower}$ corresponds to the complementary substring of the middle pair element node.

Finally, in the digraph, type 7 is different from the three types 4, 5, and 6, described above. Type 7 was generated for the attachment of the three different linked element nodes, which are (1) two different pair element nodes (starting and ending element nodes) (ζ_β, $\psi^{\zeta\beta}{}_\beta$) and ($\zeta_\alpha$, $\psi^{\zeta\alpha}{}_\alpha$); and (2) an object element node (middle element node) denoted as x_λ. A complementary substring of the middle object element node for type 7 was generated to link these three element nodes together. Type 7 is denoted as a complementary encode $5'_{lower}$-x_λ|x_λ-$3'_{lower}$, as shown in Figure 7.9. For type 7, $5'_{lower}$-1∨2∨… ∨8|1∨2∨… ∨8-$3'_{lower}$ corresponds to the complementary substring

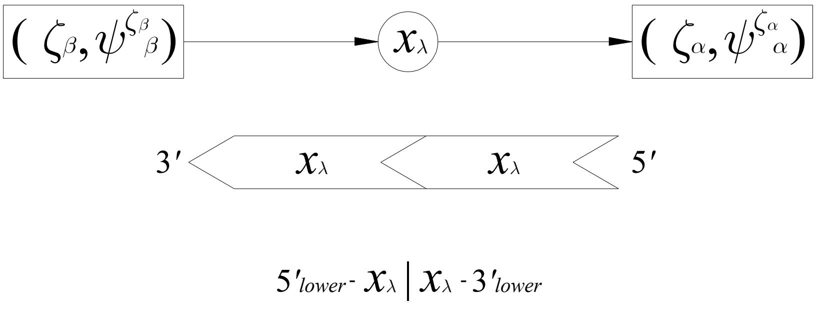

$(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$
x_λ
$(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$
3′
x_λ
x_λ
5′
$5'_{lower}$- x_λ | x_λ - $3'_{lower}$

Figure 7.9. Complementary substring of type 7.

of the middle object element node.

7.6 Experimental Studies and Results

In this chapter's study, a splicing operation model [60] is also applied to the rough DNA-based algorithm for the formation of minimising decision rules through the DNA encoding process of the proposed seven types. The splicing operation method concatenates the DNA fragments to be crosswise and linked together by a DNA ligation when there are two different encoded DNA sequences that need to be spliced.

7.6.1 Experimental studies

A program compiled using Vector NTI software was used to represent all minimal length decision rules in rough sets for the representation of the length of DNA strands. The rough DNA-based algorithm was used to minimise decision rules based on our own designed molecular computational experimentation.

For the model DNA-digraph, subsets of each object element node and pair element node are able to describe the pattern of each DNA substring, corresponding to DNA sequences. Each concatenation of DNA fragments corresponds to complementary substrings of both object and pair element nodes, which are also able to describe the pattern of each complementary substring. To encode double-encoded substrings (types 1, 2, and 3) and their complementary substrings (types 4, 5, 6, and 7), all the DNA substrings should be generated. At the same time, the fitting restriction enzymes are also added in each object element node of the DNA substrings and complementary substrings. Restriction enzymes have the main function of cleaving DNA strands in a loop. For the length measurements, all of the detected circular DNA fragments should be cleaved, denatured, and become linear DNA fragments (not circular DNA fragments). A technique of the polymerase chain reaction, which is used for the generation of DNA substrings, is associated with large quantities. For the model DNA-digraph, at least twenty eight DNA strands in each DNA substring and complementary substring.

In the model DNA-digraph, all the possible double-encoded substrings among element nodes and each complementary substring are generated to detect (1) subsets of the lower approximations in each decision class; and detect (2) all minimal length decision rules. We execute the first detection process for measuring the length of DNA strands among the given objects by using the encoded types 3 and 6. For the model decision table, as shown in Figure 7.10, the DNA fragments are associated with the given pair element nodes of each object. On the other hand, we execute the second detection process for detecting all the circular DNA fragments by using the encoded types 1, 2, 4, 5, and 7 and using types 3 and 6 (see Figure 7.13) again. The pair element nodes are connected to the object element node x_i, and they become a circular DNA fragment in the hybridisation process. The procedure of the rough DNA-based algorithm is described for the minimal decision rules in more detail as follows:

Step 1 (digraph-1)**:** A DNA-digraph can be created by understanding the relations among the given object and pair elements based on the main structure of the digraph with *n* object elements and *n* pair elements, as described and shown in Figure 7.1. The model DNA-digraph is shown in Figure 7.11(a). As shown in Figure 7.12, each DNA

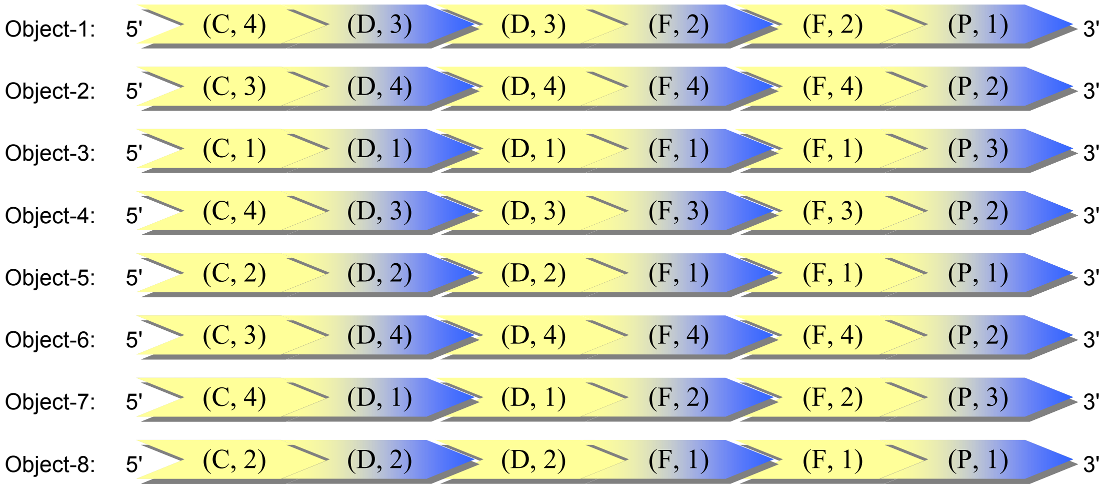

Object-1: 5' (C, 4) (D, 3) (D, 3) (F, 2) (F, 2) (P, 1) 3'
Object-2: 5' (C, 3) (D, 4) (D, 4) (F, 4) (F, 4) (P, 2) 3'
Object-3: 5' (C, 1) (D, 1) (D, 1) (F, 1) (F, 1) (P, 3) 3'
Object-4: 5' (C, 4) (D, 3) (D, 3) (F, 3) (F, 3) (P, 2) 3'
Object-5: 5' (C, 2) (D, 2) (D, 2) (F, 1) (F, 1) (P, 1) 3'
Object-6: 5' (C, 3) (D, 4) (D, 4) (F, 4) (F, 4) (P, 2) 3'
Object-7: 5' (C, 4) (D, 1) (D, 1) (F, 2) (F, 2) (P, 3) 3'
Object-8: 5' (C, 2) (D, 2) (D, 2) (F, 1) (F, 1) (P, 1) 3'

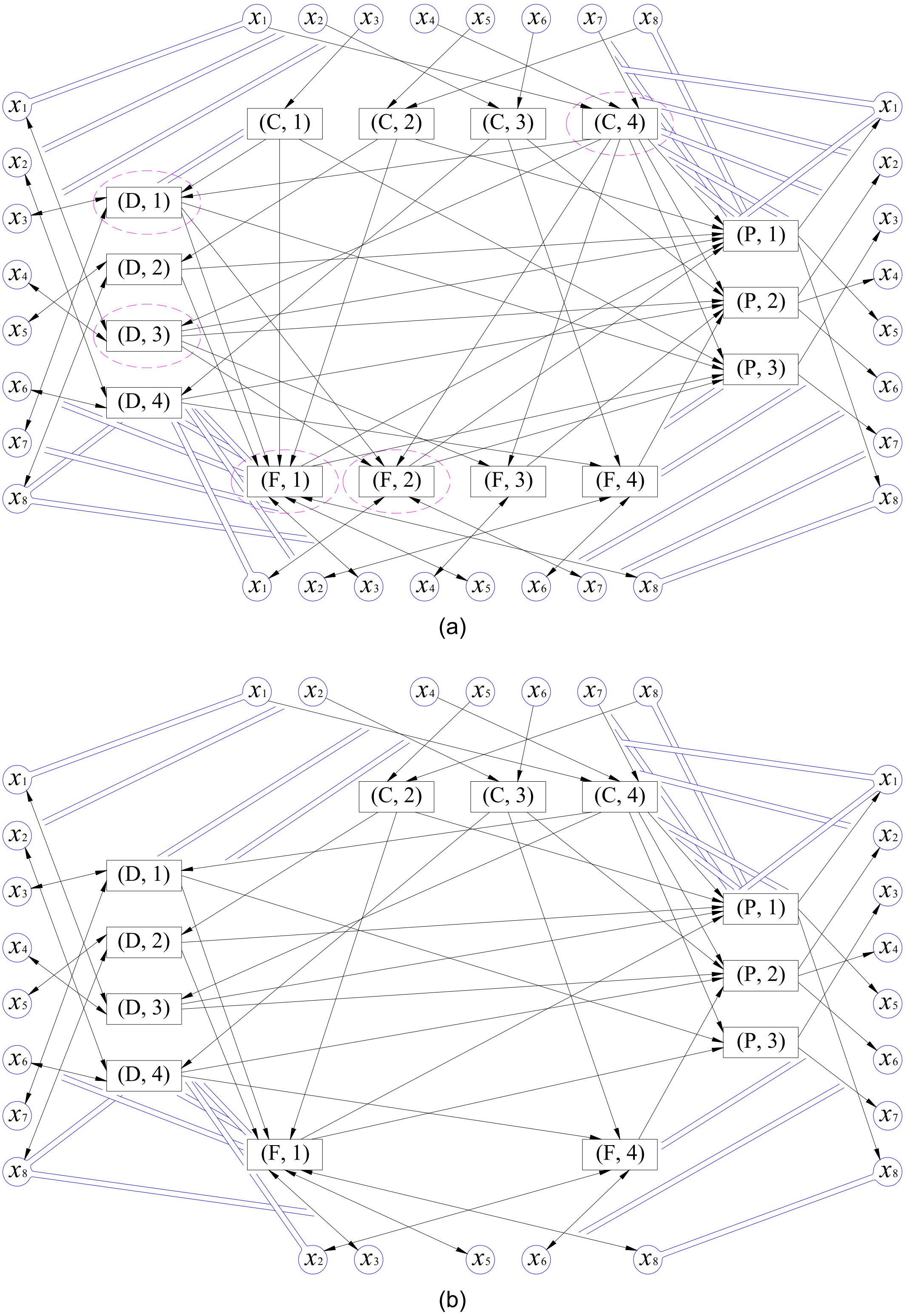
Figure 7.10. Representation of the possible arcs among the given pair element nodes of each object in the model decision table for finding out subsets of the lower approximations in each decision class.

substring of the DNA sequence for specific pair element nodes in the model DNA-digraph is composed of the arcs from the different object element nodes to the same condition attributes.

Step 2 (encoding-1)**:** For the DNA-based digraph, the double-encoded substrings and their complementary substrings are encoded based on the molecular encoding method of the directed objects and pairs as previously described. In the model decision table, DNA sequences of the existing arcs of the pair elements (see Figure 7.10) correspond to types 3 and 6, which should be first generated and encoded in ssDNA to resolve subsets of the lower approximations in each decision class.

Step 3 (hybridisation-1)**:** The entire DNA sequences of the encoded pair element node substrings (type 3) and their complementary substrings (type 6) in Step 2 are synthesised artificially and placed in a test tube. For this hybridisation process, the pair element node substrings and their complementary substrings are first heated to approximately 94˚C, and then cooled to approximately 20˚C at 1˚C/min. They are hybridised on the basis of the Watson-Crick complementary rules.

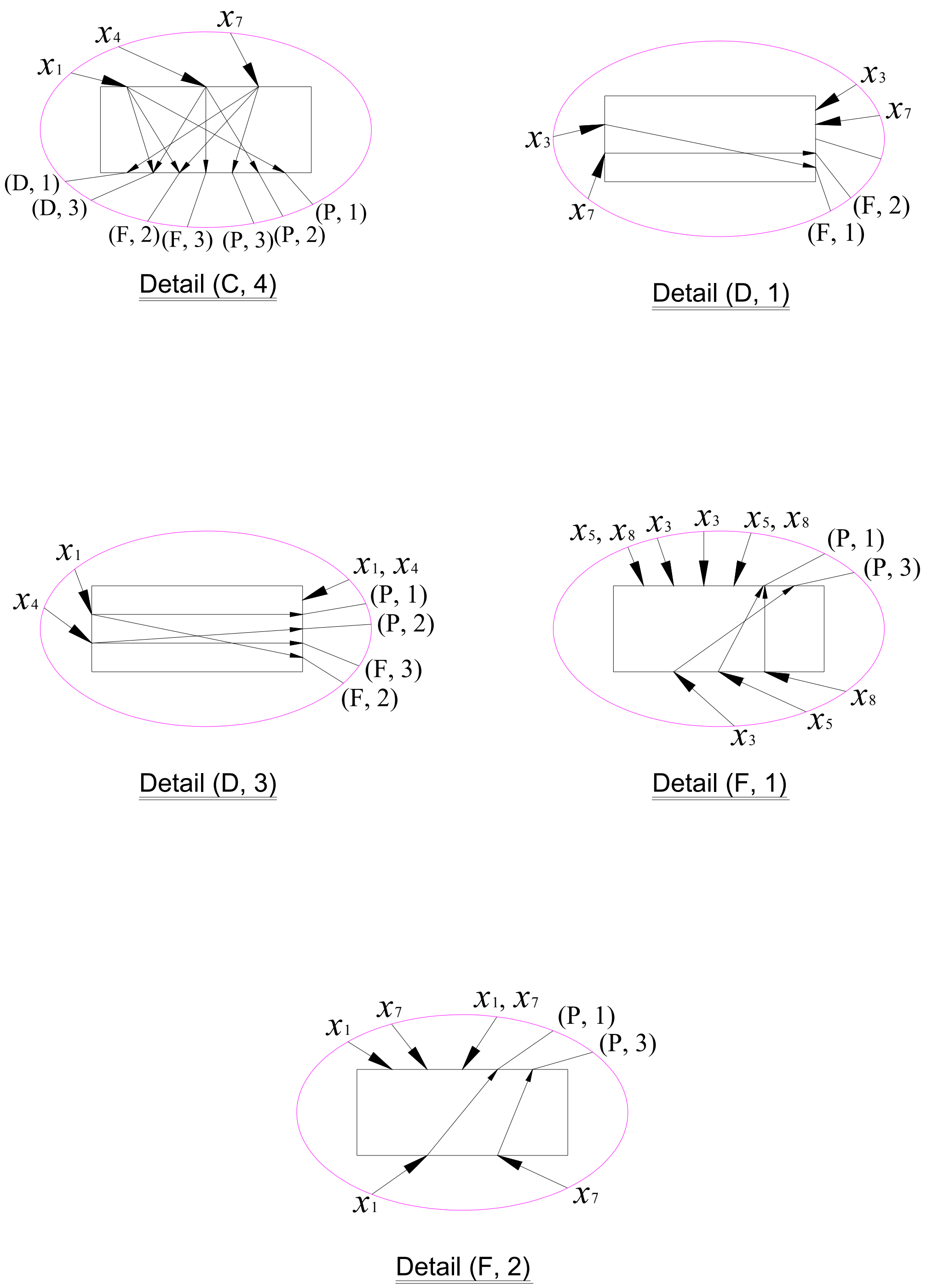
Step 4 (simulated gel electrophoresis-1 and removal-1)**:** DNA strands may be separated according to their sizes using a simulated gel electrophoresis apparatus. The specific size of the separated DNA strands is measured by comparing them to the known lengths of DNA strands. The entire lengths of the hybridised DNA strands in Step 3 should be measured by the simulated gel electrophoresis and classified into each of two decision classes for the model decision table. If there are two or more hybridised DNA strands, indicating the same length by measurement, it means that those hybridised DNA strands correspond to objects, which contain completely the same condition attribute values



(a)

(b)

Figure 7.11. Two different DNA-digraphs differing in the number of object and pair element nodes: (a) a model DNA-digraph constructed from the model decision table. The dashed circle lines are shown in more detail in Figure 7.12; (b) a new DNA-digraph constructed after removing the first condition group of decision rules.



Detail (C, 4)

Detail (D, 1)

Detail (D, 3)

Detail (F, 1)

Detail (F, 2)

Figure 7.12. Detail representations of the pair element nodes (C, 4), (D, 1), (D, 3), (F, 1), and (F, 2), in each of which the arcs are directed from the different object element nodes to the groups of the two or more pair element nodes at the same condition attribute.

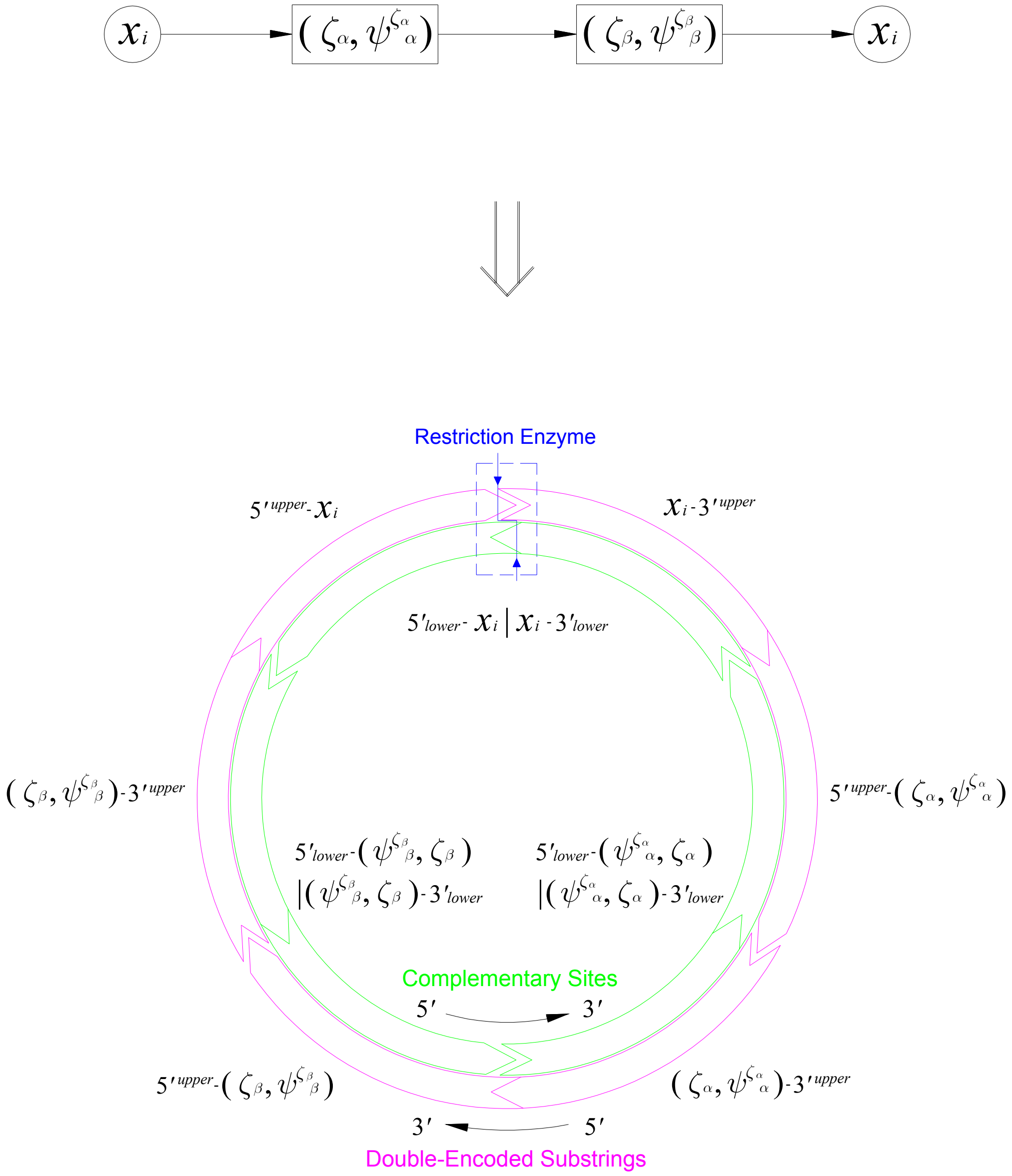

x_i
$(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$
$(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$
x_i
Restriction Enzyme
$5'^{upper}$-x_i
x_i-$3'^{upper}$
$5'_{lower}$- x_i | x_i -$3'_{lower}$
$(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$-$3'^{upper}$
$5'^{upper}$-$(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$
$5'_{lower}$-$(\psi^{\zeta_\beta}{}_\beta, \zeta_\beta)$
|$(\psi^{\zeta_\beta}{}_\beta, \zeta_\beta)$-$3'_{lower}$
$5'_{lower}$-$(\psi^{\zeta_\alpha}{}_\alpha, \zeta_\alpha)$
|$(\psi^{\zeta_\alpha}{}_\alpha, \zeta_\alpha)$-$3'_{lower}$
Complementary Sites
$5'$
$3'$
$5'^{upper}$-$(\zeta_\beta, \psi^{\zeta_\beta}{}_\beta)$
$(\zeta_\alpha, \psi^{\zeta_\alpha}{}_\alpha)$-$3'^{upper}$
$3'$
$5'$
Double-Encoded Substrings

Figure 7.13. Example representation of a circular DNA fragment for an object element node with two pair element nodes: the restriction enzyme is used for cleaving.

irrespective of decision values. This reason is that each condition attribute value has been encoded by its own length of DNA strands. For instance, if two DNA strands of the same length have been measured, but those two are not included in the same decision class but rather are included separately in each of the different decision classes despite having the same length, then those two (the same length-measured DNA strands) should be clearly removed from all the decision classes. After this process, the remaining hybridised DNA strands of each decision class can be divided into each subset of the lower approximation in each decision class. The length representation result of the first detection process in each decision class is illustrated in Figure 7.14 (lanes 1 and 2). In Figure 7.14 (lanes 1 and 2), the specific DNA strands (the same lengths) among the two given decision classes are removable groups, which cannot be included in either the lower approximation in Decision Class-1 or 2.

Step 5 (denaturing)**:** After resolving each lower approximation of each decision class, the hybridised DNA strands of types 3 and 6 in the status of dsDNA should be heated to approximately 94˚C, and they become ssDNA. The denatured DNA strands in ssDNA will be used again for the hybridisation-2 and DNA ligation process in Step 8.

Step 6 (marking-1 and digraph-2)**:** For the reduction of encoding DNA sequences, it is necessary to find some object element node or nodes, which belong only to one of the lower approximation subsets and are associated with the same decision class while they are only connecting together to a specific pair element node. In other words, any of other object element nodes, except the object element nodes meeting the above requirements, are not connected to this specific pair element node. The subsets of the arcs between objects and pair element nodes, such as those described above, are defined as

$$B_1 = \left\{ \overrightarrow{(x_i,(\zeta_\alpha,\psi^{\zeta_\alpha}{}_\alpha))} \in B^a \mid x_i \in \Xi_{\mathrm{A}}(D^v{}_\tau) \right\}, \tag{7.23}$$

which represents a subset of the arcs, each arc of which makes a direction from an object element node x_i to a pair element node (ζ_α, $\psi^{\zeta\alpha}{}_\alpha$), denoted as B_1, and

$$B_2 = \left\{ \overrightarrow{((\zeta_\beta,\psi^{\zeta_\beta}{}_\beta),x_j)} \in B^b \mid x_j \in \Xi_{\mathrm{A}}(D^v{}_\tau) \right\}, \tag{7.24}$$

which represents a subset of the arcs, each arc of which makes a direction from a pair element node (ζ_β, $\psi^{\zeta\beta}{}_\beta$) to an object element node x_j, denoted as B_2, where each subset of the object element nodes either x_i or x_j (or both) is included in one of the lower approximation subsets. At this point, each subset of the pair element nodes either (ζ_α, $\psi^{\zeta\alpha}{}_\alpha$) or ($\zeta_\beta$, $\psi^{\zeta\beta}{}_\beta$) (or both) should be first marked to be a group of single pair elements (a first condition group of decision rules) in one of the decision classes. After removing all the subsets of these arcs (B_1 and B_2), a new DNA-digraph should be constructed. This process is able to reduce encoding DNA sequences, which allows us to obtain the accurate minimal decision rules. After the removal, Figure 7.11(b) shows a new DNA-digraph that has been constructed.

Step 7 (encoding-2)**:** A new directional matrix, which is composed of the reduced object and pair element entries, can be created by transforming the new model DNA-digraph constructed in Step 6, as shown in Figure 7.11(b). All the double-encoded substrings and their complementary substrings can be encoded by using the method of encoding objects and pairs in DNA. In the model DNA-digraph, DNA substrings of the existing arcs of the object and pair element nodes, corresponding to types 1, 2, 4, 5, and 7, should be secondly generated and encoded in ssDNA while reusing the encoded DNA sequences of types 3 and 6.

Step 8 (hybridisation-2 and DNA ligation)**:** The entire encoded object and pair element node substrings (types 1 and 2) and their complementary substrings (types 4, 5, and 7) in Step 7 as well as the added pair element node substrings (type 3) and its complementary substrings (type 6) are artificially synthesised and placed in a test tube. For this hybridisation, in the process of Step 3, the element node substrings and their complementary substrings are hybridised. However, in this step, for the bonding of the different encoded DNA sequences, DNA ligases are added to ensure DNA ligation among ssDNA.

Step 9 (cleavage and affinity separation-1)**:** All of one or more circular DNA fragments should be detected and distinguished from the entire hybridised and ligated DNA strands in Step 8. Before classifying the circular DNA fragments into each object group, the circular DNA fragments are cleaved at any one point by restriction enzymes to be linear DNA fragments, and they are also heated to approximately 94˚C to become ssDNA. An affinity separation technique can be used to classify each object group by using all the object element nodes of the complementary substrings with magnetic beads.

Step 10 (simulated gel electrophoresis-2 and removal-2)**:** The entire lengths of each object group of the cleaved and denatured DNA strands are able to be measured by the simulated gel electrophoresis. As illustrated in Figure 7.14 (lanes 3 to 10), each object group of the DNA strands is classified into each test tube, and each of them is loaded in each lane. After they have been loaded, if there are two or more DNA strands, indicating the same length bp, then they should be removed; however, if two or more specific DNA strands, corresponding to two or more specific objects, are comprised in one of lower approximation subsets, then those DNA strands should not be removed. In Figure 7.14 (lanes 3 to 10), the DNA strands (the same lengths) across all the lanes are the removable groups, and they are not included in the same lower approximation subset.

Step 11 (affinity separation-2 and marking-2)**:** After the process of removing the same lengths of the DNA strands, each object of the remaining DNA fragments in each decision class contains two or more pair element nodes, which can be distinguished clearly based on the method of affinity separation. For this process, the complementary substrings of the entire pair element nodes with magnetic beads should be also prepared. Each object in each lower approximation subset contains its own subset of pair element nodes, which should be secondly marked to be a second condition group of multiple pair elements of decision rules in each decision class.

7.6.2 Results of the experimental studies

To determine the specific parts of DNA strands, representing subsets of the lower approximations and condition groups of each object in each decision class, we used the two processes of both the simulated gel electrophoresis-1 and 2. As illustrated in Figure 7.14 (lanes 1 to 10), each of the lanes represents its own group of DNA strands in different lengths of DNA strands. More detailed results are presented as follows:

First, as illustrated in Figure 7.14 (lanes 1 and 2), for the first detection process, the length representation result of the simulated gel electrophoresis-1 shows the remaining DNA fragments. These DNA fragments were transformed to the two subsets of the lower approximations in two different decision classes 1 and 2. From the result of the first detection process, the two removable groups of the same lengths were (1) Object-2 in Decision Class-2 and Object-6 in Decision Class-1; and (2) Object-5 in Decision Class-2 and Object-8 in Decision Class-1. Hence, in the model decision table, all the given objects were divided into subsets of the lower approximations in each decision class, corresponding to the decision value of the decision attributes represented in DNA sequencing directions as

(1) Decision Class-1: $\Xi_{\mathrm{A}}(D^{v}{}_{\tau 1})$ = {Object-3, Object-4} derived from
(Object-3) (C, 1)-$3'^{upper}\rightarrow 5'^{upper}$-(D, 1)|(D, 1)-$3'^{upper}\rightarrow$
$5'^{upper}$-(F, 1)|(F, 1)-$3'^{upper}\rightarrow 5'^{upper}$-(P, 3) at 762 bp and
(Object-4) (C, 4)-$3'^{upper}\rightarrow 5'^{upper}$-(D, 3)|(D, 3)-$3'^{upper}\rightarrow$
$5'^{upper}$-(F, 3)|(F, 3)-$3'^{upper}\rightarrow 5'^{upper}$-(P, 2) at 1168 bp; and

(2) Decision Class-2: $\Xi_{\mathrm{A}}(D^{v}{}_{\tau 2})$ = {Object-1, Object-7} derived from
(Object-1) (C, 4)-$3'^{upper}\rightarrow 5'^{upper}$-(D, 3)|(D, 3)-$3'^{upper}\rightarrow$
$5'^{upper}$-(F, 2)|(F, 2)-$3'^{upper}\rightarrow 5'^{upper}$-(P, 1) at 1128 bp and
(Object-7) (C, 4)-$3'^{upper}\rightarrow 5'^{upper}$-(D, 1)|(D, 1)-$3'^{upper}\rightarrow$
$5'^{upper}$-(F, 2)|(F, 2)-$3'^{upper}\rightarrow 5'^{upper}$-(P, 3) at 1032 bp.

The above two subsets of lower approximations were retrieved to encode the minimal numbers of DNA fragments and to resolve a first condition group (a group of single pair elements) of decision rules in two different decision classes on the basis of the marking-1 process, represented in DNA sequencing directions as

(1) Decision Class-1:
(Object-3) x_3-$3'^{upper}\rightarrow 5'^{upper}$-(C, 1) and
(Object-4) Both x_4-$3'^{upper}\rightarrow 5'^{upper}$-(F, 3) and (F, 3)-$3'^{upper}\rightarrow 5'^{upper}$- x_4; and

(2) Decision Class-2:
(Object-1) Both x_1-$3'^{upper}\rightarrow 5'^{upper}$-(F, 2) and (F, 2)-$3'^{upper}\rightarrow 5'^{upper}$- x_1 and
(Object-7) Both x_7-$3'^{upper}\rightarrow 5'^{upper}$-(F, 2) and (F, 2)-$3'^{upper}\rightarrow 5'^{upper}$- x_7.

The results of electropherograms are shown in Figure 7.15 for Decision Class-1 and Figure 7.16 for Decision Class-2.

Second, for the second detection process, the length representation result of the

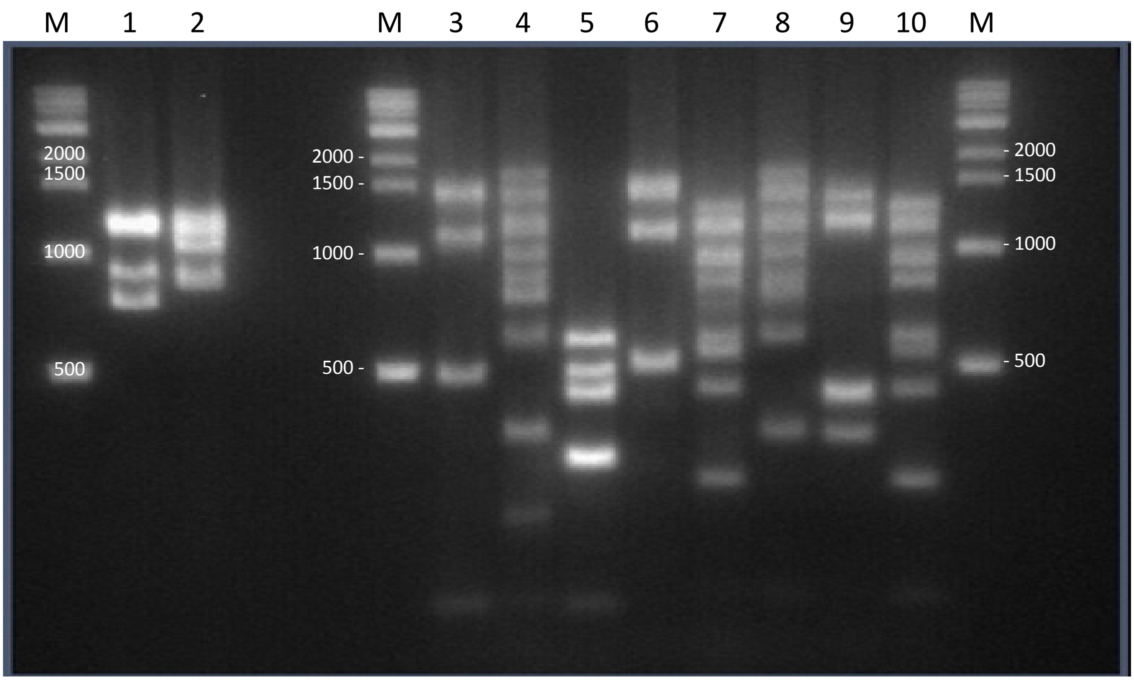


Figure 7.14. Length representation results of the simulated gel electrophoresis of (1) the first detection process in each decision class; and (2) the second detection process in each object. Lane M is for marker. For the first detection, lanes 1 and 2 indicate all the linear DNA fragments included in Decision Class-1 and 2. For the second detection, lanes 3 to 10 indicate all the cleaved and denatured circular DNA fragments included in each of Object-1 to 8.

simulated gel electrophoresis-2 shows the remaining DNA fragments, as illustrated in Figure 7.14 (lanes 3 to 10). These DNA fragments were transformed to the four subsets of the pair element nodes in four different objects (Object-1, 3, 4, and 7). From the result of the second detection process, the Object-3 and 4 in Decision Class-1 were a subset of the lower approximation and the Object-1 and 7 in Decision Class-2 were the other subset of the lower approximation. In these four different objects, the four subsets of the pair element nodes belong to two different decision classes, which were determined as a second condition group (a group of multiple pair elements) of decision rules on the basis of the marking-2 process, represented in DNA sequencing directions as

(1) Decision Class-1:
(Object-3) x_3-$3'^{upper}\rightarrow5'^{upper}$-(D, 1)|(D, 1)-$3'^{upper}\rightarrow5'^{upper}$-(F, 1)
|(F, 1)-$3'^{upper}\rightarrow5'^{upper}$-(P, 3)|(P, 3)-$3'^{upper}\rightarrow5'^{upper}$-$x_3$ at 568 bp,
(Object-3) x_3-$3'^{upper}\rightarrow5'^{upper}$-(D, 1)|(D, 1)-$3'^{upper}\rightarrow5'^{upper}$-(F, 1)
|(F, 1)-$3'^{upper}\rightarrow5'^{upper}$-$x_3$ at 472 bp,
(Object-3) x_3-$3'^{upper}\rightarrow5'^{upper}$-(F, 1)|(F, 1)-$3'^{upper}\rightarrow5'^{upper}$-(P, 3)
|(P, 3)-$3'^{upper}\rightarrow5'^{upper}$-$x_3$ at 280 bp,
(Object-4) x_4-$3'^{upper}\rightarrow5'^{upper}$-(C, 4)|(C, 4)-$3'^{upper}\rightarrow5'^{upper}$-(D, 3)
|(D, 3)-$3'^{upper}\rightarrow5'^{upper}$-(P, 2)|(P, 2)-$3'^{upper}\rightarrow5'^{upper}$-$x_4$ at 1556 bp,
(Object-4) x_4-$3'^{upper}\rightarrow5'^{upper}$-(C, 4)|(C, 4)-$3'^{upper}\rightarrow5'^{upper}$-(P, 2)

|(P, 2)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-$x_4$ at 1148 bp, and
(Object-4) x_4-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(D, 3)|(D, 3)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(P, 2)
|(P, 2)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-$x_4$ at 516 bp; and

(2) Decision Class-2:
(Object-1) x_1-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(C, 4)|(C, 4)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(D, 3)
|(D, 3)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(P, 1)|(P, 1)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-$x_1$ at 1532 bp,
(Object-1) x_1-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(C, 4)|(C, 4)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(P, 1)
|(P, 1)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-$x_1$ at 1124 bp,
(Object-1) x_1-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(D, 3)|(D, 3)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(P, 1)
|(P, 1)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-$x_1$ at 492 bp,
(Object-7) x_7-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(C, 4)|(C, 4)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(D, 1)
|(D, 1)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(P, 3)|(P, 3)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-$x_7$ at 1460 bp,
(Object-7) x_7-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(C, 4)|(C, 4)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(D, 1)
|(D, 1)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-$x_7$ at 1364 bp, and
(Object-7) x_7-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(C, 4)|(C, 4)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-(P, 3)
|(P, 3)-$3^{\prime upper}\rightarrow 5^{\prime upper}$-$x_7$ at 1172 bp.

The results of electropherograms for each of Object-1 to 8 are shown in Figures 7.17 to 7.24, respectively. In addition, each of the whole electropherogram results for Decision Class-1 to 2 and Object-1 to 8 is shown in one electropherogram, in which each of them represents different colours, as shown in Figure 7.25.

Each of the above determined groups of decision rules involved the same rules as each of the two condition groups of decision rules in each of two decision classes. Thus, the rough DNA-based algorithm results of the two condition groups of decision rules in two different decision classes were transformed to two subsets of *if-then* rules, which are represented as

(1) Decision Class-1:
if colour is blue, *then* Product-1 is selected,
if function is a bit more complex, *then* Product-1 is selected,
if design is ancient and function is simple, *then* Product-1 is selected,
if function is simple and price is expensive, *then* Product-1 is selected,
if colour is yellow and price is medium, *then* Product-1 is selected, and
if design is industrial and price is medium, *then* Product-1 is selected; and

(2) Decision Class-2:
if function is a bit simpler, *then* Product-2 is selected,
if colour is yellow and price is inexpensive, *then* Product-2 is selected,
if design is industrial and price is inexpensive, *then* Product-2 is selected,
if colour is yellow and design is ancient, *then* Product-2 is selected, and
if colour is yellow and price is expensive, *then* Product-2 is selected.

In each decision class, the above *if-then* decision rules were the minimised decision rules, corresponding to the minimal lengths of decision rules in rough sets, which were finally resolved by the rough DNA-based algorithm.

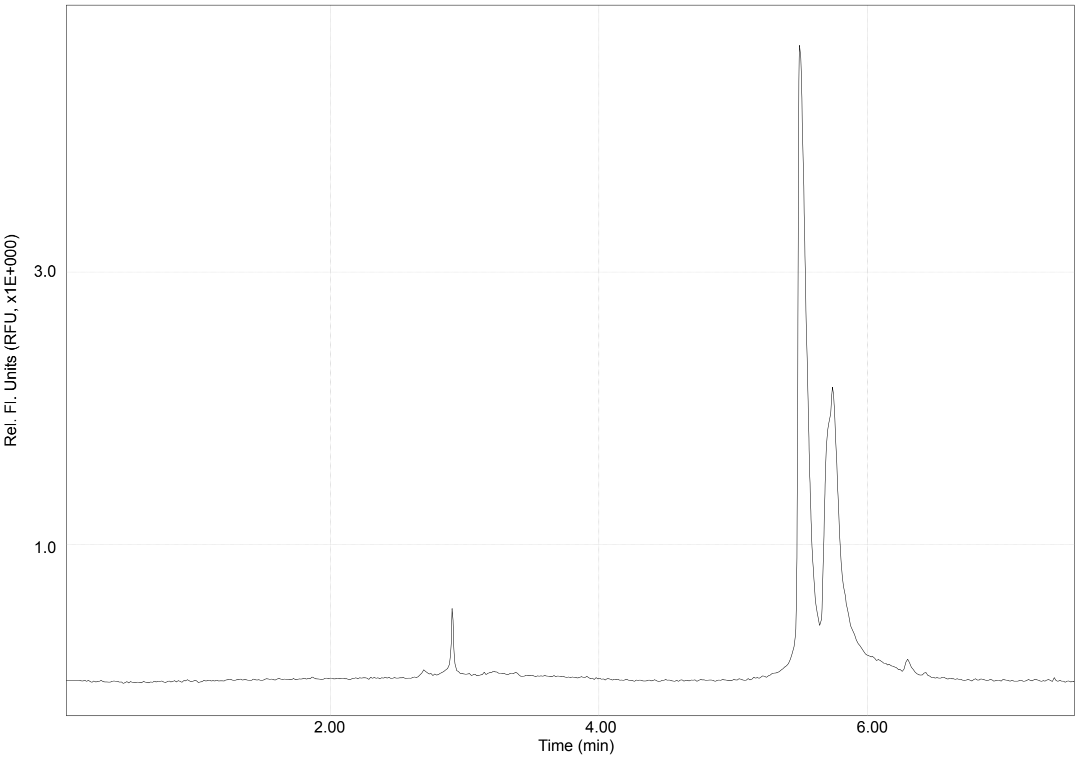


Figure 7.15. Electropherogram result obtained from lane 1.

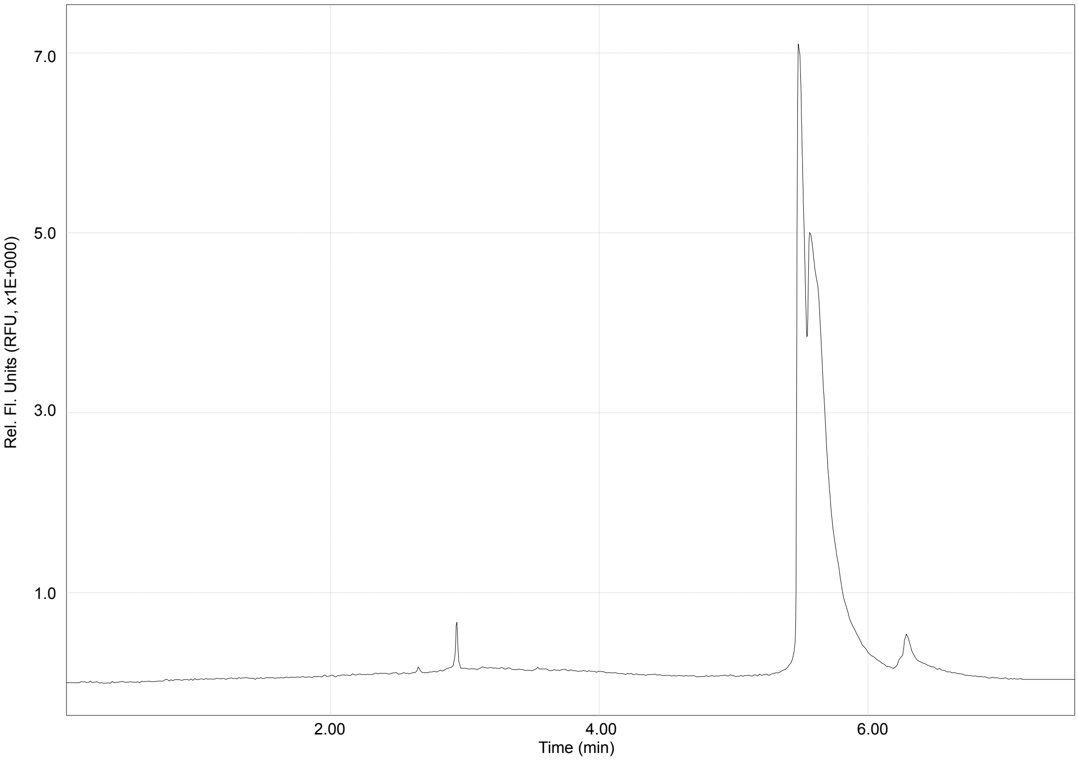


Figure 7.16. Electropherogram result obtained from lane 2.

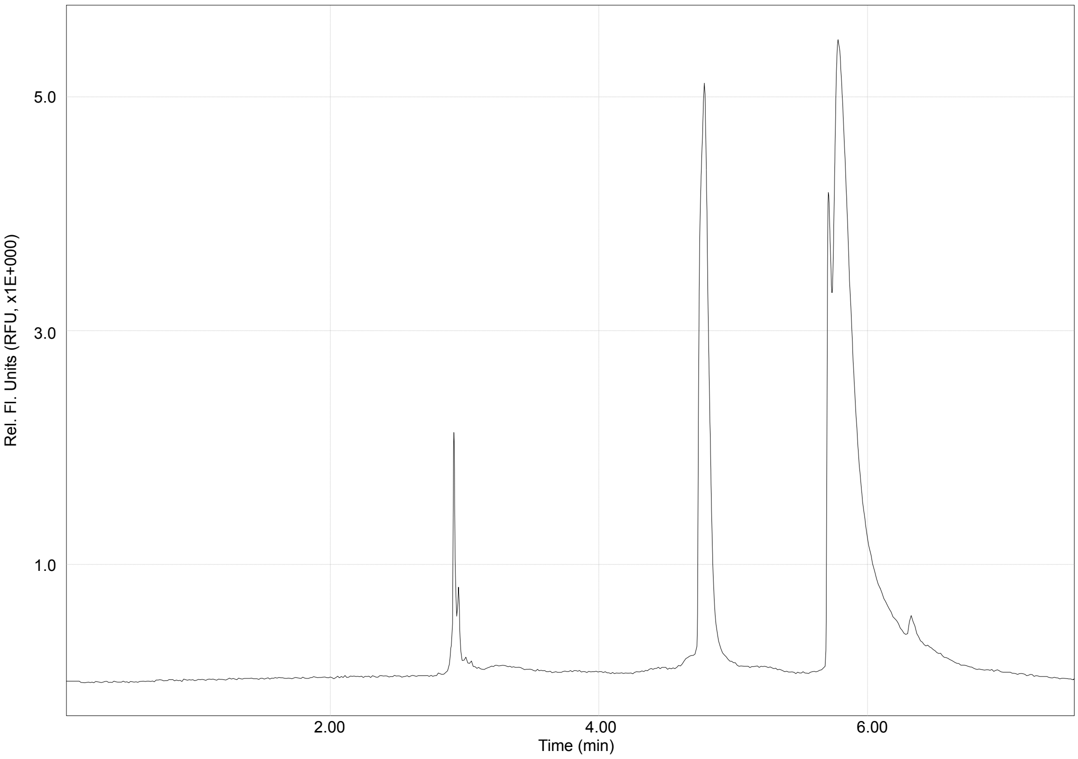


Figure 7.17. Electropherogram result obtained from lane 3.

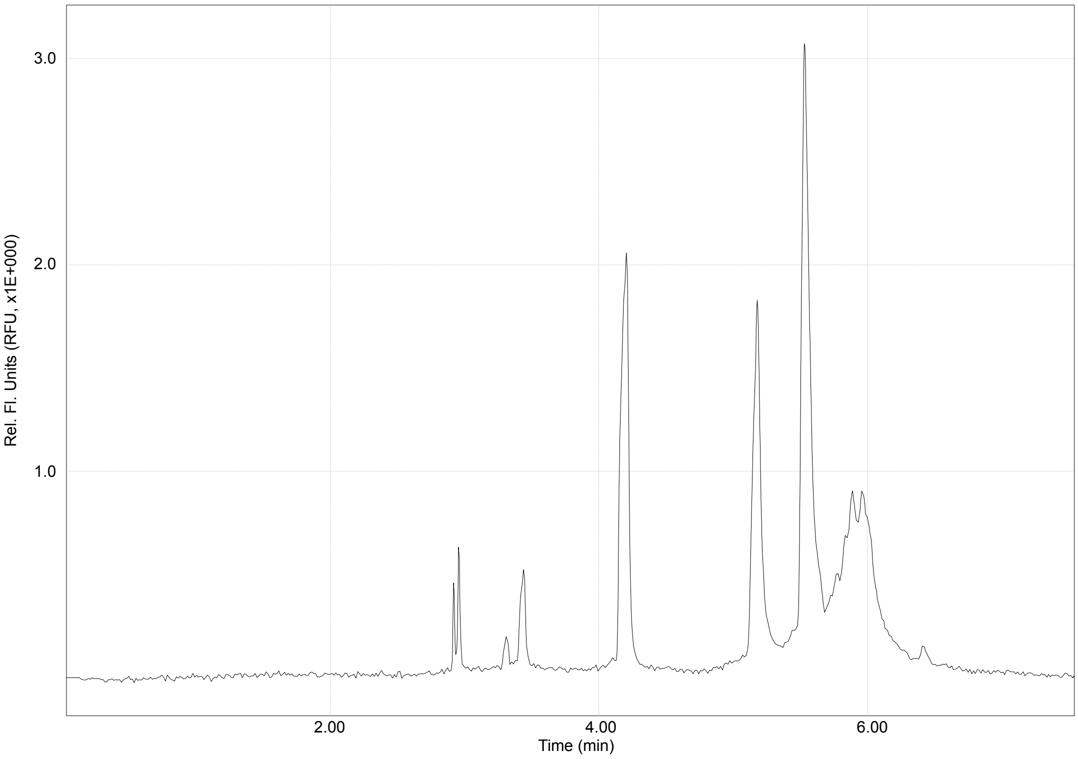


Figure 7.18. Electropherogram result obtained from lane 4.

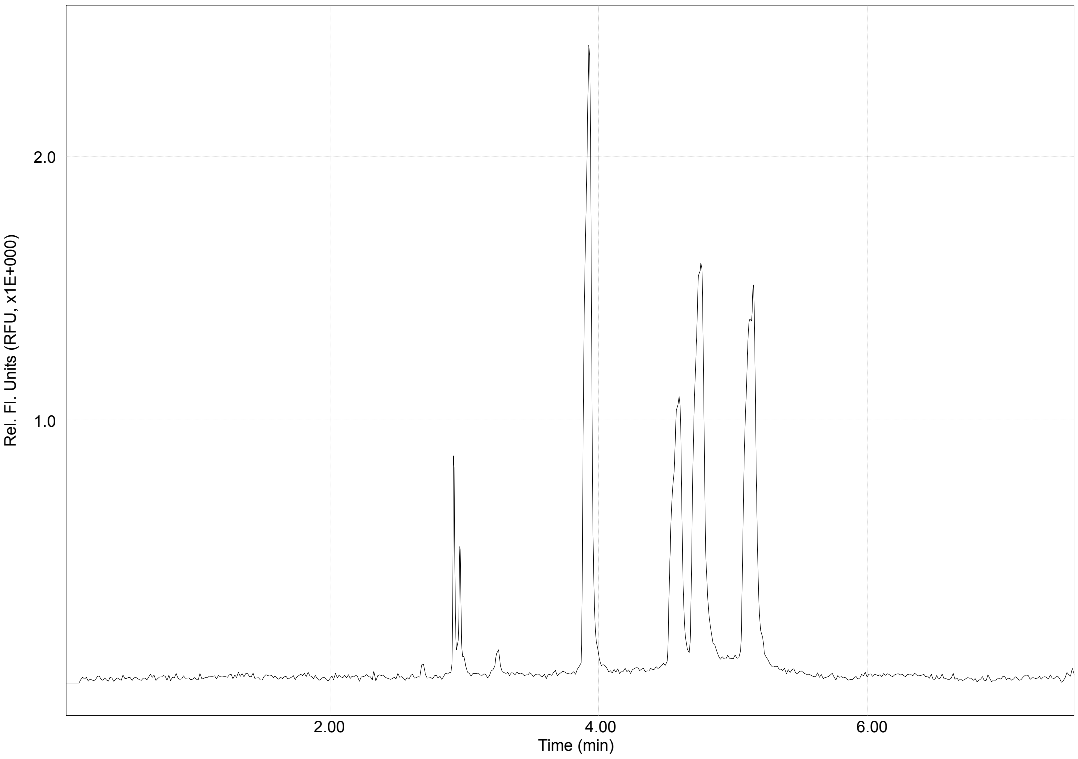


Figure 7.19. Electropherogram result obtained from lane 5.

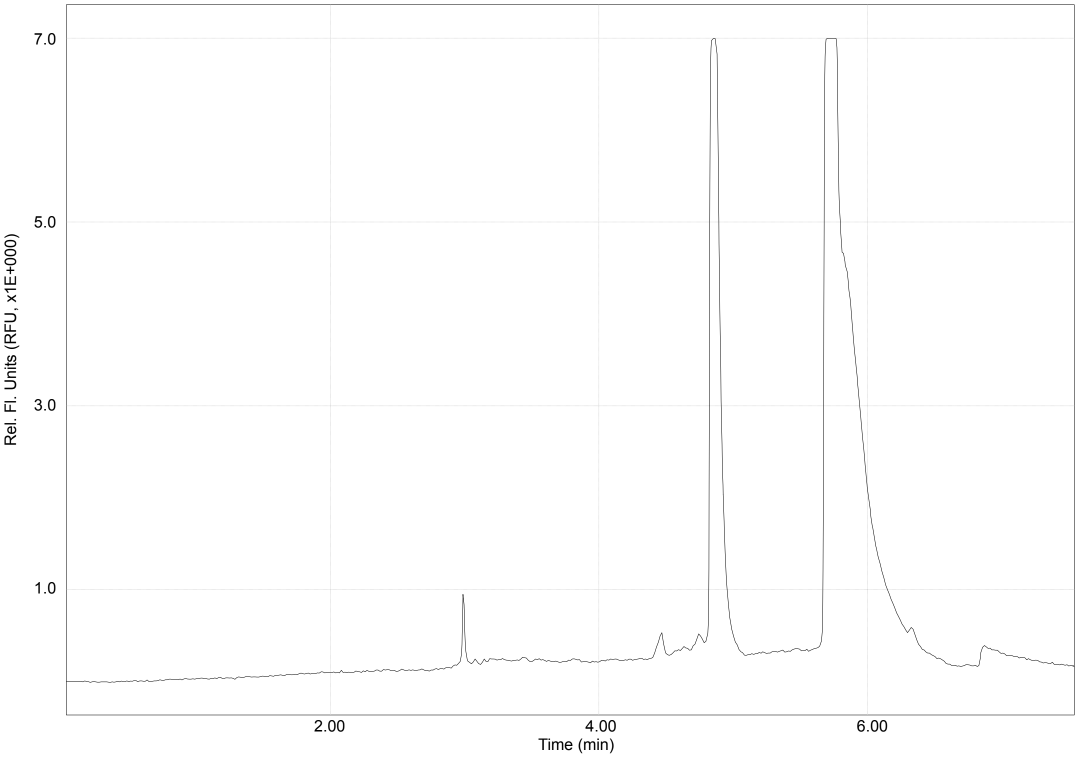


Figure 7.20. Electropherogram result obtained from lane 6.

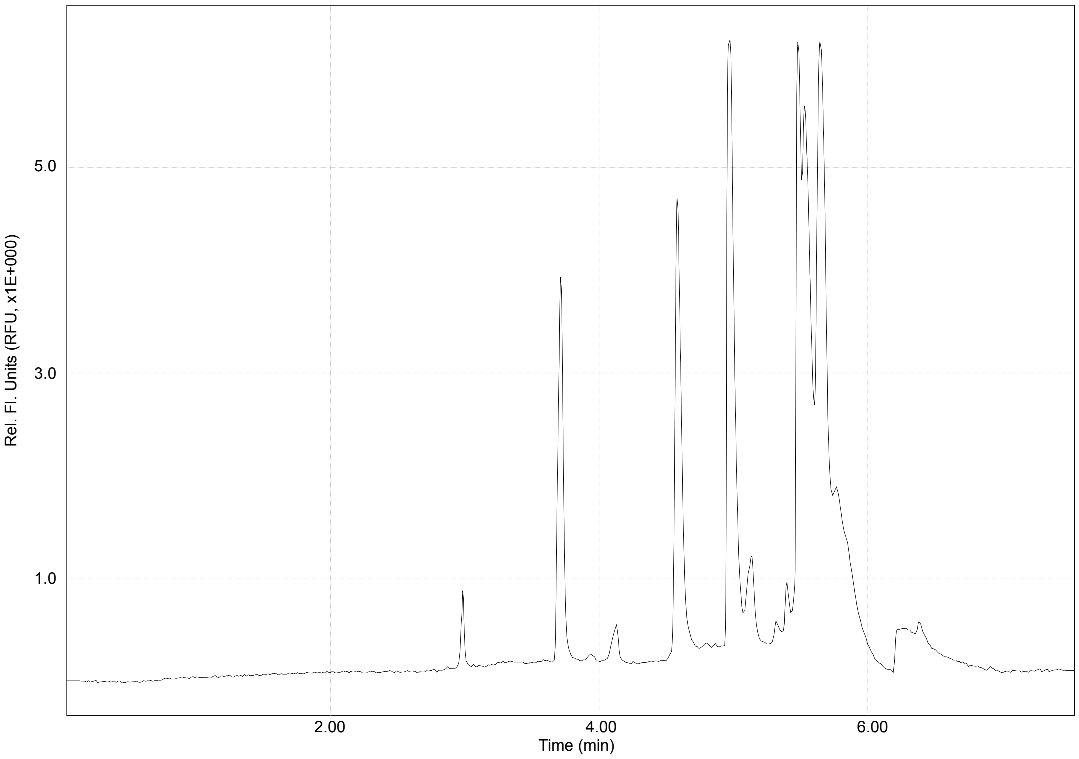


Figure 7.21. Electropherogram result obtained from lane 7.

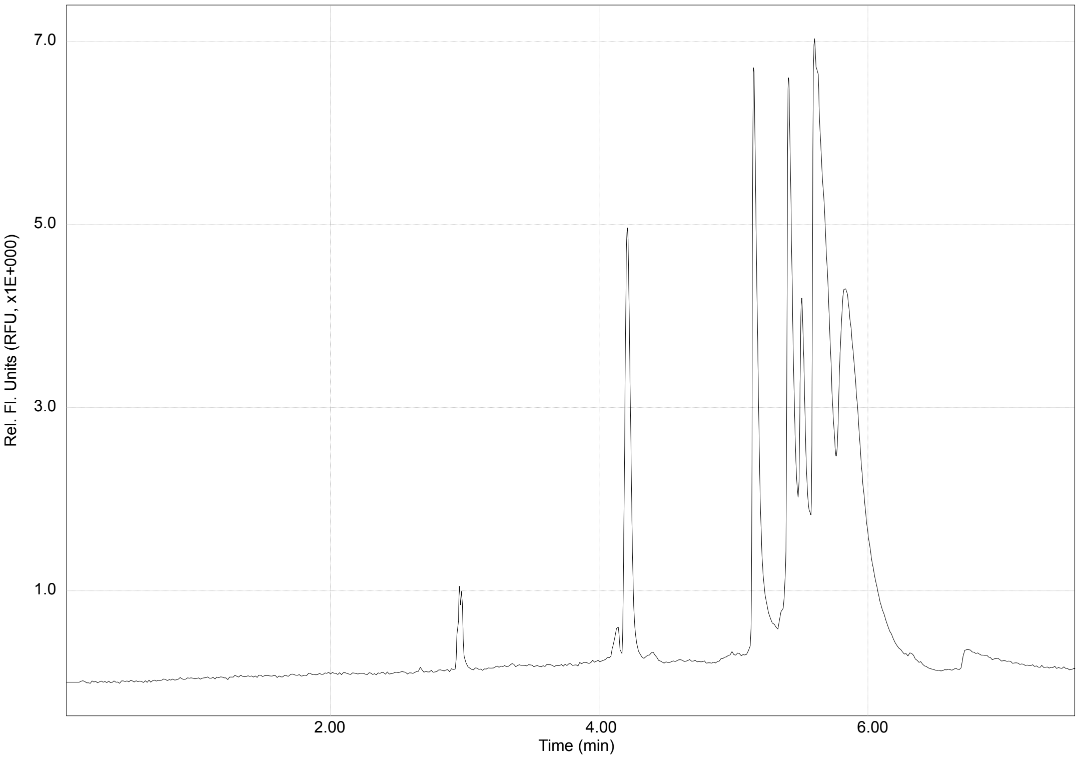


Figure 7.22. Electropherogram result obtained from lane 8.

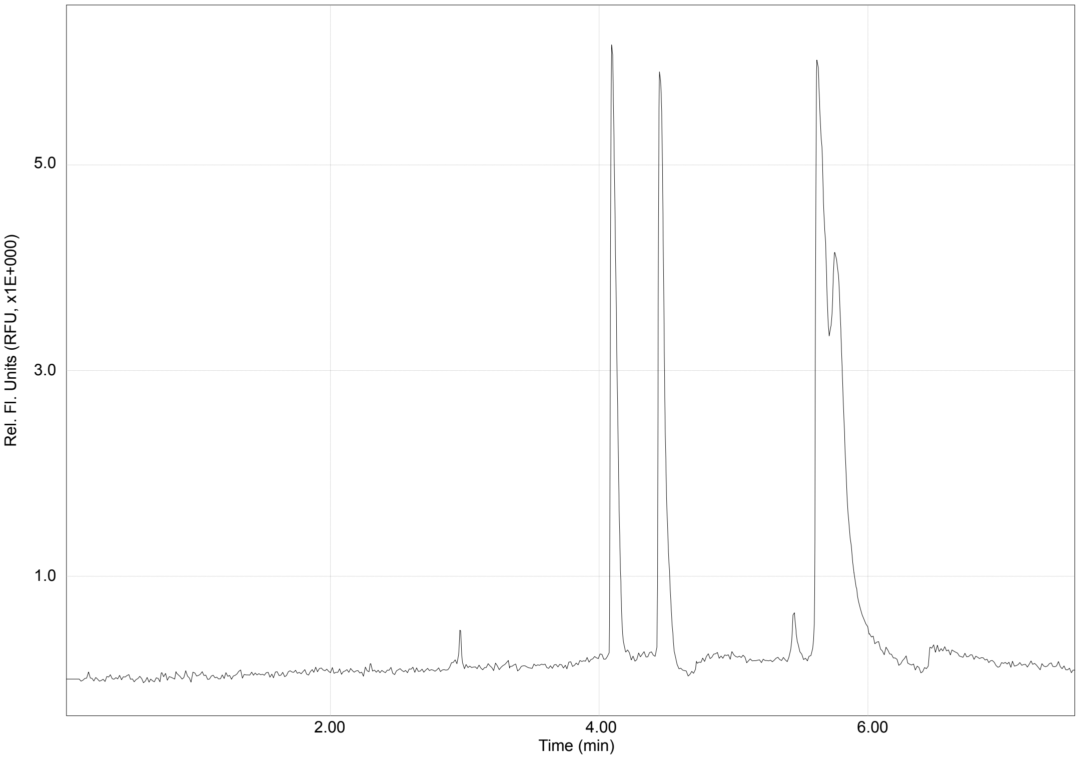


Figure 7.23. Electropherogram result obtained from lane 9.

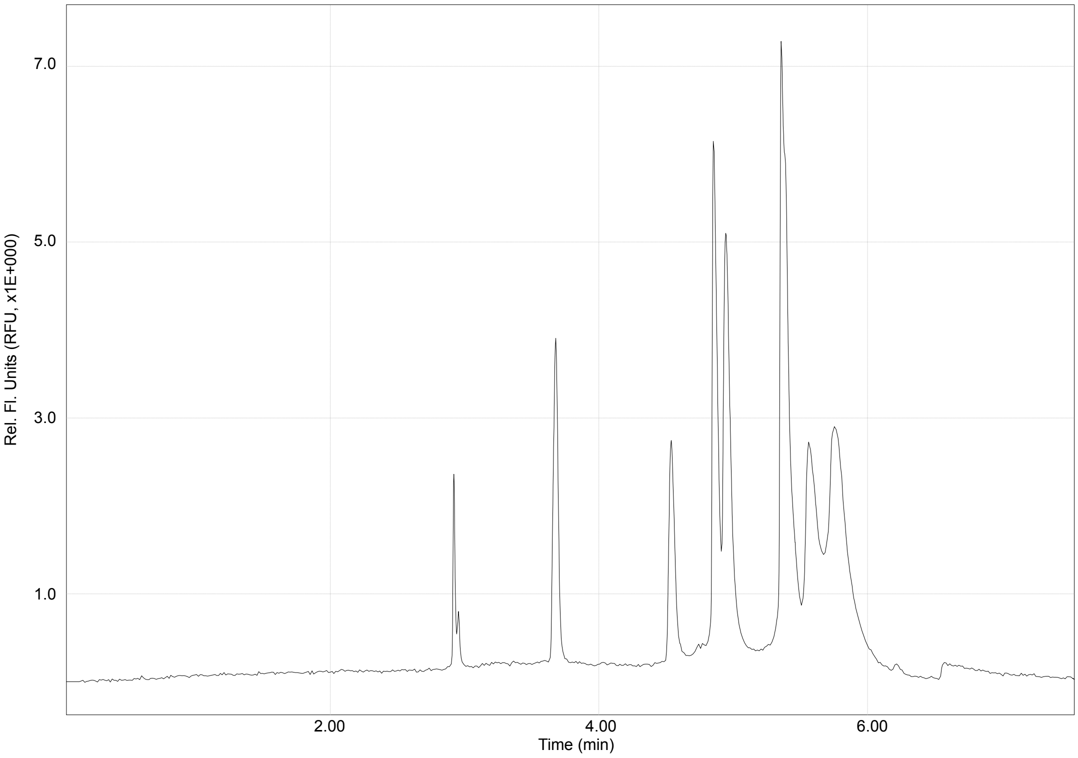


Figure 7.24. Electropherogram result obtained from lane 10.

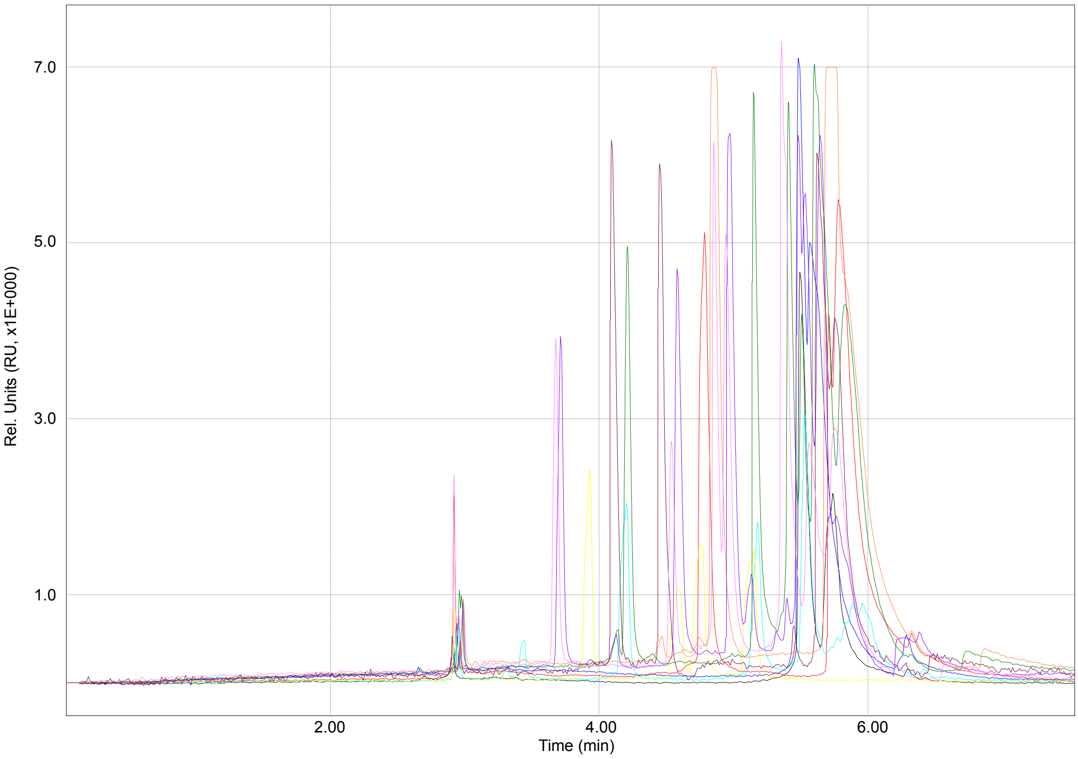


Figure 7.25. Electropherogram results obtained from lanes 1 to 10: (1) black colour indicates lane 1; (2) blue colour indicates lane 2; (3) red colour indicates lane 3; (4) cyan colour indicates lane 4; (5) yellow colour indicates lane 5; (6) orange colour indicates lane 6; (7) purple colour indicates lane 7; (8) green colour indicates lane 8; (9) brown colour indicates lane 9; and (10) pink colour indicates lane 10.

7.7 Computational Times and Solvable Sizes

A comparison of the number of elements (objects and pairs) and the number of direct relations is shown in Figure 7.26. In addition, as shown in Table 7.2, comparisons of approximated running times were created for measuring the efficiency of using the rough DNA-based algorithm. For the exponential-time algorithms and the prepared rough DNA-based algorithm (we are ready to detect solutions), the number of both objects and pairs was set to correspond to the inputs of size 10, 30, 50, and 100 separately. Figures 7.27 to 7.30 show a graphic comparison of approximated running times for the exponential-time algorithm and the prepared rough DNA-based algorithm.

We suppose that a processor executes a million high levels of instructions a second, and the exponential-time algorithm uses an operation of 2^n [106]. The approximated running times of the prepared rough DNA-based algorithm were measured and calculated on the basis of previous experimental reports, our experimental experiences, and genetic engineering notes [62-66].

In this chapter's study, the major problem is how to minimise a huge number of both objects and pairs in the rough set method to determine all minimal length decision rules; this intractable problem obviously is an NP-hard problem. To achieve a reliably optimal solution, a polynomial-time algorithm is the most reliable one, but this polynomial-time

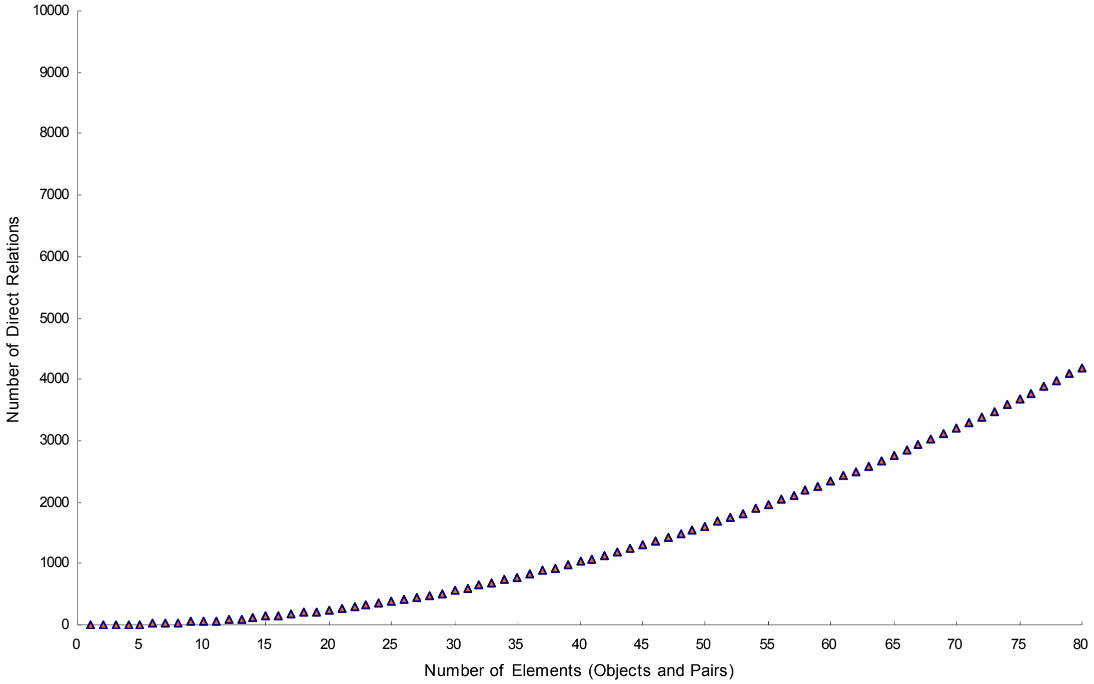


Figure 7.26. Comparison graph of the number of elements (objects and pairs) and the number of direct relations.

algorithm for this problem has not been discovered yet. The rough DNA-based algorithm has been proposed in this chapter, and this algorithm can be extended to particularly deal with a large number of both objects and pairs by taking advantage of the main characteristics of huge numbers of DNA encoding patterns.

Table 7.2. Comparisons of approximated running times for the exponential-time algorithm and the prepared rough DNA-based algorithm.

Number of Elements (Objects and Pairs)	10	30	50	100
The Exponential-Time Algorithm	< 1.00 second	18.00 minutes	36.00 years	100,000,000,000,000,000.00 years
The Prepared DNA-Based Algorithm	3.66 minutes	36.06 minutes	1.73 hours	7.04 hours

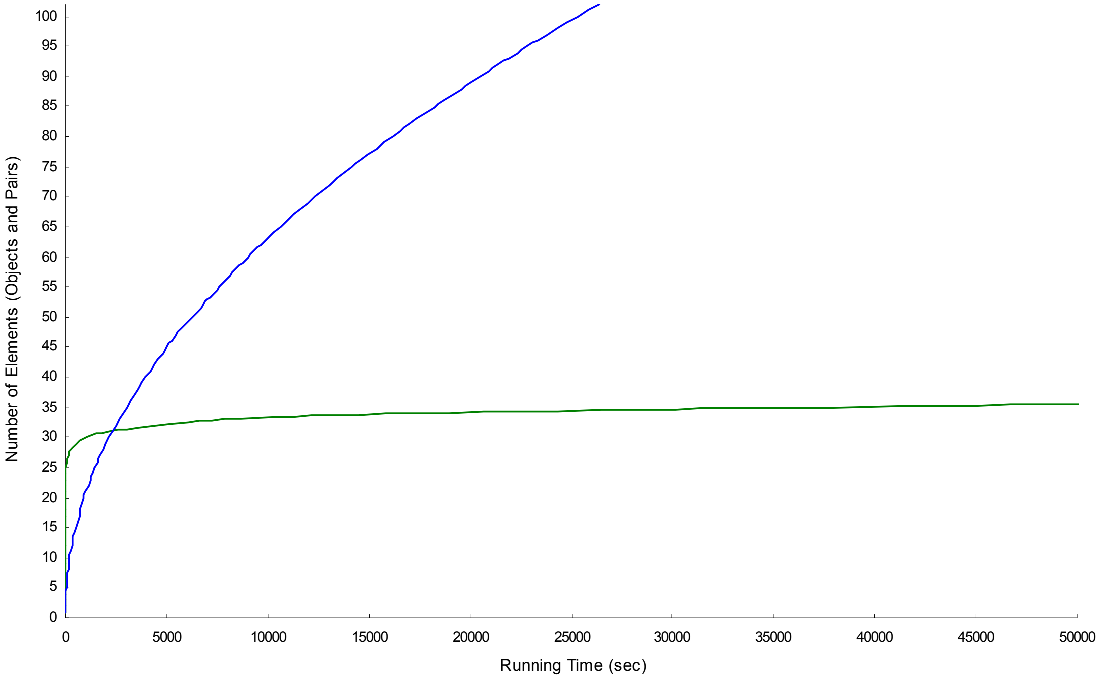


Figure 7.27. Comparison graph of approximated running times in seconds: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared rough DNA-based algorithm.

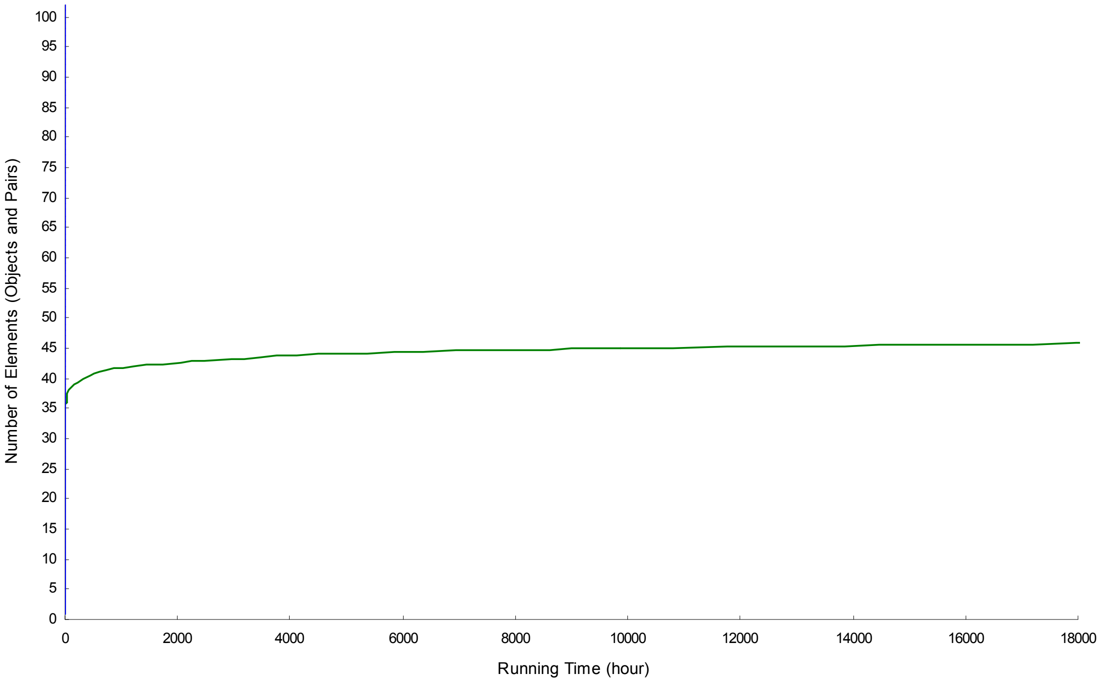


Figure 7.28. Comparison graph of approximated running times in hours: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared rough DNA-based algorithm.

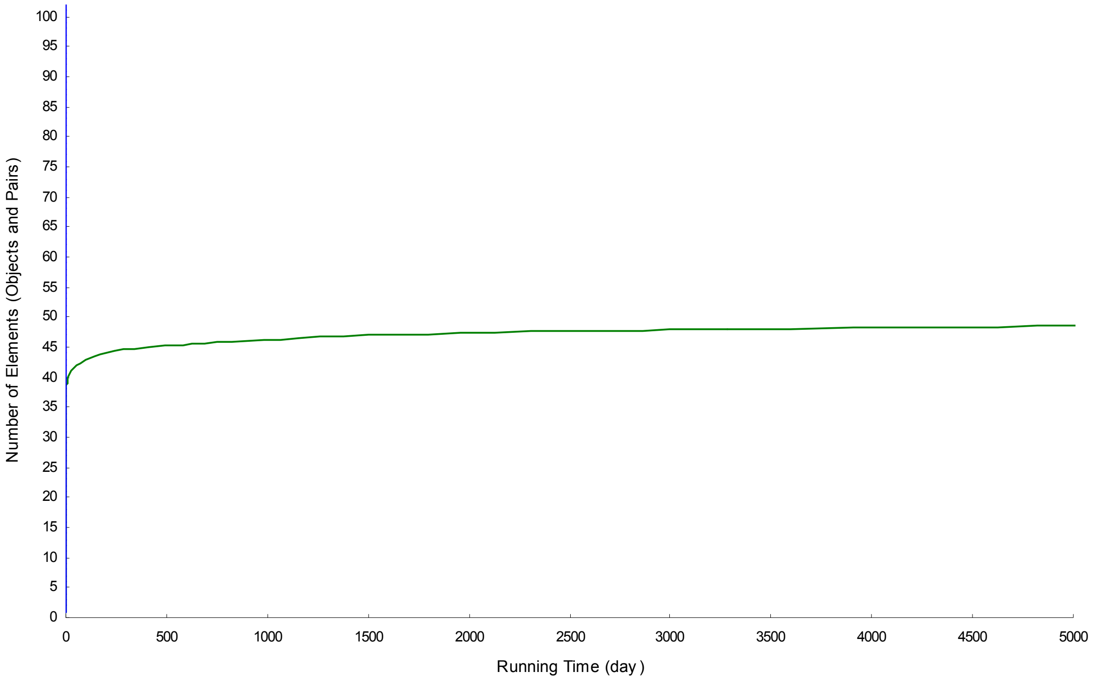


Figure 7.29. Comparison graph of approximated running times in days: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared rough DNA-based algorithm.

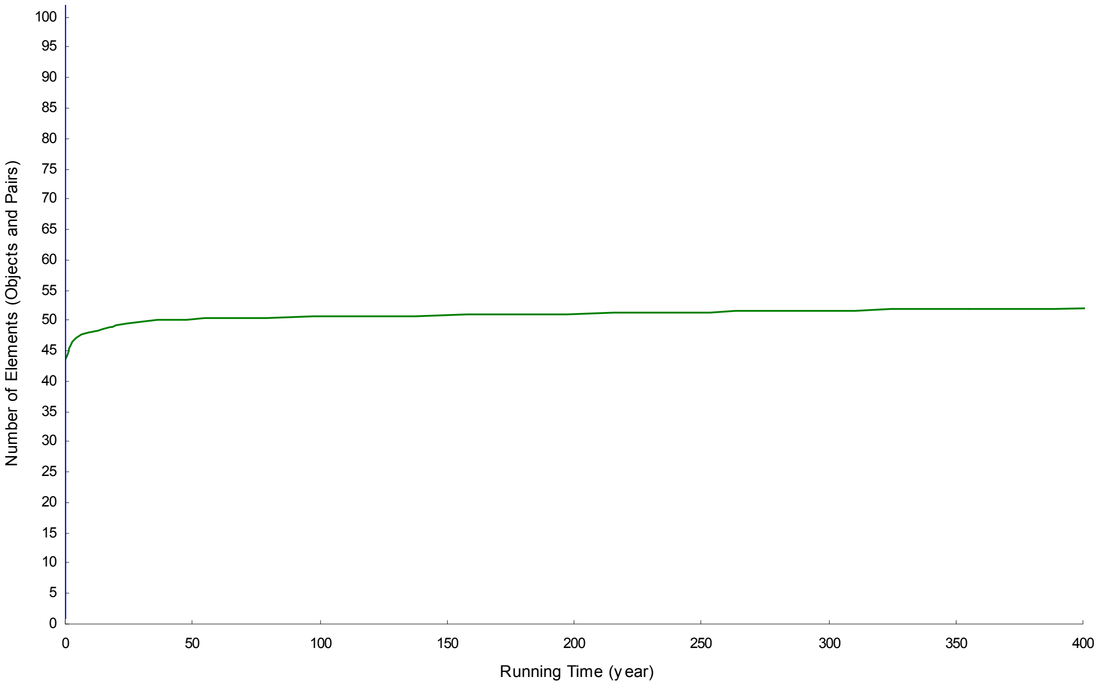


Figure 7.30. Comparison graph of approximated running times in years: (1) green colour indicates the exponential-time algorithm; and (2) blue colour indicates the prepared rough DNA-based algorithm.

7.8 Concluding Remarks

In the context of all the exponential and numerical results of the minimised decision rules by means of the rough DNA-based algorithm, we discuss the following two main issues of this chapter's study.

First, our novel processes of handling the significant information from the provided decision table were developed for the first time through the rough DNA-based algorithm, which was used for efficiently determining subsets of the lower approximations divided into each decision class and efficiently minimising decision rules. The processes of both classifying the complicated data and creating decision rules are important factors in providing information to decision makers. Additionally, the processes of updating those data and minimising decision rules also are important factors in providing useful information, which can be readily produced and handled. However, in the real world, a number of data are commonly handled, because the updating of rules and reductions poses an intractable problem. The massively parallel function of the rough DNA-based algorithm was adapted to the minimisation problem in handling a large number of objects and attributes to determine minimal decision rules in the rough set method.

Second, the rough DNA-based algorithm presented a novel molecular encoding method that performed the encoding of two different characteristics of both object and pair element nodes in DNA, and created a new type of the splicing operation method. The proposed molecular encoding method also made the best use of efficiently resolving minimal decision rules, including subsets of the lower approximations. The rough DNA-based algorithm has shown an outstanding potential to connect to other types of computer-based technologies or other sciences from other field areas to develop better cooperative data processing in interdisciplinary studies.

Chapter 8

Conclusions

The interdisciplinary forms of molecular computational algorithms for solving decision-making problems in management engineering have been presented for the first time in this dissertation. The four novel and different molecular computational algorithms with their own molecular computational experimentations were shown, and each algorithm has been designed by combining totally different types of information and mathematical scientific methods with molecular biochemistry and genetic techniques of experimentations, instrumentation, and manipulations. Further, each of the four molecular computational algorithms was designed and put into practice to solve various problems in order to measure the efficiency and practicality of these novel types of molecular computational algorithms. We conclude our discussion of the four molecular computational algorithms, introduced and presented in Chapters 4, 5, 6, and 7, in the following paragraphs.

We presented in Chapter 4 a molecular computational algorithm associated with a general molecular algorithm from the biologically computational field and a job-shop scheduling method from the management engineering field. The given machines and jobs as well as their orders and production process times can be organised and scheduled to provide feasible schedules using the present methods. However, the optimisation scheduling techniques still have problems. These problems occur when a number of the given machines and their job orders should be scheduled to minimise the maximum production completion time, because the possible solution space is small and the solution easily becomes a local optimal solution. On the other hand, a job-shop DNA-based algorithm can be used to counteract these problems. Further, the algorithm can subsequently be expanded to combine with other scheduling engineering and scientific methods and techniques.

In Chapter 5, another molecular computational algorithm was presented and associated with a general molecular algorithm from the biologically computational field and fuzzy-based methods from the management engineering field. The fuzzified numerical data can be handled by fuzzy-based methods, which are related to soft computing methods and have the potential to be integrated with other methods. Thus, we created a fuzzy DNA-based algorithm, based largely on both fuzzy-based and molecular engineering methods, to specifically identify cohesive subsets. In the fuzzy operational network, using the fuzzy DNA-based algorithm, the identification of more cohesive subgroups with more specific workforces organised into organisationally similarity subgroups was verified and its efficiency was measured.

In Chapter 6, we presented a third molecular computational algorithm associated with a general molecular algorithm from the biologically computational filed and an interpretive structural modelling method from the management engineering field. Being cognizant of this interdisciplinary challenge, molecular engineering methods provided us with powerful techniques, and by exploiting the several structural modelling methods, a hierarchical DNA-based algorithm was measured to deal with the larger number of elements in the interpretive structural modelling process. The decision-making method of interpretive structural modelling is often used for constructing a hierarchically re-structured digraph among the given elements to minimise uncertainty and risk for the problem-solving process or to provide efficient methods for decision makers. Here, the hierarchical DNA-based algorithm can be exploited to deal with a number of elements often provided and suggested by proposers, decision makers, and organisations. The hierarchical DNA-based algorithm has demonstrated the possibility for creating new interdisciplinary decision-making methods for molecular decision support computation in an efficient way.

In Chapter 7, we presented a final molecular computational algorithm associated with a general molecular algorithm from the biologically computational field and a rough set method from the management engineering field. A novel approach to handling useful information presented as a decision table was first developed by means of a rough DNA-based algorithm, which was used for deriving subsets of the lower approximations and for classifying all the objects into decision classes. The purpose of using the rough DNA-based algorithm was to minimise the number of decision rules. The rough DNA-based algorithm resulted from our new ways of encoding DNA strands based on the results of our own experimentation. This new method encoded two totally different characteristics of element nodes. The possibility of combining the rough mathematical scientific techniques and molecular engineering techniques is shown to be an interdisciplinary study that could lead to the construction of a biochemically inte-

Table 8.1. Suggested time complexities of our proposed molecular computational algorithms and the recent heuristic algorithms for Chapters 4, 5, 6, and 7. In the time complexities of Chapter 4, m is the number of machines, n is the number of jobs, and $v_{\max}$ is the maximum number of machines that are necessary in each job. In the time complexities of Chapter 5, n is the number of workforces (or nodes). In the time complexities of Chapter 6, n is the number of element nodes and a is the number of arcs. In the time complexities of Chapter 7, n is the number of condition attributes and b is the number of objects.

Chapter	Algorithm	Time Complexity
4	Job-Shop DNA-Based Algorithm	$O(mn)$
	The Recent Heuristic Algorithm	$O((nv_{\max})^2)$ [158]
5	Fuzzy DNA-Based Algorithm	$O(n)$
	The Recent Heuristic Algorithm	$O(n^3)$ [159]
6	Hierarchical DNA-Based Algorithm	$O(n)$
	The Recent Heuristic Algorithm	$O(\|n\|\|a\| \log\|n\|)$ [160]
7	Rough DNA-Based Algorithm	$O(nb)$
	The Recent Heuristic Algorithm	$O(\|n\|^3\|b\|^2)$ [161]

grated knowledge support system in decision making.

In this chapter, the suggested time complexities of each proposed molecular computational algorithm are shown in Table 8.1. Each of these suggested time complexities is also compared to the recent heuristic algorithms used for the same or similar purposes of solving our target decision-making problems (minimising the maximum production completion time in Chapter 4, identifying cohesive subsets in Chapter 5, modelling interpretive structures in Chapter 6, and minimising decision rules in Chapter 7).

For the purpose of the presented algorithms in this dissertation, we still have a biologically technical issue when encoding each of the specifically DNA-encoded nodes and their edges (or arcs) for each. Here, the nodes and their edges correspond to (1) pair nodes and their continuous arcs, respectively, in Chapter 4; (2) workforces and their operational connections, respectively, in Chapter 5; (3) element nodes and their arcs, respectively, in Chapter 6; and (4) elements and their direct relations, respectively, in Chapter 7. However, we can surely expect that high-tech molecular biological instrumentation machines are still improving and being developed for better functions by many companies around the world.

The major purpose of our proposed molecular computational algorithms is to contribute to the development of nanometric molecular computational models as molecular decision support computation for management of the engineering decision-making problems. Molecular decision support computation in management engineering can be regarded as one integrated field, mainly associated with molecular engineering mechanisms and computer science, by using both molecular engineering techniques and information technologies. Moreover, pure molecular engineering techniques are implemented not only for computer science with information technologies, but also for other various areas of engineering and science that can be turned to practical uses in developing molecular decision support computation. More flexible and practical computations using a variety of useful algorithms are needed to respond to additional management engineering problems and issues. We hope that the four proposed molecular computational algorithms, along with the four new molecular computational experimentations shown in this dissertation, could be used as either molecular or computational resources in developing further computational devices or machines for use in molecular decision support computation, exploited not only for management of the engineering decision-making problems, but also for decision-making problems in other fields.

Bibliography

[1] L. M. Adleman, “Molecular computation of solutions to combinatorial problems,” *Science*, vol. 266, no. 11, 1994, pp. 1021-1024.

[2] J. D. Watson, R. M. Myers, A. A. Caudy, and J. A. Witkowski, “Recombinant DNA: Genes and genomes - a short course, third edition,” *W. H. Freeman and Company*, 2007, pp. 5-25.

[3] D. L. Hartl and E. W. Jones, “Essential genetics: A genomics perspective, third edition,” *Jones and Bartlett Publishers, Inc.*, 2002, pp. 90-412.

[4] R. Phillips, J. Kondev, J. Theriot, and N. Orme, “Physical biology of the cell,” *Garland Science, Taylor & Francis Group, LLC*, 2002, pp. 281-325.

[5] K. Morimatsu, “It sticks to rice: Considering traditional-style Japanese food,” *Hachioji Sennin Juku*, 2005, pp. 6-11 in Japanese.

[6] B. Basham, G. P. Schroth, and P. S. Ho, “An A-DNA triplet code: Thermodynamic rules for predicting A- and B-DNA,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 92, no. 14, 1995, pp. 6464-6468.

[7] D. A. Micklos, G. A. Freyer, and D. A. Crotty, “DNA science: A first course, second edition,” *Cold Spring Harbor Laboratory Press*, 2003, pp. 30-182.

[8] J. D. Watson and F. H. C. Crick, “Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid,” *Nature*, vol. 171, no. 4356, 1953, pp. 737-738.

[9] W. L. Jorgensen and J. D. Madura, “Temperature and size dependence for Monte Carlo simulation of TIP4P water,” *Molecular Physics: An International Journal at the Interface between Chemistry and Physics*, vol. 56, issue 6, 1985, pp. 1381-1392.

[10] J. C. Lee and R. R. Gutell, “Diversity of base-pair conformations and their occurrence in rRNA structure and RNA structural motifs,” *Journal of Molecular Biology*, vol. 344, issue 5, 2004, pp. 1225-1249.

[11] E. V. Makeyev and D. H. Bamford, “Cellular RNA-dependent RNA polymerase involved in posttranscriptional gene silencing has two distinct activity modes,” *Molecular Cell*, vol. 10, issue 6, 2002, pp. 1417-1427.

[12] P. G. de Novoa and K. P. Williams, “The tmRNA website: Reductive evolution of tmRNA in plastids and other endosymbionts,” *Nucleic Acids Research, Oxford University Press, Inc.*, vol. 32, 2004, database issue D104-D108.

[13] G. M. Cooper and R. E. Hausman, “The cell: A molecular approach, third edition,” *Sinauer Associates Inc.*, 2003, pp. 261-276.

[14] R. A. Garrett, S. R. Douthwaite, A. Liljas, A. T. Matheson, P. B. Moore, and H. F. Noller, “The ribosome: Structure, function, antibiotics, and cellular interactions,” *American Society for Microbiology Press*, 2000, pp. 397-404.

[15] T. Chakraborty, “Charge migration in DNA: Perspectives from physics, chemistry, and biology,” *Nanoscience and Technology, Springer-Verlag Berlin Heidelberg*, 2007, pp. 77-175.

[16] J. Braman, “*In vitro* mutagenesis protocols, second edition,” *Methods in Molecular Biology, Humana Press Inc.*, vol. 182, 2002, pp. 7-17.

[17] D. L. Hartl and E. W. Jones, “Genetics: Analysis of genes and genomes, seventh edition,” *Jones and Bartlett Publishers, Inc.*, 2009, pp. 431-445.

[18] N. G. Cooper and P. Berg, “The human genome project: Deciphering the blueprint of heredity,” *University Science Books*, 1994, pp. 48-54.

[19] R. Tyagi, “Understanding molecular biology,” *Discovery Publishing House Pvt. Ltd.*, 2009, pp. 174-213.

[20] D. S. T. Nicholl, “An introduction to genetic engineering, second edition,” *Studies in Biology, Cambridge University Press*, 2002, pp. 43-51.

[21] J. J. Greene and V. B. Rao, “Recombinant DNA principles and methodologies,” *Marcel Dekker, Inc.*, 1998, pp. 126-134.

[22] R. S. Burlage, R. Atlas, D. Stahl, G. Geesey and G. Sayler, “Techniques in microbial ecology,” *Oxford University Press, Inc.*, 1998, pp. 299-311.

[23] J. Sambrock and D. W. Russell, “Molecular cloning: A laboratory manual, third edition,” *Cold Spring Harbor Laboratory Press*, 2001, pp. 157-161.

[24] G. Lipps, “Plasmids: Current research and future trends,” *Caister Academic Press, Norfolk, UK*, 2008, pp. 1-25.

[25] P. Rabinow, “Making PCR: A story of biotechnology,” *The University of Chicago Press*, 1996, pp. 1-17.

[26] M. J. McPherson, P. Quirke, and G. R. Taylor, “PCR 1: A practical approach,” *Oxford University Press, Inc*, 1991, pp. 1-14.

[27] P. Matejtschuk, “Affinity separations: A practical approach,” *Oxford University Press, Inc*, 1997, pp. 2-38.

[28] E. R. Goedken, M. Levitus, A. Johnson, C. Bustamante, M. O’Donnell, and J. Kuriyan, “Fluorescence measurements on the *E. coli* DNA polymerase clamp loader: Implications for conformational changes during ATP and clamp binding,” *Journal of Molecular Biology*, vol. 336, issue 5, 2004, pp. 1047-1059.

[29] M. Kinter and N. E. Sherman, “Protein sequencing and identification using tandem mass spectrometry,” *John Wiley & Sons, Inc.*, vol. 336, issue 5, 2000, pp. 117-146.

[30] B. Giffler and G. L. Thompson, “Algorithms for solving production-scheduling problems,” *Operations Research*, vol. 8, no. 4, 1960, pp. 487-503.

[31] J. Carlier and E. Pinson, “An algorithm for solving the job-shop problem,” *Management Science*, vol. 35, no. 2, 1989, pp. 164-176.

[32] P. Brucker, B. Jurisch, and B. Sievers, “A branch and bound algorithm for the job-shop scheduling problem,” *Discrete Applied Mathematics*, vol. 49, no. 1-3, 1994, pp. 107-127.

[33] M. J. Streeter and S. F. Smith, “Exploiting the power of local search in a branch and bound algorithm for job shop scheduling,” *International Conference on Automated Planning and Scheduling*, 2006, pp. 324-333.

[34] C. Artigues, M. Gendreau, L.-M. Rousseau, and A. Vergnaud, “Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound,” *Computers & Operations Research*, vol. 36, no. 8, 2009, pp. 2330-2340.

[35] T. Yamada and R. Nakano, “A genetic algorithm applicable to large-scale job-shop problems,” *Parallel Problem Solving from Nature*, 1992, pp. 283-292.

[36] T. Yamada and R. Nakano, “Genetic algorithms for job-shop scheduling problems,” *Proceedings of Modern Heuristic for Decision Support*, 1997, pp. 1-15.

[37] S.-C. Lin, E. D. Goodman, and W. F. Punch, "Investigating parallel genetic algorithms on job shop scheduling problems," *Evolutionary Programming*, 1997, pp. 383-393.

[38] W. Cheung and H. Zhou, "Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times," *Annals of Operations Research*, vol. 107, no. 1-4, 2001, pp. 65-81.

[39] A. Hertz and M. Widmer, "An improved tabu search approach for solving the job shop scheduling problem with tooling constraints," *Discrete Applied Mathematics*, vol. 65, no. 1-3, 1996, pp. 319-345.

[40] R. Thamilselvan and P. Balasubramanie, "Integrating genetic algorithm, tabu search approach for job shop scheduling," *International Journal of Computer Science and Information Security*, vol. 2, no. 1, 2009, 6 pages.

[41] U. Buscher and L. Shen, "An integrated tabu search algorithm for the lot streaming problem in job shops," *European Journal of Operational Research*, vol. 199, no. 2, 2009, pp. 385-399.

[42] M. Andersson, A. H. C. Ng, and H. Grimm, "Simulation optimization for industrial scheduling using hybrid genetic representation," *Proceedings of the 2008 Winter Simulation Conference*, 2008, pp. 2004-2011.

[43] I. Iimura, Y. Moriyama, and S. Nakayama, "Consideration on distributed immune algorithm in job-shop scheduling problem," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 12(B), 2009, pp. 5003-5010.

[44] Ph. Mauguiere, J.-C. Billaut, and J.-L. Bouquard, "New single machine and job-shop scheduling problems with availability constraints," *Journal of Scheduling, Springer Netherlands*, vol. 8, no. 3, 2005, pp. 211-231.

[45] R. Masuchun, W. Masuchun, and T. Thepmanee, "Integrating production scheduling and material requirements planning," *ICIC Express Letters*, vol. 3, no. 3(A), 2009, pp. 501-506.

[46] T. M. Willems and L. E. M. W. Brandts, "Implementing heuristics as an optimization criterion in neural networks for job-shop scheduling," *Journal of Intelligent Manufacturing, Springer Netherlands*, vol. 6, no. 6, 1995, pp. 377-387.

[47] R. Zhang and C. Wu, "A decomposition-based optimization algorithm for scheduling large-scale job shops," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 9, 2009, pp. 2769-2780.

[48] R. Zhang and C. Wu, "Bottleneck machine identification based on optimization for the job shop scheduling problem," *ICIC Express Letters*, vol. 2, no. 2, 2008, pp. 175-180.

[49] Yu. N. Sotskov and N. V. Shakhlevich, "NP-hardness of shop-scheduling problems with three jobs," *Discrete Applied Mathematics*, vol. 59, no. 3, 1995, pp. 237-266.

[50] W. J. Stevenson, "Operations management: Ninth edition," *McGraw-Hill, Irwin*, 2007, pp. 720-759.

[51] F. Hashimoto, T. Hoashi, T. Kurozawa, and K. Kato, "Production Control Systems," *Kyoritsu Shuppan Co., Ltd.*, 2001, pp. 108-120 in Japanese.

[52] J. Heizer and B. Render, "Operations management: Ninth edition," *Pearson Education, Inc.*, 2008, pp. 253-605.

[53] M. L. Pinedo, "Scheduling-theory, algorithms, and systems: Third edition,"

Springer Science+Business Media, 2008, pp. 179-216.

[54] R. W. Conway, W. L. Maxwell, and L. W. Miller, "Theory of scheduling," *Dover Publications, Inc.*, 2003, pp. 103-131.

[55] P. Patel, "Rapid analysis techniques in food microbiology," *Blackie Academic & Professional*, 1995, pp. 183-191.

[56] P. Singleton, "Dictionary of DNA and genome technology," *Wiley-Blackwell*, 2010, pp. 175-178.

[57] D. P. Clark, "Molecular biology: Understanding the genetic revolution," *Elsevier Inc.*, 2005, pp. 425-452.

[58] G. M. Malacinski, "Essentials of molecular biology: Fourth edition," *Jones and Bartlett Publishers, Inc.*, 2003, pp. 314-345.

[59] T. A. Brown, "Gene cloning and DNA analysis: An introduction, sixth edition," *Wiley-Blackwell*, 2010, pp. 15-44.

[60] C. S. Calude and G. Paun, "Computing with cells and atoms: An introduction to quantum, DNA and membrane computing," *Taylor & Francis*, 2002, pp. 77-94.

[61] J. Kleinberg and E. Tardos, "Algorithm design," *Kyoritsu Shuppan Co., Ltd.*, 2008, pp. 29-102 in Japanese.

[62] Z. F. Burton and J. M. Kaguni, "Experiments in molecular biology: Biochemical applications," *Academic Press*, 1997, pp. 11-20.

[63] J. P. Fitch, "An engineering introduction to biotechnology," *SPIE Press*, 2002, pp. 43-60.

[64] L. M. Adleman, "Computing with DNA: The manipulation of DNA to solve mathematical problems is redefining what is meant by computation," *Scientific American*, 1998, pp. 34-41.

[65] T. Tamura, "New experimental notes in genetic engineering, note 1/2," *Yodosha Co., Ltd.*, 2005, pp. 61-160 in Japanese.

[66] T. Tamura, "New experimental notes in genetic engineering, note 2/2," *Yodosha Co., Ltd.*, 2005, pp. 11-86 in Japanese.

[67] R. R. Mattu and C. G. Plaxton, "Optimal time bounds for approximate clustering," *Machine Learning*, vol. 56, no. 1-3, 2004, pp. 35-60.

[68] E. Todorov, W. Li, and X. Pan, "From task parameters to motor synergies: A hierarchical framework for approximately optimal control of redundant manipulators," *Journal of Field Robotics*, vol. 22, no. 11, 2005, pp. 691-710.

[69] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbour in high dimensions," *The 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006, pp. 459-468.

[70] S. U. Khan, "Approximate optimal sensor placements in grid sensor fileds," *The 65th IEEE Vehicular Technology Conference*, 2007, pp. 248-251.

[71] G. Agnarsson and R. Greenlaw, "Graph theory: Modeling, applications, and algorithms," *Pearson Education, Inc.*, 2007, pp. 31-131.

[72] D. B. Bernstein and R. Farrow, "Automatic maintenance of routine programming tasks based on a declarative description (experience report)," *Proceedings of the 12th International Conference on Software Engineering*, 1990, pp. 310-315.

[73] Y. Toyoura, J. Watada, Y. Yabuuchi, H. Ikegame, S. Sato, K. Watanabe, and M. Tohyama, "Fuzzy regression analysis of software bug structure," *Central European Journal of Operations Research, Springer-Verlag Berlin Heidelberg*, vol.

12, no. 1, 2004, pp. 13-23.

[74] A. R. Arredondo, Y. Yabuuchi, and J. Watada, "Analysis of safety factors by fuzzy quantification analysis type 2," *Proceedings of Workshop on Evaluation of Heart and Mind: A New Challenge*, vol. 2, 1997, pp. 45-48.

[75] F. Luthans, "Organizational behavior, tenth edition," *McGraw-Hill International Edition*, 2005, pp. 478-508.

[76] W. Pedrycz and F. Gomide, "Fuzzy systems engineering: Toward human-centric computing," *John Wiley & Sons, Inc.*, 2007, pp. 27-65.

[77] I. Kim, D. J.-F. Jeng, and J. Watada, "Redesigning subgroups in a personnel network based on DNA computing," *International Journal of Innovative Computing, Information and Control, ICIC International*, vol. 2, no. 4, 2006, pp. 885-896.

[78] R. Cross and A. Parker, "The hidden power of social networks: Understanding how work really gets done in organizations," *Harvard Business School Press*, 2002, pp. 31-166.

[79] S. Wasserman and K. Faust, "Social network analysis: Methods and applications," *Structural Analysis in the social sciences 8, Cambridge University Press*, 1994, pp. 109-290.

[80] J. Scott, "Social network analysis: A handbook, second edition," *SAGE Publications, Inc.*, 2004, pp. 100-122.

[81] D. Knoke and S. Yang, "Social network analysis, second edition," *Series: Quantitative Applications in the Social Sciences 154, SAGE Publications, Inc.*, 2008, pp. 51-85.

[82] R. D. Bock and S. Z. Husain, "An adaptation of Holzinger's *B*-coefficients for the analysis of sociometric data," *Sociometry*, vol. 13, 1950, pp. 146-153.

[83] R. D. Alba, "A graph-theoretic definition of a sociometric clique," *Journal of Mathematical Sociology*, vol. 3, 1973, pp. 113-126.

[84] P. Hage and F. Harary, "Island networks: Communication, kinship, and classification structures in Oceania," *Structural Analysis in the social sciences 11, Cambridge University Press*, 1996, pp. 22-89.

[85] G. Valiente, "Algorithms on trees and graphs," *Springer-Verlag Berlin Heidelberg*, 2002, pp. 228-350.

[86] D. S. Hochbaum, "Approximation algorithms for NP-hard problems," *PWS Publishing Company*, 1997, pp. 296-441.

[87] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 100, supplement 1, 1999, pp. 9-34.

[88] L. A. Zadeh, "From imprecise to granular probabilities," *Fuzzy Sets and Systems*, vol. 154, issue 3, 2005, pp. 370-374.

[89] J. Watada, "Trend of fuzzy multivariant analysis in management engineering," *Springer-Verlag Berlin Heidelberg*, LNAI 3682, 2005, pp. 1283-1290.

[90] C. C. Ragin, "Fuzzy-set social science," *The University of Chicago Press, Ltd.*, 2000, pp. 203-308.

[91] P. S. Nair and S-C Cheng, "Cliques and fuzzy cliques in fuzzy graphs," *IFSA World Congress and 20th NAFIPS International Conference, Proceedings*, 2001, pp. 2277-2280.

[92] P. S. Nair and S-C Cheng, "An adequate statistic for the exponentially distributed censoring data," *Computer Science and Statistics, Proceedings*, 2001, pp.

1-4.

[93] J. N. Mordeson and P. S. Nair, “Fuzzy graphs and fuzzy hypergraphs,” *Physica-Verlag Berlin Heidelberg*, 2000, pp. 1-81.

[94] X. Wang, T. Schiner, and X. Yao, “Automatic feature-queried bird identification system based on entropy and fuzzy similarity,” *Expert Systems with Applications, Elsevier B.V.*, vol. 34, issue 4, 2008, pp. 2879-2884.

[95] A. Pedrycz and M. Reformat, “An optimization of α-cuts of fuzzy sets through particle swarm optimization,” *2006 Annual Conference of the North American Fuzzy Information Processing Society, Proceedings*, 2006, pp. 57-62.

[96] A. Rosenfeld, “Fuzzy graphs, fuzzy sets and their applications,” *In: L. A. Zadeh, K. S. Fu, M. Shimura (eds), Academic Press*, 1975, pp. 77-95.

[97] W. Pedrycz, “Shadowed sets: Representing and processing fuzzy sets,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 28, no. 1, 1998, pp. 103-109.

[98] D. Dubois and H. Prade, “Fuzzy sets and systems: Theory and applications,” *Mathematics in Science and Engineering, Academic Press, Inc.*, vol. 144, 1980, pp. 19-80.

[99] V. Stix, “Finding all maximal cliques in dynamic graphs,” *Computation Optimization and Applications, Springer-Verlag Berlin Heidelberg*, vol. 27, issue 2, 2004, pp. 173-186.

[100] I. M. Bomze, M. Pelillo, and V. Stix, “Approximating the maximum weight clique using replicator dynamics,” *IEEE Transactions on Neural Networks*, vol. 11, issue 6, 2000, pp. 1228-1241.

[101] I. M. Bomze, “Evolution towards the maximum clique,” *Journal of Global Optimization*, vol. 10, issue 2, 1997, pp. 143-164.

[102] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, “Handbook of combinatorial optimization: The maximum clique problem,” *Kluwer Academic Publishers*, supplement vol. A, 1999, pp. 1-74.

[103] W. L. Toffler, A. E. Sinclair, M. S. Darr, D. L. McGinty, K. Commenford, and R. Goetz, “Using a sociomatrix to evaluate the effectiveness of small-group teaching to residents,” *Academic Medicine*, vol. 65, no. 10, 1990, pp. 654-655.

[104] Q. Ouyang, P. D. Kaplan, S. Liu, and A. Libacher, “DNA solution of the maximal clique problem,” *Science*, vol. 278, 1997, pp. 446-449.

[105] L. Kari, Gh. Paun, G. Rozenberg, A. Salomaa, and S. Yu, “DNA computing, sticker systems, and universality,” *Acta Informatica, Springer-Verlag Berlin Heidelberg*, vol. 35, 1998, pp. 401-420.

[106] J. Kleinberg and E. Tardos, “Algorithm design,” *Pearson Education, Inc.*, 2006, pp. 29-207.

[107] S. Nakano, “NP-completeness: Advanced lecture II, discrete systems engineering,” *Department of Computer Science, Gunma University*, 2006, pp. 1-5 in Japanese.

[108] J. N. Warfield, “Structuring complex systems,” *Battelle Monograph, Columbus, Ohio*, no. 4, 1974.

[109] J. N. Warfield, “On arranging elements of a hierarchy in graphic form,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 2, 1973, pp. 121-132.

[110] J. N. Warfield, “Twenty laws of complexity: Science applicable in organiza-

tions," *Systems Research and Behavioral Science*, vol. 16, no. 1, 1999, pp. 3-40.

[111] J. N. Warfield, "An introduction to systems science," *World Scientific Publishing Co. Pte. Ltd.*, 2006, pp. 38-45.

[112] J. T. Ziegenfuss and C. K. McKenna, "Ten tools of continuous quality improvement: A review and case example of hospital discharge," *American Journal of Medical Quality*, vol. 10, no. 4, 1995, pp. 213-220.

[113] R. Genma, T. Sato, and S. Mizuki, "Apprehension in students decision of seeking counseling services: The analysis of university students' narratives using KJ method," *Institute of Human Sciences, Ritsumeikan University*, vol. 17, 2008, pp. 47-60.

[114] R. Scupin, "The KJ method: A technique for analyzing data derived from Japanese ethnology," *Human Organization, The Society of Applied Anthropology*, vol. 56, no. 2, 1997, pp. 233-237.

[115] N. Souter, "Creative business solutions, breakthrough thinking: Brainstorming for inspiration and ideas," *Sterling Publishing Co., Inc.*, 2007, pp. 69-88.

[116] N. Souter, "Creative business solutions, persuasive writing: How to make words work for you," *Sterling Publishing Co., Inc.*, 2007, pp. 79-109.

[117] T. Ui, "Decision support and groupware," *Kyoritsu Shuppan Co., Ltd.*, 2002, pp. 33-58 in Japanese.

[118] E. Kinoshita, "An introduction to management science," *Kindai Kagaku sha Co., Ltd.*, 2001, pp. 215-235 in Japanese.

[119] G. F. Farris, "Executive decision making in organizations: Identifying the key men and managing the process," *MIT Sloan School Working Paper*, no. 551, 1971, pp. 11-16.

[120] R. K. F. Ip and C. Wagner, "Weblogging: A study of social computing and its impact on organizations," *Decision Support Systems*, vol. 45, issue 2, 2008, pp. 242-250.

[121] S. McLaughlin, R. A. Paton, and D. K. Macbeth, "Barrier impact on organizational learning within complex organizations," *Journal of Knowledge Management*, vol. 12, issue 2, 2008, pp. 107-123.

[122] X. Munoz, W. Unger, and I. Vrto, "One sided crossing minimization is NP-hard for sparse graphs," *Graph Drawing, LNCS 2265, Springer-Verlag Berlin Heidelberg*, 2001, pp. 115-123.

[123] S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa, "Crossing minimization in linear embeddings of graphs," *IEEE Transactions on Computers*, vol. 39, no. 1, 1990, pp. 124-127.

[124] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, "Graph drawing: Algorithms for the visualization graphs," *Prentice-Hall, Inc.*, 1999, pp. 41-325.

[125] D. M. Lee, "Structured decision making with interpretive structural modeling: Implementing the core of interactive management," *Sorach Inc.*, 2007, pp. 2-11.

[126] D. W. Malone, "An introduction to the application of interpretive structural modeling," *Proceedings of the IEEE*, vol. 63, issue 3, 1975, pp. 397-404.

[127] O. Bastert and C. Matuszewski, "Layered drawings of digraphs," *Drawing Graphs, LNCS 2025, Springer-Verlag Berlin Heidelberg*, 2001, pp. 87-120.

[128] C. Matuszewski, R. Schonfeld, and Paul Molitor, "Using sifting for *k*-layer straightline crossing minimization," *Graph Drawing, LNCS 1731, Springer-Verlag Berlin Heidelberg*, 1999, pp. 217-224.

[129] M. Junger, E. K. Lee, P. Mutzel, and T. Odenthal, "A polyhedral approach to the multi-layer crossing minimization problem," *Graph Drawing, LNCS 1353, Springer-Verlag Berlin Heidelberg*, 1997, pp. 13-24.

[130] F. Shahrokhi, O. Sykora, L. A. Szekely, and I. Vrto, "Bipartite crossing numbers of meshes and hypercubes," *Graph Drawing, LNCS 1353, Springer-Verlag Berlin Heidelberg*, 1997, pp. 37-45.

[131] T. Biedl, F. J. Brandenburg, and X. Deng, "Crossings and permutations," *Graph Drawing, LNCS 3843, Springer-Verlag Berlin Heidelberg*, 2005, pp. 1-12.

[132] M. R. Garey and D. S. Johnson, "Crossing number is NP-complete," *SIAM Journal on Algebraic and discrete methods*, vol. 4, issue 3, 1983, pp. 312-316.

[133] Japan Industrial Management Association, "Handbook of industrial management," *Maruzen Co., Ltd.*, 1994, pp. 639-650 in Japanese.

[134] J. Thakkar, A. Kanda, and S. G. Deshmukh, "Interpretive structural modeling of IT-enablers for Indian manufacturing SMEs," *Information Management & Computer Security*, vol. 16, issue 2, 2008, pp. 113-136.

[135] T. Watanabe, J. Watada, and K. Oda, "Hierarchical decision making in strategic investment by a Boltzmann machine," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 7, issue 4, 1999, pp. 429-437.

[136] K. Levitz and H. Levitz, "Logic and Boolean algebra," *Barron's Educational Series, Inc.*, 1979, pp. 1-20.

[137] G. Paun, "On the splicing operation," *Discrete Applied Mathematics*, vol. 70, issue 1, 1996, pp. 57-79.

[138] T. A. Brown, "Gene cloning and DNA analysis: An introduction, fifth edition," *Wiley-Blackwell*, 2008, pp. 54-86.

[139] W. Pedrycz, "A dynamic data granulation through adjustable fuzzy clustering," *Pattern Recognition Letters, Elsevier B. V.*, vol. 29, issue 16, 2008, pp. 2059-2066.

[140] W. Pedrycz, "Fuzzy set technology in knowledge discovery," *Fuzzy Sets and Systems, Elsevier B. V.*, vol. 98, issue 3, 1998, pp. 279-290.

[141] X. Zhu and X. Wu, "Cost-constrained data acquisition for intelligent data preparation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, 2005, pp. 1542-1556.

[142] Z. Pawlak, "Rough sets," *International Journal of Computer and Information Sciences*, vol. 11, no. 5, 1982, pp. 341-356.

[143] R. Slowinski, "Intelligent decision support: Handbook of application and advances of the rough sets theory" *Kluwer Academic Publishers*, 1992, pp. 363-372.

[144] J. W. Grzymala-Busse, "A comparison of tree strategies to rule induction from data with numerical attributes," *Electronic Notes in Theoretical Computer Science*, vol. 82, issue 4, 2003, pp. 132-140.

[145] J. W. Grzymala-Busse and T. Soe, "Partition triples: A tool for reduction of data sets," *Journal of Computer and System Sciences*, vol. 53, no. 3, 1996, pp. 575-582.

[146] W. Ziarko, "Variable precision rough set model," *Journal of Computer and System Sciences*, vol. 46, issue 1, 1993, pp. 39-59.

[147] A. Skowron and C. Rauszer, "The discernibility matrices and functions in information systems," *Intelligent Decision Support: Handbook of Application and*

Advances of the Rough Sets Theory, Kluwer Academic Publishers, 1992, pp. 331-362.

[148] N. Shan and W. Ziarko, "Data-based acquisition and incremental modification of classification rules," *Computational Intelligence*, vol. 11, issue 2, 1995, pp. 357-370.

[149] L. Polkowski, "Advances in soft computing: Rough sets," *Physica-Verlag Heidelberg*, 2002, pp. 18-45.

[150] J. F. Peters and A. Skowron, "Zdzisław Pawlak life and work (1926-2006)," *Information Sciences*, vol. 177, 2007, pp. 1-2.

[151] Z. Pawlak, "Rough classification," *International Journal of Man-Machine Studies*, vol. 20, issue 5, 1984, pp. 469-483.

[152] Z. Pawlak, S. K. M. Wong, and W. Ziarko, "Rough sets: Probabilistic versus deterministic approach," *International Journal of Man-Machine Studies*, vol. 29, issue 1, 1988, pp. 81-95.

[153] J. F. Peters and A. Skowron, "Transactions on rough sets VI," *Springer-Verlag Berlin Heidelberg*, 2007, pp. 351-396.

[154] R. J. Slater, "Experiments in molecular biology" *The Humana Press Inc.*, 1986, pp. 63-67.

[155] J. A. Rose, R. J. Deaton, M. Hagiya, and A. Suyama, "Coupled equilibrium model of hybridization error for the DNA microarray and tag-antitag systems," *IEEE Transactions on NanoBioscience*, vol. 6, no. 1, 2007, pp. 18-27.

[156] P. J. Smith and C. J. Jones, "DNA recombination and repair" *Oxford University Press*, 1999, pp. 112-129.

[157] T. A. Brown, "Essential molecular biology, second edition" *Oxford University Press*, 2000, vol. 1, pp. 143-150.

[158] A. Agnetis, M. Flamini, G. Nicosia, and A. Pacifici, "A job shop problem with one additional resource type" *Roma Tre University Report*, no. 163, 2010, pp. 1-24.

[159] H. Nakanishi and E. Tomita, "A computational complexity for finding a maximum clique in a graph with maximum degree 4" *National Institute of Informatics*, no. 107(127), 2007, pp. 1-7 in Japanese.

[160] M. Siebenhaller and M. Kaufmann, "Drawing activity diagrams" *The 2006 ACM Symposium on Software Visualization*, 2006, pp. 1-17.

[161] Z.-Y. Xu, B. Yang, W.-H. Shu, and B.-R. Yang "Efficient algorithm for attribute reduction of incomplete information systems based on assignment matrix" *Fuzzy Information and Engineering*, *Springer-Verlag Berlin Heidelberg*, 2009, vol. 2, AISC 62, pp. 787-796.

List of Publications

Refereed International Journals

1. Ikno Kim and Junzo Watada, "A Molecular Computational Approach to Solving a Work Centre Sequence-Oriented Manufacturing Problem of Classical Job Shop Scheduling," International Journal of Unconventional Computing, Old City Publishing, Inc., to appear, 2011.
2. Ikno Kim and Junzo Watada, "Combining Biological Computation and Fuzzy-Based Methods for Organisationally Cohesive Subgroups," International Journal of Unconventional Computing, Old City Publishing, Inc., Vol. 6, No. 3-4, pp. 285-300, 2010.
3. Ikno Kim and Junzo Watada, "Decision Making with an Interpretive Structural Modeling Method Using a DNA-Based Algorithm," IEEE Transactions on NanoBioscience, IEEE Publishing, Vol. 8, No. 2, pp. 181-191, 2009.
4. Ikno Kim, Junzo Watada and Witold Pedrycz, "A DNA-Based Algorithm for Arranging Weighted Cliques," Simulation Modelling Practice and Theory, Elsevier B.V., Vol. 16, Issue 10, pp. 1561-1570, 2008.
5. Ikno Kim, Junzo Watada and Ichiro Shigaki, "A Comparison of Dispatching Rules and Genetic Algorithms for Job Shop Schedules of Standard Hydraulic Cylinders," Soft Computing, Springer-Verlag Berlin Heidelberg, Vol. 12, No. 2, pp. 121-128, 2008.
6. Don Jyh-Fu Jeng, Ikno Kim and Junzo Watada, "Bio-Soft Computing with Fixed-Length DNA to a Group Control Optimization Problem," Soft Computing, Springer-Verlag Berlin Heidelberg, Vol. 12, No. 3, pp. 223-228, 2008.
7. Don Jyh-Fu Jeng, Ikno Kim and Junzo Watada, "Bio-Inspired Evolutionary Method for Cable Trench Problem," International Journal of Innovative Computing, Information and Control, ICIC International, Vol. 3, No. 1, pp. 111-118, 2007.
8. Ikno Kim, Don Jyh-Fu Jeng and Junzo Watada, "Redesigning Subgroups in a Personnel Network Based on DNA Computing," International Journal of Innovative Computing, Information and Control, ICIC International, Vol. 2, No. 4, pp. 885-896, 2006.

Refereed International Conferences, Proceedings

1. Ikno Kim and Junzo Watada, "A Novel Concept of Applying DNA Features to an Intractable Scheduling Problem in Controlling Process-Focused Manufacturing," 2011 International Conference on Computer Applications and Network Security, IEEE Publishing, to appear, May 27-29, 2011.
2. Ikno Kim, Yu-Yi Chu and Junzo Watada, "Structuralizing Complex Communications of Contextual Relations Using a Biological Encoding Method," World Automation Congress 2010, TSI Press, pp. IFMIP 99-1-6, September 19-22, 2010.
3. Ikno Kim, Junzo Watada and Jui-Yu Wu, "A DNA Encoding Method to Determine and Sequence All Cliques in a Weighted Graph," The 4th International Conference on Innovative Computing, Information and Control, IEEE Publishing, pp.

1532-1537, December 7-9, 2009.

4. Ikno Kim and Junzo Watada, “Determining Workstation Groups in a Fixed Factory Facility Based on Biological Computation,” Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering Systems, Part II, LNAI 5712, Springer-Verlag Berlin Heidelberg, pp. 188-194, September 28-30, 2009.
5. Ikno Kim and Junzo Watada, “A Bio-Inspired Evolutionary Approach to Identifying Minimal Length Decision Rules in Emotional Usability Engineering,” Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering Systems, Part II, LNAI 5712, Springer-Verlag Berlin Heidelberg, pp. 181-187, September 28-30, 2009.
6. Ikno Kim and Junzo Watada, “A Biologically Intelligent Encoding Approach to a Hierarchical Classification of Relational Elements in a Digraph,” Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering Systems, Part II, LNAI 5712, Springer-Verlag Berlin Heidelberg, pp. 174-180, September 28-30, 2009.
7. Ikno Kim and Junzo Watada, “Searching Cliques in a Fuzzy Graph Based on an Evolutionary and Biological Method,” Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering Systems, Part II, LNAI 5712, Springer-Verlag Berlin Heidelberg, pp. 166-173, September 28-30, 2009.
8. Ikno Kim and Junzo Watada, “A Hybrid Method of Biological Computation and Genetic Algorithms for Resolving Process-Focused Scheduling Problems,” Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering Systems, Part II, LNAI 5712, Springer-Verlag Berlin Heidelberg, pp. 159-165, September 28-30, 2009.
9. Ikno Kim, Junzo Watada and Don Jyh-Fu Jeng, “Determining Feasible Operating Schedules for a Job Shop Scheduling Problem Based on Bio-Soft Computing,” The 18th IEEE International Conference on Fuzzy Systems, IEEE Publishing, pp. 1426-1431, August 20-24, 2009.
10. Ikno Kim, Junzo Watada, Jui-Yu Wu and Yu-Yi Chu, “A Novel Biological Computation Method for Deriving and Resolving Discernibility Relations,” The 9th IEEE International Conference on Bioinformatics and Bioengineering, IEEE Publishing, pp. 9-14, June 22-24, 2009.
11. Ikno Kim and Junzo Watada, “Towards a New Medical Decision Support System with Bio-Inspired Interpretive Structural Modelling,” The 1st International Symposium on Intelligent Decision Technologies, New Advances in Intelligent Decision Technologies, Studies in Computational Intelligence 199, Springer-Verlag Berlin Heidelberg, pp. 459-466, April 23-24, 2009.
12. Ikno Kim and Junzo Watada, “A DNA-Based Clustering Method Based on Statistics Adapted to Heterogeneous Coordinate Data,” The 3rd International Conference on Complex, Intelligent and Software Intensive Systems, IEEE Publishing, pp. 892-897, March 16-19, 2009.
13. Ikno Kim, Don Jyh-Fu Jeng and Junzo Watada, “Analysis of Cohesive Employees in Work-Related Values by DNA Molecules,” The 7th International Conference on Intelligent Technologies, Taipei, Taiwan, pp. 75-78, December 12-15, 2006.
14. Ikno Kim, Don Jyh-Fu Jeng and Junzo Watada, “Analysing the Density of Subgroups in Valued Relationships Based on DNA Computing,” Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering

Systems, Part III, LNAI 4253, Springer-Verlag Berlin Heidelberg, pp. 964-971, October 9-11, 2006.

15. Don Jyh-Fu Jeng, Ikno Kim and Junzo Watada, "DNA-Based Evolutionary Algorithm for Cable Trench Problem," Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering Systems, Part III, LNAI 4253, Springer-Verlag Berlin Heidelberg, pp. 922-929, October 9-11, 2006.
16. Don Jyh-Fu Jeng, Ikno Kim and Junzo Watada, "Bio-Soft Computing Approach to Elevator Dispatching Problem," The 1st International Conference on Innovative Computing, Information and Control, IEEE Publishing, pp. 244-248, August 30-September 1, 2006.
17. Don Jyh-Fu Jeng, Junzo Watada and Ikno Kim, "Solving a Real Time Scheduling Problem Based on DNA Computing," 2005 International Conference on Intelligent Technologies and Applied Statistics, Taipei, Taiwan, pp. 317-322, June 24-26, 2005.

Book Chapters

1. Ikno Kim and Junzo Watada, "A Fuzzy Density Analysis of Subgroups by Means of DNA Oligonucleotides," Intelligent Systems and Technologies, Methods and Applications, Studies in Computational Intelligence 217, Springer-Verlag Berlin Heidelberg, pp. 31-45, 2009.
2. Junzo Watada, Don Jyh-Fu Jeng and Ikno Kim, "Application of DNA Computing to Group Control of Elevators," Intelligent Systems, Selected Papers, The Anniversary Symposium Celebrating 25 Years of the Seminar "Grigore Moisil" and 15 Years of the Romanian Society for Fuzzy Systems & A. I., Performantica Press, pp. 9-18, 2005.

Invited Lectures and Research Studies

1. Ikno Kim, "Molecular Engineering Experiments, Instrumentation, and Manipulations," Department of Biochemistry, School of Medicine, Taipei Medical University, Taipei, Taiwan, February 11, 2009.
2. Ikno Kim, "Biologically Inspired Computing and DNA Computing," Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada, May 25, 2006.