Chapter I

The Role of Simulated Evolutions in Bioinformatics

Andy Auyeung [I.1], Ajith Abraham [I.2]

Bioinformatics is a fast growing field in the scientific community. It involves a wide range of problems, for example, DNA sequence analyses, RNA secondary structure predictions, phylogenetic analyses and microarray analyses (just to name a few). Furthermore, new problem domains are constantly evolving, such as proteomics and biomedical applications. In fact, many of these problems are very complex in nature and are often hard to solve by conventional methods. We have learned that Evolutionary Computing (EC) is a powerful optimization technique that solves complex problems and it has been proven successful in various application domains. Therefore, in this chapter, we introduce EC applications in Bioinformatics. The chapter first gives an overview of some key concepts and classical problems in Bioinformatics. Then, it is followed by two case studies, where the problem's background, the mathematical formulation, the EC representation and the performance of the methods are presented in details.

I.1 Introduction

I.1.1 The Precursor of Bioinformatics

The origin of Bioinformatics can be traced back to the first discovery of modern genetics. In 1854, Gregor Johann Mendel (is now known as the father of genetics)

[I.1] Department of Computer Science, Oklahoma State University, Stillwater, OK 74078, USA, wingha@cs.okstate.edu, http://www.cs.okstate.edu/~wingha

[I.2] Department of Computer Science, Oklahoma State University, Tulsa, OK 74106, USA, ajith.abraham@ieee.org, http://ajith.softcomputing.net

conducted a study on inheredity of a garden pea called *Pisum sativum.* The study was to breed a population of the garden pea with different physical characteristics, such as seed shapes, seed colors and flower colors. Then the variations of these characteristics were observed in their offspring. Mendel discovered some regular patterns from the offspring by breeding parents with selected characteristics. After some careful analyses on these patterns, Mendel established two fundamental principles of genetics, which are now known as the Mendel's first and second law.

Another milestone of genetics is the discovery of DeoxyriboNucleic Acid (DNA) as genetic material. However, the discovery was not straightforward. First, in 1928, Frederick Griffith discovered an unknown agent that transforms a pneumonia causing bacteria called *Streptococcus pneumoniae* from R-type (which is harmless) to S-type (which is highly infectious). This unknown agent was believed to be responsible for the transformation of the genetic material in the bacteria. In 1940s, Oswald T. Avery and his colleagues proposed that this transformation agent was indeed DNA. Later in 1953, Alfred D. Hershey and Martha Chase provided more evidence to confirm Avery's discovery. A few years later, RiboNucleic Acid (RNA) was discovered to be the genetic material of viruses.

I.1.2 Modern Bioinformatics

The rapid growth of Bioinformatics was triggered by the introduction of the DNA sequencing technique. Walter Gilbert and Frederick Sanger were rewarded the Nobel Prize in Chemistry in 1980 for their contribution to nucleic acid sequencing. In 1981, the first complete genome sequencing was done on a circular genome of the human mitochondrion. Since then, many more genomes have been sequenced. In addition, the sequencing procedure became more time and cost effective. In 1990, the Human Genome Project (HGP) was initiated in the United States and was led by the National Institutes of Health (NIH) and the Department of Energy (DOE). After 13 years, the completion of the Human Genome Project was announced in 2003.

Bioinformatics has now become a large field with cross-expertise, such as molecular biology, genetics, biochemistry, computer science and statistics. Furthermore, this field continues to grow and evolve different sub-fields, such as proteomics and biomedical applications. In fact, more and more sub-fields are expected to evolve in the next 10 to 20 years.

I.1.3 The Chapter Goal and Layout

In this chapter, we will introduce how Evolutionary Computing (EC) can be used to solve problems in Bioinformatics. First we will discuss some important concepts in Bioinformatics. We will also introduce some key Bioinformatics problems and their mathematical formulations. Finally, we will provide two case studies and see how they can be solved using EC.

I.2 An Overview of Bioinformatics

I.2.1 Important Concepts in Bioinformatics

The cell is the fundamental unit of life (with the exception of viruses). Some organisms are single-cell and some are multi-cell. For example, a human being is composed of billions of cells. Although, all the cells are genetically identical, they can differentiate into different types of cells with different functionality to work together as a system. These complex systems not only require mechanisms that regulate the cells to work together, but also require mechanisms that regulate the processes within a cell. As a result, these intra-cellular and inter-cellular regulations in turn define the organism. In fact, all this information is encoded in the genetic material and is passed down in generations. For many organisms, DNA is the genetic material, which contains all the essential information about this organism. RNA and protein are important intermediates that carry out the 'instructions' written in DNA. Below we will discuss DNA, RNA and protein in detail and explain the relationships between them.

The DeoxyriboNucleic Acid is where the genetic information is stored. The DNA is a chain of small molecules called nucleotides. Each nucleotide is composed of three molecular components, a five-carbon sugar, a phosphate group and a nitrogenous base. In addition, there are four types of nitrogenous bases: adenine (A), thymine (T), guanine (G) and cytosine (C). Figure I.1 gives the molecular diagram of a DNA strand. The genetic information is encoded by alternating these four bases in the DNA strand. Moreover, this genetic information is usually represented as a DNA sequence, a string composed of characters from the alphabet {A, T, G, C}. A DNA strand is oriented, where one end is called the 5'-end (five prime end) and the other end is called the 3'-end (three prime end). Notice that some DNA strands are linear and some are circular. Regardless of the starting point of a DNA strand, the convention is to write the DNA sequence from the 5'-end to the 3'-end. That is, if we write TCGTA, the left end is the 5'-end and the right end is the 3'-end, i.e. 5'-TCGTA-3'. The length of a DNA strand can vary from a few thousands base pairs to a few millions base pairs in different organisms. DNA strands come in pairs and they are called complementary strands. Two complementary DNA strands complement of each other in the opposite orientation. Base A and base T are complements of each other, while base G and base C are complements. Two complementary DNA strands are bonded together in the shape of a double-helix by hydrogen bonds. Figure I.2 shows the double-helix complementary structure of DNAs. In fact, there are other higher level structures of the DNA strand, such as the supercoilng structure. Except during some cellular events (such as transcription or duplication), DNA strands are coiled and packed up with a special type of protein called histone. This super-structure is visible under the electron micrograph and it is called the chromosome.

The RiboNucleic Acid is similar to the DNA. In RNA, the five-carbon sugar has a hydroxyl group (OH) attached to the 2' carbon, instead of a hydrogen atom (H) as in the DNA. Besides, the base thymine (T) is replaced by uracil (U) in the RNA. But unlike the DNA, the RNA is usually single stranded. However, its

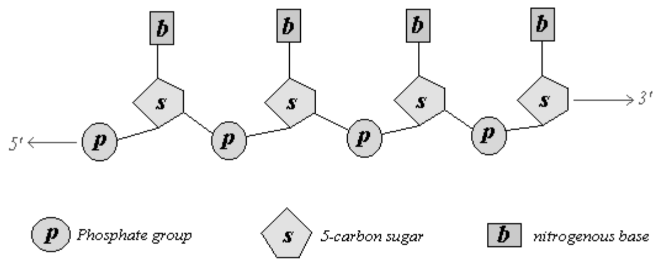


Figure I.1. The Molecular Structure of DNAs

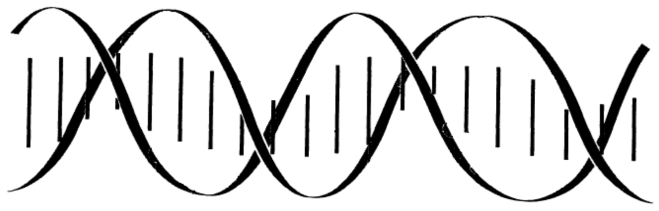
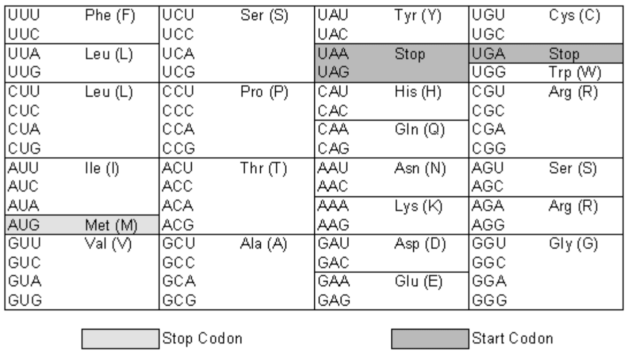


Figure I.2. The Double Helix Structure of DNAs

complementary property sometimes is exhibited within itself. That is, it 'folds up' itself when there are two segments in the RNA strand that are complementary or close to complementary. This characteristic is very important because it gives the RNA shape, which is crucial to its functionality. There are actually different types of RNAs, for example, the messenger RNA (mRNA), the transfer RNA (tRNA) and the ribosomal RNA (rRNA). One major function of the RNA is to synthesize proteins. The process is as follows. First, a segment of the DNA is 'copied' into the form of a RNA according to the complementary property. This process is called the transcription. Then, the RNA serves as the template that has encoded information carried from the DNA to produce a protein. This process is called translation.

The protein is essential for carrying out most of the mechanisms in the cell. The regulation of these mechanisms is indeed controlled by producing different proteins and in different quantities at different times. The protein is composed of a chain of amino acids. There are twenty different kinds of amino acids. Each amino acid is encoded by three RNA bases, and we call these three RNA units a codon. Since there are four different kinds of bases in RNA (and DNA), there a total of are sixty-four different combinations of codons. Therefore, some codons actually encode the same amino acid. Also, notice that some codons do not encode any amino acid, which is called the stop codon. The stop codon indicates the termination of the amino acid chain during the process of translations. Figure I.3 shows the codon/amino acid mapping table. Figure I.4 shows the relationships of DNAs, RNAs and proteins in transcriptions and translations.

A gene is a segment of the DNA that encodes proteins (i.e. a DNA segment that is transcripted to a RNA and in turn determines the protein synthesis). The



UUU	Phe (F)	UCU	Ser (S)	UAU	Tyr (Y)	UGU	Cys (C)
UUC		UCC		UAC		UGC	
UUA	Leu (L)	UCA		UAA	Stop	UGA	Stop
UUG		UCG		UAG		UGG	Trp (W)
CUU	Leu (L)	CCU	Pro (P)	CAU	His (H)	CGU	Arg (R)
CUC		CCC		CAC		CGC	
CUA		CCA		CAA	Gln (Q)	CGA	
CUG		CCG		CAG		CGG	
AUU	Ile (I)	ACU	Thr (T)	AAU	Asn (N)	AGU	Ser (S)
AUC		ACC		AAC		AGC	
AUA		ACA		AAA	Lys (K)	AGA	Arg (R)
AUG	Met (M)	ACG		AAG		AGG	
GUU	Val (V)	GCU	Ala (A)	GAU	Asp (D)	GGU	Gly (G)
GUC		GCC		GAC		GGC	
GUA		GCA		GAA	Glu (E)	GGA	
GUG		GCG		GAG		GGG	

▢ Stop Codon ▨ Start Codon

Figure I.3. The Codon/Amino Acid Mapping Table

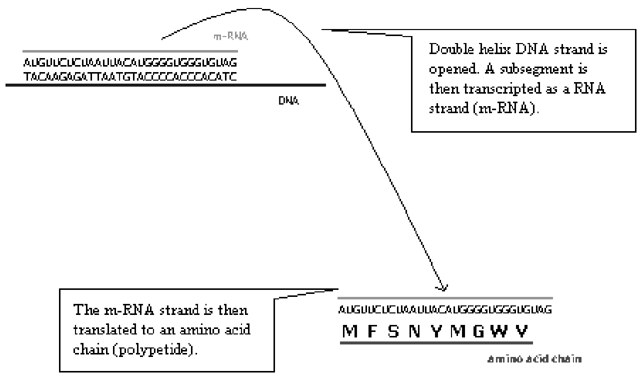

m-RNA
AUGUUCUCUAAUUACAUGGGGUGGGUGUAG
TACAAGAGATTAATGTACCCCACCCACATC
DNA
Double helix DNA strand is opened. A subsegment is then transcripted as a RNA strand (m-RNA).
The m-RNA strand is then translated to an amino acid chain (polypetide).
AUGUUCUCUAAUUACAUGGGGUGGGUGUAG
M F S N Y M G W V
amino acid chain

Figure I.4. The Relationships of DNAs, RNAs and Proteins in Transcriptions and Translations

genome is the whole set of DNAs in a cell. Although there are DNAs in some cell organelles, the term genome usually refers to the set of DNAs that serves as the genetic material.

The relationships, regulations and mechanisms between DNAs, RNAs and proteins are very complex and we understand very little of it. With the basic concepts in mind, below we will look at some of the major research areas in Bioinformatics, namely the problems of sequence alignments, multiple sequence alignments and functional site identifications.

I.2.2 Sequence Alignments

The sequence alignment is a computational model that is used to estimate the relationship between two biological sequences, such as DNA sequences and protein sequences. The alignment is used to model the imperfect matching between two sequences due to some biological events, such as mutations, additions or deletions of nucleotides. Regardless of the types of sequences to align, the basic concepts of alignments are essentially the same. Therefore, in this section the DNA sequence alignment is used to illustrate the idea.

Given two DNA sequences s_1 and s_2 {A, T, G, C}*, where * represents the kleene star, we want to extend them into s_1' and s_2' by inserting a special symbol '_', such that and have the same length. The special symbol '_' is used to represent an imaginary base that is missing from the sequence (or is added to the other sequence) due to a biological event. Therefore, our task is to find a pair of s_1' and s_2' that matches the best. That is let,

s_1 = AGTTGCA and s_2 = GTGCGA, then s_1' and s_2' can be:

$$\begin{array}{lcccccccccc} s_1' & = & \text{A} & \text{G} & \text{T} & \text{T} & \text{G} & \text{C} & \_ & \text{A} \\ s_2' & = & \_ & \text{G} & \_ & \text{T} & \text{G} & \text{C} & \text{G} & \text{A} \end{array}$$

Once we decided what s_1' and s_2' are, we can then evaluate how much they are alike. We first line up s_1' and s_2'. Then, we reward each position that has a matched base pair and penalize each position that has a mismatched base pair. We assign a $+1$ score for a matched base pairs and a -1 score for a mismatched base pairs. From the above alignment of s_1 and s_2, there are five matched base pairs and three mismatched base pairs. Therefore, it has an alignment score of $5 - 3 = 2$. Note that we do not want to have any position that has '_' on both s_1' and s_2' because it is biologically uninformative. Therefore, we could assign a negative infinity score for the position that has a '_' pair. As a result, the higher the alignment score is, the more s_1' and s_2' are alike.

In reality, the scoring scheme of matched and mismatched base pair is not that simple. Often, the scoring is based on the statistical observation of how likely it is for a certain base to undergo biological events. For computational purpose, we assume that there is a scoring function $f(a, b)$, where a and b are either a base in {A, T, G, C} or '_', that defines the score of a position of s_1' and s_2'. Therefore, the alignment score of a pair of s_1' and s_2' can be defined as $\sum_{i=1}^{l} f(s'_{1i}, s'_{2i})$, where l is the length of s'_1 (and s'_2), and s'_{1i} is the symbol at position i of s'_1 (similarly for s'_{2i}). As a result, we can describe the sequence alignment problem as follows. Given two sequences s_1 and s_2, find a pair of s'_1 and s'_2 that has the maximum alignment score.

The sequence alignment problem can be solved in $O(mn)$ time, where m and n are the length of s_1 and s_2 respectively. The $O(mn)$ time algorithm uses a dynamic programming to find the optimal alignment. We should clarify that the way we define the alignment score is just one of the many ways to do that. In fact, there are different models to evaluate the alignment score. For example, one idea is that two contiguous '_' should not be penalized the same as two separate '_'. Therefore, the concept of gap penalty was introduced.

I.2.3 Multiple Sequence Alignments

Sometimes, we want to investigate the relationships between a group of sequences. For example, a protein family is a group of proteins that are similar to each other. Moreover, a consensus sequence is a sequence that represents the motif of a protein family. The multiple sequence alignment can be used to extract the consensus sequence of a protein family. The concept of multiple sequence alignment is similar to that of the sequence alignment. Given a group of sequences s_1, s_2, ..., s_k. We want to find a set of s'_1, s'_2, ..., s'_k (all with equal length), such that $\sum_{i=1}^{k}\sum_{j=1}^{k} A(s'_i, s'_j)$ is maximized, where $A(s'_i, s'_j)$ is the maximum alignment score of s'_i and s'_j.

Although the multiple sequence alignment problem is defined similarly to the sequence alignment problem, the multiple sequence is in fact an NP-complete problem. Therefore, heuristic approaches are often used to solve this problem.

I.2.4 Functional Site Identifications

A site is a short segment of a DNA sequence that has a special function, for example, a starting site of transcription and a gene regulation region (called promoter). Often, we might know some instances of a site, and from this information, we want to identify whether a given short segment is possibly an instance of this site. For example, Table I.1 below are different putative promoter sequences preceding the messenger start point.

With the set of known instances, now we can create a profile for this site. We first create a 4 by l table T, where l is the length of this site. Each row in this table is labeled by a base and each column is labeled by a position of this site. In addition, an entry $T[b, i]$ is the number of occurrences of base b at position i. Therefore, for our example, we obtain Table I.2 below.

Sometimes, it is more convenient to calculate in terms of probability instead of number of occurrences. Therefore, if we translate the table values into a probability (that is, divide the value by the total occurrences), we have a probability table T'. Table I.3 below is the table T' for our example.

Table I.1. Putative Promoter Sequences Preceding the Messenger Start Point

GCATATAAAGT	mouse β^{maj} globin
GGGCATAAAAC	rabbit β globin
AGTTATATTAT	mouse λI Ig
GACTATAAAGC	rat insulin I & II
GGCTATAAAAG	adenovirus late
GGCTATATATT	chicken ovalbumin
TTGTATATATT	chicken ovomucoid
GTGTATAAAAA	S. purpuratus H2B
CAGTATAAAAT	B. mori silk fibroin

Table I.2. Number of Occurrences of Bases at each Position

	1	2	3	4	5	6	7	8	9	10	11
A	1	2	1	0	9	0	9	6	8	5	1
T	1	2	1	8	0	9	0	3	1	2	5
G	6	4	4	0	0	0	0	0	0	2	2
C	1	1	3	1	0	0	0	0	0	0	1

Table I.3. Probability of Occurrences of Bases at each Position

	1	2	3	4	5	6	7	8	9	10	11
A	0.11	0.22	0.11	0	1	0	1	0.67	0.89	0.56	0.11
T	0.11	0.22	0.11	0.89	0	1	0	0.33	0.11	0.22	0.56
G	0.67	0.44	0.44	0	0	0	0	0	0	0.22	0.22
C	0.11	0.11	0.33	0.11	0	0	0	0	0	0	0.11

Now we can test an unknown short sequence against the profile (represented in T') for this site. For example, given an unknown short sequence s = GGC-TATAAAAG, the probability that s is an instance of the putative promoter is:

$(0.67)(0.44)(0.33)(0.89)(1)(1)(1)(0.67)(0.89)(0.56)(0.22) = 0.0064.$

It is easy to observe that the probability of the unknown instance will be small when the site is long. In order to normalize the measure, we can calculate the maximum (max) and the minimum (min) probability of an unknown short sequence. For example,

$max = (0.67)(0.44)(0.44)(0.89)(1)(1)(1)(0.67)(0.89)(0.56)(0.56) = 0.022$, and

$min = (0.11)(0.11)(0.11)(0)(0)(0)(0)(0)(0)(0)(0.11) = 0.$

Therefore, the normalized score is:

$(0.0064 - min)/(max - min) = (0.0064 - 0)/(0.022 - 0) = 0.29.$

Also, notice that this method does not tell us whether 0.29 is high or low. Therefore, more statistical techniques are needed to solve this problem, such as using the concept of expected value.

I.2.5 Evolutionary Computing Applications in Bioinformatics

Many EC solutions have been introduced to solve problems in Bioinformatics. For example, Chellapilla and Fogel [8] proposed an evolutionary programming approach to the multiple sequence alignment problem (see multiple sequence alignment above). Later, Thomsen and his colleagues [28] extended the approach by using Clustal alignment to optimize the initial population. Platt and Dix [24] employed genetic algorithm to construct the restriction map; that is the map that indicates the relative positions of different genes on a DNA. Ando and Iba [1] used genetic algorithm to infer the gene regulatory model. Indeed, EC has also applied in many other problems such as RNA structure predictions [26], DNA motif discoveries [21], protein classifications [9] and so forth [12, 19, 20]. This chapter explains two EC applications in specific. The first case study uses EC to estimate the genome reversal distance for un-oriented genomes [2]. The second case study uses EC to find a largest compatible subset from phylogenetic data [3].

I.3 Genome Reversal Distance Estimation (case study 1)

I.3.1 Introduction

The mitochondria is a cell organelle that releases energy for the rest of the cell to consume. This organelle in fact contains a small set of DNAs for its own purpose (note that it is not the primary genetic material). In the late 1980s, Jeffrey Palmer

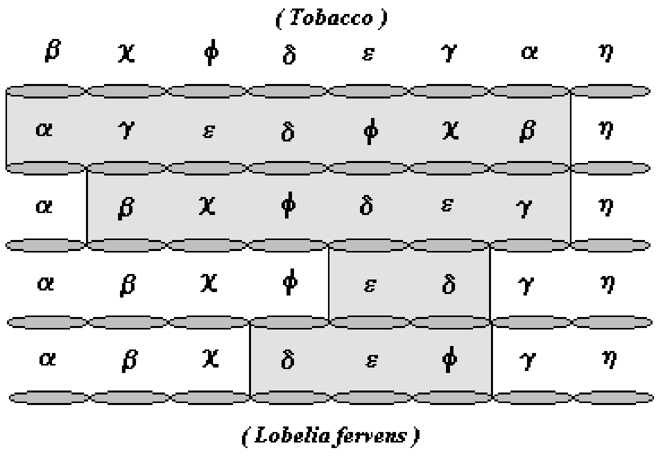


Figure I.5. The Estimated Transformation from *Tobacco* to *Lobelia fervens* by Reversals

and his colleagues discovered a gene shuffling pattern among different mitochondria genomes. This pattern is now known as the genome rearrangement.

The genome rearrangement describes a mechanism in which the genes order in the mitochondrial genome is constantly under rearrangement. Therefore, if we can estimate the rearrangement distance between two genomes, we can estimate the relationship between them. The reversal is the most commonly seen mechanism of the genome rearrangement. Figure I.5 shows the estimated transformation from *Tobacco* to *Lobelia fervens* by reversals. Due to this discovery, two mathematical models were proposed, signed permutations and unsigned permutations [5, 6, 23]. For unsigned permutations, a genome is modeled as a permutation π with an order n (i.e. a permutation of $\{1, 2, \ldots, n\}$), where n is the number of gene blocks in the genome. Let the permutation $\pi = \pi[1]\ \pi[2]\ \ldots \pi[n]$, the reversal operation $\rho(i, j)$ rearrange π into $\pi[1] \ldots \pi[i{-}1]\ \pi[j{-}1] \ldots \pi[i]\ \pi[j] \ldots \pi[n]$. For a signed permutation π', each $\pi'[k]$ has either a positive or a negative sign. Each reversal operation $\rho(i, j)$ not only rearranges π' but also negates the sign of $\pi'[k]$ for $i \leq k < j$. The problem of estimating reversal distance between two genomes is formulated as sorting the permutation by reversal operations. That is, given π (or π'), we want to find a sorting sequence that uses the minimum number of reversal to sort π (or π') into the identity permutation (i.e. the permutation, $1\ 2 \ldots n$ for unsigned permutations, and $+1\ +2\ \ldots{+n}$ for signed permutations). We called the minimum number of reversals the reversal distance, denoted by $d(\pi)$ (or $d(\pi')$). Figure I.6 provides an example of both the unsigned and the signed permutation and their sorting paths.

The problem of sorting signed permutations can be solved in $O(n^2)$ time [18]. Moreover, the problem of finding the reversal distance of a signed permutation can be solved in $O(n)$ time [4]. However, both sorting and finding the reversal distance of an unsigned permutation have been proven to be NP-hard [7]. Therefore, the EC approach can be used to search for the near-optimal solution.

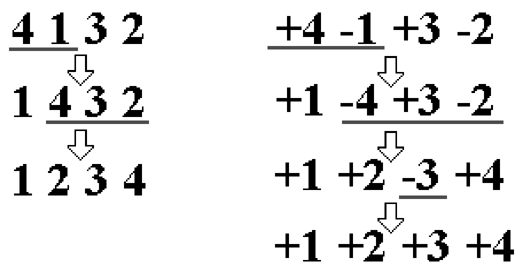


Figure I.6. An Example of the Unsigned and the Signed Permutation and their Sorting Paths

The EC approach requires an effective representation. A naïve representation may have a huge searching space, yet encounter the problem of invalid solutions. Fortunately, our knowledge of signed permutations can in fact provide us an effective EC representation for unsigned permutations. Below, we look at some background on unsigned and signed permutations.

I.3.2 The Breakpoint Graph

An unsigned permutation can be modeled by a breakpoint graph. For each gene (a number in the permutation), we create a node. The idea of the breakpoint graph is to mark the desired and realistic relationship between these nodes in the permutation. For each pair of the nodes, we draw a black edge between them if they are adjacent in the permutation, and we draw a red edge between them if they are adjacent in the identity permutation. In order to model the orientation, we expand the unsigned permutation to have a zero at the front and a n+1 at the end. An example is shown in Figure I.7. It has been shown that given a cycle decomposition of the breakpoint graph, any reversal can at most change the number of cycles by one. Besides, it has also been shown that given a cycle decomposition of the breakpoint graph, the corresponding shortest sorting sequence can then be found in polynomial time. Thus, the key problem is to find a cycle decomposition that provides the shortest sorting sequence for the unsigned permutation. However, the problem of finding an optimal cycle decomposition is NP-hard.

On the other hand, sorting signed permutations can be solved easily by using the breakpoint graph. First we create the breakpoint graph as above. Then for each gene node i, we split it into two nodes $2i$-1 and $2i$, with ascending order if i is positive, descending order otherwise. Also, note that node 0 and n+1 (the nodes we artificially added) are replaced by node 0 and $2n$+1 respectively. An example is shown in Figure I.8. The advantage is that now each node has exactly one red edge and one black edge associated with it. Because there is only one cycle decomposition of this breakpoint graph, the problem can now be solved in polynomial time. Many algorithms have been proposed to find the sorting sequence. There is an $O(n^2)$ time

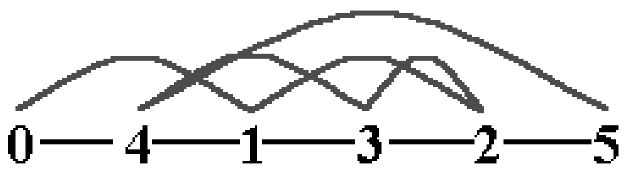


Figure I.7. The Breakpoint Graph for the Unsigned Permutation 4 1 3 2

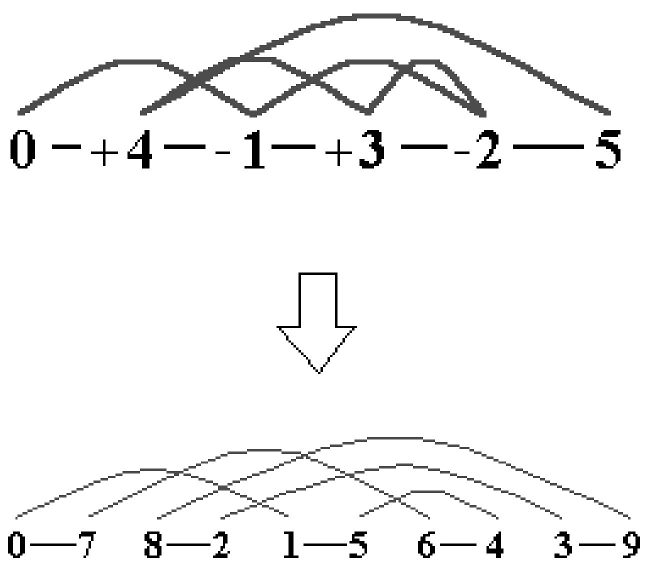


Figure I.8. The Breakpoint Graph for the Signed Permutation +4 -1 +3 -2

algorithm that finds the shortest sorting sequence. In addition, there is an $O(n)$ time algorithm that finds the reversal distance.

I.3.3 The Evolutionary Computing Solution

The idea of the EC approach [2] is to view all the possible cycle decomposition of the unsigned permutation as the signed permutations that have the same gene order. Except the node 0 and n+1, every node in the breakpoint graph has degree 2 for red edges and also for black edges. So, a cycle decomposition is used to define the (color) alternating paths. However, there is a corresponding signed permutation that actually defines the same alternating paths. Therefore, we can now turn our focus on the signed permutation instead of the cycle decomposition. We define the set *Signed*(π) be the set of signed permutations that have the same gene order as π. For example, when π is 2 1, then *Signed*(π) is {-2 -1, -2 +1, +2 -1, +2 +1}. Thus, the size of *Signed*(π) is 2^n. Furthermore, the following two observations are required for our EC approach.

Observation 1 *Each $\pi' \in Signed(\pi)$ can deduce a valid sorting sequence for π.*
Proof:

Let sorting sequence ρ sort π' into identity (i.e. $\pi'_\rho = id$). Because $\mid \pi'[i] \mid = \pi[i]$, then $\pi'_\rho[i] = \pi_\rho[i]$, for all i. Thus, ρ can also sort π (i.e. $\pi_\rho = id$).

Observation 2 *There exists a $\pi^* \in Signed(\pi)$ that deduces an optimal sorting sequence for π.*
Proof:

Let ρ be a sorting sequence for π that uses minimum number of reversals. For each $\pi[k]$, let $count[k]$ be the number of times that $\pi[k]$ is included in reversals of ρ, i.e. $\rho(i,j)$ $i \leq k < j$. Then π^* is as follows. $\pi^*[k]$ has a positive sign if $count[k]$ is even, otherwise it has a negative sign. Because $\mid \pi'[i] \mid = \pi[i]$, $\pi_\rho[i] = \mid \pi'_\rho[i] \mid$, for all i. In addition, all $\pi'_\rho[i]$, must be positive. Therefore, ρ can also sort π^* and the reversal distance for π^* is equal to the reversal distance for π.

From the two observations, we can see that the problem can be solved in $O(2^n n)$ time. That is to find the reversal distance for all 2^n corresponding π'. Let π^* be the π' that has the minimum reversal distance. Then the sorting sequence of π is the sorting sequence of π^*. However, it is not feasible to go through all 2^n π'. Therefore, genetic algorithms can be used to find π^*. We can expect the genetic algorithm to find a π' with low reversal distance. In fact, the result is an upper bound for the reversal distance of π.

The chromosome of the genetic algorithm is a binary string with length n, which represents the signs of the signed permutation. From observation 1, we know that any instance of the chromosome gives a valid solution. Therefore, it allows us to use generic bitwise crossover and mutation operations. Due to the bitwise chromosome representation and simple genetic operations, least computation is required which results in fast generations. The fitness can be some functions of $d(\pi')$, such as $\frac{1}{d(\pi')}$. Note that this fitness function requires $O(n)$ time. Figure I.9 gives a comparison between the EC approach and a heuristic approach [10]. In this experiment, single bit mutation and single point crossover operations were used.

Often, some information about the solution allows us to apply another heuristic or even prune out some search space without risking the loss of the global optimal. In fact, it has been proven that there exists a sorting sequence that does not break any sorted substring with length longer than 2. (Note that this observation is for substring straightly greater than 2. For example, when we optimally sort the permutation 3 4 1 2, one of the sorted 2-lengthed substrings (i.e. 3 4 or 1 2) has to be broken.) Therefore, for any long sorted substring, we can compress it into one number and according to whether it is ascendingly sorted or descendingly sorted, we can remove such bit from the chromosome and always assign it a positive or negative sign. Furthermore, there are different error bounded heuristics proposed. We can use these heuristic algorithms to enhance the quality of the initial population. For example, the idea of the $\frac{3}{2}$-approximation algorithm is to maximize the number of 2-lengthed (color) alternating cycles. Therefore, we can adapt the idea

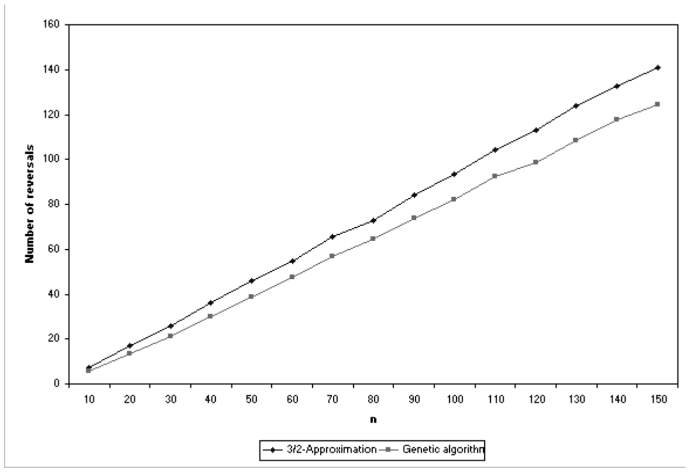


Figure I.9. A Comparison of the Evolutionary Approach and a $\frac{3}{2}$-approximation Approach

by initiating the population of different signed permutations that maximize some r-length alternating cycles (e.g. r=1, 2 or 3).

I.3.4 Open Problems

Sorting permutations by reversals is only one of the many mathematical models of the genome rearrangement. In fact, this model is the simplest one. Below we introduce two somewhat more realistic models of the genome arrangement problem. Both problems are very complex in nature and have no known efficient algorithm in general thus far.

Sorting Words by Reversals. Not all genes are uniquely existing in the genome. Sometimes, copies of the genes can be found in different locations in the genome. Sorting words by reversals is the mathematical model to address this reality. A word w is a string in *Gene**, where $Gene = \{1, 2, \ldots, n\}$. Without loss of generality, we assume that all elements in *Gene* are exhibited in w. This model is similar to the sorting permutations model, only that some genes can be duplicated. The problem is to find a sorting path using the minimum number of reversals to sort a genome π to a genome θ, where $\pi, \theta \in Gene^*$.

Sorting *A* permutations by Reversals. Another variation is sorting A permutations by reversals. This model addresses the fact that there are special sites for rearrangements. Rearrangements only occur when the beginning and the end of the subsequence allow binding for the responsible enzymes. Therefore, we now have special symbols (sites) from the set $A = \{a, b, c, \ldots\}$ which indicates different sites of the genes. For example, $3aa1baa4a52b$ is an A-permutation of 31452 with $A = \{a, b\}$. The reversal is valid only between two identical sites (i.e. $a-a$ or $b-b$).

The problem is to first identify whether the A-permutation is sortable, then to find a minimum sorting path.

I.4 The Largest Compatible Subset Problem for Phylogenetic Data (case study 2)

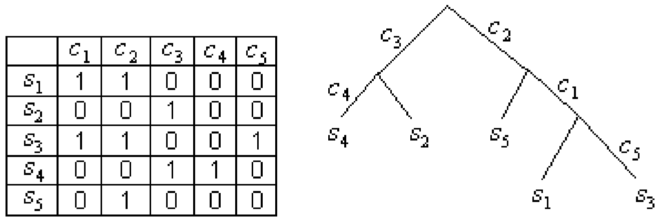
I.4.1 Introduction

The theory of evolution hypothezes that all organisms evolved from a single source, a simple organism. Therefore, under this hypothesis there is a relationship between every two species. One way to interpret this relationship is to look at the phenotypes (the features that can be observed) of the species. We call them the characters of the species. Furthermore, we expect that the more characters two species share, the closer relationship they have. The study of phylogenetics (phylogeny) is to understand these evolutionary relationships between the species. Different models have been proposed to model the evolutionary relationships between the species from different types of experimental data [16]. The perfect phylogeny [15] is a tree-based model that uses binary characters to infer phylogeny. Due to the noisy nature of the experimental data, conflicts are often found between subsets of the experimental data. Therefore, it is important to extract the motif from the imperfect data. The largest compatible subset problem is that given a set of experimental data, we want to discard the minimum such that the remaining are compatible.

In the case of the perfect phylogeny, the conflicts between subsets of data come pairwise [13, 14]. In other words, the compatibility between subset A and subset B are independent of other subsets. Moreover, if A is incompatible with B, B is also incompatible with A. From this condition, we model the incompatibility between subsets of data by a graph, where each vertex represents a disjointed data subset, and an edge (u, v) indicates that vertex u and v are incompatible.

In the context of a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the largest compatible subset problem [11] is to find a subset $U' \subseteq V$, such that $(u, v) \notin E$, $\forall u, v \in U'$ and $\mid U' \mid$ is maximum. Equivalently, we can find a subset $V' \subseteq V$ such that $\forall (u, v) \in E$, then either $u \in V'$ or $v \in V'$ (or both) and $\mid V' \mid$ is minimum. In fact, this problem has ready been studied in graph theory and is called the vertex cover problem [22]. However, the vertex cover problem has been proven to be NP-complete [17]. Therefore, the EC approach can be used to search for the near-optimal solution.

Again, a well-designed representation is critical for the EC approach. We can encode the chromosome to be a possible V'. But, there are $\sum_{i=1}^{n} \left(\begin{array}{c} n \\ i \end{array} \right)$ possible V'. Besides, some V' are in fact invalid. Here we would introduce an EC approach [3] that uses some knowledge on the vertex cover for special graphs to reduce the complexity. We also need a scheme to avoid invalid chromosomes. First, we present an overview on the perfect phylogeny and the vertex cover problem.



	c_1	c_2	c_3	c_4	c_5
s_1	1	1	0	0	0
s_2	0	0	1	0	0
s_3	1	1	0	0	1
s_4	0	0	1	1	0
s_5	0	1	0	0	0

Figure I.10. An Example of a Matrix and its Perfect Phylogenetic Tree

I.4.2 The Perfect Phylogeny

The perfect phylogeny is a phylogenetic model that uses binary characters. An m by n 0-1 matrix M records the exhibition of the n characters in the m species. That is $M[i, j] = 1$ if and only if species i exhibits character j, and zero otherwise. The definition of a perfect phylogenetic tree for matrix M is as follows. A rooted tree T is a perfect phylogenetic tree of M if it satisfies:

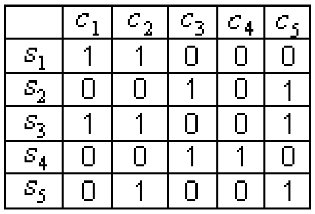
1. T has exactly m leaves;
2. each of the m species labels exactly one leaf of T;
3. each of the n characters labels exactly one edge of T; and
4. for any leaf l, the unique path from the root node to l contains labels that specify all the characters that l (its corresponding species) has.

Figure I.10 shows an example of a matrix M and its perfect phylogenetic tree T. Not all matrices are perfect phylogeny. For any two characters c_i and c_j, let o_i and o_j be the species who exhibit c_i and c_j respectively. The matrix M is a perfect phylogeny if and only if o_i and o_j are either disjoint or one contains the other, for every i, j. In other words, character c_i and c_j are incompatible when o_i and o_j are overlapped. The incompatibility between the characters in M can be visualized by a conflict graph, where each vertex represents a character and each edge represents a conflict. Figure I.11 shows an example of M that is not a perfect phylogeny and its conflict graph.

In practice, most experimental data is not a perfect phylogeny. Therefore, we want to extract a matrix M' from M, such that M' is a perfect phylogeny. The largest compatible subset problem for the perfect phylogeny is that given an m by n matrix M, find an m by n' matrix M' by removing columns from M, such that n' is maximum and M' is a perfect phylogeny. In the context of the conflict graph, we want to find the largest subset of vertices U' to keep, such that the resulting graph is edge free.

I.4.3 The Vertex Cover Problem

The largest compatible subset problem actually has already been studied in the form of the vertex cover problem in the graph theory. A vertex cover of a graph



	c_1	c_2	c_3	c_4	c_5
s_1	1	1	0	0	0
s_2	0	0	1	0	1
s_3	1	1	0	0	1
s_4	0	0	1	1	0
s_5	0	1	0	0	1

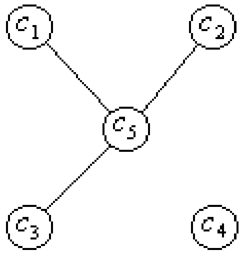


Figure I.11. An Example of a Matrix that is not Perfect Phylogeny and its Conflict Graph

$G = (V, E)$ is a set $V' \subseteq V$ such that if (u, v) is an edge of G, then either $u \in V'$ or $v \in V'$ (or both). The vertex cover problem is to find a vertex cover that minimize $\mid V' \mid$.

The largest compatible subset problem and the vertex cover problem are actually dual. The largest compatible subset problem is to find the largest subset U' to keep, such that the remaining graph is edge free, while the vertex cover problem is to find the smallest subset V' that covers all edges. That is $U' = V - V'$.

The vertex cover problem has been proven to be NP-complete. Therefore, there is no efficient algorithm to solve it (and the largest compatible subset problem too). As a result, different heuristic algorithms have been proposed to find a near-optimal solution for the vertex cover problem. For example, Algorithm I.1 will produce a vertex cover that is at most twice the size of the optimal vertex cover.

Algorithm I.1 A 2-approximation Algorithm for the Vertex Cover Problem

```
Algorithm 1 [$G = (V, E)$]
{
  $V' = 0$
  while ($E$ is not empty)
  {
     Pick an arbitrary edge $(u, v) \in E$
     $V' = V' \in u, v$
     Remove $(u, v)$ and all edges incident on $u$ or $v$ from $E$
  }
  return $V'$
}
```

Although there is no efficient algorithm to solve the vertex cover problem in general, the optimal solution can easily be found if the graph is composed of disjointed simple cycles and trees. The optimal vertex cover for a graph that is a simple cycle can be obtained by taking alternating vertices on the cycle starting from an arbitrary vertex. The optimal vertex cover for a graph that is a tree can be found by Algorithm I.2.

Algorithm I.2 A Linear-time Algorithm to find the Optimal Vertex Cover for a Tree.

```
Algorithm 2 [$G = (V, E)$]
{
  $V' = 0$
  while ($E$ is not empty)
  {
    Pick an arbitrary leaf node $u$ in $G$
    Let $v$ be the parent node of $u$
    $V' = V' \cup \{v\}$
    Removing $v$ and all edges incident on $v$ from $G$
  }
  return $V'$
}
```

I.4.4 The Evolutionary Computing Solution

The idea of the EC approach is to encode the vertex cover by a binary string, where each bit corresponds to a vertex in the graph. When a vertex u has a 1-bit, it represents including u in V' and (conceptually) excluding u from V' otherwise. However, the optimal presence can actually be found for the rest of the vertices once the presence of some 'critical vertices' is determined. Therefore, our representation of the chromosome is only composed of the set of critical vertices. We define the set of critical vertices to be the set $C \subseteq V$, where $\{u \in V \mid u$ has the edge degree larger than two and u belongs to a cycle in $G\}$. The following lemma shows the definition of critical vertices is sufficient to decompose an arbitrary graph into a graph that composes of only simple cycles or/and trees.

Lemma 1 *The set of non-critical vertices spans a graph G' that is composed of only simple cycles or/and trees.*
Proof:

If there exist a component in G' that is not a simple cycle nor a tree, then there must exist a vertex u that belongs to a cycle and has degree larger than two. But the set of non-critical vertices ($V - C$) is the set of vertices that have the degree less than three or it does not belong to any cycle in G, thus it contradicts the assumption.

Once the presence of the critical vertices is determined, the rest of the vertices can easily be determined by the methods discussed in the last section. However, some critical vertices can in fact be adjacent to each other. Thus, the effect of a vertex u with a 0-bit does not immediately exclude u from V', but it does immediately include all adjacent vertices of u from V'. Algorithm I.3 shows how the vertex cover is interpreted from a chromosome.

Once again, the EC representation and interpretation of the solution provides a better search space because it reduces the size of the search space and it ensures that all instances are valid solutions. Generic genetic operators can then be used. The fitness function can be some functions of $\mid V' \mid$, such as $\frac{1}{|V'|}$. Figure I.12 gives a

Algorithm I.3 An Algorithm to construct the Vertex Cover from a Chromosome.

```
Algorithm 3 [$G = (V, E)$]
{
  $V' = V$
  for each critical vertex $u$
  {
     if ($u$ has 1-bit)
          remove $u$ and all edges incident on $u$ from $G$
     else // $u$ has 0-bit
          remove all edges and vertices incident on $u$ from $G$
  }
  Use Algorithm 2 to find vertex cover on tree components and
  remove them from $G$
  Remove alternating vertices on simple cycles from $G$,
  starting from an arbitrary vertex
  $V' = V' - V$
  return $V'$
}
```

comparison between the EC approach and a heuristic approach. In this experiment, again single bit mutation and single point crossover operations were used.

The concept of critical vertices is the key to reduce the problem size. Figure I.13 shows the average number of critical vertices in different graph sizes. We can see that the concept of critical vertices reduces the problem size to about half, that is $(\frac{\sqrt{2}}{2})^n$ times of the search space.

I.4.5 Open Problems

Although the largest compatible subset problem for the perfect phylogeny demonstrates the fundamental concept of the compatibility problem, other variations of the models require some adaptations of the representations. Below we look at two similar compatibility problems.

Weighted Data. Sometimes, data subsets are not equally important. Biology experts might weight data subsets according to their source and characteristics. Therefore, we assign a weight for each vertex in the graph. Then, the problem is to find a vertex cover V' that minimizes the total weight, i.e. $\sum_{v \in V'} weight(v)$. Notice that this problem is inherently an NP problem, because its special case, when all weights are equal to one, is indeed the vertex cover problem.

Non-pairwise Conflicts. Sometimes, the conflicts are not in pairwise form. Therefore, we need to extend the graph definition to the hypergraph. A hypergraph is a graph that its edges can have more than two end-points. Therefore, a conflict exists between a group of species can be represented by an edge in the hypergraph.

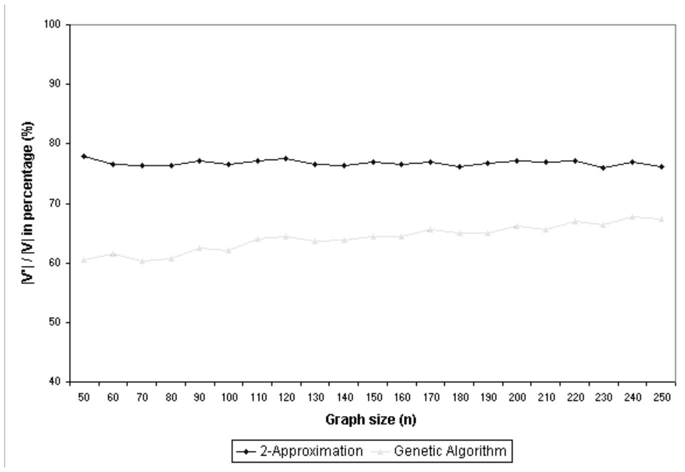


Figure I.12. A Comparison of the Evolutionary Approach and a 2-approximation Approach

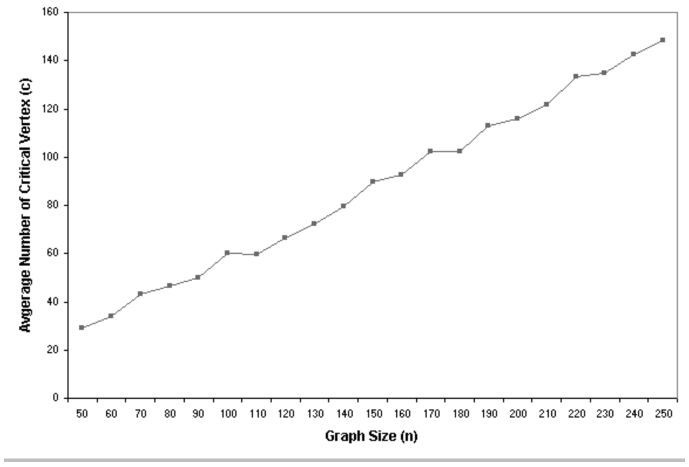


Figure I.13. The Average Number of Critical Vertices for Various Graph Size

Furthermore, by removing any vertex of this group, the conflict can be resolved. Similarly, this problem is inherently an NP problem because when all edges have exactly two end-points, it is same as the vertex cover problem.

I.5 Summary

This chapter introduced Evolutionary Computing applications in Bioinformatics. We reviewed some major events and explained some key concepts of Bioinformatics. We also gave an overview of some of the important problems. Furthermore, two case studies were presented in detail with the problem's background, the EC solution, the experimental results and the open problems.

Bibliography

[1] S. Ando and H. Iba, Inference of Gene Regulatory Model by Genetic Algorithms, The IEEE Congress on Evolutionary Computation, 1: pp. 712-719, 2001

[2] A. Auyeung and A. Abraham, Estimating Genome Reversal Distance by Genetic Algorithm, The IEEE Congress on Evolutionary Computation, pp. 1157-1161, 2003

[3] A. Auyeung and A. Abraham, The Largest Compatible Subset Problem for Phylogenetic Data, The Genetic and Evolutionary Computation Conference, 2004

[4] D. A. Bader, B. M. E. Moret and M. Yan, A Linear-Time Algorithm for Computing Inversion Distance between Signed Permutations with an Experimental Study, Journal of Computational Biology, 8(5): pp. 483-491, 2001

[5] A. Bergeron, A Very Elementary Presentation of the Hannenhalli-Pevzner Theory, The Twelfth Annual Symposium on Combinatorial Pattern Matching, pp. 106-117, 2001

[6] P. Berman and S. Hannenhalli, Fast Sorting by Reversal, The Seventh Annual Symposium on Combinatorial Pattern Matching, 1075: pp. 168-185, 1996

[7] A. Caprara, Sorting by Reversals is Difficult, The First International Conference on Computational Molecular Biology, pp. 75-83, 1997

[8] K. Chellapilla and G. B. Fogel, Multiple Sequence Alignment Using Evolutionary Programming, The IEEE Congress on Evolutionary Computation, 1: pp. 445-452, 1999

[9] S. Chiba, K. Sugawara and T. Watanabe, Classification and Function Estimation of Protein by using Data Compression and Genetic Algorithms, the IEEE Congress on Evolutionary Computation, 2: pp. 839-844, 2001

[10] D. A. Christie, A 3/2-Approximation Algorithm for Sorting by Reversals, The Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 244-252, 1998

[11] W. Day and D. Sankoff, Computational Complexity of Inferring phylogenies by Compatibility, Systematic Zoology, 35(2): pp. 224-229, 1986

[12] R. Deaton, R. C. Murphy, J. A. Rose, M. Garzon, D. R. Franceschetti and S. E. Stevens Jr., A DNA Based Implementation of an Evolutionary Search for Good Encodings for DNA Computation, The IEEE International Conference on Evolutionary Computation, pp. 267-271, 1997

[13] G. F. Estabrook, C. Johnson and F. R. McMorris, A Mathematical Foundation for Analysis of Cladistic Character Compatibility, Math Bioscience, 29: pp. 181-187, 1976

[14] G. F. Estabrook and F. R. McMorris, When are Two Qualitative Taxonomic Characters Compatible, Journal of Mathematical Biology, 4: pp. 195-200, 1977

[15] D. Gusfield, Efficient Algorithm for Inferring Evolutionary History, Networks, 21: pp. 19-28, 1991

[16] D. Gusfield, Algorithm on Strings Trees, and Sequences - Computer Science and Computational Biology, Cambridge University Press, 1997

[17] E. Halperin, Improved Approximation Algorithm for the Vertex Cover Problem in Graphs and Hypergraphs, SIAM Journal of Computing, 31(5): pp. 1608-1623, 2002

[18] H. Kaplan, R. Shamir andR. E. Tarjan, Faster and Simpler Algorithm for Sorting Signed Permutations by Reversals, The Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 344-351, 1997

[19] S. Kikuchi, D. Tominaga, M. Arita, K. Takashi and M. Tomita, Dynamic modeling of genetic networks using genetic algorithm and S-system, Bioinformatics 19(5): pp. 643-650, 2003

[20] A. Meade, D. Corne and R. Sibly, Applying Evolutionary Computation to Understanding the Insertion Behavior of LINE-1 Retrotransposons in Human DNA. The IEEE Congress on Evolutionary Computation, pp. 836-843, 2000

[21] A. Meade, D. Corne and R. Sibly, Discovering Patterns in Microsatellite Flanks with Evolutionary Computation by Evolving Discriminatory DNA Motifs. The IEEE Congress on Evolutionary Computation, pp. 1-6, 2002

[22] V. T. Paschos, A Survery of Approximately Optimal Solution to Some Covering and Packing Problems, ACM Computing Surveys, 29(2): pp. 171-209, 1997

[23] P. A. Pevzner, Computational Molecular Biology An Algorithmic Approach, MIT Press, 2001

[24] D. M. Platt and T. I. Dix, Construction of Restriction Maps Using a Genetic Algorithm, The Twenty-Sixth International Conference on System Sciences, 1: pp. 756-762, 1993

[25] P. J. Russell, iGenetics, Benjamin Cummings, 2002

[26] A. B. Shapiro, J. C. Wu, D. Bengali and M. J. Potts, The Massively Parallel Genetic Algorithm for RNA Folding: MIMD Implementation and Population Variation. Bioinformatics 17(2): pp. 137-148, 2001

[27] J. Tao, X. Ying and M. Q. Zhang, Current Topics in Computational Molecular Biology, MIT Press, 2002

[28] R. Thomen, G. B. Fogel, T. Krink, A Clustal Alignment Improver using Evolutionary Algorithms, The IEEE Congress on Evolutionary Computation, 1: pp. 121 -126, 2002