# Biological computation and computational biology: survey, challenges, and discussion

Zaineb Chelly Dagdia, Pavel Avdeyev, Md. Shamsuzzoha Bayzid

**Noname manuscript No.**
(will be inserted by the editor)

# Biological Computation and Computational Biology: Survey, Challenges, and Discussion

**Zaineb Chelly Dagdia · Pavel Avdeyev · Md. Shamsuzzoha Bayzid**

Received: 30 March 2020 / Accepted: date

**Abstract** Biological computation involves the design and development of computational techniques inspired by natural biota. On the other hand, computational biology involves the development and application of computational techniques to study biological systems. We present a comprehensive review showcasing how biology and computer science can guide and benefit each other, resulting in improved understanding of biological processes and at the same time advances in the design of algorithms. Unfortunately, integration between biology and computer science is often challenging, especially due to the cultural idiosyncrasies of these two communities. In this study, we aim at highlighting how nature has inspired the development of various algorithms and techniques in computer science, and how computational techniques and mathematical modeling have helped to better understand various fields in biology. We identified existing gaps between biolog-

The discussion in this paper about the challenges and importance of filling the gaps between the biological computation and computational biology communities represents the outcome of an analysis that has been done in the $5^{th}$ Heidelberg Laureate Forum; specifically during a workshop `https://scilogs.spektrum.de/hlf/experience-learn-share-heidelberg-laureate-forum/` organized by Dr. Zaineb Chelly Dagdia and mentored by Professor Stephen Smale (Fields Medal awardee). After the workshop, a collaboration was formed between Dr. Zaineb Chelly Dagdia who works on biological computation and two participants and contributors to the workshop, Pavel Avdeyev and Dr. Md. Shamsuzzoha Bayzid, who work on different areas in computational biology.

Zaineb Chelly Dagdia
Université Paris-Saclay, UVSQ, DAVID
LARODEC, ISG, Université de Tunis, Tunis, Tunisia
E-mail: zaineb.chelly-dagdia@uvsq.fr

Pavel Avdeyev
Department of Mathematics and Computational Biology Institute,
The George Washington University, USA,
E-mail: avdeyev@gwu.edu

Md. Shamsuzzoha Bayzid
Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology, Bangladesh,
E-mail: shams_bayzid@cse.buet.ac.bd

ical computation and computational biology and advocate for bridging this gap between "wet" and "dry" research.

**Keywords** Biological Computation · Computational Biology · Survey

## 1 Introduction

Computer science and biology have communally lived a long history together while enjoying a productive and fruitful collaboration for several decades. For many years, computer scientists have designed several computational methods and algorithms all inspired by the biological systems. Likewise, biologists relied on computer science frameworks to retrieve and analyze biological data; among other needs. Nonetheless, in spite of the convergence of these two directions, both produce questions that require us to consider their deep implications to not only build successful applications and more accurate models but also in providing a further fine-grained biological understanding leading to new biological insights.

It is known that nature has always served as inspiration for scientists across various disciplines and computer science is, definitely, no exemption. Computer scientists have relied on biological systems for inspiration to develop new, nature based techniques for solving difficult computational problems. Back in the 60s, works from artificial intelligence highlighted the development of a class of computational methods known as neural networks which are related to the activity of neurons in the brain. These have been used in many machine learning applications such as financial predictions and miscellaneous applications. Other works were inspired by common operations in *Deoxyribonucleic Acid* (DNA) sequence evolution which led to the development of genetic algorithms used to solve optimization problems. A number of additional inspired methods, including what the human immune system can teach us about protecting computer networks has also capitalized on biological insights to derive new computing paradigms.

The reverse direction, that of applying studies and ideas from theoretical computer science to improve our understanding of biological phenomena, is called computational biology. This field has emerged in the early 1950s when the British mathematician and logician Alan Turing used early computers to implement a model of biological morphogenesis. By 1953, computers had been applied to deal with much more-varied sets of analyses, namely those examining protein structure, leading to the discovery of the structure of DNA. Computational biology offers the development and application of data-analytical and theoretical methods, mathematical modeling and computational simulation techniques to the study of biological systems. Great strides have been made in understanding the complexity and diversity of biological phenomena. It became essential to use various techniques from computer science, math and statistics in biological research. The recent unprecedented growth in the size and scope of biological data sets completely changed the landscape of regular biological research. In our days, it is almost impossible to make a discovery in biology without representing, manipulating, analyzing, and interpreting biological data. For helping biologists to address emerged challenges dealing with data, among other challenges, computational biology community has developed profound approaches in the past four decades.

Although biological computation and computational biology have much in common, the two directions - relying on biological ideas to develop computational

methods and using computational methods to study biology - remained somehow disconnected. For example, while computational methods inspired by nature led to fruitful applications, unfortunately they only relied on a high-level and a general understanding of the biological principals they were based on, and therefore most of the time they did not directly lead to new biological insights. Likewise, although the development of novel computational methods that aim at helping researchers learn new biology, the application of these methods to the biological problem rarely fed back to help computer scientists design better algorithms.

In this paper, we identify and deeply discuss various challenges in building a fruitful collaboration between the biological computation and the computational biology communities. This discussion motivates the global contributions of this work which are (i) an up-to-date review on nature inspired algorithms, referred to as biological computation, while describing how nature has inspired the development of different algorithms and computational techniques, (ii) an up-to-date review on how computational techniques and mathematical modeling have helped to study biological systems, and (iii) highlighting the impact of each of these two disciplines, i.e., biology and computer science, on each other and how they can guide and benefit each other to not only improve the understanding of biological processes but also to make notable advances in the design of algorithms. In this concern, we have also discussed the possible reasons for such gap between biological computation and computational biology while addressing the following questions: How can we apply our knowledge of computer science, math and statistics to enhance the study of biology? What challenges do both disciplines share? What challenges may both directions offer to each other? How can we further boost the communication between the two directions?

For seeking answers to all these questions, we will first provide a comprehensive review of the concepts encountered in biological computation by describing different kinds of nature-inspired models. This will be presented in Section 2. Note that the aim of this section is not to provide a comparative study by determining the strengths and pitfalls of the bio-inspired algorithms or to guide researchers into which techniques to use for various problems. Instead, the objective of this up-to-date survey is to show the tremendous impact and contribution that nature has on developing computational techniques. In Section 3, we provide a review on various important topics in computational biology with an emphasis on how the mathematical and computational techniques have been used to tackle various problems in biology. Since the number of questions addressed by computational biology is almost the same as the number of biological phenomena, we focus on a few of the major research areas, especially the topics related to computational and evolutionary genomics. In Section 4, we will discuss the existing gaps and challenges with respect to conducting research in the interface of computational techniques and biology. We will identify a set of challenges that needs to be considered for better synergies between the computational biology and the biological computation communities. Finally, we draw our conclusions in Section 5.

## 2 Biological Computation

Nature has always been a rich source of inspiration for many researchers in several and different ways. By far a large set of nature-inspired algorithms are those which

are based on special characteristics of the biological system. Thus, the largest fraction of nature-inspired algorithms are biology-inspired, or bio-inspired for short. Even with emphasizing the source of inspiration, we can still have various ways to categorize the nature-inspired algorithms, depending on the details and the sub-sources to be used. In this section, we will categorize the nature-inspired algorithms into four main branches namely *Artificial Immune Systems*, *Connectionist Systems*, *Evolutionary Computing*, and *Swarm Intelligence*. A summary of the main biological systems that have inspired computational algorithms and their related application domains will be given at the end of this section in Table 1. As previously mentioned, the aim of this section is not to provide a comparative study of the bio-inspired algorithms. The aim is to show the tremendous impact and contribution that nature has on developing the up-to-date discussed computational techniques.

Please, also, note that different versions of the biological computation algorithms were proposed and a discussion of these and their representative software is out of scope of this Section. For a specific bio-inspired algorithm, the different proposed algorithmic versions are similar in their basic approach and in making use of the bio-inspired concepts, but they mainly differ either in the way they represent the information or in the way they are structured, e.g., optimized version, additional components, etc. In what follows, we will give an overview of the basic bio-inspired techniques within the defined four main branches.

## 2.1 Artificial Immune Systems

Although we are in permanent contact with innumerable germs in the environment of which some are pathogenic, the infections which we develop are relatively rare. The reason is that our organism has multiple means of defense which constitute the immune system (IS). Notable and remarkable characteristics are expressed by the IS as it is highly distributed, highly adaptive, self-organizing in nature, maintains a memory of past encounters, and has the ability to continually learn about new invaders. From a computational viewpoint, the natural immune system has much to offer by way of inspiration for the creation of novel approaches to computational problems. This field of research is referred to as *"Immunological Computation"* (IC) or *"Artificial Immune System"* (AIS) [82]. The study and design of AIS still represent a relatively new area of research that tries to build bio-inspired computational systems. In what follows, we give an overview of the main natural immunological concepts used by AIS as well as a synopsis of the bio-inspired approaches.

### *2.1.1 Immunological Concepts*

The natural immune system is a network of cells, tissues, and organs that work together to defend the body against attacks by "foreign" invaders that are trying to do harm to it. This main task is achieved thanks to its capability to recognize the presence of infectious foreign cells and substances, known as "non-self" elements and to respond to them by eliminating them or by neutralizing them. This distinction between the "non-self" and the body's "self" cells is based on a process called "self-non-self discrimination" [163]. The non-self elements, also called

"antigens" or "pathogens", are mainly microbes; tiny organisms such as bacteria, parasites, and fungi. All of these can, under the right conditions, cause damage and destruction to parts of the body and if these were left unchecked, the human body would not be able to function appropriately. Thus, it is the purpose of the immune system to act as the body's own army by preventing antigens from doing harm to it. This is achieved through several lines of defense of the immune system and several immune strategies. These are explained as follows:

The protection layers can be divided as physical barriers such as the skin and the respiratory system that provide effective barriers to the entry of most microorganisms, physiological barriers such as destructive enzymes and stomach acids that provide unsuitable living conditions for foreign pathogens, and the immune system which can be broadly divided into two heads which are the innate immunity and the adaptive immunity. These are interlinked and they influence each other.

The innate immunity is the first line of defense against invading antigens. It is those parts of the immune system that work no matter what the damage is caused by. They are always at work and do not need to have seen the offending invader before to be able to start attacking it. The innate immune system, which protects the body non specifically, includes phagocytic barriers, blood proteins, and cytokines. The phagocytic barriers work on ingesting and destroying microbes, and they ensure the interaction with the rest of the immune system. These tasks are mediated by antigen presenting cells and phagocytes such as neutrophils, macrophages and natural killer cells. Blood proteins, known as "complement proteins", perform four major functions including the lysis of infectious organisms, i.e., rupturing membranes of foreign cells, the activation of inflammation, the opsonization, i.e., enhancing phagocytosis of antigens, and ensuring immune clearance. The last components of the innate immunity are the cytokines which are proteins produced in response to antigens. They mediate and regulate immune and inflammatory reactions, and allow cells to communicate with each other.

On the other hand, the second line of defense is the adaptive immune system which affords protection against re-exposure to the same pathogen. The adaptive immune system is called into action against pathogens that are able to evade or overcome innate immune defenses. The cells of the adaptive immune system are mainly lymphocytes, the B-cells and the T-cells, but there are also other important parts of the adaptive immune system such as the complement cascade and the production of antibodies. These mentioned elements of the immune system do not work separately, but all work together in a co-operative fashion. If they have to work effectively then they need a good system for communicating messages. This system is provided by the released cytokines.

Based on these different lines of defense and immune strategies, we refer to a multi-layered immune system.

### *2.1.2 Synopsis of Artificial Immune Systems*

An inspiration from the remarkable properties expressed by the natural immune system led to the conception and the design of artificial immune systems exhibiting similar functionalities. These systems are discussed in what follows.

*Clonal Selection Theory* Clonal selection theory [47] is the algorithm used by the immune system to clarify the basic response of the adaptive immune system to

antigenic stimulus. Clonal selection involves two main concepts which are cloning (or proliferation) and selection (or affinity maturation). More precisely, it establishes the idea that only those cells capable of recognizing an antigen will proliferate while other cells are selected against. Clonal selection calls both B-cells and T-cells. When B-cells antibodies – which are a group of proteins – interact with an antigen (also referred to as *antigen binding*), cells become activated and they differentiate either to be plasma cells or memory cells. The closer the matching between an antibody and a specific antigen is, the stronger is the bind. This property is called "affinity". Plasma cells make large amounts of specific antibodies that work against specific antigens to destroy them. As the genes for antibodies in B-cells frequently suffer mutation, the antibody response improves after repeated immunizations; this phenomenon is called "affinity maturation" (see the part below the differentiation line). On the other hand, in the upper part of the differentiation line, memory cells remain with the host and promote a rapid secondary response. However, before this process, clones of B-cells are produced and undergo somatic hypermutation. Consequently, diversity is introduced into the B-cell population.

Based on this theory, various clonal selection algorithms have been proposed in the literature and most of them are devoted to perform computational optimisation and pattern recognition tasks [356]. In particular, inspiration has been taken from the antigen driven affinity maturation process of B-cells with its associated hypermutation mechanism. These key features are exploited from a computational viewpoint where an antibody, for instance, represents a solution to the problem being solved, the affinity defines the fitness function as a quantitative measure of effectiveness of the system, the antigen represents the value of the fitness function to optimize, the cloning process is seen as the reproduction of solutions, the somatic hypermutation represents the mutation of a solution, and the affinity maturation represents the mutation and the selection of best solutions. Adding to these, the developed clonal selection based algorithms utilize the idea of memory cells to retain good solutions to the problem being solved. A detailed description and applications of AIS clonal selection algorithms can be found in [356].

*Immune Network Theory* The immune system, being a dynamic and auto-regulated system, is a network of cells and antibodies that have a profound sense of self and the ability to remember and learn. The immune network theory states that the recognizers of the immune system, the B-cells and antibodies, not only recognize foreign particles but also recognize and interact with each other. This created network is based on interconnected B-cells for antigen recognition. The strength of the B-cells connections is directly proportional to the affinity that they share. Indeed, B-cells can stimulate, activate, and suppress each other even in the absence of antigens in order to stabilize the network.

Basic concepts of the immune network theory have been implemented leading to several immune algorithms dedicated to data analysis, unsupervised clustering, data visualization, and to solve optimization problems [139]. From a computational viewpoint, the main objective of the immune network based algorithms is to prepare a repertoire of pattern detectors for a given problem domain. In such configuration, the better performing cells will suppress low-affinity (or similar) cells in the network. This process is achieved through an interactive procedure of exposing the population to external information to which it responds. A detailed description of AIS immune network algorithms can be found in [139].

*Self-Non-Self Theory* The self-non-self theory is able to tell the difference between what is foreign and potentially harmful, and what is actually a part of its own system. The representative self-non-self theories are the negative selection and positive selection theories. The purpose of the negative selection theory is to provide tolerance for self cells. During the process of generating T-cells, and based on a pseudo-random genetic rearrangement process, a set of receptors are made. After that, and in the thymus, these receptors undergo a censoring process that is named the negative selection. There, the T-cells which react against self-proteins get destroyed. Therefore, only a limited set of T-cells which do not bind to self-proteins leave the thymus. These circulate throughout the human body to protect it against foreign antigens; by performing immunological functions [4]. As for the positive selection theory, it works as the opposite of the negative selection process. Specifically, positive selection is a process which is responsible for controlling survival: T-cells are tested for recognition of MHC molecules which are a group of genes that code for proteins found on the surfaces of cells that help the immune system recognize foreign substances. If a T-cell fails to recognize any of the MHC molecules, it is discarded; otherwise, it is kept.

From a computational perspective, the negative selection based algorithms focus on the principles of T-cells maturation which are capable of binding only non-self antigens, and on the fact of considering these cells a kind of anomaly detectors. This principle is achieved by building a model of anomalous or unknown (non-self) data through the generation of a set of patterns that do not match an existing corpus of available self (or normal) patterns. The prepared non-self model is then used to monitor the existing self data by seeking matches to the non-self patterns. An inspiration from the negative selection and positive selection theories gave rise to numerous AIS algorithms which are mainly applied to computer security, network intrusion detection problems [150], classification problems [127] and to solve optimization problems [52]. A review of the progress of negative selection algorithms can be found in [164], and a survey of methods and tools to detect recent and strong positive selection can be found in [282].

*Danger Theory* The danger theory [234] is a new theory that has become popular amongst immunologists. It was proposed to explain current anomalies in the understanding of how the immune system recognizes foreign invaders. The central idea in danger theory is that the immune system does not respond to non-self but to danger. Thus, just like the self-non-self theory, it fundamentally supports the need for discrimination. However, it differs in the answer to what should be responded to. Instead of responding to foreignness, the immune system reacts to danger based on environmental context (signals) rather than the simple self-non-self principle. Specifically, upon cell death, cells release specific signals that urge the behavior of the system since they are a reflection of the state of the environment. There are three main categories of released signals namely the Safe Signals (SSs) which are released as a result of normal cell death (apoptosis), Pathogen Associated Molecular Pattern Signals (PAMPs) which are essential molecules produced by microbes but not produced by the host, and Danger Signals (DSs) which are released as a result of an accidental cell death by harmful pathogens (necrosis). Those signals are recognized by antigen presenting cells, specifically by Dendritic Cells (DCs), and based on this phenomena the immune system recognizes the danger zone and then evaluates the vulnerability of the system. AIS algorithms based on danger

theory have been mainly used to solve computer security and machine learning problems [392,315].

Meanwhile, it is important to mention that danger theory is still considered as a relatively new addition to the field of immunology, and therefore danger theory inspired algorithms are still in their infancy. The key algorithm which was proposed in the literature and which introduced the concepts of danger signals is the Dentritic Cell Algorithm (DCA) [135]. The algorithm is inspired by the role and behavior of natural DCs. These cells are in charge of catching, processing and revealing antigens to T-cells. They, also, express receptors on their surfaces to receive signals (SS, DS and PAMPs) from their neighborhood. The first status of a DCs is the immature state (iDCs) where they collect antigens and signals. Based on the combination of the various signals received, the cells differentiate into two different maturity levels. Under the reception of safe signals, iDCs migrate to the semi-mature state and they cause antigens tolerance. iDCs migrate to the mature state (mDCs) if they are more exposed to danger signals and to PAMPs than to safe signals. In this case, they present the collected antigens in a dangerous context.

Based on these DCs immunological concepts, the information processing objective of the DCA is to prepare a set of mature DCs (seen as prototypes) that offer context specific information about how to classify normal (safe) and anomalous (dangerous) input patterns. This is achieved based on three key asynchronous steps, namely migrating sufficiently stimulated iDCs, promoting migrated cells to either the semi-mature (safe) or to the mature (danger) context depending on their accumulated response, and finally labeling the patterns as safe or as dangerous based on the composition of the sub-population of cells that respond to each pattern.

As noticed, the DCA was originally designed and used as an anomaly detection algorithm. In literature, a variety of DCA versions were developed aiming at improving the algorithm by conducting investigations on it, and by addressing and resolving its limitations. A full review of the DCA and its applications can be found in [60] while some other recent DCA versions can be found in [77,76].

## 2.2 Connectionist Systems

In recent years, connectionism has re-emerged as a dynamic and active area of research. Initially, the main motivation of connectionist systems was the aspiration to comprehend and mimic the information processing principles of the biological nervous system. By now, these systems have grown and spread into various subdisciplines for which cognitive science no longer represent the lone concern. From a methodological perspective, connectionist systems aim at studying cognitive phenomena using architectures of processing units which are interrelated via weighted connections. These architectures, presenting analogies to the natural neural systems, are referred to as "Artificial Neural Networks" (ANN) [312]. In literature, several neural network models have been proposed and implemented to explain various aspects of cognition. In what follows, we will outline the key functioning of the biological neural network and then discuss the main ANN taxonomy and types.

### *2.2.1 Biological Neural Network*

Up to now, much is still unknown about how the human brain trains itself to process information. In the brain, a neural network is seen as a chain of interlinked neurons. Its activation defines a recognizable linear pathway. More precisely, a neuron gathers signals from other neurons by means of a host of fine branches called "dendrites". The neuron sends out pulses of signals, over a long and thin stand termed "axon" that splits into many branches. A structure named "synapse" is at the end of each of these branches. Synapses are membrane-to-membrane connections containing molecular machinery that allow a fast transmission of signals. If the neuron receives a high enough level of input signals within a definite period of time, the neuron sends an action potential, in the form of an electrical pulse, at the axon and transmits this signal along the axon into the terminals. These outgoing signals are then received by other neurons.

### *2.2.2 Artificial Neural Networks*

Biological neural networks have inspired the design of Artificial Neural Networks (ANNs) [312] as a novel structure for information processing. ANNs are made of artificial neurons, considered as processing elements, and are prearranged in three inter-connected layers; namely the input layer, the hidden layer that may include more than one single layer, and the output layer. The first layer encloses input neurons that send information to the hidden layer that in turn sends data to the output layer. The basic structure of an ANN is given in Figure 1.

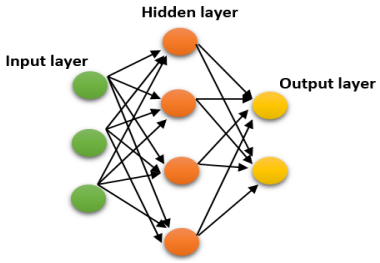


**Fig. 1** Basic structure of an artificial neural network: input layer, the hidden layer(s), and the output layer .

In the ANNs' structure, every single neuron has its weighted inputs which are mapped as synapses, an activation function which defines the output given the collected inputs, and some output(s) to produce. In such structure, the synapses represent the adjustable parameters that convert a neural network to a parameterized system. Based on the weighted sum of the inputs, an activation signal is generated that is transferred to the activation function, e.g., rectified linear activation function, radial basis function, logistic function, linear function, etc., to obtain some output(s) from the neuron. The training process consists in optimizing the weights in a way that the error of predictions is minimized and the network reaches a definite level of accuracy. The approach that is commonly used to determine the error contribution of each neuron is named "backpropagation" and it

aims at computing the gradient of the loss function with respect to the weights of the network for a single input–output example.

As mentioned, any ANNs structure may be composed of several additional hidden layers. This is to make the system more flexible and more powerful. ANNs with more than two hidden layers between the input layer and the output layer are named "deep neural networks". These can model more complex relationships. Numerous ANNs architectures and approaches have been proposed in the literature [128], and are exponentially growing. In what follows, we will discuss the most common taxonomy of artificial neural networks.

*Feed Forward Networks* Feed Forward neural networks (FF) [306] are very simple and straightforward architectures. They allow signals (information) to travel in one direction only between the nodes. A layer alone never has connections and in general two adjacent layers are fully connected (every neuron form one layer to every neuron to another layer)[1]. In such architectures, data move from the input nodes to the output nodes, passing through the hidden nodes (if any). The activation flows from the input to the output layer without any feedback (no cycles or back loops), i.e., the output of any layer does not affect that same layer. In a supervised learning, the network is given a paired data sets of "what goes in" and "what the user want to have as output" while in an unsupervised learning only the input is given to the network which will try to learn and figure out the most appropriate output. Given that the network has enough hidden neurons, it can theoretically always model the relationship between the input and output. Practically, the use of these ANNs is somehow limited but they are popularly hybridized with other ANNs to build new networks.

Various version of the FF network have been proposed where these use different activation functions. Among the various functions and networks, we mention the Radial Basis Function (RBF) network [45] which is a FF network applying the radial basis function. It functions exactly as a FF network but its specificity is the use of this specific activation function.

*Multilayer Perceptron* Multilayer Perceptron (MLP) networks opened the gate to deep learning. This class of networks involves multiple layers of computational units which are usually interconnected in a feed forward way. It applies a nonlinear activation function, e.g., hyperbolic tangent, logistic function, etc., that allows the network to classify data that is not linearly separable. In an MLP architecture, every node in a layer connects to each node in the next layer making the network fully connected. MLP is trained using the backpropagation approach involving the adjustment of the parameters (weights) of the model in order to minimize the error. The error being backpropagated is usually the variation of the difference between the input and the output, and it can be measured in a variety of ways, e.g., Mean Squared Error, Root Mean Squared Error, etc.

*Auto-encoders* The main idea of Auto-encoders (AEs) [40] is to automatically encode information, i.e., as in compress, not encrypt. The AE network has an hourglass shape where the hidden layers are smaller than the input and the output layers. AEs are also symmetrical around the middle layer(s). In such architecture,

[1] `https://www.asimovinstitute.org/blog/`

the hidden layers represent the checkpoints of the network where the information is most compressed. The part of the network which is up to the middle is named the "encoding part", the part which is after the middle is called the "decoding" part, and the middle reflects the "code". AEs may be trained using the backpropagation approach.

Variants of AEs have emerged and among these, we mention the Sparse Auto-Encoders (SAEs) [292], the Variational Auto-Encoders (VAEs) [180] and the Denoising Auto-Encoders (DAEs) [358]. SAE can be seen as the opposite architecture of AEs. More precisely, the network instead of learning and representing the information in less "space" or nodes, it will try to encode information in more space. Consequently, the network instead of converging in the middle and then expanding back to the input size, the structure will grow in the middle. SAE networks can be used to extract many small features from a given data set. On the other hand, VAEs are based on the same architecture as AEs but are trained differently as they compress an approximated probability distribution of the input samples. They rely on Bayesian mathematics regarding probabilistic inference and independence, as well as a re-parametrization trick to achieve this different representation. The key idea is to take into account the influence that one node may have on another. Practically, if one thing happens in one part of the network and something else happens in another part, then they are not essentially correlated. If they are not linked then the error propagation should consider this. Because ANNs are generally large graphs, VAEs become very useful as they help to rule out influence from some nodes to other nodes as one dives into deeper layers. As the name suggests, DAEs are based on noise. More precisely, DAEs take a partially corrupted input (also referred to as noise) and are trained to recover the original undistorted input. The objective of DAEs is to clean the corrupted input; or denoising. This is a key characteristic of DAEs where the network is encouraged to not learn details but to learn wider features. This is because learning smaller features often turns out to be mistaken or insufficient due to its constant change with noise.

*Convolutional Neural Network* Convolutional Neural Networks (CNN) [200] are relatively different from most other neural networks. They feature convolutional layers, pooling layers and fully connected layers; each perform a different task. Convolution layers apply a convolution operation to the input data and pass the result to the following layer. This operation allows the network to be deeper with much fewer parameters. Pooling layers simplify the network by reducing unnecessary features. This is achieved by combining the outputs of a convolutional layer into a single output and pass it to the next convolutional layer. Fully connected layers connect every neuron in one layer to every neuron in another layer. CNNs are typically used for image processing and specifically for image recognition.

They tend to start with an input "scanner" which is not intended to parse all the training data at once, i.e., the input window is sliding along the image, pixel by pixel. This input data is passed afterwards to the convolutional layers. These convolutional layers can be seen as a funnel since they compress the features which are detected and filter out details.

Real world CNNs implementations often attach deep feed forward networks, e.g., MLP, to the final convolutional layer to further process the data.

Among the variants of CNNs, we mention Deconvolutional Networks (DNs) [382] and Deep Convolutional Inverse Graphics Networks (DCIGNs) [191]. DNs

are seen as reversed CNNs. In the context of image processing, for instance, the structure is seen as if the network is fed the output, e.g., the word cat, and trained to generate the right corresponding input image, e.g., cat-like pictures. DCIGNs, on the other hand, are essentially VAEs but with CNNs and DNs used as encoders and decoders, respectively. DCIGNs can learn to model complex transformations on images such as changing the source of light or the rotation of a 3D object. These networks tend to be trained with backpropagation.

*Recurrent Neural Networks* Unlike feed forward neural networks, the Recurrent Neural Networks (RNNs) [103] are variants of recursive ANNs that allow for a bi-directional flow of data, i.e., in which connections between neurons make a directed cycle. In such architecture, neurons are fed information not only from the previous layer but also from themselves from the previous pass. Accordingly, the order in which the input is fed and the network is trained matters. For instance, in the context of texts, a word can be analyzed only in context of previous words or sentences. This RNNs memory characteristic lets users solve natural language processing problems such as handwriting recognition and speech recognition.

*Long Short-term Memory* Long Short-Term Memory (LSTM) networks [149] are specific RNNs that introduce a memory cell, which is a special cell capable of processing data when data have time gaps. These cells comprise components named "gates" which are of three main types; namely input, output and forget gates. The key function of these recurrent gates is to safeguard and control the information by stopping or passing the flow forward, erase memory, etc. The input gate determines the information, received from the previous layer, that will be kept in memory (gets stored in the cell), the output gate regulates the amount of data passed to the next layer, and the forget gate controls the tearing rate of the stored memory. Each of these gates has its own weight and sometimes activation functions. This makes them requiring more resources to run. LSTM networks are also widely used for writing and speech recognition.

Many other architectures exist for the LSTM networks such as the Gated Recurrent Units networks (GRU) [62]. GRUs are slightly different from LSTMs as they have different gates. GRUs lack the output gate, and have an update gate and a reset gate. The update gate decides both how much information to keep from the last state and how much information to pass forward to the next layer. The reset gate functions similarly to LSTM forget gate. Generally, in comparison to LSTMs, GRUs are slightly faster as they are less resource consuming, and are easier to run.

Further ANNs have been proposed in literature, and a mostly complete chart of these topologies accompanied by their illustrations was given by *Fjodor van Veens*[2]. It is notable that ANNs have this remarkable and unique ability to derive meaning from complex and imprecise data. ANNs can extract patterns and detect trends that are too complicated and excessively difficult to be perceived by humans or other computer techniques. They are, also, adaptive learning techniques capable of learning how to perform tasks based on the input training data, self-organized as they can create their own representation of the information received during learning time, and fault tolerant via the redundant information coding characteristic.

[2] `https://www.asimovinstitute.org/author/fjodorvanveen/`

The surge of interest in artificial neural networks has created an impact across a remarkably broad range of disciplines and applications such as in classification, prediction, decision-making, visualization, and many others [326].

## 2.3 Evolutionary Computing

Evolutionary computation includes a set of metaheuristics that are inspired by biological evolution. These metaheuristics seek to emulate the mechanisms and the main principles inferred from Darwin's theory of natural selection and population [79]; aiming to solve complex optimization problems. In what follows, we will describe the main principles of the natural evolution and then discuss the most popular evolutionary computing metaheuristics.

### *2.3.1 Natural Evolution*

From the beginning, biological life has been evolving every year; where unicellular life organisms get progressively mutated to complex life form organisms. This gradual change is evolved by the process of genetic evolution. Evolution refers to the change over time that occurs in populations of species (organisms) in reaction to environmental changes. The changes which are coded in the molecules of species' Deoxyribonucleic Acid (DNA) are transferred from one generation to another, and over the history of primitive life have resulted in gradually more complex life forms. These changes cause organisms to adapt and familiarize to their environment, and turn out to be extinct. In this concern, Darwin's theory of evolution [79] asserts that species survive through a process named "natural selection". The theory states that individuals in a species show a varied range of variation caused by the differences in their genes. Those individuals that successfully adapt, or evolve, to meet the changing requirements of their natural habitat are more likely to survive and reproduce. The genes that allow these individuals to be successful are passed to their offspring. On the other hand, individuals that fail to evolve and reproduce die off. Consequently, their genes are less likely to be passed to the next generation [130]. Given enough time, a species will gradually evolve.

### *2.3.2 Evolutionary Computing Metaheuristics*

Inspired by biological evolution, several potentials arise to design advanced evolutionary algorithms, i.e., metaheuristics that are able to efficiently explore complex search spaces in order to solve complex optimization problems. These mathematical models operate on a population involving individuals or chromosomes. Each individual represents a potential solution to the problem being solved and is codified according to the problem's specific requirements. The goodness of an individual that reflects how apt that individual is to survive in the environment is represented by an objective function which is a quantitative measure of effectiveness of the system, and the constraints to the problem which describe the relationships between the system's variables and define the allowable values to be taken by the variables [96]. Those individuals having lower fitness value are gradually eliminated by the dominant competitors. Within a population and for each individual, re-production

operators, typically chromosomal crossover and mutation, are applied over some individuals (parents) to produce new features in the chromosome. These new features represent new individuals (offspring) that maintain some properties of their ancestors which are conserved or are eliminated via a selection. This evolution process repeats itself during a certain number of cycles or generations; where species continuously strive to reach a specific genetic structure of the chromosomes that maximizes their probability of survival in a given environment. Specifically, this process continues until an acceptable result is achieved, i.e., the maximal fitness solution is found. An illustration of a generation in evolutionary algorithms is given in Figure 2. As presented in Figure 2, the key components of any evolutionary algorithm are the selection, the reproduction which includes the crossover and the mutation, and the replacement. Each of these components can be realized using different operators. For instance, selection can be performed using the roulette wheel, the tournament or the ranking approach, crossover can be achieved via single/multi-point or a uniform method, and mutation can be performed using bit-flipping, uniform or the shrink approach. A full review of the operators can be found in [84,360].

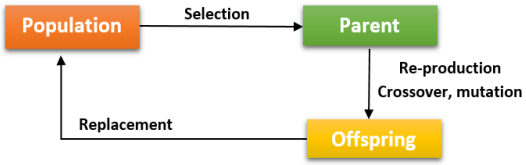


**Fig. 2** A generation in evolutionary algorithms: Within a population and for each individual, re-production operators are applied over some individuals (parents) to produce new individuals (offspring) that maintain some properties of their ancestors which are conserved or are eliminated via a selection .

In the literature, a wide range of different evolutionary optimization algorithms have been proposed [342,85] and applied to various domains [80,64]. All of these algorithms are similar in their basic approach and in making use of the bio-inspired concepts, but they essentially differ in the way they represent the information. In what follows, we will give an overview of the most popular metaheuristics.

*Genetic Algorithm* Genetic Algorithms (GAs) have been firstly proposed in [152] to understand the adaptive processes of natural systems. After that, they have been applied to solve optimization and machine learning problems [151] [83]. The classical versions of GAs use a binary representation where the chromosome representation is based on a binary string of fixed length. However, the later implementations of GAs make use of several other types of representations such as integer or real-valued representations, order-based representations or chromosomes of variables length and many more [55]. For the selection process, the algorithm originally uses the roulette wheel operator. A replacement selection is also performed, i.e., the selection of survivors of the parent and the offspring populations. Basically, there are two kinds of replacement strategies: *generational replacement* and *steady state replacement*. In generational replacement, the parents are replaced systematically by the offspring. In the steady state replacement, only a small fraction of

parents is replaced by the offspring during each iteration. Concerning the reproduction process, it is traditionally made via the crossover and the mutation operators with a fixed probability for each of them. Nevertheless, the algorithm emphases more the importance of the crossover operator over mutation. The crossover operator is based on the single/multi-point or uniform crossover while the mutation is generally bit-flipping.

*Evolutionary Strategy* Unlike GAs which are mostly applied to discrete optimization problems, Evolution Strategies (ESs) [300] are mostly applied to continuous optimization where representations are based on real-valued vectors. In ES, the representation of an individual is made by its genetic material and by a so-called strategy parameter which determines the behavior of the individual or define the individual in more detail. The genetic material is represented by floating-point variables while the strategy parameter is, generally, defined by the standard deviation of a Gaussian distribution associated with each individual. In many ESs, the selection operator is mostly deterministic and is based on the fitness ranking. Two types of mutation operators are commonly used namely the discrete mutation, in which the gene value of the offspring is the gene value from the parent, and intermediate mutation in which the midpoint between the gene value of the parents gives the gene value of the offspring. The mutation operator has a special implementation in ES as it mutates both the strategy parameter and the genetic material. Therefore, the evolution process evolves the genetic material and the strategy parameter at the same time; and accordingly, ES is considered to be a "self-adaptive" mechanism [242]. The main ESs advantage is their efficiency in terms of time complexity.

*Evolutionary Programming* Evolutionary Programming (EP) [115] is a stochastic optimization strategy similar to GA. EP emphasizes the behavioral linkage between parents and their offspring, rather than seeking to emulate specific genetic operators as observed in nature. Its working is similar to the working of ESs. It uses Gaussian distributed mutations and the self-adaptation paradigm. In EP, the parent selection mechanism is deterministic, and the replacement process is probabilistic and is based on a stochastic tournament selection [100]. Unlike ES, it has no restriction regarding the use of data types of attributes. EP has fixed structure of program and it allows numerical parameters to evolve. Like both the ESs and GAs, EP is a useful method of optimization when other techniques such as gradient descent or direct analytical discovery are not possible. Specifically, combinatory and real-valued function optimization – in which the optimization surface or fitness landscape is "rugged" – having several local optimal solutions are well suited for EP. However, it is important to note that EP has rather slow convergence rates, on some function optimization problems.

*Genetic Programming* Genetic Programming (GP) [187] is considered to be a GA extension, in which the structures in the population are not fixed length strings that encode candidate solutions to a problem but programs expressed as syntax trees, i.e., nonlinear representation based on trees. In GP, generally, the parent selection is a fitness proportional and the replacement selection is a generational one. The crossover operator exchanges parts of two parent trees resulting in two new trees and the mutation randomly changes a function of the tree into another

function, or a terminal into another terminal. One of the main problems in GP is the uncontrolled growth of trees which is a phenomenon called "bloat". The cause of bloat, as stated by the Crossover-Bias Theory, is that the distribution of program sizes during evolution is skewed in a way that encourages bloat to appear, by punishing small individuals and favoring larger ones. Indeed, GPs need a huge population and then they are very computationally intensive.

### 2.4 Swarm Intelligence

Swarming behavior is common in biology. An inspiration from the collective behavior of social swarms in nature such as flocks of birds, honey bees, schools of fish, and ant colonies led to the development of an innovative intelligent paradigm called "swarm intelligence" [38]. Swarm intelligence is based on the common comportment of these species that compete for food. The main features of swarm intelligence based algorithms are their simplicity and their particle aspect. They are based on agents, i.e., insects or swarm individuals, which are relatively unsophisticated and which cooperate, by doing movements in the decision space, to achieve tasks necessary for their survival. Among the most successful swarm intelligence based algorithms are *Ant Colony*, *Particle Swarm Optimization*, and *Bee Colony Optimization*. These will be detailed in what follows.

*Ant Colony Optimization* Ant Colony Optimization (ACO) algorithms are based on the idea of imitating the foraging behavior of real ants [93]. The inspiration comes from the collective behavior of ants to perform complex tasks such as transportation of food and finding shortest paths to the food sources. In nature, ants communicate by means of chemical trails; called "pheromone". This substance assists ants in finding the shortest paths between their nest and food. In a natural observation, ants usually wander randomly. When they find food, they return to their nest while laying down pheromone trails on the ground. This chemical, if found by other ants, will not keep them wander at random, but will help them to follow the trail and to quickly return to their nest, i.e., this trail will guide the other ants toward the target point. Meanwhile, these ants will reinforce the path if they find food. However, ants have to travel the path back and forth and as the pheromone is olfactive and volatile, it has to evaporate. Hence, the path becomes less prominent. In such situation, ants will look for the path having a higher density of pheromone. This means that this particular path was visited by more ants and is definitely the shortest path to take. In other words, the larger the amount of pheromone on a particular path, the larger the probability that the ants will select that specific path. Hence, there is an emergence of the shortest path. Based on this inspiration, ant colony optimization algorithms can be seen as multi-agent systems in which each single agent is inspired by the behavior of a real ant. In literature, there are numerous successful implementations of the ACO metaheuristic. A review of their applications to a wide range of different optimization problems can be found in [94].

*Particle Swarm Optimization* Another successful swarm intelligence model is particle swarm optimization [63]. It draws inspiration from the sociological behavior of natural organisms such as bird flocking and fish schooling to find a place with

sufficient food. Within these swarms' populations, a synchronized behavior using local movements emerges without any dominant control. Each individual within its community (population) is moved to a good area based on its fitness for the environment, i.e., its flexible velocity (position change) in the search space. Indeed, based on the particle's memory, the best position the individual has ever visited in the search space is remembered. Following this natural observation, the movements of swarms is seen as an aggregated acceleration towards their best previously visited position and towards the best particle of a topological neighborhood, i.e., the social influence between the particles. This phenomenon led to several efficient particle swarm optimization algorithms which are mainly applied to solve optimization problems [63].

*Bees Algorithm* The behavior of bee colonies exhibits various features that can be used as models for intelligent and collective behavior. Among these features, we mention nectar search, mating during flight, food foraging, and the waggle dance. More precisely, in nature, a bee colony holds a single reproductive female called the "queen", males known as "drones", and many sterile females called the "workers". After mating with some drones, the queen breeds numerous young bees called "broods". Based on the behavior of these bees, three of their main activities have inspired researchers to develop bee colony based optimization algorithms [172]. These activities are essentially the search for the nest site, the food foraging and the marriage behavior.

To search and select the appropriate nest site, bees go through a pre-organized process. In a colony, some scout bees leave to explore few nest sites while the remaining bees stay quiescent in the colony; possibly, to conserve the swarms' energy supply until decision-making, and then migrate to the selected nest site. Once the foraging bees come back from their search, they indicate various nest sites by following different dances which have eight patterns called waggle dances [317]. The quality of the nest site, e.g., entrance area, entrance height, cavity volume, is related to the speed and the orientation of the dance. To select the new nest site among the possible proposed options, scouts vote by dancing for their favorite site and then they make a group decision via a quorum sensing and not via consensus sensing. Afterwards, scouts inform their nest mates by waggle dances. At the end, the entire bee colony migrates toward the new nest site [317]. Little research has been conducted in proposing optimization models based on the nest site searching. One basic study which established this principle was proposed in [298] to deal with making decision strategy of the bees over the new nest site selection and to solve the resource allocation problems as a numerical optimization.

Similar to nest site search, for food foraging, some scout bees navigate and explore the region with the aim to find a food source. When they discover an appropriate source, they come back to the hive and enter a place called the dance floor to transmit and share their discovery with the others through the waggle dance related to the discovery distance. When it happens, some bees are recruited and become foragers. The number of the recruited bees is proportional to the food quantity information communicated by the scouts. This step is called the exploration phase which is followed by another phase called the exploitation. In the later step, bees collect food and determine their quantity to make a new decision; either they continue collecting food by the memorization of this best location, or they leave the source and return to the hive as simple bees. Most research has

studied the food source searching behavior and based on this several bee colony optimisation algorithms have been proposed to mainly solve optimization problems, whether combinatorial or numerical. The configuration of such algorithms is based on mapping the food source as a possible solution to the problem being solved and the quality of the food source as the fitness function.

For instance, to solve the travelling salesman problem, the algorithm proceeds as follows: First, a group of bees (a population) is created that contains active, inactive and scout bees. This constitutes the initialization phase. After that, the foraging process is initiated. During this process, bees explore and exploit the search space, try to collect as much nectar as possible, to find the optimal solution for the problem. Bees explore the search space in a random way during the first bee cycle since they do not have any waggle dance to follow upon. In the next step, cities data are given as an input parameter together with the number of cities that are to be visited by the salesman. Next, a population of bees is created each of which has a random solution. As bees consider that the shorter the link, the higher the nectar quantity collected along that link when flying, the solution having least value of distance is taken to be the best solution. The bee optimisation is set to find the best solution. Then, the three different categories of bees are made to do their part of work. Both of the active and the scout bees observe the waggle dance. The active bee first gets a neighbour solution tied to its current solution stored in its memory, and then determines the quality of that neighbour. If the current bee finds a better neighbour solution, then the algorithm determines whether the bee has made a mistake and, hence, rejects the better neighbour or if the bee has to accept the better found neighbour solution. Likewise, in case where the current bee did not find a better neighbour solution, then the algorithm determines if the bee has made a mistake and, hence, has to accept the worse neighbour solution or if the bee did not make a mistake and, hence, has to reject the neighbour. In the hypothesis where the bee has exhausted a specific food source without finding the better neighbour solution then the active bee is transformed to an inactive bee. In this case, a scout bee generates a random solution, checks if the random solution is better than the current solution in memory, and, if it is the case, copies the random solution into memory. In case where the scout bee has found a better solution, then the algorithm determines if the new solution is a global best solution. An active bee or a scout bee returns to the hive and performs a waggle dance to inactive the bees in order to convey information about the location and quality of a food source. The termination criterion is the number of iterations when completed and the optimal result for the travelling salesman problem is given by the obtained result [250].

In literature, there are several other applications of the bees' colony optimization algorithms that are based on this specific bees' behavior; applications to solve stochastic vehicle routing problems for instance or the routing in wired computer networks. A full review of the applications can be found in [35].

Another inspiring behavior of bees is the marriage behavior where, to reproduce, the queen has to mate several drones. The reproduction phenomenon is carried out far from the hive and happens in the air. More precisely, the queen performs a special dance engaging other drones to follow her and mate with her. They will mate until the queen's spermatheca will be full. After that, the queen lays her eggs. The unfertilized eggs will generate drones, while the fertilized eggs will give birth to workers or queens depending on food quality given to larvae.

There are several implementations of bees colony optimization algorithms which are based on the marriage phenomenon. Among these, we mention the basic study that was proposed in [29]. MBO is based on the mating flight that can be visualized as a set of transitions in a state space, i.e., the environment, where the queen moves between the different states in the space in some speed and mates with the drones encountered at each state probabilistically. Several extensions of MBO have been introduced where all of these try to improve some MBO algorithmic aspects to improve the sough results, e.g by using more than one worker or more than one queen. In literature, there are several applications of MBO and its extensions where the algorithms have been applied to data mining clustering problems and to the water resources management problems. A full review of MBOs applications can be found in [35].

Another set of metaheuristics has been proposed lately in literature, which were inspired by different aspects from nature. Among the most popular algorithms dedicated to solve NP-hard optimization problems, we mention the Firefly Algorithm [352] which is a metaheuristic inspired by the flashing behaviour of fireflies to carry out nonlinear design optimization. Basically, the algorithm mimics how fireflies interact with each other using their flashing lights. The algorithm assumes that all fireflies are unisex, which means that any firefly can be attracted by any other firefly; the attractiveness of a firefly is directly proportional to its brightness which depends on the objective function. In other words, a firefly will be attracted to another brighter firefly. The brightness decreases with distance. From an algorithmic perspective, a randomly generated feasible solution, called fireflies, will be assigned with a light intensity based on their performance in the objective function. This intensity will be used to compute the brightness of the firefly, which is directly proportional to its light intensity. For minimization problems, a solution with smallest functional value will be assigned with highest light intensity. Once the intensity or brightness of the solutions is assigned, each firefly will follow fireflies with better light intensity. For the brightest firefly, it will perform a local search by randomly moving in its neighbourhood. These updates of the location of fireflies continue with iteration until a termination criterion is met. The termination criterion can be maximum number of iterations, a tolerance from the optimum value if it is known or no improvement is achieved in consecutive iterations [178]. It was shown that the Firefly Algorithm is potentially more powerful than other existing algorithms such as particle swarm optimization [352] but at the same time it was criticized as differing from the well-established PSO only in a negligible way [219]. Another nature-inspired recent algorithm is the Bat Algorithm [379] which is inspired by the echolocation behaviour of microbats, the Cuckoo Search optimization algorithm [119] which is inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species), the krill herd biologically-inspired algorithm which is based on the simulation of the herding behavior of krill individuals for solving optimization tasks [362], and the Shuffled Frog-Leaping Algorithm [104] for solving combinatorial optimization problems and which is based on observing, imitating, and modelling the behavior of a group of frogs when searching for the location that has the maximum amount of available food.

Based on the descriptions of all these bio-inspired algorithms from *Artificial Immune Systems*, *Connectionist Systems*, *Evolutionary Computing*, and *Swarm Intelligence*, we can clearly see that nature has been a great source of inspiration

for several researchers in many different ways. These computational algorithms have an important impact in solving a variety of real-world problems, as presented in Table 1, and among these issues are the challenges faced in the biological domain. Further discussions about this will be given in the remaining sections.

| Biological system | Bio-inspired algorithms | Main computational applications |
|---|---|---|
| **Immune System** | **Artificial Immune System** | |
| ***Key features:*** cloning, selection, memory cells, affinity maturation. | Clonal selection algorithm: a detailed description and applications of various clonal selection algorithms can be found in [356]. | Modeling and optimization, dimensionality reduction, pattern recognition. |
| ***Key features:*** autoregulation, cells interaction, cells and antigens recognition, cells stimulation, activation and suppression, network stabilization. | Immune network algorithm: a detailed description of numerous immune network algorithms can be found in [139]. | Data analysis, clustering, data visualization, modeling and optimization. |
| ***Key features:*** self-non-self discrimination, negative selection, positive selection. | Negative/Positive selection algorithms: a review of the progress of negative selection algorithms can be found in [164]. | Computer security, network intrusion detection, classification, optimization. |
| ***Key features:*** discrimination, environmental context, signals, danger, dendritic cells. | The dendritic cell algorithm: a full review of the DCA and its applications can be found in [60]. | Computer security, anomaly detection, classification. |
| **Connectionist System** | **Artificial Neural Network** | |
| ***Key features:*** information processing, interlinked neurons, memory, learning. | Feed forward, auto-encoders, convolutional neural network, etc. Further ANNs are highlighted in Section 2.2. | Image recognition, handwriting recognition, speech recognition, network design, classification, prediction, decision-making, data visualization. |
| **Biological Evolution** | **Evolutionary Computing** | |
| ***Key features:*** environmental changes, adaptation, natural selection, reproduction. | Genetic algorithms, evolutionary strategy, evolutionary programming, etc. Further evolutionary algorithms are highlighted in Section 2.3 and a wider range of other evolutionary optimization algorithms can be found in [342, 85]. | Safety systems, power extraction, flow shop problems, flow-shop scheduling problem, biometric Systems. |
| **Social Swarms** | **Swarm Intelligence** | |
| ***Key features:*** cooperative behavior, foraging behavior, marriage behavior, nest search. | Ant colony optimization algorithms, particle swarm optimization algorithms, etc. A review of swarm based algorithms as well as their application domains can be find in [94,35,188]. | Search optimization, network design, graph matching, resource allocation, decision-making, data mining. |

Table 1: Main biological systems that have inspired computational algorithms and their related application domains.

## 3 Computational Biology

Computational Biology is the study of biology using computational, statistical, and mathematical methods. This field significantly relies on analyzing biological

data. One of the main sources of biological data is molecular data that comes from sequencing three major *macromolecules*, namely DeoxyriboNucleic Acid (DNA), RiboNucleic Acid (RNA), and protein. *Sequencing* is a process of determining the sequence of nucleotides in the case of DNA and RNA and the sequence of amino acids in the case of protein. During the last decades, for each main macromolecule, a number of the sequencing technologies have been developed. The progress in such technologies facilitate the availability of biological data leading to easier and comprehensive analyses of molecular data.

The Human Genome Project (HGP) [72] – an international research effort to determine the sequence of the human genome – was one of the great feats of exploration in history. Beginning on October 1, 1990 and completed in April 2003, the HGP gave us the ability, for the first time, to understand the blueprint for building a person. This international effort to sequence the 3 billion DNA letters in the human genome had a major impact in the fields of medicine, biotechnology, and the life sciences, as the human genome holds an extraordinary trove of information about human development, physiology, medicine and evolution. In addition to the human genome, the international network of researchers has produced a series of amazing advances [350]: an advanced draft of the mouse genome sequence, published in December 2002; an initial draft of the rat genome sequence, produced in November 2002; the identification of more than 3 million human genetic variations, called single nucleotide polymorphisms (SNPs); and the generation of full-length complementary DNAs (cDNAs) for more than 70 percent of known human and mouse genes. As other great projects like 1000 Genomes Project [67,68,69], International HapMap Project [70], and the Cancer Genome Atlas (TCGA) [349] unfolded, sparked by HGP, scientists entered into the era of "genomics" [124]. Subsequently, the need to manage and analyze copious amounts of digital genome data drove the growth of computational biology.

Advanced technologies for genome sequencing and assembly were fundamental to the success of the human genome project. The first sequencing technology that has revolutionized biological research was Sanger sequencing technology invented in 1977 (*first-generation* sequencing technology) for DNA and RNA [310]. Further innovations in reagents, biotechnologies, and computational methods allowed to accomplish the HGP [72]. The release of the first high-throughput DNA sequencing platforms, such as 454 sequencing by Roche in 2005 and Solexa 1G by Illumina in 2006, have remarkably increased the sequencing data produced per instrument and dramatically decreased its cost [181,240]. The developed DNA and RNA sequencing platforms with the number of related biotechnologies in the next five years are referred to as *second-generation* sequencing (sometimes called *next-generation* sequencing) technologies [231]. Since the second-generation technologies have enabled sequencing of many new genomes and performing the population-scale analysis of genomic diversity [69], they have facilitated an explosion in biological knowledge in the following decade. Although further significant progress in increasing the scale of data production and decreasing its cost, some technological limitations remain [212]. One of the key limitations is that the genomes sequenced by second-generation platforms are often of lower quality as compared to those sequenced using more expensive first-generation technologies. The appearance of single-molecule sequencing from Pacific Biosciences (PacBio) [303] in 2010 and nanopore-based sequencing from Oxford Nanopore Technologies [162] in 2014 addressed some of the existing limitations of second-generation sequencing

platforms. These two technologies together with the Chromium technology from 10X Genomics [390] and Hi-C technologies [297] are referred to as *third-generation* sequencing technologies [316,232,132].

Unlike DNA and RNA sequencing technologies, there are much fewer of them available for protein sequencing. The first automated technology for protein sequencing (*protein sequenator*) based on the *Edman degradation reaction* was invented in 1967 [98]. The release of mass spectrometry (MS) techniques [174,173, 92] transformed proteomics studies. Further improvements in the sensitivity and resolution of mass spectrometry instrumentation and the associated advances in technologies for sample preparation have created the most comprehensive and versatile technology in large-scale proteomics that is available in our days. Two main techniques in the modern MS technologies are matrix-assisted laser desorption ionization (MALDI) [175] and electrospray ionization (ESI) [112]. Using nanopore-based technology for proteins may lead to creating new techniques for protein sequencing [353,304,376].

The aforementioned modern sequencing technologies generate a tremendous amount of biological data from living species. As a consequence, the number of biological questions that can be addressed with mathematical, statistical, and computational methods has significantly increased. Computation biology has been emerged as a huge interdisciplinary field that develops and applies computational methods to analyze large collections of biological data to make new predictions, and discover new biology. It includes, but not limited to, various analytical methods, mathematical modeling and simulation for a number of broad areas, including analysis of protein and nucleic acid sequence, structure and function, evolutionary genomics and proteomics, systems biology, population genomics, regulatory and metabolic networks, biomedical image analysis and modeling, gene-disease associations, and development and spread of disease. Since it is not feasible to give an overview of all of these areas, we focus on a few of the major research areas in computational biology, and discuss how the integration of mathematical modeling, computer science and statistics with biology contributes towards significant advances in answering various biological questions, and in general the advancement of computational biology. We begin with a discussion on genome assembly which is one of the most important research areas in computational biology. Since existing sequencing instruments typically cannot determine the entire sequence of a genome, it requires computational methods to assemble the whole genome. We present various state-of-the-art techniques for genome assembly in Section 3.1. Another important field of research in computational biology is comparative genomics and evolution which we will discuss in Section 3.2, followed by a brief discussion of genome rearrangement. Finally, in Section 3.3, we give an overview of Genome-Wide Association Studies (GWAS) due to its immense applications and impact in identifying genetic variations responsible of various traits (phenotypes).

### 3.1 Genome Assembly

Despite tremendous advances in sequencing technologies described above, modern instruments can read only small segments of genomes [232]. A single continuous segment of a DNA sequence produced by a sequencing instrument is called a *read* [316]. The length of a DNA sequence or segment is often measured as the

number of complementary nucleotide pairs (base pairs, bp). The lengths of the reads produced by existing sequencing platforms range from approximately $100 - 200$ bp (e.g., Illumina technology) to approximately $10 \times 10^3 - 20 \times 10^3$ bp (e.g., Pacific Biosciences and Oxford Nanopore technologies) [232]. By contrast, the human genome comprises more than $3 \times 10^9$ bp ($\approx 3$ Gb) [72]. Moreover, even some bacteria genomes have a length more than $1 \times 10^6$ bp ($\approx 1$ Mb). Reconstructing an entire genome thus requires merging together many reads into longer segments. The simple approach that does not require time-consuming and labor-intensive procedures, is to develop a computer program to do the merging. That leads to the development of the scientific field called *genome sequence assembly* [236]. There are two main directions of genome sequence assembly depending on the availability of sequenced genomes of closely related organisms [322]. The first one is *de novo* genome assembly, where only the reads from the sequenced organism are used to assemble a whole genome. The second one is *reference-guided genome assembly* (sometimes called *assisted assembly*), where the genome of a related organism is used to aid the genome assembly of the organism being sequenced. In both cases, as we will see in subsequent sections, applications of computational techniques, including various algorithmic techniques, mathematical and graph modeling have made notable progress towards assembling genome sequences.

### *3.1.1 De Novo Genome Assembly*

The "simplicity" of the biological problem formulation of de novo genome assembly captured the interest of mathematicians and computer scientists and led to the development of the theoretical foundations for genome sequence assembly [195]. One of the earlier attempts to formalize the genome assembly problem was solving the problem of finding the shortest super-string that encompasses all the reads as substrings [228]. The main assumption of such a formalization is that the reconstructed genome is the parsimonious explanation of the set of reads [347,299]. However, such an assumption ignores the observation about the presence of identical or nearly identical segments of a DNA sequence, called *repeats*. Since most genomes of living species have a complex structure which is comprised of nonrandom repeating elements, the corresponding DNA sequences stray from parsimony. In other words, the correct assembly of a genome sequence may not be the shortest super-string [177,134,237,371].

The requirement to account for repeats has led to new graph-based assembly models of genome sequence assembly [177,237,253]. Graph-based assembly models represent sequence of reads and their inferred relationship to one another as vertices and edges in the graph, respectively. In addition to such representations, by contrary to the requirement to reconstruct a single string, the formalization of genome assembly problem for graph-based models was shifted to strings reconstruction that are substrings of a DNA sequence, called *contigs* (also known as *unitigs* or *omnitigs*) [236]. Such graph-based models play a key role in virtually all modern genome assemblers in our days [329]. Moreover, these graph-based models allow to "measure" the "practical" complexity of genome assemblers that depends on the ratio between read lengths and repeat lengths in the genome [256]. In other words, longer reads imply better and more complete solution of genome assembly problem. However, the practical success of genome assemblers even for Illimuna reads has been observed over the past few decades [329]. In the rest of this section,

we will overview two main graph-based assembly models. More detailed surveys of De Novo genome assemblers, De Novo assembly algorithms, and theoretical results can be found in [322,329,101,244,257,313,361].

*Overlap-layout Consensus (OLC) Graphs* In the OLC graphs, a read is a vertex, and a pair of vertices are linked with an edge if they overlap [254]. Alternative formulations have a pair of vertices for each read, one representing the start of the read and one representing the end of the read, with an edge linking these vertices that carries the read sequence. Read overlaps are represented by edges from the terminal vertex of one read to the terminal vertex of another read. Regardless of the representation of the OLC graph, the assembly process consists of four main steps: detect overlapping pairs of reads, construct the OLC graph from the obtained pairs, find the appropriate ordering and orientation of reads, and compute the consensus sequence from the resulting ordering and orientation [321,253,322]. The overlap computation step is a particular bottleneck in this approach because naive methods are scaled quadratically regarding the total number of sequenced bases. Techniques that allow drastically reduce computational time for the overlap computation step, have been proposed in [255,267,32,208].

From the given OLC graphs, it is easy to obtain the *string graph* as follows [254]. First, reads that are substrings of some other read are removed from the OLC graph. Next, transitive edges in the OLC graph are removed from it. String graphs have several advantages in comparison to OLC graphs. Surprisingly, for construction of string graphs, there exist several fast and memory-efficient algorithms [321, 320,203,30,89,126]. Moreover, the string graph shares many properties with the de Bruijn graph without the need to break the reads into $k$-mers (see below for a discussion on de Brujn graphs) [254].

*De Bruijn Graphs* For constructing De Bruijn graphs, each read is broken into a sequence of overlapping *k-mers* (a k-mer is just a string of of length $k$) [159, 285,287]. Then the distinct $k$-mers are assigned as vertices and two vertices are connected by an edge if the corresponding $k$-mers originate from adjacent positions in a read in the De Bruijn graph. The assembly problem corresponding to De Bruijn graphs can be then formulated as finding a walk through the graph that visits each edge in the graph once (also known as a Eulerian path problem) [287, 286,385]. Due to the presence of repeats, there is a potentially exponential number of Eulerian traversals of the graph, only one of which is correct. In most practical instances, assemblers attempt to construct contigs consisting of the unambiguous, unbranching regions (i.e., contigs) of the De Bruijn graph [385,19,51,54,74,221, 238,323].

### *3.1.2 Reference-Guided Genome Assembly*

When a related genome is available, one may use this genome to guide the assembly of the target genome. However, in nature, two identical genomes do not exist even among individuals of one species. Such genomic variation is caused by *point mutations* and *genome rearrangements* (see Section 3.2 for detailed discussions about the reasons for genomic diversity). The necessity to account for genomic differences poses challenges in developing approaches to assisted assembly.

Reference-assisted assembly tools aligned reads against the reference and ordered them according to their positions in the reference genome [343,289,21,147, 328]. While this approach is still commonly used, it addresses genomic differences caused by point mutations only. As a consequence, this approach introduces errors when genome rearrangements between the reference and the assembled (target) genome occur. In an attempt to address this problem, *the ordering problem* was formulated [120], which asks for the order of reads so that the *Double-Cut-and-Join distance* (see Section 3.2) between the resulting target genome and the reference genome is minimized. This formulation has been further applied in some reference-guided assembly methods [301,302]. Despite advances made by these tools, these methods still generate erroneous contigs when there are rearrangements between the target and reference genomes.

Thanks to increasing availability of good quality assembled genomes, an important step toward a reliable reconstruction of the target genome has been made [179]. In contrast to previous methods, which use only one reference, the new approach utilizes multiple reference genomes to guide the assembly. This approach proved to be valuable since the order information in the reference genomes can also help infer more accurate order information in the target assembly [179,183,20,39].

The rapid progress in the sequencing technologies poses many novel problems in developing reference-guided assembly tools. For example, since the quality of target assembly is often improved when multiple references are used, this approach requires to develop new data structures that would allow storing and working with a large number of genomes [245,246,320,28,53]. Moreover, better models that capture the most possible different types of genome rearrangements are needed (see Section 3.2 for detailed discussion about models of genome rearrangements).

### *3.1.3 Chromosome Scaffolding*

*Scaffolds*, like contigs, represent the assembled portion of a chromosome [316]. They are formed by ordering and orienting contigs. Scaffolds are often much longer than contigs, in some cases they include entire chromosomes. Unlike contigs, a sequence of scaffolds may contain gap letters (often represented as Ns) where regions of potentially unknown size remain unassembled [257]. The process of ordering and orienting contigs into scaffolds is called *scaffolding*. Filling the gaps in scaffolds is often referred to as *gap filling*. As with genome assembly, repeat regions represent the most challenging regions to resolve.

Due to the similar nature between scaffolds and contigs, all the approaches that have been developed for scaffolding can be divided into two main methodologies. One of the methodologies is to use one or more reference genomes for ordering and orienting contigs [252,3,58]. The methods and ideas in this direction are extremely similar to those that are used in the reference-guided genome assembly [207].

The other type of methodology is scaffolding without reference genomes [23, 329,257]. In this approach, additional supporting data from the assembled genome are needed [158]. Typical examples of such data are *optical maps* (produced by BioNanoGenomics technology) [258,345,225,239], *linked reads* (produced by 10X Chromium Genomics technology) [373,190,380], *long reads* (produced by PacBio

and Oxford Nanopore technologies) [367,19,23,37,193], *jumping libraries*[3] (produced by Illumina technology) [158,323,184,136,221,81,36], *higher order chromatin interactions* (produced by Hi-C technology) [48], or other long-range sequencing information. Scaffolding algorithms that utilize one or more types of data commonly use either "greedy" approaches that iteratively join together contigs with the strongest linking support, or a "global optimization" that tries to best satisfy all of the linked information at once. Among all methods, the combination of long reads with Hi-C based data has lead to high-quality chromosome-length scaffolds [123,33]. By accurate usage of the linked read data, the assembler (that includes the scaffolder method as a step) from the 10X Chromium Genomics technology called Supernova has provided scaffolds with lengths among the best available for any human genome [373].

### *3.1.4 Error-Correction*

In order to overview the genome assembly, a strong simplifying assumption about the sequencing technologies was made [236] where the fact that reads may have errors was ignored. If a read has an error in it (e.g., one position is a C instead of an A), then it is no longer a substring of the genome. For example, reads produced by the Illumina sequencer machine have an error rate of $\approx 0.1\%$ [314,11]. Moreover, reads produced by Pacific Biosciences and Oxford Nanopore technologies have an error rate of $\approx 10-15\%$ [303] and $\approx 10-20\%$ [198,226], respectively. Therefore, the sequencing errors should be corrected for more accurate and contiguous De Novo assembly. The essential problem [329] of correcting sequencing errors is identifying sequencing errors and distinguishing them from the heterozygous alleles[4].

Most of the error correction tools implement $k$-mer counting methods to detect sequencing errors. In $k$-mer counting methods, *low-depth* $k$-mers (i.e., the $k$-mers that occur relatively less frequently in the sequencing reads) are assumed to be erroneous [329,314,11]. In contrast to this method, the multiple sequence alignment-based method detects sequencing errors by directly aligning reads with each other and they are corrected by consensus. Despite the multiple sequence alignment-based methods may be computationally expensive, it is commonly used for error correction of long reads [321,319]. Since the error rate for long reads is higher than for short ones, hybrid error correction approaches that use both short and long reads are proposed to correct sequencing errors in long reads [226,161,131,308, 138,201,243]. Recently, methods for Oxford Nanopore reads have been proposed that polish the resulting assemblies according to the raw signal from its sequencer machine [218,324]. These methods make it possible to assemble highly contiguous sequences with 99.9% accuracy or greater from Oxford Nanopore reads, even though the initial sequencing reads may have 20% sequencing errors or worse [218].

[3] A jumping library is a set of pairs mate-pair reads derived from long fragments of DNA. A mate-pair read is a pair of sequence reads from a single fragment of DNA (often the distance between the reads is approximately known).

[4] An allele is a variant form of a given gene. Individuals who are heterozygous for a certain gene carry two different alleles.

#### *3.1.5 Discussion*

Regardless of the sequencing technologies that are used, the basic strategy for assembly comprises three steps [316,329,322]: contig assembly, scaffolding, and gap filling. In the contig assembly step, the reads are assembled into contigs without gaps. Then, in the scaffolding step, the contigs are connected into scaffolds according to the information from various technologies. After that, gaps remain between the contigs in scaffolds if there is no overlap between the contigs and approximate distances are estimated using different technologies. The gaps are carefully filled by using other independent reads at the gap-filling step to complete the assembly. The scaffolding and gap-filling steps can be performed iteratively to enhance the quality of the resulted assembly until no contigs are scaffolded or no additional gaps are resolved. The error correction must be performed before, during, or after [325,218] assembly because the errors might prevent extension of contigs or scaffolds.

It should be noted that long read technologies have not made the assembly problem irrelevant [208,185,170,204]. Instead, they have simply shifted the focus from short repeats to longer repeats comparable in length to the median long read size. Further progress in such sequencing technologies and appropriate assembly algorithms may lead to significant improvements in genome assembly.

### 3.2 Comparative Genomics and Evolutionary Biology

The genomes of different organisms are widely different and at the same time amazingly similar. For example, the human genome comprises more than 3 billion bases [357], while a virus genome may range from only a few thousands to a couple of million base pairs (the HIV genome has a scant 10,000 bases, while *Pandoravirus salinus* genome has around 2.4 million base pairs) [288]. Despite the difference in lengths, various genomes may have similar genes. For example, many genes in Humans and Fruit flies (fruit fly has around 140 million bases [102]) are similar. Moreover, almost 99% of all human genes are conserved across all mammals [168].

One of the main processes happening in the cells that contributes to genetic difference across species and individuals in a particular species is the *replication* (copying) of the genome. Using layman's terms, copying can be described as follows: two complementary strands of a DNA sequence are separated, and then each of them spontaneously attracts "spare" bases that are present in the cell's environment. Since the nucleotide $A$ will successfully pair only with its complementary base pair $T$, and $G$ only with $C$, each strand of DNA can serve as a "template" and can specify the sequence of nucleotides in its complementary strand by DNA base-pairing [6]. A double-helical DNA molecule can be precisely replicated in this way.

A single strand of DNA or RNA whose nucleotide sequence acts as a guide for the synthesis of a complementary strand.

Due to the complexity of involving machinery in the replication process, it is also prone to errors. Such occurring errors in the replication process in general explain the diversity of living organisms. This inaccuracy of the replication process and natural selection is the main principle of molecular *evolution*. While the former accounts for the possibility of obtaining new or loosing existing functionality, the

latter ensures that only those mutations that are beneficial in terms of adaptation and survival are preserved throughout the generations.

A DNA sequence may evolve by means of *point mutation*, also known as *single nucleotide polymorphisms* (SNP) (i.e., a mutation at the level of nucleotides), and *rearrangements*, also known as *structural variations* (i.e., a mutation that modifies sequence organization at a larger scale). There are three different kinds of point mutations, namely:

- *Substitutions*: a nucleotide is replaced with another one,
- *Insertions*: a nucleotide is added to the DNA sequence,
- *Deletions*: a nucleotide is removed from the DNA sequence.

Due to a redundancy of the genetic code, point mutation events are often neutral in the sense of evolution. Therefore, SNPs are the most common events that one can detect based on genomes of living species. The main rearrangements that are being observed and studied today are the following:

- *Reversal* or *inversion*: a contiguous segment of a chromosome is reversed, and strands are exchanged respectively;
- *Deletion*: a contiguous segment of a chromosomes is deleted;
- *Insertion*: a new contiguous DNA segment is introduced into a chromosome;
- *Fission*: a single chromosome is split into two;
- *Fusion*: two chromosomes are merged together into a single larger chromosome;
- *Translocation*: a chromosome breaks and a portion of it reattaches to a different chromosome;
- *Transposition*: a contiguous segment of genome DNA is relocated into a location on the same or a different chromosome;
- *Tandem duplication* a contiguous DNA segment is copied and inserted next to the original fragment;
- *Whole genome duplication (WGD)*: each chromosome of a genome is simultaneously duplicated;
- *Retrotranspositional duplication*: a copied DNA fragment is inserted into an arbitrary position in the genome that is not directly adjacent to the original fragment;
- *Horizontal (or lateral) gene transfer*: a contiguous fragment of genome is copied from one genome to another.

These rearrangements operate on contiguous parts of the genomes, rather than on nucleotides. This is why, in studies of rearrangements, genomes are often represented as sequences of *conserved segments* (almost identical subsequences that are found in an almost identical state in several species and not cut by rearrangements). We will often refer to such conserved segments as genes, since genes play a vital role in storing critical "instructions" regarding the cell's lifecycle, and thus are rarely the place for a breakpoint to occur and destroy them[5]. Two genes are called *homologous* if they derive from a common ancestor and are distinguished by a speciation (i.e., found in two different species) or a duplication (i.e., found within

[5] Mechanisms of breakpoint origination remain a mystery, but the idea of genes playing the role of "solid" regions can be explained by the fact that their functionality, if lost, may result in the death of the cell, and thus a cell whose genomes have such "broken" genes would most likely not procreate.

the same genome) event. Unlike point mutation events, large-scale rearrangement events are very rare.

The emergence of many newly assembled genomes allows us to address various questions related to evolution. For example, comparative genomics is a field of biological research in which the genome sequences of different species are compared to understand how organisms are related to each other at the genetic level. The detecting of point mutation events is the goal of *sequence alignment* studies. Comparison of segments of genomes is mostly done by aligning "homologous sites" from different species, which is known as Multiple Sequence Alignment (MSA). Sequence alignments and the information that can be gained by comparing two genomes together is largely dependent on the evolutionary history and distance between them, which can be represented by phylogenetic trees. In this section, we will discuss the MSA phenomenon, various key components of phylogenomic analyses, genome rearrangements and how research in computational biology helped developing efficient tools for estimating multiple sequence alignments, phylogenetic trees, genomic rearrangements, etc.

### *3.2.1 Multiple Sequence Alignment*

Gene sequences evolve under processes that include events such as substitutions, insertions and deletions (jointly called "indels") that can change the length of the sequences, and must be accounted for comparative genomics. Indels have the effect that they blur what parts of sequences from various organisms are related to each other. Two characters in a sequence are called homologous if they are both derived from a common character in an ancestor, and this relationship is called homology. A Multiple Sequence Alignment (MSA) of a set of sequences is defined by the evolutionary history relating the sequences. MSA lines up individual nucleotides or amino acids from biomolecular sequences collected from organisms into a matrix structure such that every column in the true alignment indicates shared ancestry (homologies). That means, for a multiple sequence alignment of a set $S$ of sequences, the rows are the sequences in $S$ and the characters within each column are homologous. For evolutionary reconstruction, this refers to positional homology meaning that all the characters (nucleotides) in the same column have evolved from a common ancestor. However, for protein sequences, MSA can be defined as structural homology where the entries (residues) within each column produce identical structural features in protein folding. Figure 3 shows an example of two sequences evolved with substitutions, insertions and deletions and the corresponding multiple sequence alignment.

Multiple sequence alignment is one of the most useful tools in computational biology having immense applications in phylogenetic analyses, analyzing various attributes of protein (e.g., protein structure, function, folding, binding sites), protein family identification, orthology identification, etc. A number of MSA tools have been developed over the past few decades using the biological insights of sequence evolution and various aspects of computer science, mathematics and statistics. Computational techniques that have been applied to this area include dynamic programming, divide-and-conquer, probabilistic approaches (maximum likelihood, Markov chain Monte Carlo (MCMC), Bayesian statistics, etc.), and various discrete optimization search techniques. Application of these techniques have resulted in many efficient tools for computing multiple sequence alignments.

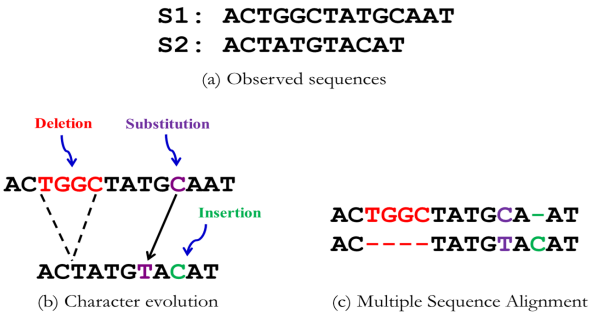


**Fig. 3 Character evolution and multiple sequence alignment.** (a) Two observed sequences, (b) Character evolution with substitution and indels which can change the sequence length and blur the homology, and (c) Multiple sequence alignment of the two sequences capturing the underlying character evolution where each site consists of homologous characters.

To name a few, PASTA [247], SATé [210], T-Coffee [269], MAFFT [176], Clustal W [351], Clustal $\Omega$ [318], MUSCLE [97], PRANK [220] Kalign [197], RetAlign [339] are being widely used.

### *3.2.2 Phylogenomics*

[6] A phylogeny is a representation of the evolutionary relationships of a set of entities, e.g., species, genes, languages, etc. Phylogenetic entities are commonly known as *taxa*. The simplest and the most useful representation of such evolutionary history is a "tree", which we call *phylogenetic tree*. A *tree* $T$ is a connected acyclic graph with a set of vertices $V$ and a set of edges $E$. A leaf in a phylogenetic tree represents a *taxon* that typically exists in the present day. The internal nodes represent the hypothetical ancestral taxa from which the descendant taxa evolved. The internal nodes typically represent extinct species that existed in the past, but do not exist anymore. An edge $e = (u, v) \in E$ represents an evolutionary relationship between the two taxa at the vertices $u$ and $v$. The length of the edges (branches) in an evolutionary tree is known as *branch length*. Branch length is a non-negative real number that represent various quantities measured on a branch. Most often, a branch length represent the amount of evolutionary change or the amount of time between two nodes. When trees are not provided with branch lengths, we generally refer to them as *topologies*. Figure 4 shows an example of a phylogenetic tree that illustrates the evolutionary history of humans, chimpanzees, gorillas and orangutans. It shows that humans are more closely related to chimpanzees than they are to gorillas and orangutans.

Phylogenetic trees provide insights into basic biology, including how life evolved, the mechanisms of evolution and how it modifies function and structure, orthology detection, medical diagnosis, drug design, criminal investigation, etc. In fact, there is a famous saying by Dobzhansky that "Nothing in biology makes sense except in the light of evolution" [90]. Phylogenetic analysis is mostly used for a comparative study [5,30], where a particular question is addressed by how certain biological

[6] Most of the figures and some of the materials presented in this section are taken from [24].

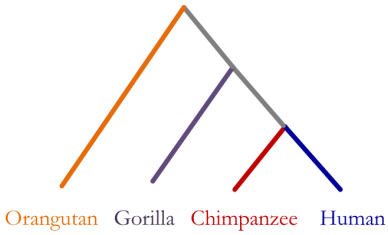


**Fig. 4 Phylogenetic tree.** A phylogenetic tree relating four species: humans, chimpanzees, gorillas and orangutans.

characters have evolved in different lineages of a phylogeny [209]. Phylogenetic analysis is also commonly used to test bio-geographic hypothesis [75,209], which is the study of the distribution of species, extant and extinct.

Phylogenetic analysis is useful in drug design. Phylogenies can be used to represent the evolution of diseases to identify the currently dominant strains so that appropriate drugs and vaccines can be designed for the currently dominant strains. For example, scientists try to keep track of the evolution of influenza types so that effective flu vaccines can be designed for the types that are most likely to dominate in a particular flu season [49]. Phylogenetic analyses have been featured in criminal investigations to assess DNA evidence presented in court cases. Most famously, phylogeny was used in a case where a doctor in Louisiana was accused of having deliberately infected his girlfriend with HIV [241,359,1,78]. The prosecution tried to trace the origin of the victim's HIV infection. However, since HIV evolves rapidly, each HIV-infected individual may have viruses distinct from others. But they could be genetically similar enough to determine the ancestry and therefore virus transmission history. The phylogenetic evidence featured prominently in the trial and the doctor was ultimately convicted of attempted second degree murder, and is now serving a 50-year prison sentence for the crime [359, 78].

Evidence from morphological, and gene sequence data suggests that all organisms on earth are genetically related, and the relationships of living things can be represented by a vast evolutionary tree – the "Tree of Life". The *Tree of Life* is one of the most ambitious goals and grand challenges of modern science [140]. Central to assembling this tree of life is the ability to efficiently analyze the vast amount of genomic data available these days due to the tremendous advancement in sequencing techniques, and computer hardware and software.

In phylogenomic analyses, data from many genes sampled throughout the whole genome are used for reconstructing the challenging parts of the tree of life. However, combining individual gene histories to infer a coherent species tree is a very challenging task [224]. We now discuss the key components of phylogenomic analyses: gene trees and species trees, how species tree construction is complicated due to the "gene tree-species tree discordance", and the standard computational approaches that are being used for estimating species trees from multiple genes, etc.

*Gene Tree and Species Tree* Most often, the goal of a phylogenetic reconstruction is to infer an evolutionary tree depicting the history of speciation events that lead

to a currently extant set of taxa. A *species tree* can be defined as the pattern of branching of species lineages via the process of speciation. A *gene tree* represents the evolution of a particular "gene" within a group of species. When species are split by speciation, the gene copies within species are also split into separate lineages of descent. Thus, gene trees are contained within species trees [224]. Interestingly, due to various biological processes, different genes (i.e., different parts of the whole genome) may have discordant evolutionary histories. Figure 5 shows an example of discordance between a species tree and a gene tree. Here, species $C$ and species $B$ are "sister" species in the species history, whereas $C$ is closer to $D$ than $B$ in the gene history. This is called the *gene tree incongruence/discordance*, and can arise from incomplete lineage sorting, gene duplication and loss, horizontal gene transfer, hybridization, etc. [224]. This disparity among the gene trees makes the species tree construction complicated. We now briefly describe various biological reasons for gene tree discordance.

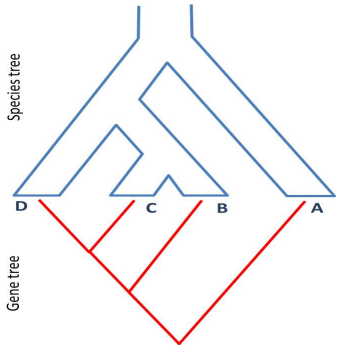


**Fig. 5 Gene tree-species tree incongruence.** A species tree (given in blue) and a gene tree (given in red) on the same set $\{A, B, C, D\}$ of taxa with different topologies.

*Gene Duplication and Loss* Gene duplication is the process of generating multiple gene lineages in coexisting in a species lineage [276]. A gene duplication event causes a second "locus", and these duplicated loci evolve independent of each other – resulting in incongruence between gene tree and the containing species tree [129]. Moreover, some of the gene lineages could go extinct if it decayed into a "pseudo-gene", or if it evolved a new function and diverged [224]. This phenomenon is knows as *gene loss* which may result in gene tree discordance. Figure 6 shows how gene duplication and loss can cause gene tree discordance. Alternatively, this figure shows how to explain the discordance between a gene tree and a species tree using gene duplication and loss events. Such embedding of a gene tree inside a species tree is called "reconciliation". Therefore, estimating species from a collection of gene trees when gene trees are discordant due to gene duplication and loss requires appropriate algorithms that take gene duplication and loss into consideration [224, 56,25,42,27].

*Incomplete Lineage Sorting* Incomplete Lineage Sorting (ILS), also known as deep coalescence, is best understood under the coalescent model [87,88,156,264,265,

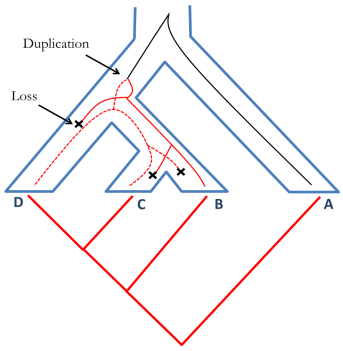


**Fig. 6 Reconciliation under gene duplication and loss model.** We show a reconciliation of the discordant gene tree-species tree pair shown in Fig. 5 using gene duplication and loss. We embed the reconciled tree inside the species boundaries. Duplicated gene copes (dashed and solid red lines) evolve independently and can go extinct. This reconciliation requires one duplication and three gene losses.

305,340,341]. The *coalescent model* describes the evolutionary process as if it operates backwards in time, and connects gene lineages to a common ancestor through a process of "coalescence" of lineage pairs. This model treats a species as a population of individuals, having a pair of alleles for each gene. The coalescent process traces the present day variants of a gene (known as *alleles*) back in time across successive generations by following the ancestral alleles in the previous generation from which this given alleles evolved. Eventually, we reach a point where two alleles coalesce (i.e., they find a common ancestor). The *Multi-Species Coalescent* (MSC) model is the extension of this general coalescent framework where multiple randomly mating populations corresponding to multiple species are present. Thus, multi-species coalescent represents a gene tree inside a species tree.

Incomplete lineage sorting or deep coalescence refers to the case in which two lineages fail to coalesce at their speciation point. Under the coalescent model, deep coalescence can be a source of discordance, because the common ancestry of gene copies at a single locus can extend deeper than speciation events. Figure 7 shows an example of discordance due to ILS. Therefore, in the presence of gene tree-species tree discordance due to incomplete lineage sorting, estimating species trees from a set of gene trees should take the reason of discordance into account [224,348,381, 26].

*Horizontal Gene Transfer* In many organisms (bacteria for example), a significant level of genetic exchange occurs between lineages, and lineages can combine to produce new independent lineages. *Horizontal Gene Transfer* (HGT), also known as *lateral gene transfer*, is the process that causes the genes to be transferred across species. These exchanges and combinations result in discordance between gene trees and species trees, and the accurate representation of evolutionary history requires a phylogenetic network instead of a tree [224,260,229]. Figure 8 shows how HGT transforms a tree into a network, which results in gene tree-species tree discordance.

With the abundance of molecular data available, species tree reconstruction from genes sampled from throughout the whole genome has drawn significant at-

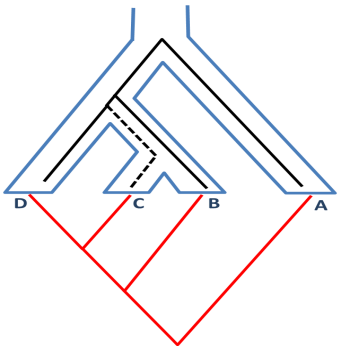


**Fig. 7 Reconciliation under incomplete lineage sorting**. We show a reconciliation of the discordant gene tree-species tree pair shown in Fig. 5 using incomplete lineage sorting. We embed the reconciled tree inside the species boundaries. Going back in time, the gene copies within species $B$ and $C$ first meet at their corresponding speciation point (i.e, the most recent common ancestor of species $B$ and $C$), but fail to coalesce at the speciation point. Both of these copies go further back in time, and hence we have two gene lineages (dashed and solid black lines) on deeper ancestral branch. Therefore, we have one *extra lineage* on the ancestral branch. The gene from $C$ first coalesces with the gene from species $D$, and subsequently with the gene from $B$.

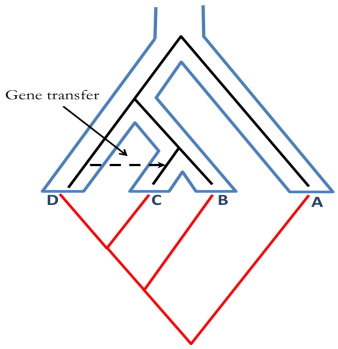


**Fig. 8 Reconciliation under horizontal gene transfer**. We show a reconciliation of the discordant gene tree-species tree pair shown in Fig. 5 using horizontal gene transfer. Here, the gene lineage from species D moves horizontally across species boundaries and enters into the species boundary of C. This "foreign" gene lineage is maintained and spread into the receiving species population. If the receiving lineage (C) goes extinct or is not sampled, then there will be discordance between the species tree and the gene tree.

tention from systematists. However, species tree reconstruction from a set of gene trees, in the presence of different biological processes causing gene tree discordance, is a challenging task. Central to addressing this challenge is to develop mathematical models to explain (or reconcile) gene tree-species tree incongruence assuming specific reasons for discordance (as shown in Figures. 6, 7, and 8). This concept of reconciling gene trees inside a species tree dates from Goodman *et al.*'s [129] attempt to find the most parsimonious reconciliation of a gene tree within a species tree under duplication and loss events. Later on, this concept of reconciliation was explored quite extensively [137,249,273,274,275,133,386]. Two of the most

popular approaches for estimating species trees from a collection of gene trees are *concatenation* (also known as "combined analysis") and *summary* methods.

- **Combined Analysis:** Concatenation or combined analysis is the most basic and simple pipeline for phylogenomic analysis where alignments are estimated for each gene and concatenated into a supermatrix, which is then used to estimate the species tree. Concatenation does not consider gene tree discordance as it combines all the gene alignments into a supermatrix. Implicit in this analyses is the assumption that all the genes have the same evolutionary history. Therefore, this approach can return incorrect trees with high confidence in the presence of gene tree discordance [86,99,145,196,199,214].
- **Summary Methods:** Summary methods refer to a broad class of phylogenomic methods that construct a species tree by summarizing a collection of gene trees. Gene tree parsimony methods such as estimating species trees by Minimizing Deep Coalescence (MDC) and Minimizing Gene Duplication and Loss (MGD/MGDL) are examples of summary methods [348,25,27]. Unlike concatenation, summary methods are not necessarily agnostic about the reason of discordance and can be statistically consistent. Therefore, summary methods are becoming more popular and gaining much attention from systematists, and therefore many summary methods have been developed over the last decade [145,12,214,248,211,56,348,25,27,266,160]. Some of them have the nice theoretical guarantee that they are proven to reconstruct the true species tree with arbitrarily high probability, given a sufficiently large number of true gene trees [196,251,215,189,213,214,248,160]. Unfortunately, however, we do not know the true gene histories and the number of genes is limited. Thus, techniques that have nice statistical guarantee might perform poorly on biological data sets.

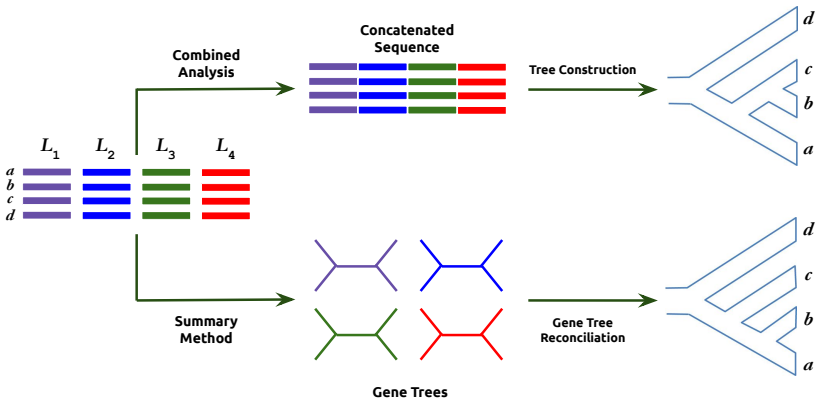


**Fig. 9 Two approaches for constructing species trees from gene trees.** Here we have four genes ($L_1, L_2, L_3, L_4$) across four species ($a, b, c, d$). Combined analysis combines all the gene sequence alignments and analyzes the supergene matrix, while summary methods reconcile individual gene trees.

Figure 9 demonstrates these two approaches of species tree estimation. There is another class of techniques (known as "co-estimation" techniques) that takes gene sequence alignments as input and co-estimates both the gene trees and species trees [145,42]. Co-estimation techniques are usually highly accurate but not scalable to large numbers of genes and species [26]. Thus, computer science and mathematical modeling have a great impact on the advancement of phylogenomics and more accurate and scalable techniques need to be developed to achieve the ambitious goal of *Tree of Life* which will undoubtedly require the integration of advanced computational techniques with biological insights.

### *3.2.3 Genome Rearrangements*

In 1936, Dorzhansky and Sturtevant proposed [336,91] that evolutionary remoteness between different species can be measured by the level of disorder between the organization of genes. They also suggested that the number of genome rearrangements explaining such organization disorder may represent a rough approximation of an *evolutionary distance* between organisms. Dorzhansky and Sturtevant studied multiple groups of flies. They proposed a *scenario* of inversions (i.e., a sequence of inversions transforming one genome into another) to explain chromosome difference, as well as the possible large-scale structure of the ancestral genomes for groups flies. In many subsequent studies, molecular biologists used techniques like in-situ hybridisation and chromosome banding to determine possible genome rearrangements in the evolutionary history of closely related species [290]. As we entered the genome sequence era, the importance of rearrangements in evolution was shown by examining the difference in the gene order of the mitochondrial genomes of cabbage and turnip, which have very similar nucleotide sequences of genes but dramatically diverge in the gene order [277].

Since genome rearrangements are relatively rare events, in 1941 Sturtevant and Novitski [337] formulated the problem of minimizing the number of inversions that may explain the difference between gene orders of two species. Typically, the measurement of the number of genome rearrangements is based on the *maximum parsimony assumption*, where the evolutionary distance is estimated/approximated as the minimum number of rearrangements between genomes.

In the early 1980's, mathematicians and computer scientists began to formalize the biological problem of finding evolutionary distance using mathematical notation. Watterson et al. [369] proposed to represent the relative gene orders in different genomes as *permutations*, and thus the original problem was translated into the problem of transforming one permutation into another with a minimum number of inversions. However, the proposed formulation ignores the fact that DNA molecule has two-strands, and some rearrangements may change the strand that a gene belongs to. Therefore, taking into account this fact, each gene may be assigned a "+" or a "−" sign to indicate the strand it resides on. That leads to the model, where relative gene order is represented as a *signed permutation*. Hannenhalli and Pevzner [143] showed that the problem of finding the minimal number of inversions for signed permutations has a polynomial-time algorithm in contrast to the unsigned variant, which is NP-hard. In subsequent studies, the initial algorithm was extended to translocations and resulted in the theory that is known in our days as the Hannenhalli-Pevzner theory [143].

In this section, we will overview the number of genome rearrangements models that are widely used in our days. Moreover, we will describe how these models can be used to measure the evolutionary distance between species. We will discuss the application of genome rearrangement models to the reconstruction of ancestral gene organization based on genomes of extant species. It should be noted that the application of genome rearrangement models is not restricted only on the ancestral genome reconstruction. These models have been used in many other fields of computational biology, such as scaffolding [2], reference guided genome assembly [183], phylogenomics [279], and cancer genomics [384]. Comprehensive reviews of genome rearrangement models and its applications can be found in [113, 142,383,15,144,311].

*Genome Rearrangement Models* In the genome rearrangement studies, a *genome* is often viewed as a set of linear or circular gene sequences representing *chromosomes*. Depending on the assumptions that are made on the genomes and the set of genome rearrangements that are considered, different genome rearrangements models are used. We will overview the simplest possible models. We assume that (*i*) the order of genes in each genome is known; (*ii*) all genomes share the same set of genes; (*iii*) all genomes contain a single copy of each gene. Since many of the results are based on graph structures, we first discuss the graphical representation in this context.

Under the assumptions made, a genome can be viewed as a graph, called *genome graph*, that can be constructed as follows. Each circular chromosome consisting of $n$ genes is represented as a graph cycle with $n$ directed edges encoding genes and their strands, which alternate with $n$ undirected edges connecting the extremities of adjacent genes [169,66] (see Figure 10a). Each gene $x$ is represented as a directed edge $(x^t, x^h)$, where $x^t$ and $g^h$ refer to the endpoints of the genes representing its tail and head, respectively. Similarly, each linear chromosome consisting of $n$ genes is represented as a path with $n$ directed edges alternating with $(n+1)$ undirected edges (see Figure 10b). We label each directed edge with the corresponding gene $x$, and further label its tail and head endpoints with $x^t$ and $x^h$, respectively. A collection of such cycles and paths representing the chromosomes forms the *genome graph* (see Figure 10a,b). The endpoints of linear chromosomes in a genome (i.e., the endpoints of paths in the corresponding genome graph) are called *telomeres*. We call an edge *irregular* if one of its endpoints is a telomer.

A major tool for the analysis of genome rearrangements between two genomes is the *breakpoint graph* [18]. For two genomes, the *breakpoint graph* can be constructed by gluing the identically labeled directed edges in the corresponding genome graphs (see Figure 10c,d). The breakpoint graph can be also easily defined for three or more genomes [16]. We will consider three most commonly used rearrangement models:

1. *Single-Cut-or-Join (SCJ) model [110].* The SCJ model includes two types of operations: a single "cut" in a genome, and "gluing" two telomeres in a genome. The SCJ operations mimic reversals, translocations, fissions, fusions, and transposition since each rearrangement can be represented by compositions of cuts and joins.
   A SCJ operation in genome $P$ corresponds in the genome graph to the replacement either a single undirected edge with two irregular edges or two irregular

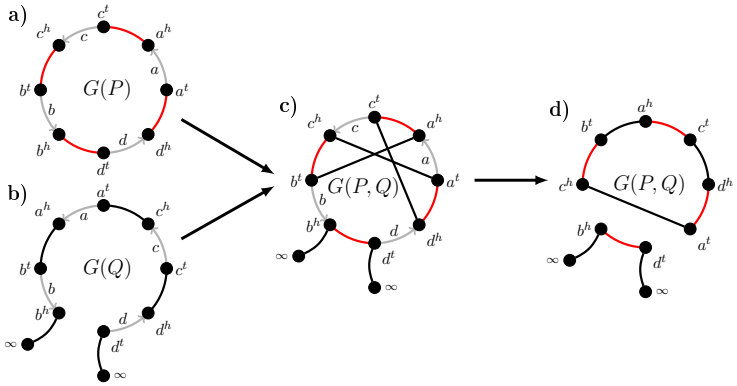


**Fig. 10** **a)** Genome graph $G(P)$ for unichromosomal circular genome $P = \{+d + a + c + b\}$, where undirected $P$-edges are colored red. **b)** Genome graph $G(Q)$ for unichromosomal linear genome $Q = (+d+c+a+b)$, where undirected $Q$-edges are colored black. **c)** The superposition of genome graphs $G(P)$ and $G(Q)$. **d)** The breakpoint graph $G(P, Q)$ (two layouts) is obtained from the superposition of $G(P)$ and $G(Q)$ with removal of directed edges. The graph $G(P, Q)$ is formed by a black-red path and cycle.

edges with a single undirected edge (see Figure 11a). The evolutionary distance between genomes $P$ and $Q$ on the same $n$ genes under parsimony assumption and Double-Cut-and-Join (DCJ) model (or simply *DCJ distance*— discussed below) equals $d_{SCJ}(P, Q) = |E(P) \triangle E(Q)|$, where $\triangle$ denotes the symmetric difference and $E(P), E(A)$ are the sets of undirected edges in the breakpoint graphs of genomes $P$ and $Q$, respectively. While the SCJ model is a simplistic approximation of genome rearrangement evolution, this model has the advantage to allow for an efficient solution of many core problems of genome rearrangement analysis [109].

2. *Double-Cut-and-Join (DCJ) model [378,31]* (also known as 2-*break* model [9]). The most common rearrangements are reversals, translocations, fissions, and fusions. All these rearrangements can be conveniently modeled by DCJ operations, which make up to two "cuts" in a genome and "glue" the resulting genomic fragments in a new order. The transpositions are included indirectly via two consecutive DCJ operations: the excision of a circular intermediate chromosome and the reinsertion at a different position.
   A DCJ operation in genome $P$ corresponds in the genome graph to the replacement of a pair of undirected edges with a different pair of undirected edges (see Figure 11b). The evolutionary distance between genomes $P$ and $Q$ on the same $n$ genes under parsimony assumption and DCJ model equals $d_{DCJ}(P, Q) = n - c(P, Q) - \frac{p(P,Q)}{2}$, where $c(P, Q)$ and $p(P, Q)$ are the number of cycles and even paths (i.e., paths with even number of edges) in the corresponding breakpoint graph for genomes $P$ and $Q$ [378,31]. By various experiments, it was shown that the DCJ model represents a realistic evolutionary model. However, the majority core problems of genome rearrangement analysis cannot be solved efficiently (i.e., these problems are NP-hard) [346].
3. *k-break model [9].* The most common rearrangements can be modeled by making two "cuts" in a genome and "gluing" the resulting fragments in a new order, i.e., by DCJ operations. However, the rearrangement operations that

create three breakpoints, e.g., transpositions, can be modeled only indirectly when at most two cuts are allowed. So, one can imagine a hypothetical *k-break rearrangement* that makes $k$ "cuts" in a genome and further "glue" the resulting genomic pieces in a new order [8,9] (Fig. 11c). While $k$-break rearrangements for $k > 3$ have not been observed in evolution, they are used in cancer genomics to model *chromothripsis*, a catastrophic event of multiple breakages happening simultaneously in a genome [334,372]. It is known that the evolutionary distance under parsimony assumption and $k$-break model (or *k-break distance*) between two genomes can be computed in terms of cycle lengths in the breakpoint graph of these genomes [8]. Namely, while the 2-break distance depends only on the number of cycles in this graph, the $k$-break distance in general depends on the distribution of the cycle lengths. It should be noted that among the aforementioned rearrangements models, the $k$-break model is the most poorly studied model.

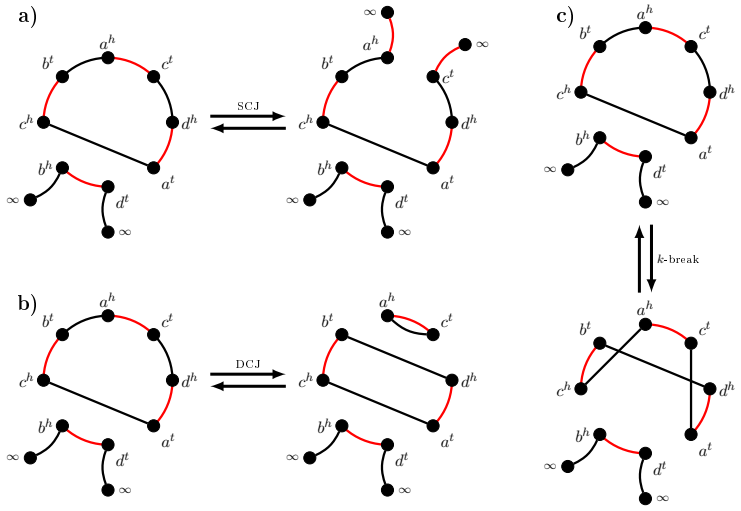


**Fig. 11** **a)** A SCJ in genome $P$ replaces a single red edge in the breakpoint $G(P,Q)$ with two irregular red edges. **b)** A DCJ in genome $Q$ replaces a pair of black edges in the breakpoint $G(P,Q)$ with another pair of black edges forming matching on the same 4 vertices. **c)** A 3-break in genome $Q$ replaces a triple of black edges in the breakpoint $G(P,Q)$ with another triple of black edges forming matching on the same 6 vertices.

While the modern rearrangement models have allowed us to achieve remarkable results, there are a lot of open questions in genome rearrangement analysis. The main problem of all the existing models is that it is impossible to reconstruct the "true" scenario of genome (i.e., a sequence of genome rearrangements that transform one genome into the other) happening in the evolution of given genomes [44]. It is happening because for a given evolutionary distance under all models, there exist several equally likely genome rearrangements scenarios. Recently, several initial results have been proposed, which are attempted to weigh different rearrangements depending on the location in the genomes [34,338]. Another open question is that in reality, the parsimony assumption may not always

hold, emphasizing the need for estimation of evolutionary distance that does not rely on such assumption. The distance that accounts for the actual (rather than minimal) number of rearrangements between two genomes is often referred to as the true evolutionary distance [206,10,34]. Finding true evolutionary distance under different rearrangement models is a challenging problem. Finally, developing rearrangement models that include different types of events significantly improves the accuracy of genome rearrangement analysis.

*Reconstruction of Ancestral Genomes* With the increased availability of assembled genomes, the development of computational methods for reconstructing ancestral genomes becomes increasingly relevant. There have been tremendous methodological developments over the last 10 – 15 years in this direction. The state-of-the-art methods in our days are able to propose the ancestral genome organization in mammals [16,223,7,59], insects [263], fungi [57,111], plants [309], bacteria [366, 280], amniotes [182,259], vertebrates [182,259], chordates [296], or dinosaurs [272].

The main computational problem of great interest in the reconstructing ancestral genomes is the *Small Parsimony Problem* (SPP), which asks to reconstruct ancestral genomes at the internal nodes of a given phylogenetic tree from the extant genomes at the leaves of that tree. There are two main computational approaches for the SPP, namely *rearrangement-based* [16,7,41,388,377,389] and *homology-based* [59,223,167,222,153,335,121,284,117]. The rearrangement-based methods for a given rearrangement model reconstruct ancestral genomes from the given extant genomes by minimizing the total rearrangement distance between genomes along the branches of the phylogenetic tree. On the other hand, the homology-based methods do not assume any rearrangement model. Instead of minimization the total rearrangement distance along the branches, the homology-based methods reconstruct ancestral genomes from the given extant genomes, by minimizing the total amount of *homoplasy* phenomenon observed in the phylogenetic tree. *Homoplasy* is a phenomenon by which two genomes in different lineages acquire independently a same feature that is not shared and derived from a common ancestor.

Both types of methods have pros and cons. For example, the results of rearrangement-based methods are complete genomes. However, the results of homology-based methods are contiguous ancestral features that contain only reliable features, called *Contiguous Ancestral Regions* (CARs) [223]. While the rearrangement-based methods rely on the current state-of-the-art rearrangement models, the homology-based methods can produce results even for genomes undergoing genome rearrangements that are not included in the state-of-the-art rearrangement models [111]. Furthermore, in various studies [108], it was shown that homology-based methods are able to reconstruct reliable CARs even if parsimony assumption on the evolution between genomes does not hold.

Despite the recent advances in solving SPP, this problem still poses many challenges. For example, while there exist genome rearrangement models that include indels and various duplication events, there are no rearrangement-based methods that work with duplication events. On the other hand, the accuracy of the homology-based methods are still worse than the rearrangement-based methods on genomes for which parsimony assumption holds [111,108]. Recently, the promising direction of developing tools which combine the advantages of the homology- and rearrangement-based methods are emerging [107,108,17].

### 3.3 Genome-Wide Association Studies (GWAS)

Although genomes of individuals of a species are quite similar, there are variations in the sequence of DNA across individuals. These variations are broadly of two types – sequence variations and structural variations. Sequence variations include *substitutions* (mutation at a single base from one base into another), and insertion or deletion of a few bases (jointly known as indels). In some cases, both variants or alleles resulting from a mutation persist in populations. These are known as Single Nucleotide Polymorphisms (SNPs). On the other hand, structural variations are long-range variations in chromosomes including insertions, deletions, Copy Number Variations (CNVs), inversions and translocations.

Some of these variations or genotypes may result in changes in an idividual's traits or phenotypes and can cause diseases through alteration of the structure of the proteins encoded, change in regulation of expression, or other mechanisms. *Association mapping* refers to associating regions or variations in genome to various phenotypes. Although association does not necessarily imply causality, the associated regions may then be investigated for causal variants. Association mapping is typically done in the form of Genome Wide Association Study (GWAS), which considers all the variants in the whole genome to see if any variant is associated to a particular trait or phenotype. GWAS is performed in a *case-control* setup. For a particular phenotype, two groups of individuals are selected – individuals who have the trait or the disease of interest (cases) and who do not (controls). From the SNP array constructed from the individuals being considered, each SNP is tested for association with the phenotype by computing a $P$-value using a statistical test. If certain genetic variations are found to be significantly more frequent in people with the trait (disease) compared to people without disease, the variations are said to be "associated" with the disease.

The impact of GWAS in medical science could potentially be substantial. With the success of the Human Genome Project [72], 1000 Genomes Project [67,68, 69], International HapMap Project [70] and other community effort to collect phenotype-genotype data [65,363], researchers are now able to find various genetic variations associated with a particular disease. Subsequently, numerous important GWA studies have been performed to date [46]. These will help the researchers develop better customized strategies to treat and prevent the disease, and will pave the way for modern research in precision medicine.

Performing GWAS in a correct manner is not easy and requires specific knowledge of genetics, statistics, and bioinformatics. We now briefly discuss a few of the key concepts related to a genome wide association studies.

*$P-value$* The SNPs are tested for association with the phenotype by computing a $P$-value using a statistical test. $P$-value is the probability of observing an outcome which is "at least as extreme" as the one being observed if the null hypothesis is true. The null hypothesis in this context is that the SNP is not associated with the phenotype, and the the alternate hypothesis is that it is associated with the phenotype. Therefore, small $P$-value is taken as an evidence that the null hypothesis may not be true. If $P$-value is very small, then it would be very unlikely to observe the data under the null hypothesis. Subsequently, either null hypothesis is not true or an unlikely event has been encountered.

For genetic association analyses of complex traits, determining the correct $P$-value threshold for statistical significance is critical to control the number of false-positive associations [105]. Since, in a typical GWAS, $P$-values of millions of SNPs are computed, the $P$-value threshold for significance must be corrected for multiple testing and SNPs with $P$-values less than $5 \times 10^{-8}$ are commonly considered signifcant for humans [105,283,71]. However, this commonly used $P$-value threshold needs to be updated to account for the lower allele frequency spectrum used in many recent array-based GWA studies [374,283,105]. Fadista *et al.* [105] confirmed that the $5 \times 10^{-8}$ $P$-value threshold to be valid for common (Minor Allele Frequency (MAF) $> 5\%$) genetic variation in the European population. However, for lower frequency variants, the genome-wide $P$-value threshold needs to be more stringent for studies with European ancestry.

*Minor Allele Frequency (MAF)* MAF is the frequency of the least often occurring allele at a specific location. SNPs with a low MAF are rare, and it is quite challenging to detect associations with SNPs with very low MAF and therefore most studies exclude SNPs with relatively low MAF [233]. The HapMap project [71] considered the SNPs with a minor allele frequency of $\geq 0.05$.

*Population stratification* Population stratification refer to the presence of multiple subpopulations (e.g., individuals with different ethnic background) in a study. Such allele frequency differences between cases and controls, due to systematic ancestry differences, is an important source of systematic bias in GWAS. Since there can be difference in allele frequencies across populations, population stratification can lead to false positive associations and/or mask true associations [194,217,116,230,146]. An interesting example is the "chopstick gene", where a SNP, due to population stratification, accounted for nearly half of the variance in the capacity to eat with chopsticks [141]. Therefore, correcting for population structure and other confounding factors such as age and sex is often needed for $P$-value computation. Principal component analysis (PCA) based approach [294,281] is commonly used to identify and correct for population stratification. PCA based approach uses genome-wide genotype data to estimate principal components which can be used as covariates in the association analyses. EIGENSTRAT [294] is a widely used PCA based approach to correct for population stratification.

*Linkage disequilibrium (LD)* Linkage disequilibrium refers to the nonrandom association between alleles at different loci in a given population. Individuals who carry a particular SNP allele at one site often predictably carry specific alleles at other nearby variant sites. This correlation is known as linkage disequilibrium, which exists because of the shared ancestry of contemporary chromosomes [71,327]. LD can be considered to be linkage between markers on a population scale [50]. The concept of linkage disequilibrium is important in GWAS as it allows identifying genetic markers that tag the actual causal variants. Therefore, systematic studies of common genetic variants are facilitated by the concept of LD [71,327]. "Clumping" is a procedure used in GWAS to identify and select the most significant SNPs in each LD block for further analyses [233].

*Manhattan plot* Manhattan plot is a popular approach to visualize the resulting $P$-values across the genome, where negative logarithm of $P$-values are plotted against genomic co-ordinates of SNPs. Manhattan plot is one type of scatter plot where each "dot" corresponds to a SNP. Since the strongest associations have the smallest $P$-values, their negative logarithms will be relatively higher.

PLINK [295], SUGEN [205], SNPtest [73], GWASTools [125], EMMAX [171], and GEMMA [391] are a few most commonly used tools for performing GWA studies. METAL [375], GWAMA [227] and FUMA [368] are widely used for meta-analysis of GWAS results and summary statistics. We do not pursue this further here, but see [50,148,365,235] for more details on various components of GWAS and different challenges in performing successful GWAS.

### 3.4 Conclusion

On an ending note, from these descriptions of various important fields in computational biology, we can comprehend how mathematical modeling, algorithms, simulation and statistical analyses have helped the biologists translate biological processes into computational models. Efficient computational techniques have become an integral part for solving various problems in biology, especially when we need to handle large amount of biomolecular data. Thus, we should propel our effort in utilizing the power of computer science to better understand the nature.

## 4 Algorithms in Biology: Challenges and Discussion

The mutual relationship between biological computation and computational biology goes a long way back. Various computational techniques have been inspired by the nature and these nature-inspired algorithms have been successfully used to solve different biological problems. Computational biology has greatly harnessed various nature-inspired algorithms. Notably, various evolutionary algorithms (e.g., genetic algorithms) and simulated annealing have been used for multiple sequence alignment and phylogenetic tree estimation [268,270,262,331,186,393,202], artificial neural networks and deep learning has been widely used in predicting various protein attributes (e.g., secondary structures of proteins) [106,330,363,166, 387,165,364,355]. In particular, deep learning is being extensively used in various domains in computational biology. For example, DeepVariant [291] (a deep learning-based variant caller), and DeepFold [216] (a deep convolutional neural network model to extract structural motif features of a protein structure) are two notable deep learning based methods which demonstrate the power and efficacy of deep neural networks in solving biological questions. The wide-ranging impact and application of deep learning methods in various fields of computational biology underscore the importance of separate comprehensive reviews. We do not pursue this further here, but see [13,370,122,278,14,61,354]. Thus, methods like artificial neural networks have been inspired by biological neural networks and subsequently they are being successfully used to solve various biological questions. Nevertheless, up to now, the application of computational methods to study, explore and deeply understand biological processes, and the impact of biological systems on the development and improvement of computational techniques remained mostly separate.

Guaranteeing a strong bond between these two research directions involves considering biological processes as bio-inspired algorithms, aiming at solving challenging real-world problems under a range of constraints and conditions. The fact of considering these algorithms as information processing systems, allows a much better and a deeper understanding of their biological characteristics and features; while the gained knowledge empowers the design and the enhancement of computational systems.

Seeking the fortification of this convergence between biological computation and computational biology, new biological challenges have emerged requiring the contribution of biological computation techniques. Such need has been also noticed in the biological computation direction where some methods showed some limitations in solving real-world problems, and hence a call for a profounder connection with computational biology seems crucial.

## 4.1 Challenge 1: The explosive increase of biological data

Nowadays, with the explosive increase of biological data, biologists are faced with the challenges derived from gathering and processing enormous data. An excessive data growth is witnessed in DNA sequencing, for instance, in which thousands of genomes need to be explored and analyzed in concert. Similarly, cellular imaging and the organisms' phenotypes require to be systematically assessed in a high-throughput format. The biological data are not only characterized by their large volume and their velocity but also by their variety and veracity aspects, making the task for pure biologists harder, challenging and more complicated. Indeed, the biological data are commonly characterized by their imperfection, their imprecision (e.g., missing/vague values), their uncertainty (e.g., probable, possible values), and their inconsistency (e.g., conflicting or incoherent data values). Biologists need to carefully deal with this veracity aspect to avoid any possibly faulty analyses of the conducted experiments. Dealing with this aspect remains among the major worries for biologists, as they should evade any bias in the interpretations of their results.

### *4.1.1 The need to re-design biological computation techniques using distributed parallel infrastructures*

It is based on this first set of challenges, which are related to the urgent need for dealing with big biological data, that biological computation comes into action. This need merited serious attention from computer scientists, statisticians, and mathematicians, and have prompted several achievements in bio-inspired intelligent systems from large-scale data curation, i.e., content creation, selection, classification, transformation, validation, and preservation to data mining, data visualization, and optimization. Several of the biological computation techniques, discussed in Section 2, that originally deal with a manageable amount of data, found their limits and could not work efficiently and satisfactorily when the amount of biological data exceeded the capabilities of a given system to process the data in terms of time and/or memory consumption. Therefore, it became necessary to re-design the biological computation techniques to excavate big data to enable enhanced decision making, insight discovery and process optimization. Specifically,

these developed bio-inspired techniques and prototypes which are tested on proof-of-concept biological examples must scale smoothly to real-life case studies. The context of big data has also demanded serious efforts in worldwide infrastructure to support this revolution, where a new generation of robust fault-tolerant systems based on parallel and distributed computer architectures have been established. These parallel frameworks allow distributed processing of large data sets across clusters of computers using simple programming models. They are designed to scale up from single server to thousands of machines, each offering local computation and storage. In this sense, the biological computation techniques were re-modelled and developed in a distributed and parallel way [5] using a set of distributed parallel infrastructures such as Apache Hadoop[7], Apache Storm[8], Apache Samza[9], Apache Spark[10] and Apache Flink[11]. The distributed bio-inspired techniques via the integration of sophisticated modeling capabilities and high performance computing can help biologists in coherently studying complex biological systems and behaviors, and extracting and analyzing detailed biological principles that underlie the huge amount of data acquired. With these facilities, new opportunities for discovering new biological values from massive data sets can be sought, helping to gain an in-depth understanding of the hidden values, and also to incur new challenges. This clearly shows the mutual relationship between biological computation and computational biology and the convergence between these two directions. Computational biology, as shown in Section 3, has much to offer as inspiration to derive powerful biological computation techniques. These bio-inspired techniques can also promote the computational biology research by producing large amount of knowledge from biological data analyses and even through data visualization.

#### *4.1.2 The need to re-build more powerful software biology tools to deal with real biological problems*

Still, it is important to mention that biological computation comes with a major challenge in addressing real biological problems, i.e., real-life biological cases. In this concern, a set of software tools have been released specifically designed for extracting the significant information from biological data, and to carry out functional, structural or interactional analysis. These tools are dedicated to perform several tasks such as gene identification and function prediction, structure prediction and modeling, molecular interaction prediction, biological network prediction and modeling, cell level modeling and simulation, mechanical stress to cell signaling modeling, and modeling of ecosystems. These can be classified as homology and similarity tools, protein functional analysis tools, sequence analysis tools and miscellaneous tools. Among the popular and most used open-source software tools, we mention BLAST[12] a sequence-comparison program, BEAST [95][13] a cross-platform program for Bayesian analysis of molecular sequences, MEGA [192,344][14]

[7] `http://hadoop.apache.org/`
[8] `http://storm.apache.org/`
[9] `http://samza.apache.org/`
[10] `https://spark.apache.org/`
[11] `https://flink.apache.org/`
[12] `https://blast.ncbi.nlm.nih.gov/Blast.cgi`
[13] `https://beast.community/`
[14] `https://www.megasoftware.net/`

a tool for estimating evolutionary distances, reconstructing phylogenetic trees and computing basic statistical quantities from molecular data, RAxML [333,332][15] and MrBayes [157] – two widely used tools for likelihood-based and Bayesian inference of phylogenetic trees, AMPHORA[16] an automated phylogenomic inference application for large-scale protein phylogenetic analysis, Ascalaph Designer[17] a software for general purpose molecular modelling for molecular design and simulations, DNASTAR Lasergene Molecular Biology Suite[18] a software to align DNA, RNA, protein, or DNA+ protein sequences via pairwise and multiple sequence alignment algorithms, DECIPHER[19] a software toolset that can be used to decipher and manage biological sequences efficiently, MEME[20] a tool for discovering motifs in a group of related DNA or protein sequences, and many more. These computational techniques have substantially sped up the revolution of computational biology, expanded the scope of biological research by having a remarkable impact on the scientific community, and truly fostered innovation. However, when it comes to big biological data sets, few of these software biology tools are capable of dealing with such large amount of data, as when increasing the amount of data the processing task of these tools becomes prohibitively slow. This is explained by the fact that these classical software tools can neither deal with the data in an interoperable mode nor conduct analyses in an integrative way. In computational biology, it is hard to make sense of lots of data if these are dealt with separately or in their own. The data should be linked to other sources of information (e.g., other data sets) that all together can be handled by the same software once at a time. These limit the use of these standard tools and urge the call for the parallel infrastructures. The discussed distributed frameworks offer plenty of features that solve the problems of the current standard biological software tools. Apart from being lightning-fast cluster computing technologies specifically designed to handle big data, they offer an interoperable working environment allowing an integrative analyses of data, and a robust and reusable software infrastructure on which software biology can be built.

#### *4.1.3 Highlights*

At this point, we notice that there are some gaps between the work and advances in enhancing the development of distributed biological computation techniques, the available parallel frameworks in the business market and the set of the used biological software tools. Up to now, biological computation techniques have been developed to solve a variety of problems, but when it comes to solve specific problems in computational biology, these techniques are mainly designed as sequential methods, handling manageable data sizes, or as distributed methods, handling big biological data while making use of the emerging new parallel infrastructures. On the other hand, only a limited number of bioinformatics software (that we also call "software biology tools") are making use of the parallel frameworks, while properly

[15] `https://cme.h-its.org/exelixis/software.html/`
[16] `http://wolbachia.biology.virginia.edu/WuLab/Software.html`
[17] `http://www.biomolecular-modeling.com/Ascalaph/`
[18] `http://www.dnastar.com/`
[19] `http://www2.decipher.codes/`
[20] `http://meme-suite.org/`

involving biological computation techniques in their mechanisms. This gives rise to a new challenge in biological computation that requires further attention from computer scientists, statisticians, and mathematicians. The sought biological software tools, that we can call *"a second generation of biological software tools"*, need to be reusable and portable. They should take into consideration all of the discussed aspects ranging from flexibility, scalability, interoperability, and should fade into the working background so that the user can mainly focus on more interesting tasks such as data analysis and interpretation.

### 4.2 Challenge 2: The larger dimension and complexity of scientific questions

Another main challenge faced by biologists and to be addressed by the computational community, i.e., computer scientists, mathematicians, statisticians, is the need to address the increasingly larger dimension and complexity of their scientific questions. Indeed, with the aim to develop a large ecosystem involving several biological components, it will easily become impractical in computational biology to have the whole system, intended to be designed, available for testing, and hence biologists need to find alternatives for studying and validating their systems behavior. As another aspect of the already mentioned convergence between the two research directions (i.e., biological computation and computational biology), quantitative computing in biological computation comes into play to foster the move toward a simulation-based science that is needed to address these new challenges.

Simulation systems in biology are built to help biologists study the dynamic behavior of biological objects in response to an environmental setting that cannot be easily or safely accessed in real life. Simulations are particularly beneficial in assisting biologists to measure and predict how the functioning of an entire biological system may be affected by mutating some of the system's components. More precisely, simulations of biological systems, via the quantitative computing in biological computation, have a great forecasting power as they allow biologists to forecast scenarios that have never happened before and to run new setups outside of the known biological historical bounds.

These simulation systems provide biologists with highly flexible practices for addressing and answering extremely relevant research questions such as "How does order emerge from disorder?". The answers to this scientific question, for instance, can offer biologists some new ways of organizing robust and self-adapting natural and technological networks. Indeed, practical feedbacks are provided to biologists when designing their biological systems allowing them to determine the correctness and effectiveness of their proposed design before the system is actually constructed. Subsequently, biologists may explore the merits of some other possible designs while avoiding the need to physically build their systems. Furthermore, by exploring the properties of specific design decisions during the design/simulation phase rather than in the construction phase, the overall cost of constructing the system reduces considerably. Consequently, in comparison to the cost of experimenting in the real world, the use of simulations of biological systems requires very little time and resources.

Another benefit that the biological computation community offers to biologists via the simulators is the ability to study the biological problem at different levels of abstraction. If biologists interact with the system at a higher level of abstraction

then they can better understand the interactions and behaviors of all the high level biological system's components. This interaction is clearly displayed via the use of computer graphics and animations, modules of the simulation systems, to dynamically show the behavior and relationship of all the simulated system's components, thereby a meaningful understanding of the system's nature can be provided. From a high level of abstraction, biologists will be better prepared to counteract the complexity of their overall system. This complexity may overwhelm biologists if they have approached the system from a lower level. By using the simulation systems biology, biologists can better understand the operation of the higher level components, and therefore can simply deal with the lower level components.

To enable a team to test and answer more scientific questions in a great value-add, to deal with the growth of complex ideas in research projects, and with the desire to keep testing a broad range of hypotheses and more scenarios, it is only recently that a set of powerful simulation systems in biology has emerged. Among these, we mention Kappa[21] [43] an integrated suite of analysis and visualization techniques for building and interactively exploring rule-based models, BioNetGen[22] a software for generating mathematical/computational models that account comprehensively and precisely for the full spectrum of molecular species implied by user-specified activities, potential modifications and interactions of the domains of signaling molecules, BioSPICE Dashboard[23] intended to assist biological researchers in the modeling and simulation of spatio-temporal processes in living cells, INSILICO discovery[24] an advanced computational tool for network oriented "in silico" analysis and design of cellular properties, and many more. Although the biological computation community was making many efforts in developing a wide range of simulation systems for biology, the use of these software tools remained somehow restricted, and called for further improvements. More precisely, like with most tools, the main problem is linked to the computationally intensive processing which is required by simulators. This computationally intensive processing which is required by simulators can be either tied to the large amount of the biological data being simulated or due to the complex interactions that occur between the system's entities. Consequently, a considerable delay may occur to get the results of the simulation readily available after launching the simulation — an event that may happen instantly in the real world may actually take hours to mimic in a simulated environment. Accordingly, the biological computation community is encouraged to improve these biological systems that are restricted by limited hardware platforms by making use of more powerful distributed infrastructures as these are available in these days.

Now from a technical perspective which gives rise to another challenge, simulation software tools use a set of mathematical descriptions (or a model) of a real system in the form of a computer program to describe the functional relationships between the systems' components. In this concern, and to ease the task for biologists to better use simulators, it will be a great addendum if the computational community work on hiding as many formal details as possible from biologists, and offer a wider palette of software visualization; an improvement that can play a

[21] https://kappalanguage.org/
[22] https://cellsignaling.lanl.gov/bionetgen/
[23] http://www.biospice.org/
[24] http://www.insilico-biotechnology.com/discovery_en.html

critical role. More specifically, such simulation software should be user-friendly and have good Graphical User Interfaces. Actually, the visual metaphors of algorithm animations can help biologists understand how systems evolve, even while they remain bound to their standard "picture" representations. Adding to this, a formalized set of descriptors (i.e., software manuals) and best practices of these software tools, which is not systematically and always readily available (but there is an effort in this direction noticed recently), is required to further help biologists with their work and to contribute in their achievements.

Based on these challenges in designing and developing simulators for biological problems, a fundamental question is whether these virtual mountains of proteomics, expressions, sequences, images, and other biological data can be converted into biological knowledge in such way that it is trusted to biologists. Today, some biologists are still skeptical of simulation. We believe that this skepticism will subside when more success stories and validated forecasts will be witnessed and highlighted in biology.

### 4.3 Challenge 3: The paradigm-shifting of the research process

From another perspective, while excessive biological data growth poses a lot of hard challenges for the mathematical, statistical, and computer science communities, it is crucial to recall the importance of mathematically modeling the biological systems and problems. In fact, the usefulness of the mathematical models for the biological systems and problems is dictated by the rapid evolution of the technologies for gathering biological data. The consequence of such technological evolution is that the quality and properties of biological data obtained by different methods are also changing. The new type of data, also, requires the development of novel computational, statistical, and mathematical approaches. This technological race pushes the invention of methods to solve biological problems without properly formulating the mathematical model of a problem. Sometimes, researchers do not dig into real-case data, and so "quick-and-dirty" methods are used. Such methods are usually not solving a well-formulated mathematical problem and make it hard to use known mathematical, statistical, or computer science techniques. Nevertheless, these methods can perform well on real-case data. As a result, a well-formulated mathematical model of a specific biological problem is usually given for challenging cases where a "quick-and-dirty" method does not work. It is true that such a scientific process allows obtaining results, but it seems necessary that the computational biology and the biological computation communities need to change this practice towards the research process. In this new practice, the rigorous mathematical modelling of any biological problem should come before focusing on solutions or methods. Such a paradigm-shifting of the research process will help both communities in several ways.

Firstly, the suggested practice would help to enhance the collaboration between computational biology and biological computation. For example, if a computational biologist formulates the correct computational and mathematical problem, it would be much easier for a researcher in biological computation to understand which method should be used for solving this problem. Moreover, if a problem is formulated mathematically, the next generation of researchers from both communities can pay more attention towards the development of the effective methods

to solve various problems without calling for a specific expertise in the deep understanding of the biological aspect behind it. Such a mathematical formalization will also help both communities complement their background in biology if it is little. In addition, it will make the communication between these two communities fast and efficient. Computational biology and biological computation have common roots in computer science. Therefore, discussing existing research problems would be faster without recalling and explaining biological terms since mathematical formulations are much easier for computer scientists, mathematicians and statisticians.

Secondly, both of the computational biology and biological computation communities should bear in mind that working together will endorse the effectiveness and the universality of the developed methods; specifically when it comes to real-world applications. For instance, if one community develops a method that solves the mathematical formalization of a biological problem, then it will be feasible for the other community to verify the correctness of the obtained solution. This, in its turn, supports task partitioning in the collaborative projects because researchers (for example, from computational biology community) who pose the mathematical version of the problem can easily verify the correctness of the solution proposed by researches from the second community.

Thirdly, new approaches would favor capturing the common aspects of various seemingly unrelated biological problems and systems. Having such models for various biological problems would allow abstraction from imperfection, imprecision, uncertainty, and inconsistency of real-case data for each biological problem and focus on discovering the common phenomena among these problems. Mathematicians and computer scientists specifically could help in such discoveries. This is because of the nature of their research which permits a deeper and a detailed analysis of the problem to solve; with this capability to approach the problem from different angles using appropriate tools and techniques.

Despite the clear benefits of the described paradigm-shifting, in some cases, it is not so easy or straightforward to perform it. There are many remaining challenges in moving towards mathematical modeling of some biological problems that both communities are encouraged to address in the foreseeable future. These issues are raised because computational biology and biological computation are highly interdisciplinary fields and many researchers from different scientific cultures are usually involved in the same research process. In the rest of this section, we will discuss the most significant challenges in our opinion that make it difficult to achieve the desired goals.

One of the frequent issues occurring when a researcher tries to develop a mathematical version of a biological system or problem is that his/her model—even if it is considered as the most well-established solution—cannot “perfectly" describe the real-life biological cases. This situation pushes biologists to ask mathematical, statistician, and computer science communities to develop more sophisticated models that would allow a complete modeling of the complex biological processes. Unfortunately, this is often almost impractical due to time requirements and/or complexity of the biological systems. However, this challenge can be dealt with when opting for a significant simplification strategy of the biological systems in mathematical modeling which usually allows capturing the key aspects of the interesting biological problem. Therefore, the computational biology community needs to work on leveraging the power and usefulness of simplification in biological stud-

ies (mostly in mathematical modeling of biological systems) without ignoring the key aspects of the biological systems. Simplification is not only a powerful and useful technique, but simplifications are unavoidable in many biological studies.

From the perspective of mathematicians, statisticians, or computer scientists, it would be very hard to understand all the details behind the biological problem to be solved. Even if a biologist wants to describe every single detail about the biological phenomenon, there are details at the chemical and/or physical level that even a biology researcher might not fully grasp. Fundamentally, when a biological process is described in words (even in full details), the words are, by definition, a simplification of reality. Thus, even if mathematical models cannot describe completely the real-life biological systems and problems, such models can still significantly contribute to discover various mechanisms in the biological systems.

The complexity of biological systems and the race in technology also require the reorganization of the existing standard research workflow and the process of disseminating the result as publications. For example, the rapid pace of the technological race requires to quickly publish papers on new discoveries. However, the publishing cycle still can take a significant amount of time. Recently, researchers addressed this issue and launched in 2013 an open access preprint repository for biological science, called bioRxiv[25]. bioRxiv is an analog of the well-known preprint repository arXiv[26] that consists of scientific papers in the fields of mathematics, physics, astronomy, electrical engineering, computer science, quantitative biology, statistics, and quantitative finance. This initiative is a positive sign showing how mathematicians, statisticians, and computer scientists may influence the other discipline; helping the latter to solve some issues and improve its research workflow.

While open access preprint repositories are developed for dealing with the technological race, the problem with the description of complex biological systems remains somehow unaddressed. There are dozens, sometimes hundreds of publications dedicated to one biological phenomena. So, understanding the state-of-the-art information about some biological process in full details for a non-expert (for example, computer scientists) becomes an almost inaccessible problem. Therefore, communities who are involved in biological studies are encouraged to develop a web-based platform to represent the current knowledge about biological phenomena in a concise, complete, and explicit manner using the uniform view. Similarly as arXiv inspired the creation of bioRxiv, mathematical web-sites such as a wiki for research-level notes in mathematics, physics, and philosophy, called $n$Lab[27], and the On-Line Encyclopedia of Integer Sequences (OEIS[28]), may be used as an inspiration for such platforms. Such platforms would allow the biological computation community to more easily find information about biological systems and to use it for developing more sophisticated bio-inspired algorithms. On the other hand, computational biologists may use this system as a source of open problems for filling gaps in the biological system understanding.

Another example of demands in changing research workflow is that the results of the computational biology research are, often, not only the research paper but also some software dedicated to the specific problem. A software program has a dif-

[25] https://www.biorxiv.org/
[26] https://arxiv.org
[27] https://ncatlab.org/nlab/show/HomePage
[28] https://oeis.org/

ferent nature in comparison to a paper. It usually takes time that a research paper gets published after a reviewing process where authors get feedback to improve the quality of their research. Today, most of the authors are in line with the practice of sharing their source code and hence their research work is not restricted to the feedback of the reviewers; they are benefiting from a large feedback given by interested parties who used the code. This practice which aims at making the solution source code publicly available to all interested committees, is referred to as "open source" software. To endorse the adoption of such practice, there are several version control software such as Git[29], Fossil[30], Veracity[31], Mercurial[32], Monotone[33] and many more. These version control software also allow to help reporting bugs and errors. It is to offer the possibility to report some code bugs to technicians, to PhD students, to Postdocs and to academics that they can correct whenever appropriate and release their new versions that better meet the second community's needs and requirements. In this concern, it would be interesting to point out that implementing scientific software should become more rewarded in academia, so that also more researchers do coding, not only computer programmers.

However, it is important to mention that recently, several conferences and journals explicitly require the source code as well as all the needed materials to be publicly available upon the publication of the research; which pushes forward the open access and open source movements. For example, within the computational biology community, a considerable effort is being put into projects such as BioPerl [34], BioPython[35], etc., and dedicated groups and conferences such as BOSC[36]/Open Bioinformatics Foundation[37].

Yet, making software open source highly adopted by both communities, it is not enough. Researchers are encouraged to opt and endorse practices that support already developed software. In addition, scientists are encouraged to make software programs easily installable for non-professionals. Computational biology community recently addressed this issue by creating a package manager specializing in computational biology software, called BioConda[38], and a workflow management system, called Snakemake[39]. Still with the same idea aiming at providing and endorsing shared practices, the synthetic biology community proposed the Synthetic Biology Open Language (SBOL) [118] – that provides a community standard for communicating designs in synthetic biology – and the systems biology markup language (SBML) [155,154] – used for representation and exchange of biochemical network models.

Of course, the process of mathematical modeling requires a team that can understand both biology and either mathematics, computer science, or statistics. A breadth of mathematics, computer science, or statistics knowledge is needed to

[29] https://git-scm.com/
[30] https://fossil-scm.org/
[31] http://veracity-scm.com/
[32] https://www.mercurial-scm.org/
[33] https://www.monotone.ca/
[34] https://bioperl.org/
[35] https://biopython.org/
[36] https://www.open-bio.org/wiki/BOSC
[37] https://www.open-bio.org/
[38] https://bioconda.github.io/
[39] https://snakemake.readthedocs.io

guide them to a formulation that is mathematically rigor, while an understanding of the biology is needed to guide the team to a formulation that performs well in experimental evaluations. However, creating such teams is by itself another challenging goal due to differences in the scientific cultures of these various disciplines. In order to solve this challenging goal, communities from different disciplines are encouraged to move in the direction of each other. For example, biologists need to recognize the usefulness of mathematical rigidity of problem formulation with which statistician, mathematician, and computer science people usually work. Moreover, biologists need to gain their knowledge of statistics and mathematics and to acquire some skills in programming.

On the other hand, it will be a great addendum if mathematicians, statisticians, and computer scientists learn "deeper" biology; rather than the basic concepts. They need to be aware of the possible complexities that may arise during data generation processes. In addition, it will be very promising, if mathematical, statistical, and computer science communities continue improving skills on how to work in big collaborative projects containing hundred of researchers. Based on this, a stronger collaboration is required between these communities. The lack of such collaborative aspect as well as the lack of a strong and sufficient or constructive communication between the two communities in the past can be clearly viewed via this gap between the biological computation, computational biology, and mathematical biology disciplines. While our paper addresses only the goal of seeking reasons for the gap between biological computation and computational biology, the existence of the gap between computational biology and mathematical biology remains rather mysterious and requires additional discussion in the future between these two communities.

## 5 Conclusions

In this paper, we presented a review on biological computation and various important sectors of computational biology, discussed the interface between these two research areas and finally we identified a set of challenges for better synergies between biology and computation. We aimed at presenting how various computational techniques have been inspired from the nature, and, on the other hand, how computational techniques and mathematical modeling have been used to solve various biological questions. We presented a reasonably comprehensive review on biological computation, showing the tremendous impact and contribution this nature has on developing computational techniques. On the other hand, since computational biology is a huge interdisciplinary field covering numerous topics and areas in the interface of biology and computation, our goal was to highlight only a few of them with an emphasis on describing the application and impact of mathematics, algorithms and statistical modeling. Note that we did not aim for providing any suggestions or guidance for the researchers in choosing various techniques to solve different problems.

Research in biological systems and computational methods are converging and significantly contributing towards the advancement of science and technology. We believe that there is still a lot of room for improvement in the integration of computational techniques and life sciences. Although many algorithms have been developed using the concept from biological systems (as outlined in Section 2),

they are not necessarily being improved by incorporating more detailed knowledge of biological systems. For example, neural networks and genetic algorithms are undoubtedly two of the most important computational techniques based on biological processes. However, they only capture the high level understanding of the human brain and DNA sequence evolution, respectively. Taking a closer look at these, it would be really interesting to see how more detailed knowledge about the human brain and DNA sequence evolution can be incorporated to make further improvements. As our knowledge on various biological systems are getting better day by day, we should continue improving the bio-inspired techniques as we know more and more about the corresponding biological systems.

Similarly, the field of computational biology may be benefited by using biological insights. Currently various standard algorithmic techniques are being used to handle a wide range of biological problems. However, we have not seen much notable improvements of the existing algorithmic techniques and data structures by incorporating biological insights. We believe that it is possible and would be very interesting to develop novel data structures, programming languages, database systems, operating systems, fast and scalable algorithms for big data, etc. by looking into nature. For instance, looking into how and in what format human brains store and analyze data, can help in the design of such systems. In such a way, algorithmic systems biology can contribute to the advancement of computer science in a broad way.

Research in biological computation and computational biology is highly interdisciplinary which requires close collaboration between biologists and computer scientists. While experts from these two fields have been working together and making great strides towards the advancement of science and technology, we believe more involved collaboration is required. This sort of collaborative efforts will introduce a nice balance between the "wet" lab and "dry" lab research which will ultimately result in an improved understanding of biological processes as well as fast and accurate computational techniques. Therefore, along with others [114, 22,271,293,261,307], we are advocating for the need of developing advanced and appropriate computational tools to better understand and model life sciences, as well as the need for a deeper integration of computer science in biology.

## Acknowledgement

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 702527. Authors would also like to thank Professor Stephen Smale, Fields Medal awardee, who mentored the workshop organized by the first author, Dr. Zaineb Chelly Dagdia, at the $5^{th}$ Heidelberg Laureate Forum. The outcome of an analysis that has been done during this workshop presents the current research manuscript. Additional thanks to all organizers of the $5^{th}$ Heidelberg Laureate Forum, to the Heidelberg Institute for Theoretical Studies, Mathematisches Forschungszentrum Oberwolfach and Schloss Dagstuhl - Leibniz Center for Informatics, and to the Heidelberg Laureate Forum Foundation's Scientific Committee.

## References


1. Scientists use virus to trace assault suspect. ABC News (2006). URL https://abcnews.go.com/Technology/story?id=97856&page=1
2. Aganezov, S., Sitdykova, N., Alekseyev, M.A.: Scaffold assembly based on genome rearrangement analysis. Computational Biology and Chemistry **57**, 46 – 53 (2015). DOI https://doi.org/10.1016/j.compbiolchem.2015.02.005. URL http://www.sciencedirect.com/science/article/pii/S1476927115000225. 13th Asia Pacifi c Bioinformatics Conference, HsinChu, Taiwan, 21-23 January 2015
3. Aganezov, S., Sitdykova, N., Alekseyev, M.A., Consortium, A., et al.: Scaffold assembly based on genome rearrangement analysis. Computational biology and chemistry **57**, 46–53 (2015)
4. Aickelin, U., Dasgupta, D.: Artificial immune systems. In: Search methodologies, pp. 375–399. Springer (2005)
5. Alba, E.: Parallel evolutionary computations, vol. 22. springer (2006)
6. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: Integrins. In: Molecular Biology of the Cell. 4th edition. Garland Science (2002)
7. Alekseyev, M., Pevzner, P.A.: Breakpoint graphs and ancestral genome reconstructions. Genome research pp. gr–082784 (2009)
8. Alekseyev, M.A., Pevzner, P.A.: Whole genome duplications, multi-break rearrangements, and genome halving problem. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), pp. 665–679. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2007)
9. Alekseyev, M.A., Pevzner, P.A.: Multi-break rearrangements and chromosomal evolution. Theoretical Computer Science **395**(2), 193–202 (2008). DOI 10.1016/j.tcs.2008.01.013
10. Alexeev, N., Alekseyev, M.A.: Estimation of the true evolutionary distance under the fragile breakage model. BMC Genomics **18**(4), 356 (2017). DOI 10.1186/s12864-017-3733-3. URL https://doi.org/10.1186/s12864-017-3733-3
11. Alic, A.S., Ruzafa, D., Dopazo, J., Blanquer, I.: Objective review of de novo stand-alone error correction methods for NGS data. Wiley Interdisciplinary Reviews: Computational Molecular Science **6**(2), 111–146 (2016)
12. Ané, C., Larget, B., Baum, D.A., Smith, S.D., Rokas, A.: Bayesian estimation of concordance among gene trees. Molecular Biology and Evolution **24**, 412–426 (2007)
13. Angermueller, C., Pärnamaa, T., Parts, L., Stegle, O.: Deep learning for computational biology. Molecular systems biology **12**(7) (2016)
14. Angermueller, C., Pärnamaa, T., Parts, L., Stegle, O.: Deep learning for computational biology. Molecular systems biology **12**(7), 878 (2016)
15. Anselmetti, Y., Luhmann, N., Bérard, S., Tannier, E., Chauve, C.: Comparative Methods for Reconstructing Ancient Genome Organization, pp. 343–362. Springer New York, New York, NY (2018). DOI 10.1007/978-1-4939-7463-4_13. URL https://doi.org/10.1007/978-1-4939-7463-4_13
16. Avdeyev, P., Jiang, S., Aganezov, S., Hu, F., Alekseyev, M.A.: Reconstruction of ancestral genomes in presence of gene gain and loss. Journal of Computational Biology **23**(3), 150–164 (2016). DOI 10.1089/cmb.2015.0160
17. Avdeyev, P. and Alexeev, N. and Rong, Y. and Alekseyev, M. A.: A Unified ILP Framework for Genome Median, Halving, and Aliquoting Problems Under DCJ. In: J. Meidanis, L. Nakhleh (eds.) Proceedings of 15th International Workshop on Comparative Genomics (RECOMB-CG), *Lecture Notes in Computer Science*, vol. 10562, pp. 156–178 (2017)
18. Bafna, V., Pevzner, P.: Genome rearrangements and sorting by reversals. SIAM Journal on Computing **25**(2), 272–289 (1996). DOI 10.1137/S0097539793250627. URL https://doi.org/10.1137/S0097539793250627
19. Bankevich, A., Nurk, S., Antipov, D., Gurevich, A.A., Dvorkin, M., Kulikov, A.S., Lesin, V.M., Nikolenko, S.I., Pham, S., Prjibelski, A.D., et al.: Spades: a new genome assembly algorithm and its applications to single-cell sequencing. Journal of computational biology **19**(5), 455–477 (2012)
20. Bao, E., Jiang, T., Girke, T.: Aligngraph: algorithm for secondary de novo genome assembly guided by closely related references. Bioinformatics **30**(12), i319–i328 (2014). DOI 10.1093/bioinformatics/btu291. URL http://dx.doi.org/10.1093/bioinformatics/btu291

21. Bartels, D., Kespohl, S., Albaum, S., Drüke, T., Goesmann, A., Herold, J., Kaiser, O., Pühler, A., Pfeiffer, F., Raddatz, G., et al.: Baccardi—a tool for the validation of genomic assemblies, assisting genome finishing and intergenome comparison. Bioinformatics **21**(7), 853–859 (2004)
22. Bartocci, E., Lió, P.: Computational modeling, formal analysis, and tools for systems biology. PLoS computational biology **12**(1), e1004591 (2016)
23. Bashir, A., Klammer, A.A., Robins, W.P., Chin, C.S., Webster, D., Paxinos, E., Hsu, D., Ashby, M., Wang, S., Peluso, P., et al.: A hybrid approach for the automated finishing of bacterial genomes. Nature Biotechnology **30**(7), 701–707 (2012)
24. Bayzid, M.S.: Estimating species trees from gene trees despite gene tree incongruence under realistic model conditions. Ph.D. thesis (2016)
25. Bayzid, M.S., Mirarab, S., Warnow, T.: Inferring optimal species trees under gene duplication and loss. In: Proc. of Pacific Symposium on Biocomputing (PSB), vol. 18, pp. 250–261 (2013)
26. Bayzid, M.S., Warnow, T.: Naive binning improves phylogenomic analyses. Bioinformatics **29**(18), 2277–2284 (2013)
27. Bayzid, M.S., Warnow, T.: Gene tree parsimony for incomplete gene trees: addressing true biological loss. Algorithms for Molecular Biology **13**, 1 (2018)
28. Beller, T., Ohlebusch, E.: Efficient construction of a compressed de bruijn graph for pan-genome analysis. In: Annual Symposium on Combinatorial Pattern Matching, pp. 40–51. Springer (2015)
29. Below, N.C.A.L.: Simulated annealing and boltzmann machines a stochastic approach to combinatorial optimization and neural computing (1989)
30. Ben-Bassat, I., Chor, B.: String graph construction using incremental hashing. Bioinformatics **30**(24), 3515–3523 (2014)
31. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: International Workshop on Algorithms in Bioinformatics, pp. 163–173. Springer (2006)
32. Berlin, K., Koren, S., Chin, C.S., Drake, J.P., Landolin, J.M., Phillippy, A.M.: Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nature biotechnology **33**(6), 623 (2015)
33. Bickhart, D.M., Rosen, B.D., Koren, S., Sayre, B.L., Hastie, A.R., Chan, S., Lee, J., Lam, E.T., Liachko, I., Sullivan, S.T., et al.: Single-molecule sequencing and chromatin conformation capture enable de novo reference assembly of the domestic goat genome. Nature genetics **49**(4), 643 (2017)
34. Biller, P., Guéguen, L., Knibbe, C., Tannier, E.: Breaking good: Accounting for fragility of genomic regions in rearrangement distance estimation. Genome Biology and Evolution **8**(5), 1427–1439 (2016). DOI 10.1093/gbe/evw083. URL `http://dx.doi.org/10.1093/gbe/evw083`
35. Bitam, S., Batouche, M., Talbi, E.g.: A survey on bee colony algorithms. In: Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, pp. 1–8. IEEE (2010)
36. Boetzer, M., Henkel, C.V., Jansen, H.J., Butler, D., Pirovano, W.: Scaffolding preassembled contigs using SSPACE. Bioinformatics **27**(4), 578–579 (2011)
37. Boetzer, M., Pirovano, W.: SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information. BMC Bioinformatics **15**, 211 (2014)
38. Bonabeau, E., Marco, D.d.R.D.F., Dorigo, M., Théraulaz, G., Theraulaz, G., et al.: Swarm intelligence: from natural to artificial systems. 1. Oxford university press (1999)
39. Bosi, E., Donati, B., Galardini, M., Brunetti, S., Sagot, M.F., Lió, P., Crescenzi, P., Fani, R., Fondi, M.: Medusa: a multi-draft based scaffolder. Bioinformatics **31**(15), 2443–2451 (2015)
40. Bourlard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biological cybernetics **59**(4-5), 291–294 (1988)
41. Bourque, G., Pevzner, P.A.: Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome research **12**(1), 26–36 (2002)
42. Boussau, B., Szöllősi, G.J., Duret, L., Gouy, M., Tannier, E., Daubin, V.: Genome-scale coestimation of species and gene trees. Genome research **23**(2), 323–330 (2013)
43. Boutillier, P., Maasha, M., Li, X., Medina-Abarca, H.F., Krivine, J., Feret, J., Cristescu, I., Forbes, A.G., Fontana, W.: The kappa platform for rule-based modeling. Bioinformatics **34**(13), i583–i592 (2018)
44. Braga, M.D., Stoye, J.: The solution space of sorting by DCJ. Journal of Computational Biology **17**(9), 1145–1165 (2010)

45. Broomhead, D.S., Lowe, D.: Radial basis functions, multi-variable functional interpolation and adaptive networks. Tech. rep., Royal Signals and Radar Establishment Malvern (United Kingdom) (1988)
46. Buniello, A., MacArthur, J.A.L., Cerezo, M., Harris, L.W., Hayhurst, J., Malangone, C., McMahon, A., Morales, J., Mountjoy, E., Sollis, E., et al.: The nhgri-ebi gwas catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. Nucleic acids research **47**(D1), D1005–D1012 (2019)
47. Burnet, S.F.M., et al.: The clonal selection theory of acquired immunity (1959)
48. Burton, J.N., Adey, A., Patwardhan, R.P., Qiu, R., Kitzman, J.O., Shendure, J.: Chromosome-scale scaffolding of de novo genome assemblies based on chromatin interactions. Nature Biotechnology **31**(12), 1119–1125 (2013)
49. Bush, R.M., Bender, C.A., Subbarao, K., Cox, N.J., Fitch, W.M.: Predicting the evolution of human influenza a. Science **286**(5446), 1921–1925 (1999)
50. Bush, W.S., Moore, J.H.: Genome-wide association studies. PLoS Comput Biol **8**(12), e1002822 (2012)
51. Butler, J., MacCallum, I., Kleber, M., Shlyakhter, I.A., Belmonte, M.K., Lander, E.S., Nusbaum, C., Jaffe, D.B.: Allpaths: de novo assembly of whole-genome shotgun microreads. Genome research (2008)
52. Cao, X., Qiao, H., Xu, Y.: Negative selection based immune optimization. Advances in Engineering Software **38**(10), 649–656 (2007)
53. Cazaux, B., Lecroq, T., Rivals, E.: From indexing data structures to de bruijn graphs. In: Symposium on Combinatorial Pattern Matching, pp. 89–99. Springer (2014)
54. Chaisson, M.J., Pevzner, P.A.: Short read fragment assembly of bacterial genomes. Genome research **18**(2), 000–000 (2007)
55. Chambers, L.D.: The practical handbook of genetic algorithms: applications. Chapman and Hall/CRC (2000)
56. Chaudhary, R., Bansal, M.S., Wehe, A., Fernández-Baca, D., Eulenstein, O.: iGTP: a software package for large-scale gene tree parsimony analysis. BMC Bioinformatics **1**(1), 574 (2010)
57. Chauve, C., Gavranovic, H., Ouangraoua, A., Tannier, E.: Yeast ancestral genome reconstructions: the possibilities of computational methods ii. Journal of Computational Biology **17**(9), 1097–1112 (2010)
58. Chauve, C., Ponty, Y., Zanetti, J.P.P.: Evolution of genes neighborhood within reconciled phylogenies: an ensemble approach. BMC bioinformatics **16**(19), S6 (2015)
59. Chauve, C., Tannier, E.: A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genomes. PLoS computational biology **4**(11), e1000234 (2008)
60. Chelly, Z., Elouedi, Z.: A survey of the dendritic cell algorithm. Knowledge and Information Systems **48**(3), 505–535 (2016)
61. Ching, T., Himmelstein, D.S., Beaulieu-Jones, B.K., Kalinin, A.A., Do, B.T., Way, G.P., Ferrero, E., Agapow, P.M., Zietz, M., Hoffman, M.M., et al.: Opportunities and obstacles for deep learning in biology and medicine. Journal of The Royal Society Interface **15**(141), 20170387 (2018)
62. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
63. Clerc, M.: Particle swarm optimization, vol. 93. John Wiley & Sons (2010)
64. Coello, C.A.C., Lamont, G.B.: Applications of multi-objective evolutionary algorithms, vol. 1. World Scientific (2004)
65. Collins, F.S., Varmus, H.: A new initiative on precision medicine. New England Journal of Medicine **372**(9), 793–795 (2015)
66. Compeau, P., Pevzner, P.: Bioinformatics Algorithms: An Active Learning Approach. La Jolla. CA: Active Learning Publishers (2018)
67. Consortium, .G.P., et al.: A map of human genome variation from population-scale sequencing. Nature **467**(7319), 1061 (2010)
68. Consortium, .G.P., et al.: An integrated map of genetic variation from 1,092 human genomes. Nature **491**(7422), 56 (2012)
69. Consortium, .G.P., et al.: A global reference for human genetic variation. Nature **526**(7571), 68 (2015)
70. Consortium, I.H., et al.: The international hapmap project. Nature **426**(6968), 789 (2003)
71. Consortium, I.H., et al.: A haplotype map of the human genome. Nature **437**(7063), 1299 (2005)

72. Consortium, I.H.G.S., et al.: Initial sequencing and analysis of the human genome. Nature **409**(6822), 860 (2001)
73. Consortium, W.T.C.C., et al.: Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. Nature **447**(7145), 661 (2007)
74. Conway, T.C., Bromage, A.J.: Succinct data structures for assembling large genomes. Bioinformatics **27**(4), 479–486 (2011)
75. Crisp, M.D., Trewick, S.A., Cook, L.G.: Hypothesis testing in biogeography. Trends in ecology & evolution **26**(2), 66–72 (2011)
76. Dagdia, Z.C.: A distributed dendritic cell algorithm for big data. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 103–104 (2018)
77. Dagdia, Z.C.: A scalable and distributed dendritic cell algorithm for big data classification. Swarm and Evolutionary Computation **50**, 100432 (2019)
78. Dalke, K.: In court, scientists map a murder weapon. Genome News Network (2003). URL `http://www.genomenewsnetwork.org/articles/01_03/hiv.shtml`
79. Darwin, C.: On the origin of species, 1859. Routledge (2004)
80. Dasgupta, D., Michalewicz, Z.: Evolutionary algorithms in engineering applications. Springer Science & Business Media (2013)
81. Dayarian, A., Michael, T.P., Sengupta, A.M.: SOPRA: Scaffolding algorithm for paired reads via statistical optimization. BMC Bioinformatics **11**, 345 (2010)
82. De Castro, L.N., Timmis, J.: Artificial immune systems: a new computational intelligence approach. Springer Science & Business Media (2002)
83. De Jong, K.: Genetic algorithms: a 30 year perspective. Perspectives on Adaptation in Natural and Artificial Systems **11** (2005)
84. De Jong, K.A., Spears, W.M.: An analysis of the interacting roles of population size and crossover in genetic algorithms. In: International Conference on Parallel Problem Solving from Nature, pp. 38–47. Springer (1990)
85. Deb, K.: Multi-objective optimization using evolutionary algorithms, vol. 16. John Wiley & Sons (2001)
86. DeGiorgio, M., Degnan, J.H.: Fast and consistent estimation of species trees using supermatrix rooted triples. Molecular Biology and Evolution **27**(3), 552–569 (2010)
87. Degnan, J.H., Rosenberg, N.A.: Discordance of species trees with their most likely gene trees. PLoS Genetics **2**, 762 – 768 (2006)
88. Degnan, J.H., Salter, L.A.: Gene tree distributions under the coalescent process. Evolution : International Journal of Organic Evolution **59**(1), 24–37 (2005). URL `http://view.ncbi.nlm.nih.gov/pubmed/15792224`
89. Dinh, H., Rajasekaran, S.: A memory-efficient data structure representing exact-match overlap graphs with application for next-generation dna assembly. Bioinformatics **27**(14), 1901–1907 (2011)
90. Dobzhansky, T.: Nothing in biology makes sense except in the light of evolution. The american biology teacher **75**(2), 87–91 (2013)
91. Dobzhansky, T., Sturtevant, A.H.: Inversions in the chromosomes of drosophila pseudoobscura. Genetics **23**(1), 28 (1938)
92. Dole, M., Mack, L.L., Hines, R.L., Mobley, R.C., Ferguson, L.D., Alice, M.B.: Molecular beams of macroions. The Journal of Chemical Physics **49**(5), 2240–2249 (1968). DOI 10.1063/1.1670391. URL `https://doi.org/10.1063/1.1670391`
93. Dorigo, M., Di Caro, G.: Ant colony optimization: a new meta-heuristic. In: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, vol. 2, pp. 1470–1477. IEEE (1999)
94. Dorigo, M., Stützle, T.: The ant colony optimization metaheuristic: Algorithms, applications, and advances. In: Handbook of metaheuristics, pp. 250–285. Springer (2003)
95. Drummond, A.J., Rambaut, A.: Beast: Bayesian evolutionary analysis by sampling trees. BMC evolutionary biology **7**(1), 214 (2007)
96. Eberbach, E.: Toward a theory of evolutionary computation. BioSystems **82**(1), 1–19 (2005)
97. Edgar, R.C.: Muscle: multiple sequence alignment with high accuracy and high throughput. Nucleic acids research **32**(5), 1792–1797 (2004)
98. Edman, P., Begg, G.: A protein sequenator. European Journal of Biochemistry **1**(1), 80–91 (1967)
99. Edwards, S.V., Liu, L., Pearl, D.K.: High-resolution species trees without concatenation. Proceedings of the National Academy of Sciences **104**(14), 5936–5941 (2007)

100. Eiben, A.E., Smith, J.E., et al.: Introduction to evolutionary computing, vol. 53. Springer (2003)
101. El-Metwally, S., Hamza, T., Zakaria, M., Helmy, M.: Next-generation sequence assembly: four stages of data processing and computational challenges. PLoS computational biology **9**(12), e1003345 (2013)
102. Ellis, L.L., Huang, W., Quinn, A.M., Ahuja, A., Alfrejd, B., Gomez, F.E., Hjelmen, C.E., Moore, K.L., Mackay, T.F., Johnston, J.S., et al.: Intrapopulation genome size variation in d. melanogaster reflects life history variation and plasticity. PLoS genetics **10**(7), e1004522 (2014)
103. Elman, J.L.: Finding structure in time. Cognitive science **14**(2), 179–211 (1990)
104. Eusuff, M., Lansey, K., Pasha, F.: Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Engineering optimization **38**(2), 129–154 (2006)
105. Fadista, J., Manning, A.K., Florez, J.C., Groop, L.: The (in) famous gwas p-value threshold revisited and updated for low-frequency variants. European Journal of Human Genetics **24**(8), 1202–1205 (2016)
106. Fang, C., Shang, Y., Xu, D.: Mufold-ss: New deep inception-inside-inception networks for protein secondary structure prediction. Proteins: Structure, Function, and Bioinformatics **86**(5), 592–598 (2018)
107. Feijão, P.: Reconstruction of ancestral gene orders using intermediate genomes. BMC bioinformatics **16**(Suppl 14), S3 (2015)
108. Feijão, P., Araujo, E.: Fast ancestral gene order reconstruction of genomes with unequal gene content. BMC Bioinformatics **17**(14), 413 (2016)
109. Feijão, P., Meidanis, J.: Scj: A variant of breakpoint distance for which sorting, genome median and genome halving problems are easy. In: S.L. Salzberg, T. Warnow (eds.) Algorithms in Bioinformatics, pp. 85–96. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
110. Feijao, P., Meidanis, J.: Scj: A breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans. Comput. Biol. Bioinformatics **8**(5), 1318–1329 (2011). DOI 10.1109/TCBB.2011.34. URL `http://dx.doi.org/10.1109/TCBB.2011.34`
111. Feng, B., Lin, Y., Zhou, L., Guo, Y., Friedman, R., Xia, R., Hu, F., Liu, C., Tang, J.: Reconstructing yeasts phylogenies and ancestors from whole genome data. Scientific Reports **7**(1), 15209 (2017)
112. Fenn, J.B., Mann, M., Meng, C.K., Wong, S.F., Whitehouse, C.M.: Electrospray ionization for mass spectrometry of large biomolecules. Science **246**(4926), 64–71 (1989)
113. Fertin, G., Labarre, A., Rusu, I., Vialette, S., Tannier, E.: Combinatorics of genome rearrangements. MIT press (2009)
114. Fisher, J., Henzinger, T.A.: Executable cell biology. Nature biotechnology **25**(11), 1239 (2007)
115. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial intelligence through simulated evolution (1966)
116. Freedman, M.L., Reich, D., Penney, K.L., McDonald, G.J., Mignault, A.A., Patterson, N., Gabriel, S.B., Topol, E.J., Smoller, J.W., Pato, C.N., et al.: Assessing the impact of population stratification on genetic association studies. Nature genetics **36**(4), 388–393 (2004)
117. Gagnon, Y., Blanchette, M., El-Mabrouk, N.: A flexible ancestral genome reconstruction method based on gapped adjacencies. In: BMC bioinformatics, vol. 13, p. S4. BioMed Central (2012)
118. Galdzicki, M., Clancy, K.P., Oberortner, E., Pocock, M., Quinn, J.Y., Rodriguez, C.A., Roehner, N., Wilson, M.L., Adam, L., Anderson, J.C., et al.: The synthetic biology open language (sbol) provides a community standard for communicating designs in synthetic biology. Nature biotechnology **32**(6), 545 (2014)
119. Gandomi, A.H., Yang, X.S., Alavi, A.H.: Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with computers **29**(1), 17–35 (2013)
120. Gaul, É., Blanchette, M.: Ordering partially assembled genomes using gene arrangements. In: RECOMB Workshop on Comparative Genomics, pp. 113–128. Springer (2006)
121. Gavranović, H., Chauve, C., Salse, J., Tannier, E.: Mapping ancestral genomes with massive gene loss: a matrix sandwich problem. Bioinformatics **27**(13), i257–i265 (2011)
122. Gawehn, E., Hiss, J.A., Schneider, G.: Deep learning in drug discovery. Molecular informatics **35**(1), 3–14 (2016)

123. Ghurye, J., Pop, M., Koren, S., Bickhart, D., Chin, C.S.: Scaffolding of long read assemblies using long range contact information. BMC genomics **18**(1), 527 (2017)
124. Gibbs, R.A.: The human genome project changed everything. Nature Reviews Genetics pp. 1–2 (2020)
125. Gogarten, S.M., Bhangale, T., Conomos, M.P., Laurie, C.A., McHugh, C.P., Painter, I., Zheng, X., Crosslin, D.R., Levine, D., Lumley, T., et al.: Gwastools: an r/bioconductor package for quality control and analysis of genome-wide association studies. Bioinformatics **28**(24), 3329–3331 (2012)
126. Gonnella, G., Kurtz, S.: Readjoiner: a fast and memory efficient string graph-based sequence assembler. BMC bioinformatics **13**(1), 82 (2012)
127. González, F.A., Dasgupta, D.: Anomaly detection using real-valued negative selection. Genetic Programming and Evolvable Machines **4**(4), 383–403 (2003)
128. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
129. Goodman, M., Czelusniak, J., Moore, G., Romero-Herrera, E., Matsuda, G.: Fitting the gene lineage into its species lineage: a parsimony strategy illustrated by cladograms constructed from globin sequences. Systematic Zoology **28**(2), 132–163 (1979)
130. Goodwin, B.C.: Development and evolution. Journal of Theoretical Biology **97**(1), 43–55 (1982)
131. Goodwin, S., Gurtowski, J., Ethe-Sayers, S., Deshpande, P., Schatz, M.C., McCombie, W.R.: Oxford nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. Genome research (2015)
132. Goodwin, S., McPherson, J.D., McCombie, W.R.: Coming of age: ten years of next-generation sequencing technologies. Nature Reviews Genetics **17**(6), 333 (2016)
133. Górecki, P.: Reconciliation problems for duplication, loss and horizontal gene transfer. In: Proceedings of the 8th Annual International Conference on Computational Molecular Biology, pp. 316 – 325 (2004)
134. Green, P.: Against a whole-genome shotgun. Genome Research **7**(5), 410–417 (1997)
135. Greensmith, J., Aickelin, U., Twycross, J.: Articulation and clarification of the dendritic cell algorithm. In: International Conference on Artificial Immune Systems, pp. 404–417. Springer (2006)
136. Gritsenko, A.A., Nijkamp, J.F., Reinders, M.J., de Ridder, D.: GRASS: a generic algorithm for scaffolding next-generation sequencing assemblies. Bioinformatics **28**(11), 1429–1437 (2012)
137. Guigo, R., Muchnik, I., Smith, T.: Reconstruction of ancient molecular phylogeny. Molecular Phylogenetics and Evolution **6**(2), 189–213 (1996)
138. Hackl, T., Hedrich, R., Schultz, J., Förster, F.: proovread: large-scale high-accuracy pacbio correction through iterative short read consensus. Bioinformatics **30**(21), 3004–3011 (2014)
139. Hajela, P., Yoo, J.S.: Immune network modelling in design optimization. In: New ideas in optimization, pp. 203–216. McGraw-Hill Ltd., UK (1999)
140. Halanych, K.M., Goertzen, L.R.: Grand challenges in organismal biology: the need to develop both theory and resources. Integrative and Comparative Biology **49**(5), 475–479 (2009)
141. Hamer, D.H.: Beware the chopsticks gene. Molecular psychiatry **5**(1), 11–13 (2000)
142. Hannenhalli, S., Pevzner, P.A.: Towards a computational theory of genome rearrangements, pp. 184–202. Springer Berlin Heidelberg, Berlin, Heidelberg (1995). DOI 10.1007/BFb0015244. URL `https://doi.org/10.1007/BFb0015244`
143. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. J. ACM **46**(1), 1–27 (1999). DOI 10.1145/300515.300516. URL `http://doi.acm.org/10.1145/300515.300516`
144. Hartmann, T., Middendorf, M., Bernt, M.: Genome Rearrangement Analysis: Cut and Join Genome Rearrangements and Gene Cluster Preserving Approaches, pp. 261–289. Springer New York, New York, NY (2018). DOI 10.1007/978-1-4939-7463-4_9. URL `https://doi.org/10.1007/978-1-4939-7463-4_9`
145. Heled, J., Drummond, A.J.: Bayesian inference of species trees from multilocus data. Molecular Biology and Evolution **27**(3), 570–580 (2010)
146. Helgason, A., Yngvadottir, B., Hrafnkelsson, B., Gulcher, J., Stefánsson, K.: An icelandic example of the impact of population structure on association studies. Nature genetics **37**(1), 90–95 (2005)

147. van Hijum, S.A., Zomer, A.L., Kuipers, O.P., Kok, J.: Projector 2: contig mapping for efficient gap-closure of prokaryotic genome sequence assemblies. Nucleic acids research **33**(suppl_2), W560–W566 (2005)
148. Hirschhorn, J.N., Daly, M.J.: Genome-wide association studies for common diseases and complex traits. Nature reviews genetics **6**(2), 95–108 (2005)
149. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
150. Hofmeyr, S.A., Forrest, S.: Architecture for an artificial immune system. Evolutionary computation **8**(4), 443–473 (2000)
151. Holland, J., Goldberg, D.: Genetic algorithms in search, optimization and machine learning. Massachusetts: Addison-Wesley (1989)
152. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press (1992)
153. Hu, F., Zhou, J., Zhou, L., Tang, J.: Probabilistic reconstruction of ancestral gene orders with insertions and deletions. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) **11**(4), 667–672 (2014)
154. Hucka, M., Bergmann, F.T., Hoops, S., Keating, S.M., Sahle, S., Schaff, J.C., Smith, L.P., Wilkinson, D.J.: The systems biology markup language (sbml): language specification for level 3 version 1 core. Journal of integrative bioinformatics **12**(2), 382–549 (2015)
155. Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., Bray, D., Cornish-Bowden, A., et al.: The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. Bioinformatics **19**(4), 524–531 (2003)
156. Hudson, R.R.: Testing the constant-rate neutral allele model with protein sequence data. Evolution **37**(1), 203 – 217 (1983)
157. Huelsenbeck, J.P., Ronquist, F.: Mrbayes: Bayesian inference of phylogenetic trees. Bioinformatics **17**(8), 754–755 (2001)
158. Hunt, M., Newbold, C., Berriman, M., Otto, T.D.: A comprehensive evaluation of assembly scaffolding tools. Genome Biology **15**(3), 1–15 (2014)
159. Idury, R.M., Waterman, M.S.: A new algorithm for dna sequence assembly. Journal of computational biology **2**(2), 291–306 (1995)
160. Islam, M., Sarker, K., Das, T., Reaz, R., Bayzid, M.S.: Stelar: A statistically consistent coalescent-based species tree estimation method by maximizing triplet consistency. BMC Genomics **21**(1), 1–13 (2020)
161. Jain, M., Fiddes, I.T., Miga, K.H., Olsen, H.E., Paten, B., Akeson, M.: Improved data analysis for the minion nanopore sequencer. Nature methods **12**(4), 351 (2015)
162. Jain, M., Olsen, H.E., Paten, B., Akeson, M.: The oxford nanopore minion: delivery of nanopore sequencing to the genomics community. Genome Biology **17**(1), 239 (2016). DOI 10.1186/s13059-016-1103-0. URL `https://doi.org/10.1186/s13059-016-1103-0`
163. Janeway Jr, C.A.: The immune system evolved to discriminate infectious nonself from noninfectious self. Immunology today **13**(1), 11–16 (1992)
164. Ji, Z., Dasgupta, D.: Revisiting negative selection algorithms. Evolutionary Computation **15**(2), 223–251 (2007)
165. Jiménez, J., Skalic, M., Martinez-Rosell, G., De Fabritiis, G.: K deep: protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks. Journal of chemical information and modeling **58**(2), 287–296 (2018)
166. Jo, T., Hou, J., Eickholt, J., Cheng, J.: Improving protein fold recognition by deep learning networks. Scientific reports **5**, 17573 (2015)
167. Jones, B.R., Rajaraman, A., Tannier, E., Chauve, C.: Anges: reconstructing ancestral genomes maps. Bioinformatics **28**(18), 2388–2390 (2012)
168. Jones, N.C., Pevzner, P.A., Pevzner, P.: An introduction to bioinformatics algorithms. MIT press (2004)
169. Jones, N.C., Pevzner, P.A., Pevzner, P.: An introduction to bioinformatics algorithms. MIT press (2004)
170. Kamath, G.M., Shomorony, I., Xia, F., Courtade, T., David, N.T.: Hinge: long-read assembly achieves optimal repeat resolution. Genome research pp. gr–216465 (2017)
171. Kang, H.M., Sul, J.H., Service, S.K., Zaitlen, N.A., Kong, S.y., Freimer, N.B., Sabatti, C., Eskin, E., et al.: Variance component model to account for sample structure in genome-wide association studies. Nature genetics **42**(4), 348–354 (2010)
172. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (abc) algorithm. Applied soft computing **8**(1), 687–697 (2008)

173. Karas, M., Bachmann, D., Bahr, U., Hillenkamp, F.: Matrix-assisted ultraviolet laser desorption of non-volatile compounds. International journal of mass spectrometry and ion processes **78**, 53–68 (1987)
174. Karas, M., Bachmann, D., Hillenkamp, F.: Influence of the wavelength in high-irradiance ultraviolet laser desorption mass spectrometry of organic molecules. Analytical chemistry **57**(14), 2935–2939 (1985)
175. Karas, M., Hillenkamp, F.: Laser desorption ionization of proteins with molecular masses exceeding 10,000 daltons. Analytical chemistry **60**(20), 2299–2301 (1988)
176. Katoh, K., Misawa, K., Kuma, K.i., Miyata, T.: Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. Nucleic acids research **30**(14), 3059–3066 (2002)
177. Kececioglu, J.D., Myers, E.W.: Combinatorial algorithms for dna sequence assembly. Algorithmica **13**(1-2), 7 (1995)
178. Khan, W.A., Hamadneh, N.N., Tilahun, S.L., Ngnotchouye, J.: A review and comparative study of firefly algorithm and its modified versions. Optimization Algorithms-Methods and Applications pp. 281–313 (2016)
179. Kim, J., Larkin, D.M., Cai, Q., Zhang, Y., Ge, R.L., Auvil, L., Capitanu, B., Zhang, G., Lewin, H.A., Ma, J., et al.: Reference-assisted chromosome assembly. Proceedings of the National Academy of Sciences **110**(5), 1785–1790 (2013)
180. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
181. Kircher, M., Kelso, J.: High-throughput dna sequencing – concepts and limitations. BioEssays **32**(6), 524–536 (2010). DOI 10.1002/bies.200900181. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/bies.200900181`
182. Kohn, M., Högel, J., Vogel, W., Minich, P., Kehrer-Sawatzki, H., Graves, J.A., Hameister, H.: Reconstruction of a 450-my-old ancestral vertebrate protokaryotype. TRENDS in Genetics **22**(4), 203–210 (2006)
183. Kolmogorov, M., Raney, B., Paten, B., Pham, S.: Ragout—a reference-assisted assembly tool for bacterial genomes. Bioinformatics **30**(12), i302–i309 (2014). DOI 10.1093/bioinformatics/btu280. URL `http://dx.doi.org/10.1093/bioinformatics/btu280`
184. Koren, S., Treangen, T.J., Pop, M.: Bambus 2: scaffolding metagenomes. Bioinformatics **27**(21), 2964–2971 (2011)
185. Koren, S., Walenz, B.P., Berlin, K., Miller, J.R., Bergman, N.H., Phillippy, A.M.: Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Genome research pp. gr–215087 (2017)
186. Kosakovsky Pond, S.L., Posada, D., Gravenor, M.B., Woelk, C.H., Frost, S.D.: Gard: a genetic algorithm for recombination detection. Bioinformatics **22**(24), 3096–3098 (2006)
187. Koza, J.R.: Genetic Programming II, Automatic Discovery of Reusable Subprograms. MIT Press, Cambridge, MA (1992)
188. Krause, J., Cordeiro, J., Parpinelli, R.S., Lopes, H.S.: A survey of swarm algorithms applied to discrete optimization problems. In: Swarm Intelligence and Bio-Inspired Computation, pp. 169–191. Elsevier (2013)
189. Kubatko, L.S., Carstens, B.C., Knowles, L.L.: STEM: species tree estimation using maximum likelihood for gene trees under coalescence. Bioinformatics **25**(7), 971–973 (2009). DOI 10.1093/bioinformatics/btp079. URL `http://www.ncbi.nlm.nih.gov/pubmed/19211573`
190. Kuleshov, V., Snyder, M.P., Batzoglou, S.: Genome assembly from synthetic long read clouds. Bioinformatics **32**(12), i216–i224 (2016)
191. Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.: Deep convolutional inverse graphics network. In: Advances in Neural Information Processing Systems, pp. 2539–2547 (2015)
192. Kumar, S., Tamura, K., Nei, M.: Mega: molecular evolutionary genetics analysis software for microcomputers. Bioinformatics **10**(2), 189–191 (1994)
193. Lam, K.K., LaButti, K., Khalak, A., Tse, D.: FinisherSC: a repeat-aware tool for upgrading de novo assembly using long reads. Bioinformatics **31**(19), 3207–3209 (2015)
194. Lander, E.S., Schork, N.J.: Genetic dissection of complex traits. Science **265**(5181), 2037–2048 (1994)
195. Lander, E.S., Waterman, M.S.: Genomic mapping by fingerprinting random clones: a mathematical analysis. Genomics **2**(3), 231–239 (1988)
196. Larget, B., Kotha, S.K., Dewey, C.N., Ané, C.: BUCKy: Gene tree/species tree reconciliation with the Bayesian concordance analysis. Bioinformatics **26**(22), 2910–2911 (2010)

197. Lassmann, T., Frings, O., Sonnhammer, E.L.: Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features. Nucleic acids research **37**(3), 858–865 (2008)
198. Laver, T., Harrison, J., O'neill, P., Moore, K., Farbos, A., Paszkiewicz, K., Studholme, D.J.: Assessing the performance of the oxford nanopore technologies minion. Biomolecular detection and quantification **3**, 1–8 (2015)
199. Leaché, A.D., Rannala, B.: The accuracy of species tree estimation under simulation: a comparison of methods. Systematic Biology **60**(2), 126–137 (2011)
200. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
201. Lee, H., Gurtowski, J., Yoo, S., Marcus, S., McCombie, W.R., Schatz, M.: Error correction and assembly complexity of single molecule sequencing reads. BioRxiv p. 006395 (2014)
202. Lewis, P.O.: A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data. Molecular biology and evolution **15**(3), 277–283 (1998)
203. Li, H.: Exploring single-sample snp and indel calling with whole-genome de novo assembly. Bioinformatics **28**(14), 1838–1844 (2012)
204. Li, H.: Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. Bioinformatics **32**(14), 2103–2110 (2016)
205. Lin, D.Y., Tao, R., Kalsbeek, W.D., Zeng, D., Gonzalez II, F., Fernández-Rhodes, L., Graff, M., Koch, G.G., North, K.E., Heiss, G.: Genetic association analysis under complex survey sampling: the hispanic community health study/study of latinos. The American Journal of Human Genetics **95**(6), 675–688 (2014)
206. Lin, Y., Moret, B.M.: Estimating true evolutionary distances under the dcj model. Bioinformatics **24**(13), i114–i122 (2008). DOI 10.1093/bioinformatics/btn148. URL `http://dx.doi.org/10.1093/bioinformatics/btn148`
207. Lin, Y., Nurk, S., Pevzner, P.A.: What is the difference between the breakpoint graph and the de bruijn graph? BMC genomics **15**(6), S6 (2014)
208. Lin, Y., Yuan, J., Kolmogorov, M., Shen, M.W., Chaisson, M., Pevzner, P.A.: Assembly of long error-prone reads using de bruijn graphs. Proceedings of the National Academy of Sciences **113**(52), E8396–E8405 (2016)
209. Linder, C.R., Warnow, T.: An overview of phylogeny reconstruction (2001)
210. Liu, K., Raghavan, S., Nelesen, S., Linder, C.R., Warnow, T.: Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. Science **324**(5934), 1561–1564 (2009)
211. Liu, L.: BEST: Bayesian estimation of species trees under the coalescent model. Bioinformatics **24**(21), 2542–2543 (2008)
212. Liu, L., Li, Y., Li, S., Hu, N., He, Y., Pong, R., Lin, D., Lu, L., Law, M.: Comparison of next-generation sequencing systems. BioMed Research International **2012** (2012)
213. Liu, L., Yu, L.: Estimating species trees from unrooted gene trees. Systematic Biology **60**(5), 661–667 (2011). DOI 10.1093/sysbio/syr027
214. Liu, L., Yu, L., Edwards, S.V.: A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. BMC Evolutinary Biology **10**(1), 302 (2010)
215. Liu, L., Yu, L., Pearl, D.K., Edwards, S.V.: Estimating species phylogenies using coalescence times among sequences. Systematic Biology **58**(5), 468–477 (2009)
216. Liu, Y., Ye, Q., Wang, L., Peng, J.: Learning structural motif representations for efficient protein structure search. Bioinformatics **34**(17), i773–i780 (2018)
217. Lohmueller, K.E., Pearce, C.L., Pike, M., Lander, E.S., Hirschhorn, J.N.: Meta-analysis of genetic association studies supports a contribution of common variants to susceptibility to common disease. Nature genetics **33**(2), 177–182 (2003)
218. Loman, N.J., Quick, J., Simpson, J.T.: A complete bacterial genome assembled de novo using only nanopore sequencing data. Nature methods **12**(8), 733 (2015)
219. Lones, M.A.: Metaheuristics in nature-inspired algorithms. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 1419–1422. ACM (2014)
220. Löytynoja, A., Goldman, N.: An algorithm for progressive multiple alignment of sequences with insertions. Proceedings of the National academy of sciences of the United States of America **102**(30), 10557–10562 (2005)
221. Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., He, G., Chen, Y., Pan, Q., Liu, Y., Tang, J., Wu, G., Zhang, H., Shi, Y., Liu, Y., Yu, C., Wang, B., Lu, Y., Han, C., Cheung, D.W., Yiu, S.M., Peng, S., Xiaoqian, Z., Liu, G., Liao, X., Li, Y., Yang, H., Wang, J., Lam, T.W., Wang, J.: Soapdenovo2: an empirically improved memory-efficient short-read

de novo assembler. GigaScience **1**(1), 18 (2012). DOI 10.1186/2047-217X-1-18. URL https://doi.org/10.1186/2047-217X-1-18
222. Ma, J.: A probabilistic framework for inferring ancestral genomic orders. In: Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference On, pp. 179–184. IEEE (2010)
223. Ma, J., Zhang, L., Suh, B.B., Raney, B.J., Burhans, R.C., Kent, W.J., Blanchette, M., Haussler, D., Miller, W.: Reconstructing contiguous regions of an ancestral genome. Genome research **16**(11), 000–000 (2006)
224. Maddison, W.P.: Gene trees in species trees. Systematic Biology **46**(3), 523–536 (1997)
225. Madoui, M.A., Dossat, C., d'Agata, L., van Oeveren, J., van der Vossen, E., Aury, J.M.: MaGuS: a tool for quality assessment and scaffolding of genome assemblies with Whole Genome Profiling™Data. BMC Bioinformatics **17**, 115 (2016)
226. Madoui, M.A., Engelen, S., Cruaud, C., Belser, C., Bertrand, L., Alberti, A., Lemainque, A., Wincker, P., Aury, J.M.: Genome assembly using nanopore-guided long and error-free dna reads. BMC genomics **16**(1), 327 (2015)
227. Mägi, R., Morris, A.P.: Gwama: software for genome-wide association meta-analysis. BMC bioinformatics **11**(1), 288 (2010)
228. Maier, D.: The complexity of some problems on subsequences and supersequences. Journal of the ACM (JACM) **25**(2), 322–336 (1978)
229. Makarenkov, V., Kevorkov, D., Legendre, P.: Phylogenetic network construction approaches. In: Applied mycology and biotechnology, vol. 6, pp. 61–97. Elsevier (2006)
230. Marchini, J., Cardon, L.R., Phillips, M.S., Donnelly, P.: The effects of human population structure on large genetic association studies. Nature genetics **36**(5), 512–517 (2004)
231. Mardis, E.R.: Next-generation sequencing platforms. Annual review of analytical chemistry **6**, 287–303 (2013)
232. Mardis, E.R.: Dna sequencing technologies: 2006–2016. Nature protocols **12**(2), 213 (2017)
233. Marees, A.T., de Kluiver, H., Stringer, S., Vorspan, F., Curis, E., Marie-Claire, C., Derks, E.M.: A tutorial on conducting genome-wide association studies: Quality control and statistical analysis. International journal of methods in psychiatric research **27**(2), e1608 (2018)
234. Matzinger, P.: Essay 1: the danger model in its historical context. Scandinavian journal of immunology **54**(1-2), 4–9 (2001)
235. McCarthy, M.I., Abecasis, G.R., Cardon, L.R., Goldstein, D.B., Little, J., Ioannidis, J.P., Hirschhorn, J.N.: Genome-wide association studies for complex traits: consensus, uncertainty and challenges. Nature reviews genetics **9**(5), 356–369 (2008)
236. Medvedev, P.: Modeling biological problems in computer science: a case study in genome assembly. Briefings in bioinformatics (2017)
237. Medvedev, P., Georgiou, K., Myers, G., Brudno, M.: Computability of models for sequence assembly. In: International Workshop on Algorithms in Bioinformatics, pp. 289–301. Springer (2007)
238. Melsted, P., Pritchard, J.K.: Efficient counting of k-mers in dna sequences using a bloom filter. BMC Bioinformatics **12**(1), 333 (2011). DOI 10.1186/1471-2105-12-333. URL https://doi.org/10.1186/1471-2105-12-333
239. Mendelowitz, L., Pop, M.: Computational methods for optical mapping. GigaScience **3**(1), 33 (2014)
240. Metzker, M.L.: Sequencing technologies—the next generation. Nature reviews genetics **11**(1), 31 (2010)
241. Metzker, M.L., Mindell, D.P., Liu, X.M., Ptak, R.G., Gibbs, R.A., Hillis, D.M.: Molecular evidence of hiv-1 transmission in a criminal case. Proceedings of the National Academy of Sciences **99**(22), 14292–14297 (2002)
242. Meyer-Nieberg, S., Beyer, H.G.: Self-adaptation in evolutionary algorithms. In: Parameter setting in evolutionary algorithms, pp. 47–75. Springer (2007)
243. Miclotte, G., Heydari, M., Demeester, P., Rombauts, S., Van de Peer, Y., Audenaert, P., Fostier, J.: Jabba: hybrid error correction for long sequencing reads. Algorithms for Molecular Biology **11**(1), 10 (2016)
244. Miller, J.R., Koren, S., Sutton, G.: Assembly algorithms for next-generation sequencing data. Genomics **95**(6), 315–327 (2010)
245. Minkin, I., Patel, A., Kolmogorov, M., Vyahhi, N., Pham, S.: Sibelia: a scalable and comprehensive synteny block generation tool for closely related microbial genomes. In: International Workshop on Algorithms in Bioinformatics, pp. 215–229. Springer (2013)

246. Minkin, I., Pham, S., Medvedev, P.: Twopaco: An efficient algorithm to build the compacted de bruijn graph from many complete genomes. Bioinformatics **33**(24), 4024–4032 (2016)
247. Mirarab, S., Nguyen, N., Guo, S., Wang, L.S., Kim, J., Warnow, T.: Pasta: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences. Journal of Computational Biology **22**(5), 377–386 (2015)
248. Mirarab, S., Reaz, R., Bayzid, M.S., Zimmermann, T., Swenson, M.S., Warnow, T.: ASTRAL: genome-scale coalescent-based species tree estimation. Bioinformatics **30**(17), i541–i548 (2014)
249. Mirkin, B., Muchnik, I., Smith, T.: A biologically consistent model for comparing molecular phylogenies. Journal of Computational Biology **2**(4), 493–507 (1995)
250. Mittal, S., Nirwal, N., Sardana, H.: Enhanced artificial bees colony algorithm for traveling salesman problem. Journal of Advanced Computing and Communication Technologies **2**(2), 2347–2804 (2014)
251. Mossel, E., Roch, S.: Incomplete lineage sorting: consistent phylogeny estimation from multiple loci. IEEE/ACM Transactions on Computational Biology and Bioinformatics **7**(1), 166–171 (2011)
252. Muñoz, A., Zheng, C., Zhu, Q., Albert, V.A., Rounsley, S., Sankoff, D.: Scaffold filling, contig fusion and comparative gene order inference. BMC bioinformatics **11**(1), 304 (2010)
253. Myers, E.W.: Toward simplifying and accurately formulating fragment assembly. Journal of Computational Biology **2**(2), 275–290 (1995)
254. Myers, E.W.: The fragment assembly string graph. Bioinformatics **21**(suppl_2), ii79–ii85 (2005)
255. Myers, E.W., Sutton, G.G., Delcher, A.L., Dew, I.M., Fasulo, D.P., Flanigan, M.J., Kravitz, S.A., Mobarry, C.M., Reinert, K.H., Remington, K.A., et al.: A whole-genome assembly of drosophila. Science **287**(5461), 2196–2204 (2000)
256. Nagarajan, N., Pop, M.: Parametric complexity of sequence assembly: theory and applications to next generation sequencing. Journal of computational biology **16**(7), 897–908 (2009)
257. Nagarajan, N., Pop, M.: Sequence assembly demystified. Nature Reviews Genetics **14**(3), 157 (2013)
258. Nagarajan, N., Read, T.D., Pop, M.: Scaffolding and validation of bacterial genome assemblies using optical restriction maps. Bioinformatics **24**(10), 1229–1235 (2008)
259. Nakatani, Y., Takeda, H., Kohara, Y., Morishita, S.: Reconstruction of the vertebrate ancestral genome reveals dynamic genome reorganization in early vertebrates. Genome research **17**(9), 000–000 (2007)
260. Nakhleh, L., Sun, J., Warnow, T., Linder, C.R., Moret, B.M., Tholse, A.: Towards the development of computational tools for evaluating phylogenetic network reconstruction methods. In: Biocomputing 2003, pp. 315–326. World Scientific (2002)
261. Navlakha, S., Bar-Joseph, Z.: Algorithms in nature: the convergence of systems biology and computational thinking. Molecular systems biology **7**(1), 546 (2011)
262. Nayeem, M.A., Bayzid, M.S., Rahman, A.H., Shahriyar, R., Rahman, M.S.: A'phylogeny-aware'multi-objective optimization approach for computing msa. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 577–585. ACM (2019)
263. Neafsey, D.E., Waterhouse, R.M., Abai, M.R., Aganezov, S.S., Alekseyev, M.A., Allen, J.E., Amon, J., Arcà, B., Arensburger, P., Artemov, G., et al.: Highly evolvable malaria vectors: the genomes of 16 anopheles mosquitoes. Science **347**(6217), 1258522 (2015)
264. Nei, M.: Stochastic errors in DNA evolution and molecular phylogeny. Progress in Clinical and Biological Research **218**, 133 – 147 (1986)
265. Nei, M.: Molecular evolutionary genetics. Columbia University Press (1987)
266. Nguyen, N., Mirarab, S., Warnow, T.: MRL and SuperFine+MRL: new supertree methods. Algorithms for Molecular Biology **7**, 3 (2012)
267. Ning, Z., Cox, A.J., Mullikin, J.C.: Ssaha: a fast search method for large dna databases. Genome research **11**(10), 1725–1729 (2001)
268. Notredame, C., Higgins, D.G.: Saga: sequence alignment by genetic algorithm. Nucleic acids research **24**(8), 1515–1524 (1996)
269. Notredame, C., Higgins, D.G., Heringa, J.: T-coffee: a novel method for fast and accurate multiple sequence alignment1. Journal of molecular biology **302**(1), 205–217 (2000)
270. Notredame, C., O'Brien, E.A., Higgins, D.G.: Raga: Rna sequence alignment by genetic algorithm. Nucleic acids research **25**(22), 4570–4580 (1997)

271. Nurse, P.: Life, logic and information. Nature **454**(7203), 424 (2008)
272. O'Connor, R.E., Romanov, M.N., Kiazim, L.G., Barrett, P.M., Farré, M., Damas, J., Ferguson-Smith, M., Valenzuela, N., Larkin, D.M., Griffin, D.K.: Reconstruction of the diapsid ancestral genome permits chromosome evolution tracing in avian and non-avian dinosaurs. Nature communications **9**(1), 1883 (2018)
273. Page, R.: Genetree: comparing gene and species phylogenies using reconciled trees. Bioinformatics **14**(9), 819–820 (1998)
274. Page, R., Charleston, M.: From gene to organismal phylogeny: reconciled trees and the gene tree/species tree problem. Molecular Phylogentics and Evolution **7**(2), 231–240 (1997)
275. Page, R., Charleston, M.: Reconciled trees and incongruent gene and species trees. Mathematical Hierarchies in Biology **37**, 57–70 (1997)
276. Page, R.D.: Genes, organisms, and areas: the problem of multiple lineages. Systematic Biology **42**(1), 77–84 (1993)
277. Palmer, J.D., Herbon, L.A.: Plant mitochondrial dna evolved rapidly in structure, but slowly in sequence. Journal of Molecular Evolution **28**(1), 87–97 (1988). DOI 10.1007/BF02143500. URL `https://doi.org/10.1007/BF02143500`
278. Park, Y., Kellis, M.: Deep learning for regulatory genomics. Nature biotechnology **33**(8), 825 (2015)
279. Patané, J.S.L., Martins, J., Setubal, J.C.: Phylogenomics, pp. 103–187. Springer New York, New York, NY (2018). DOI 10.1007/978-1-4939-7463-4_5. URL `https://doi.org/10.1007/978-1-4939-7463-4_5`
280. Patterson, M., Szöllősi, G., Daubin, V., Tannier, E.: Lateral gene transfer, rearrangement, reconciliation. In: BMC bioinformatics, vol. 14, p. S4. BioMed Central (2013)
281. Patterson, N., Price, A.L., Reich, D.: Population structure and eigenanalysis. PLoS genet **2**(12), e190 (2006)
282. Pavlidis, P., Alachiotis, N.: A survey of methods and tools to detect recent and strong positive selection. Journal of Biological Research-Thessaloniki **24**(1), 1–17 (2017)
283. Pe'er, I., Yelensky, R., Altshuler, D., Daly, M.J.: Estimation of the multiple testing burden for genomewide association studies of nearly all common variants. Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society **32**(4), 381–385 (2008)
284. Perrin, A., Varré, J.S., Blanquart, S., Ouangraoua, A.: Procars: Progressive reconstruction of ancestral gene orders. BMC genomics **16**(5), S6 (2015)
285. Pevzner, P.A.: 1-tuple dna sequencing: computer analysis. Journal of Biomolecular structure and dynamics **7**(1), 63–73 (1989)
286. Pevzner, P.A., Tang, H.: Fragment assembly with double-barreled data. Bioinformatics **17**(suppl_1), S225–S233 (2001)
287. Pevzner, P.A., Tang, H., Waterman, M.S.: An eulerian path approach to dna fragment assembly. Proceedings of the National Academy of Sciences **98**(17), 9748–9753 (2001)
288. Philippe, N., Legendre, M., Doutre, G., Couté, Y., Poirot, O., Lescot, M., Arslan, D., Seltzer, V., Bertaux, L., Bruley, C., et al.: Pandoraviruses: amoeba viruses with genomes up to 2.5 mb reaching that of parasitic eukaryotes. Science **341**(6143), 281–286 (2013)
289. Pop, M., Kosack, D.S., Salzberg, S.L.: Hierarchical scaffolding with bambus. Genome research **14**(1), 149–159 (2004)
290. Popescu, P., Hayes, H.: Techniques in animal cytogenetics. Springer Science & Business Media (2000)
291. Poplin, R., Chang, P.C., Alexander, D., Schwartz, S., Colthurst, T., Ku, A., Newburger, D., Dijamco, J., Nguyen, N., Afshar, P.T., et al.: A universal snp and small-indel variant caller using deep neural networks. Nature biotechnology **36**(10), 983–987 (2018)
292. Poultney, C., Chopra, S., Cun, Y.L., et al.: Efficient learning of sparse representations with an energy-based model. In: Advances in neural information processing systems, pp. 1137–1144 (2007)
293. Priami, C.: Algorithmic systems biology. Communications of the ACM **52**(5), 80–88 (2009)
294. Price, A.L., Patterson, N.J., Plenge, R.M., Weinblatt, M.E., Shadick, N.A., Reich, D.: Principal components analysis corrects for stratification in genome-wide association studies. Nature genetics **38**(8), 904–909 (2006)
295. Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M.A., Bender, D., Maller, J., Sklar, P., De Bakker, P.I., Daly, M.J., et al.: Plink: a tool set for whole-genome association and population-based linkage analyses. The American journal of human genetics **81**(3), 559–575 (2007)

296. Putnam, N.H., Butts, T., Ferrier, D.E., Furlong, R.F., Hellsten, U., Kawashima, T., Robinson-Rechavi, M., Shoguchi, E., Terry, A., Yu, J.K., et al.: The amphioxus genome and the evolution of the chordate karyotype. Nature **453**(7198), 1064 (2008)
297. Putnam, N.H., O'Connell, B.L., Stites, J.C., Rice, B.J., Blanchette, M., Calef, R., Troll, C.J., Fields, A., Hartley, P.D., Sugnet, C.W., et al.: Chromosome-scale shotgun assembly using an in vitro method for long-range linkage. Genome research (2016)
298. Quijano, N., Passino, K.M.: Honey bee social foraging algorithms for resource allocation, part i: Algorithm and theory. In: American Control Conference, 2007. ACC'07, pp. 3383–3388. IEEE (2007)
299. Räihä, K.J., Ukkonen, E.: The shortest common supersequence problem over binary alphabet is np-complete. Theoretical Computer Science **16**(2), 187–198 (1981)
300. Rechenberg, I.: " evolutionsstrategie-optimierung technischer systems nach prinzipien der biologischen evolution, stuttgart: Frommannholzboog, 1973. New York: John Wiley (1981)
301. Richter, D.C., Schuster, S.C., Huson, D.H.: Oslay: optimal syntenic layout of unfinished assemblies. Bioinformatics **23**(13), 1573–1579 (2007)
302. Rissman, A.I., Mau, B., Biehl, B.S., Darling, A.E., Glasner, J.D., Perna, N.T.: Reordering contigs of draft genomes using the mauve aligner. Bioinformatics **25**(16), 2071–2073 (2009)
303. Roberts, R.J., Carneiro, M.O., Schatz, M.C.: The advantages of smrt sequencing. Genome Biology **14**(6), 405 (2013). DOI 10.1186/gb-2013-14-6-405. URL https://doi.org/10.1186/gb-2013-14-6-405
304. Rosen, C.B., Rodriguez-Larrea, D., Bayley, H.: Single-molecule site-specific detection of protein phosphorylation with a nanopore. Nature biotechnology **32**(2), 179 (2014)
305. Rosenberg, N.: The probability of topological concordance of gene trees and species trees. Theoretical Population Biology **61**(2), 225–247 (2002). URL http://dx.doi.org/10.1006/tpbi.2001.1568
306. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review **65**(6), 386 (1958)
307. Rubinstein, A., Chor, B.: Computational thinking in life science education. PLoS computational biology **10**(11), e1003897 (2014)
308. Salmela, L., Rivals, E.: Lordec: accurate and efficient long read error correction. Bioinformatics **30**(24), 3506–3514 (2014)
309. Salse, J.: Ancestors of modern plant crops. Current Opinion in Plant Biology **30**, 134 – 142 (2016). DOI https://doi.org/10.1016/j.pbi.2016.02.005. URL http://www.sciencedirect.com/science/article/pii/S136952661630022X. SI: 30: Genome studies and molecular genetics
310. Sanger, F., Nicklen, S., Coulson, A.R.: Dna sequencing with chain-terminating inhibitors. Proceedings of the national academy of sciences **74**(12), 5463–5467 (1977)
311. Sankoff, D., Nadeau, J.H.: Comparative Genomics, pp. 3–7. Springer Netherlands, Dordrecht (2000). DOI 10.1007/978-94-011-4309-7_1. URL https://doi.org/10.1007/978-94-011-4309-7_1
312. Schalkoff, R.J.: Artificial neural networks, vol. 1. McGraw-Hill New York (1997)
313. Schatz, M.C., Delcher, A.L., Salzberg, S.L.: Assembly of large genomes using second-generation sequencing. Genome research pp. gr–101360 (2010)
314. Schirmer, M., Ijaz, U.Z., D'Amore, R., Hall, N., Sloan, W.T., Quince, C.: Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. Nucleic acids research **43**(6), e37–e37 (2015)
315. Secker, A., Freitas, A.A., Timmis, J.: A danger theory inspired approach to web mining. In: International Conference on Artificial Immune Systems, pp. 156–167. Springer (2003)
316. Sedlazeck, F.J., Lee, H., Darby, C.A., Schatz, M.C.: Piercing the dark matter: bioinformatics of long-range sequencing and mapping. Nature Reviews Genetics p. 1 (2018)
317. Seeley, T.D., Visscher, P.K., Passino, K.M.: Group decision making in honey bee swarms: When 10,000 bees go house hunting, how do they cooperatively choose their new nesting site? American Scientist **94**(3), 220–229 (2006)
318. Sievers, F., Wilm, A., Dineen, D., Gibson, T.J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., et al.: Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. Molecular systems biology **7**(1), 539 (2011)
319. Simpson, J.T.: Exploring genome characteristics and sequence quality without a reference. Bioinformatics **30**(9), 1228–1235 (2014)

320. Simpson, J.T., Durbin, R.: Efficient construction of an assembly string graph using the fm-index. Bioinformatics **26**(12), i367–i373 (2010)
321. Simpson, J.T., Durbin, R.: Efficient de novo assembly of large genomes using compressed data structures. Genome research **22**(3), 549–556 (2012)
322. Simpson, J.T., Pop, M.: The theory and practice of genome sequence assembly. Annual review of genomics and human genetics **16**, 153–172 (2015)
323. Simpson, J.T., Wong, K., Jackman, S.D., Schein, J.E., Jones, S.J., Birol, I.: Abyss: a parallel assembler for short read sequence data. Genome research pp. gr–089532 (2009)
324. Simpson, J.T., Workman, R.E., Zuzarte, P., David, M., Dursi, L., Timp, W.: Detecting dna cytosine methylation using nanopore sequencing. Nature methods **14**(4), 407—410 (2017). DOI 10.1038/nmeth.4184. URL https://doi.org/10.1038/nmeth.4184
325. Simpson, J.T., Workman, R.E., Zuzarte, P., David, M., Dursi, L., Timp, W.: Detecting dna cytosine methylation using nanopore sequencing. Nature methods **14**(4), 407 (2017)
326. Simpson, P.K.: Neural networks applications. IEEE Press (1997)
327. Slatkin, M.: Linkage disequilibrium—understanding the evolutionary past and mapping the medical future. Nature Reviews Genetics **9**(6), 477–485 (2008)
328. Soderlund, C., Bomhoff, M., Nelson, W.M.: Symap v3. 4: a turnkey synteny system with application to plant genomes. Nucleic acids research **39**(10), e68–e68 (2011)
329. Sohn, J.i., Nam, J.W.: The present and future of de novo whole-genome assembly. Briefings in bioinformatics **19**(1), 23–40 (2016)
330. Spencer, M., Eickholt, J., Cheng, J.: A deep learning network approach to ab initio protein secondary structure prediction. IEEE/ACM transactions on computational biology and bioinformatics (TCBB) **12**(1), 103–112 (2015)
331. Stamatakis, A.: An efficient program for phylogenetic inference using simulated annealing. In: 19th IEEE International Parallel and Distributed Processing Symposium, pp. 8–pp. IEEE (2005)
332. Stamatakis, A.: Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics **22**(21), 2688–2690 (2006)
333. Stamatakis, A., Ludwig, T., Meier, H.: Raxml-iii: a fast program for maximum likelihood-based inference of large phylogenetic trees. Bioinformatics **21**(4), 456–463 (2005)
334. Stephens, P.J., Greenman, C.D., Fu, B., Yang, F., Bignell, G.R., Mudie, L.J., Pleasance, E.D., Lau, K.W., Beare, D., Stebbings, L.A., McLaren, S., Lin, M.L., McBride, D.J., Varela, I., Nik-Zainal, S., Leroy, C., Jia, M., Menzies, A., Butler, A.P., Teague, J.W., Quail, M.A., Burton, J., Swerdlow, H., Carter, N.P., Morsberger, L.A., Iacobuzio-Donahue, C., Follows, G.A., Green, A.R., Flanagan, A.M., Stratton, M.R., Futreal, P.A., Campbell, P.J.: Massive genomic rearrangement acquired in a single catastrophic event during cancer development. Cell **144**(1), 27 – 40 (2011). DOI https://doi.org/10.1016/j.cell.2010.11.055. URL http://www.sciencedirect.com/science/article/pii/S0092867410013772
335. Stoye, J., Wittler, R.: A unified approach for reconstructing ancient gene clusters. IEEE/ACM Transactions on Computational Biology and Bioinformatics **6**(3), 387–400 (2009)
336. Sturtevant, A.H., Dobzhansky, T.: Inversions in the third chromosome of wild races of drosophila pseudoobscura, and their use in the study of the history of the species. Proceedings of the National Academy of Sciences **22**(7), 448–450 (1936). DOI 10.1073/pnas.22.7.448. URL http://www.pnas.org/content/22/7/448
337. Sturtevant, A.H., Novitski, E.: The homologies of the chromosome elements in the genus drosophila. Genetics **26**(5), 517 (1941)
338. Swenson, K.M., Blanchette, M.: Models and algorithms for genome rearrangement with positional constraints. In: M. Pop, H. Touzet (eds.) Algorithms in Bioinformatics, pp. 243–256. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
339. Szabó, A., Novák, Á., Miklós, I., Hein, J.: Reticular alignment: A progressive corner-cutting method for multiple sequence alignment. BMC bioinformatics **11**(1), 570 (2010)
340. Tajima, F.: Evolutionary relationship of DNA sequences in finite populations. Genetics **105**(2), 437–460 (1983). URL http://www.genetics.org/cgi/content/abstract/105/2/437
341. Takahata, N.: Gene geneaology in three related populations: consistency probability between gene and population trees. Genetics **122**(4), 957–966 (1989)
342. Talbi, E.G.: Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons (2009)

343. Tamazian, G., Dobrynin, P., Krasheninnikova, K., Komissarov, A., Koepfli, K.P., O'Brien, S.J.: Chromosomer: a reference-based genome arrangement tool for producing draft chromosome sequences. GigaScience **5**(1), 38 (2016). DOI 10.1186/s13742-016-0141-6. URL https://doi.org/10.1186/s13742-016-0141-6
344. Tamura, K., Dudley, J., Nei, M., Kumar, S.: Mega4: molecular evolutionary genetics analysis (mega) software version 4.0. Molecular biology and evolution **24**(8), 1596–1599 (2007)
345. Tang, H., Zhang, X., Miao, C., Zhang, J., Ming, R., Schnable, J.C., Schnable, P.S., Lyons, E., Lu, J.: ALLMAPS: robust scaffold ordering based on multiple maps. Genome Biology **16**, 3 (2015)
346. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal median and halving problems under different genomic distances. BMC Bioinformatics **10**, 120 (2009)
347. Tarhio, J., Ukkonen, E.: A greedy approximation algorithm for constructing shortest common superstrings. Theor. Comput. Sci. **57**(1), 131–145 (1988)
348. Than, C.V., Nakhleh, L.: Species tree inference by minimizing deep coalescences. PLoS Computational Biology **5**(9), e1000501 (2009)
349. The Cancer Genome Atlas Program: https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga. Accessed: September 10, 2020
350. The Human Genome Project: https://www.genome.gov/human-genome-project. Accessed: September 10, 2020
351. Thompson, J.D., Higgins, D.G., Gibson, T.J.: Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic acids research **22**(22), 4673–4680 (1994)
352. Tilahun, S.L., Ngnotchouye, J.M.T., Hamadneh, N.N.: Continuous versions of firefly algorithm: A review. Artificial Intelligence Review **51**(3), 445–492 (2019)
353. Timp, W., Nice, A.M., Nelson, E.M., Kurz, V., McKelvey, K., Timp, G.: Think small: nanopores for sensing and synthesis. IEEE Access **2**, 1396–1408 (2014)
354. Torrisi, M., Pollastri, G., Le, Q.: Deep learning methods in protein structure prediction. Computational and Structural Biotechnology Journal (2020)
355. Uddin, M.R., Mahbub, S., Rahman, M.S., Bayzid, M.S.: SAINT: self-attention augmented inception-inside-inception network improves protein secondary structure prediction. Bioinformatics p. btaa531 (2020). DOI 10.1093/bioinformatics/btaa531. URL https://doi.org/10.1093/bioinformatics/btaa531
356. Ulutas, B.H., Kulturel-Konak, S.: A review of clonal selection algorithm and its applications. Artificial Intelligence Review **36**(2), 117–138 (2011)
357. Venter, J.C., Adams, M.D., Myers, E.W., Li, P.W., Mural, R.J., Sutton, G.G., Smith, H.O., Yandell, M., Evans, C.A., Holt, R.A., et al.: The sequence of the human genome. science **291**(5507), 1304–1351 (2001)
358. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning, pp. 1096–1103. ACM (2008)
359. Vogel, G.: HIV strain analysis debuts in murder trial. Science (1998). URL https://www.sciencemag.org/news/1998/10/dna-strain-analysis-debuts-murder-trial
360. Voigt, H.M., Anheyer, T.: Modal mutations in evolutionary algorithms. In: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, pp. 88–92. IEEE (1994)
361. Wajid, B., Serpedin, E.: Review of general algorithmic features for genome assemblers for next generation sequencers. Genomics, proteomics & bioinformatics **10**(2), 58–73 (2012)
362. Wang, G.G., Gandomi, A.H., Alavi, A.H., Gong, D.: A comprehensive review of krill herd algorithm: variants, hybrids and applications. Artificial Intelligence Review **51**(1), 119–148 (2019)
363. Wang, S., Jiang, X., Tang, H., Wang, X., Bu, D., Carey, K., Dyke, S.O., Fox, D., Jiang, C., Lauter, K., et al.: A community effort to protect genomic data sharing, collaboration and outsourcing. NPJ genomic medicine **2**(1), 33 (2017)
364. Wang, S., Peng, J., Ma, J., Xu, J.: Protein secondary structure prediction using deep convolutional neural fields. Scientific reports **6**(1), 1–11 (2016)
365. Wang, W.Y., Barratt, B.J., Clayton, D.G., Todd, J.A.: Genome-wide association studies: theoretical and practical concerns. Nature Reviews Genetics **6**(2), 109–118 (2005)
366. Wang, Y., Li, W., Zhang, T., Ding, C., Lu, Z., Long, N., Rose, J.P., Wang, B.C., Lin, D.: Reconstruction of ancient genome and gene order from complete microbial genome sequences. Journal of theoretical biology **239**(4), 494–498 (2006)

367. Warren, R.L., Yang, C., Vandervalk, B.P., Behsaz, B., Lagman, A., Jones, S.J., Birol, I.: LINKS: Scalable, alignment-free scaffolding of draft genomes with long reads. GigaScience **4**, 35 (2015)
368. Watanabe, K., Taskesen, E., Van Bochoven, A., Posthuma, D.: Functional mapping and annotation of genetic associations with fuma. Nature communications **8**(1), 1–11 (2017)
369. Watterson, G., Ewens, W., Hall, T., Morgan, A.: The chromosome inversion problem. Journal of Theoretical Biology **99**(1), 1 – 7 (1982). DOI https://doi.org/10.1016/0022-5193(82)90384-8. URL `http://www.sciencedirect.com/science/article/pii/0022519382903848`
370. Webb, S.: Deep learning for biology. Nature **554**(7693) (2018)
371. Weber, J.L., Myers, E.W.: Human whole-genome shotgun sequencing. Genome research **7**(5), 401–409 (1997)
372. Weinreb, C., Oesper, L., Raphael, B.J.: Open adjacencies and k-breaks: detecting simultaneous rearrangements in cancer genomes. BMC Genomics **15**(6), S4 (2014). DOI 10.1186/1471-2164-15-S6-S4. URL `https://doi.org/10.1186/1471-2164-15-S6-S4`
373. Weisenfeld, N.I., Kumar, V., Shah, P., Church, D.M., Jaffe, D.B.: Direct determination of diploid genome sequences. Genome research (2017)
374. Welter, D., MacArthur, J., Morales, J., Burdett, T., Hall, P., Junkins, H., Klemm, A., Flicek, P., Manolio, T., Hindorff, L., et al.: The nhgri gwas catalog, a curated resource of snp-trait associations. Nucleic acids research **42**(D1), D1001–D1006 (2014)
375. Willer, C.J., Li, Y., Abecasis, G.R.: Metal: fast and efficient meta-analysis of genomewide association scans. Bioinformatics **26**(17), 2190–2191 (2010)
376. Wu, D., Bi, S., Zhang, L., Yang, J.: Single-molecule study of proteins by biological nanopore sensors. Sensors **14**(10), 18211–18222 (2014)
377. Xu, A.W., Moret, B.M.: Gasts: Parsimony scoring under rearrangements. In: International Workshop on Algorithms in Bioinformatics, pp. 351–363. Springer (2011)
378. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)
379. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010), pp. 65–74. Springer (2010)
380. Yeo, S., Coombe, L., Warren, R.L., Chu, J., Birol, I.: Arcs: scaffolding genome drafts with linked reads. Bioinformatics **34**(5), 725–731 (2017)
381. Yu, Y., Warnow, T., Nakhleh, L.: Algorithms for mdc-based multi-locus phylogeny inference: beyond rooted binary gene trees on single alleles. Journal of Computational Biology **18**(11), 1543–1559 (2011)
382. Zeiler, M.D., Krishnan, D., Taylor, G.W., Fergus, R.: Deconvolutional networks. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 2528–2535. IEEE (2010)
383. Zeira, R., Shamir, R.: Genome rearrangement problems with single and multiple gene copies: A review (2018)
384. Zeira, R., Shamir, R.: Sorting cancer karyotypes using double-cut-and-joins, duplications and deletions. Bioinformatics p. bty381 (2018). DOI 10.1093/bioinformatics/bty381. URL `http://dx.doi.org/10.1093/bioinformatics/bty381`
385. Zerbino, D., Birney, E.: Velvet: algorithms for de novo short read assembly using de bruijn graphs. Genome research pp. gr–074492 (2008)
386. Zhang, L.: From gene trees to species trees II: Species tree inference by minimizing deep coalescence events. IEEE/ACM Transactions on Computational Biology and Bioinformatics **8**(9), 1685–1691 (2011)
387. Zhang, S., Zhou, J., Hu, H., Gong, H., Chen, L., Cheng, C., Zeng, J.: A deep learning framework for modeling structural features of rna-binding protein targets. Nucleic acids research **44**(4), e32–e32 (2015)
388. Zhao, H., Bourque, G.: Recovering true rearrangement events on phylogenetic trees. In: RECOMB International Workshop on Comparative Genomics, pp. 149–161. Springer (2007)
389. Zheng, C., Sankoff, D.: On the pathgroups approach to rapid small phylogeny. BMC bioinformatics **12**(1), S4 (2011)
390. Zheng, G.X., Lau, B.T., Schnall-Levin, M., Jarosz, M., Bell, J.M., Hindson, C.M., Kyriazopoulou-Panagiotopoulou, S., Masquelier, D.A., Merrill, L., Terry, J.M., et al.: Haplotyping germline and cancer genomes with high-throughput linked-read sequencing. Nature biotechnology **34**(3), 303 (2016)

391. Zhou, X., Stephens, M.: Genome-wide efficient mixed-model analysis for association studies. Nature genetics **44**(7), 821–824 (2012)
392. Zhu, Y., Tan, Y.: A danger theory inspired learning model and its application to spam detection. In: International Conference in Swarm Intelligence, pp. 382–389. Springer (2011)
393. Zwickl, D.J.: Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion. Ph.D. thesis (2006)