ANALYZING LATERAL GENE TRANSFER WITH MACHINE LEARNING AND PHYLOGENETIC METHODS

LU BINGXIN

(M.Eng, ECNU, China)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2017

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Lu Bingxin

4 August 2017

Acknowledgements

First, I would like to express heartfelt gratitude to my supervisor Prof. Hon Wai Leong for his great guidance and kind support. His intuitive and patient explanations always helped me to understand the problems better. He asked many thought-provoking questions to inspire me. He encouraged me to study the problems in depth and sharpen the saw for solving specific problems. He assisted me to see the big picture and avoid not seeing the wood for the trees. He generously gave me sufficient freedom to work on my own and provided help when needed. His high enthusiasm in teaching and supporting students left a deep impression on me. He also introduced various interesting stuff which largely broadened my horizons. In short, he not only taught me basic research skills but also serves as a role model of good supervisor.

Next, I am extremely grateful to Prof. Louxin Zhang, who kindly introduced me to the field of phylogenetic networks. A large part of this thesis is impossible without his generous help and detailed guidance. I have learnt a lot from working with him on related problems. His rigorousness and thoughtfulness in research deeply influenced me, which keeps encouraging me to do good research.

My great appreciations also go to Prof. Limsoon Wong and Prof. Wing Kin Sung, who provided insightful comments and invaluable suggestions for my graduate research paper and thesis proposal. Their suggestive feedback helped me to adjust my research direction and further improve my work.

I appreciate the constant help from former and present members of the RAS group as well. The stimulating discussions and pleasant chat with them brought me lots of inspirations and fun. Among them, I offer special thanks to Sriganesh Srihari who suggested my research topic.

I would like to thank other lab mates in the Computational Biology Lab for their kind help and encouragement, especially Abha Belorkar and Iana Pirogova.

I would also like to thank other friends for their sincere concern and help, particularly those in NUS who accompanied me on the long PhD journey.

Last but not least, I am indebted to my parents and sisters for their unconditional love and support.

Contents

Summary

Lateral gene transfer (LGT), the transfer of genetic materials between two reproductively isolated organisms, is an important process in evolution. LGT is also related to the spread of antibiotic resistance and pathogenicity. To further understand the impact of LGT, it is necessary to characterize the prevalence of LGT quantitatively. In this thesis, we mainly study three related problems: how to detect large genomic regions originated from LGT; how to model LGT with phylogenetic networks; and how to detect LGT events. The aim of our research is to develop computational methods to help solving these problems.

A large continuous genomic region acquired by LGT is called a genomic island (GI). The accurate inference of GIs is important for both evolutionary study and medical research. But the available GI detection methods still do not have desirable performances and they may not be easily applied on newly sequenced microbial genomes. So we developed two machine learning methods for better GI detection: GI-SVM which utilizes one-class SVM based on k-mer frequencies and GI-Cluster which utilizes consensus clustering based on multiple GI-related evidence. These two methods provide researchers with better alternative tools to detect GIs. GI-SVM serves as a more sensitive method for a first-pass detection of GIs. GI-Cluster brings a widely applicable framework for GI analysis, which can generate more accurate results.

LGT is one kind of reticulate evolutionary events that is suitable to be modeled with phylogenetic networks. But it is still challenging to reconstruct rooted phylogenetic networks, including LGT networks. Since the relationships among phylogenetic networks, phylogenetic trees and clusters serve as a basis for reconstructing phylogenetic networks, we focus on two fundamental problems arising in network reconstruction: the tree containment problem (TCP) and the cluster containment problem (CCP). Both the TCP and CCP are NP-complete. We implemented fast exponential-time programs for solving the two problems on arbitrary phylogenetic networks. The resulting CCP program is further extended into a program for fast computation of the Soft Robinson–Foulds distance between phylogenetic networks. The evaluation results show that these programs are fast enough for use in practice. So they are likely to be valuable for the application of phylogenetic networks in LGT modelling and evolutionary genomics.

To detect LGTs, numerous computational methods of different categories have been developed. However, known estimates obtained from different methods are often dis-

crepant, and most methods are believed to be complementary. Since there are very few studies that systematically investigated the complementary performances of diverse methods in practice, we conducted a case study on cyanobacterial genomes, which have been well studied in terms of LGT. Our results indicate very low overlap among predictions from different methods, especially from methods of different categories, which is consistent with previous discoveries. Therefore, to get more reliable LGT detections, it is really necessary to prudently apply multiple methods of different kinds and carefully examine their predictions whenever possible.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Lateral gene transfer and related computational problems

In vertical inheritance or reproduction, genetic materials are transferred from ancestry to descendants. However, genetic materials can also be transferred from one organism to another in a lateral manner. This is often referred to as lateral or horizontal gene transfer (LGT or HGT). LGT was reported to be pervasive and have important biological consequences, particularly in prokaryotes (Zhaxybayeva and Doolittle, 2011). In this thesis, we will focus on computational methods for the quantitative analysis of LGTs.

1.1.1 What is LGT

Broadly speaking, LGT is an evolutionary process in which genes are transferred between two reproductively isolated organisms. LGT can occur either between different species (interspecies LGT) or in the same species (intraspecies LGT), such as LGTs between different strains of bacteria. LGT often refers to complete LGT in which the laterally transferred gene replaces an orthologous gene in the recipient genome (orthologous replacement) or added to the recipient genome (novel acquisition). There is also partial or within gene LGT in which a fragment of the gene is replaced by transferred sequences. As a result of LGTs, recipient genomes often show mosaic composition, in which different regions may have originated from different donors. Figure 1.1 briefly shows LGTs from two donor organisms to one recipient organism. Usually LGT means the exchanges which have already fixed in recipient lineages, rather than other physical mechanisms transferring DNA into prokaryotic cells (Doolittle et al., 2003). So genetic

engineering, which can be seen as artificial LGT, is not in our research scope.

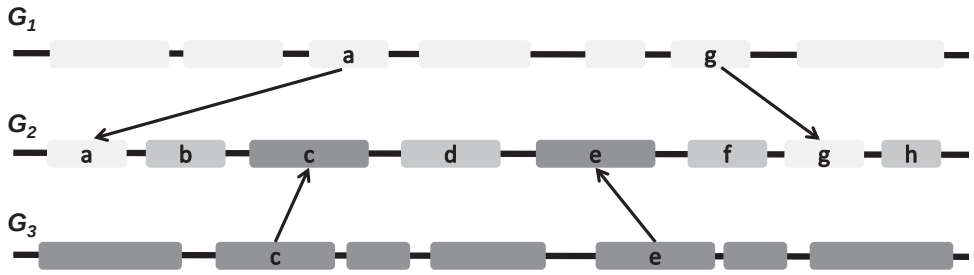

Figure 1.1: The schematic representation of LGT among three organisms. Each rectangular bar represents a gene. Gene a and g were transferred from donor organism G_1 to recipient organism G_2. Gene c and e were transferred from donor organism G_3 to recipient organism G_2. Different colors represent different origins of genes.

The study of LGT begins as early as 1928, when Frederick Griffith's experiment of *Streptococcus pneumonia* suggests genetic materials can be laterally transferred between bacteria (Griffith, 1928). It has been documented for a long time that microbes can integrate alien DNAs into their genomes from sources other than parents (Avery et al., 1944). The detection of LGT can be traced back to 1990s when the availability of many prokaryotic genome sequences exposed unexpected sequence similarities among quite different species (Zhaxybayeva and Doolittle, 2011). The well-supported inconsistencies between gene tree and species tree which were recalcitrant to improvement of phylogenetic methods also prompt the discovery of LGT (Gogarten et al., 2002). Due to the complexity of LGT, there are still lots of open issues, including its process, mechanism, impact, and quantification (Ragan and Beiko, 2009).

The mechanisms of LGT in eukaryotes are still poorly known despite that several possible gene transfer pathways are recently uncovered (Soucy et al., 2015). But there are several well-known mechanisms in prokaryotes (Ochman et al., 2000; Popa and Dagan, 2011; Soucy et al., 2015). Three major mechanisms are transformation, conjugation, and transduction. Transformation is the uptake of raw DNA directly from the environment, which can exchange DNAs between evolutionarily distantly related organisms. Conjugation is featured by the physical contact between donor and recipient cells which can then causes integration of donor plasmid into recipient chromosomes. Transduction is mediated by phages which incorporates DNA fragments in previous hosts and then integrates their DNAs into recipient chromosomes. Two other mechanisms include gene transfer agents and cell fusion (Popa and Dagan, 2011; Soucy et al., 2015). Gene transfer

agents are phage-like elements which lack the typical features of phages but can carry random DNAs from host to recipient. Cell fusion is similar to conjugation, but differs in that the exchange of DNA is bi-directional after cell contact.

1.1.2 The prevalence and impact of LGT

There have been plenty of evidences showing the prevalence of LGT in and between all the three domains of life: bacteria, archaea, and eukaryotes (Gogarten and Townsend, 2005; Keeling and Palmer, 2008; Soucy et al., 2015). In prokaryotes (bacteria and archaea), LGT may even blur the boundaries of species (Gogarten and Townsend, 2005). In particular, LGT is most common in bacteria, where the spread of antibiotics is attracting increasing interest (Ochman et al., 2000; Skippington and Ragan, 2011). The cases of LGT between prokaryotes and eukaryotes and even among eukaryotes are increasing, particularly in unicellular eukaryotes and plants (Keeling and Palmer, 2008; Soucy et al., 2015). LGT in virus is also common (Metzler, 2014).

Generally speaking, genetic materials resulted from LGTs may be important in promoting genome innovation, and hence they may bring selective advantages to the host organisms and facilitate their adaptation to new niches. For instance, Zhaxybayeva and Doolittle (2011) summarized four novel adaptations of organisms which cannot be explained without LGT acting as an important factor: an archaea-to-archaea transfer to allow *Salinibacter ruber* to survive in environments with high salt concentrations, a bacterium-to-bacterium transfer to help the gut bacterium *Bacteroides plebeius* digest carbohydrates from plants, a bacteria-to-animal transfer to assist the nematode *Meloidogyne incognita* in parasitizing plants, and a fungus-to-animal transfer contributing to the colourations in the pea aphids.

However, LGT has fuelled strong debates about its impact and frequency in evolution (Dagan and Martin, 2007; Vernikos, 2008; Ragan and Beiko, 2009; Sjöstrand et al., 2014). The views span from one extreme that LGT hardly exists and has insignificant impact to the other extreme that LGT is so rampant that the traditional tree of life may not be a meaningful representation of organismal evolution (Dagan and Martin, 2007; Tofigh et al., 2011). Despite these debates, LGT has been widely accepted as an important process in molecular evolution. Thus, the quantitative characterization of LGT events can help us to better understand the extant genomes in their historical context.

On the other hand, LGT has the potential to quickly spread antibiotic resistance and pathogenic elements. The spread of antibiotic resistance brings obvious implications for public health (Skippington and Ragan, 2011). There are a few examples. Epidemic strains of *Staphylococcus aureus* acquired resistance to methicillin via frequent lateral transfers of methicillin resistance genes and brought problems to hospitals worldwide (Enright et al., 2002; Awadalla, 2003). Several *Escherichia coli* outbreaks were reported to be related to LGT. An outbreak in Japan in 1996 was caused by *Escherichia coli* O157:H7 Sakai strain which arose from *Escherichia coli* O55:H7 strain by acquiring Shiga toxin-encoding prophage via LGT (Kyle et al., 2012; Trappe et al., 2016). Another outbreak in Germany in 2011 (Frank et al., 2011) was caused by *Escherichia coli* O104:H4 strain whose ability to cause hemorrhagic colitis and antibiotic resistance was probably acquired by several LGT events (Brzuszkiewicz et al., 2011). So the reliable and rapid detection of LGT is of potential medical importance, which may help to identify drug-resistance genes and develop vaccine targets.

1.1.3 The computational problems related to LGT

For the quantitative study of LGT, many important computational problems have arisen, such as how to identify LGT events and how to estimate LGT rate. In this work, we mainly focus on three related problems, which are discussed below.

(1) Which large genomic regions originated from LGT?

Some DNA sequences acquired via LGT appear as clusters in the genome. These clusters of sequences were initially referred to as *pathogenicity islands* (PAIs) (Hacker et al., 1990), which are large virulence-related inserts present in pathogenic bacterial strains but absent from other non-pathogenic strains. Later, the discoveries of regions similar to PAIs but encoding different functions in non-pathogenic organisms lead to the designation of *genomic islands* (GIs) (Hacker and Kaper, 2000). GIs are then found to be common in both pathogenic and environmental microbes (Dobrindt et al., 2004).

Specifically, a GI is a large continuous genomic region arisen by LGT, which can contain tens to hundreds of genes. The size of known GIs varies from less than 4.5 kb to 600 kb (Vernikos, 2008). Laterally acquired genomic regions shorter than a threshold are also called *genomic islets* (Juhas et al., 2009; Bellanger et al., 2014). GIs often

have phylogenetically sporadic distribution. Namely, they are present in some particular organisms but absent in several closely related organisms. As shown in Figure 1.2, GIs have several other well-known features to distinguish them from the other genomic regions (Hacker et al., 1997; Schmidt and Hensel, 2004; Juhas et al., 2009), such as different sequence composition from the core genome, the presence of mobility-related genes, flanking direct repeats (DRs), and specific integration sites. For example, tDNA (tRNA or tmRNA gene) is well known as a hotspot for GI insertion (Williams, 2002; Bellanger et al., 2014). However, not all these features are present in a GI, and some GIs lack many of these features. As a consequence, GIs were also considered as a superfamily of mobile elements with core and variable structural features (Vernikos and Parkhill, 2008).

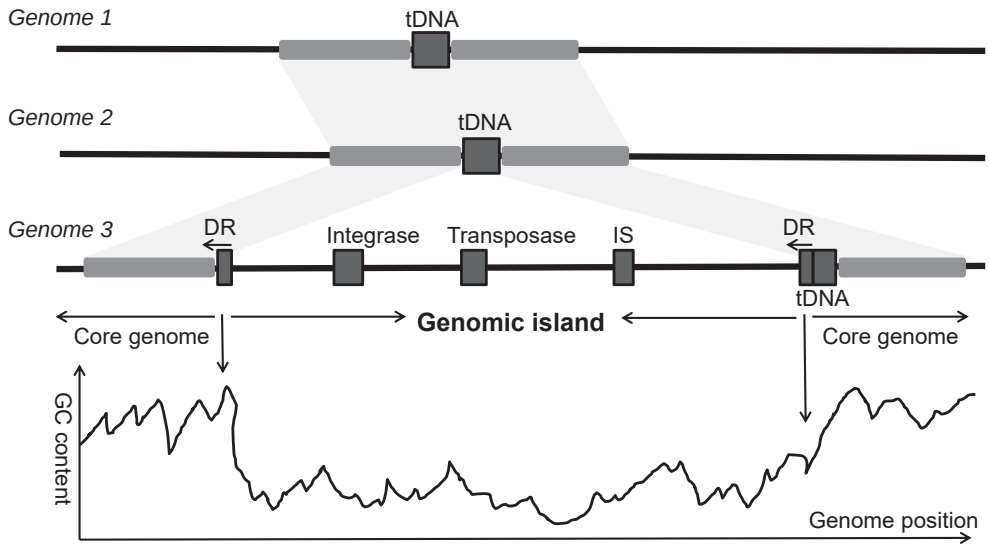

Figure 1.2: The schematic representation of several features associated with a genomic island (GI). A GI is often absent in closely related genomes. It may also have atypical compositional characteristics compared with the core genome, such as lower GC content. The presence of several sequence elements are indicators of a GI: flanking conserved regions (dark grey rounded rectangles connected by light grey regions), direct repeats (DRs), insertion sequence (IS) elements and mobility-related genes encoding integrase and transposase.

In addition to the restricted GI definition in (Boyd et al., 2009), GIs are often seen as a broad category of mobile genetic elements (MGEs) (Langille et al., 2010). They can be further grouped into subcategories by mobility: some GIs are mobile and hence can further transfer to a new host, such as integrative and conjugative elements (ICEs), conjugative transposons and prophages; but other GIs are not mobile any more (Juhas

et al., 2009). GIs can also be classified by the function of genes within as follows: PAIs with genes encoding virulence factors; *resistance islands* (REIs) with genes responsible for antibiotic resistance; *metabolic islands* with genes related to metabolism; and so on (Dobrindt et al., 2004). However, the latter classification may not be definite since the functions of genes within GIs may not be clear-cut in practice.

GIs play crucial roles in microbial genome evolution and adaptation of microbes to environments. As part of a flexible gene pool (Hacker and Carniel, 2001), the acquisition of GIs can facilitate evolution in quantum leaps, allowing bacteria to gain large numbers of genes related to complex adaptive functions in a single step and thereby confer evolutionary advantages (Dobrindt et al., 2004; Juhas et al., 2009). Remarkably, the genes inside GIs can influence a wide range of important traits: virulence, antibiotic resistance, symbiosis, fitness, metabolism, and so on (Dobrindt et al., 2004; Juhas et al., 2009). In particular, PAIs can carry many genes contributing to pathogen virulence (Hacker et al., 1997; Schmidt and Hensel, 2004; Ho Sui et al., 2009), and potential vaccine candidates were suggested to locate within PAIs (Moriel et al., 2010). Thus, the accurate identification of GIs is important not only for evolutionary study but also for medical research.

The in silico prediction of GIs can be summarized as follows: given the genome sequence of a query organism, identify the positions of GIs along the query genome via computer programs alone. Additional input information may also be incorporated in prediction, such as the genomes of other related organisms and genome annotations.

(2) How to model LGT with phylogenetic networks?

The accumulating evidence of LGT between lineages, particularly in prokaryotes (Gogarten and Townsend, 2005; Soucy et al., 2015; Doolittle and Brunet, 2016), keeps challenging the simplified phylogenetic trees which have been adopted to model evolutionary history of life since Darwin's *The Origin of Species*. Additionally, other reticulate evolutionary events also cause complications in constructing tree-like evolutionary models, such as hybridization and introgression between species (Mallet, 2007; Fontaine et al., 2015), and recombination of various forms (Martin et al., 2011). When the real organismal phylogeny is a complex web woven with reticulation events, it seems more suitable to build a species network rather than a species tree (Gogarten and Townsend, 2005).

Consequently, researchers started adopting phylogenetic networks to model reticulation events (Doolittle and Bapteste, 2007; Bapteste et al., 2013).

Phylogenetic networks can be used to either visualize conflicting phylogenetic information or model reticulation events explicitly. The former are typically unrooted, whereas the latter are rooted. LGTs are often modeled as additional edges added to a base phylogenetic tree in a rooted phylogenetic network. Therefore, we focus on rooted phylogenetic networks for the modeling of LGT.

According to Morrison (2014), the first published LGT network is a network showing the transfer of an endogenous primate virus from the Old World monkeys to an ancestor of the domestic cat (Benveniste and Todaro, 1974) (Figure 1.3). Kunin et al. (2005) reconstructed the first large LGT network to represent the history of 165 microbial genomes, termed as "the net of life", in which vertical inheritance constitutes a tree and multiple tiny vines of LGT events interconnect the tree. They used established tree reconstruction methods to model vertical inheritance and added instances of LGT after identifying LGT events based on phylogenetic distribution of gene families.

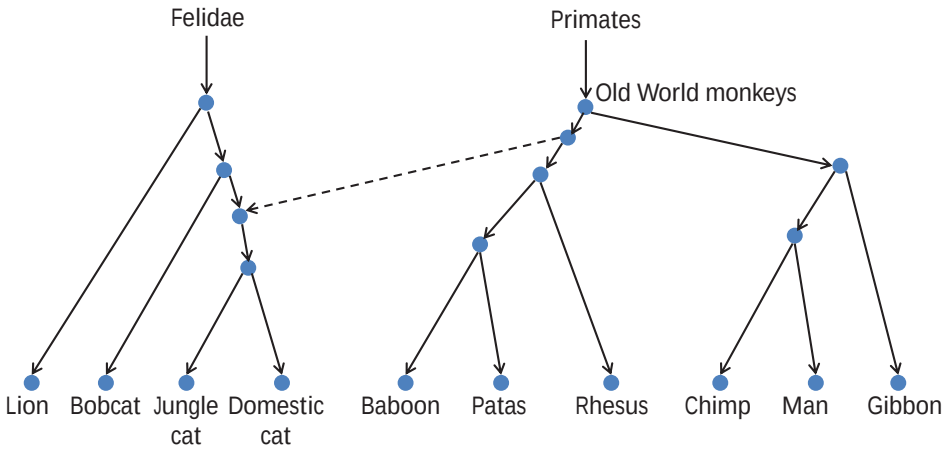

Figure 1.3: The first LGT network, redrawn from Figure 3 in (Benveniste and Todaro, 1974). An endogenous primate virus was transferred to an ancestor of the domestic cat after the divergence of the Old World monkeys and the apes. The LGT is denoted by the dashed arrow.

Owing to the laboriousness of tree reconstruction and LGT identification, it is desirable to reconstruct a LGT network directly from the data, such as a set of trees or sequences. But it is still challenging to build phylogenetic networks (Huber et al., 2015),

especially LGT networks. Almost all existing methods for reconstructing phylogenetic networks are not commonly used in practice (Huson et al., 2010). Therefore, some fundamental problems have to be resolved to facilitate the development of practical methods for network reconstruction.

The trees and (soft) clusters displayed in a network are of great significance in the studies of phylogenetic networks. As a result, two related problems have arisen from reconstructions of phylogenetic networks (Kanj et al., 2008): the tree containment problem (TCP) and the cluster containment problem (CCP). The TCP asks whether a phylogenetic tree is displayed in a phylogenetic network. The CCP asks whether a cluster is a soft cluster displayed in (of some tree node in) a phylogenetic network. Efficient solutions to the TCP and CCP can provide foundations for reconstructing, comparing, and validating phylogenetic networks. Consequently, they may contribute to network-based LGT modeling and detection. In the following paragraphs, we explain the TCP and CCP in more detail.

A cluster is any subset of a set of taxa. In a phylogenetic tree, the taxa below a node form a unique subset of the taxa, called its cluster. A phylogenetic tree is uniquely determined by the set of “nested" clusters in the tree (Huson et al., 2010). A phylogenetic network is a generalization of a phylogenetic tree in which there are additional reticulate nodes, which are the nodes with an indegree of at least two. Formally, a phylogenetic network is a rooted acyclic digraph with uniquely labeled leaves. Since most gene families have tree-like evolutionary histories, a network model of the evolution of a set of genomes is often built and validated by checking its consistency with the available related gene trees and/or clusters (Huson et al., 2010).

A phylogenetic tree is said to be displayed in a phylogenetic network if it can be obtained by deleting all but one incoming edges from each reticulate node and then contracting all the nodes of degree two. In a phylogenetic network, a non-reticulate node is called a tree node. Each tree node represents a cluster and a set of soft clusters. Similar to the case of phylogenetic tree, a node’s cluster consists of all taxa below it, whereas its soft clusters are the clusters represented by this node in the phylogenetic trees that are displayed in the network. The cluster and soft cluster are also called hardwired cluster and softwired cluster, respectively (Huson et al., 2010).

Measuring the dissimilarity between phylogenetic networks is important for assess-

ing a network reconstruction method. The solutions to the TCP and CCP also relate to the computing of distances (based on the number of trees or soft clusters displayed in a network) between two phylogenetic networks. In particular, the efficient solution to the CCP can help to develop better methods for computing the distance based on soft clusters. The distance based on soft clusters is a distance related to the Robinson–Foulds (RF) distance which is a generalization of the same metric for phylogenetic trees. Simply put, the RF distance is the half of the cardinality of the symmetric difference of the two sets of clusters respectively contained in the two networks (Cardona et al., 2009a). By replacing clusters with soft clusters, we obtain the Soft Robinson–Foulds (SRF) distance (Huson et al., 2010). The RF and SRF distances are two possible extensions of the RF distance for trees. If a reconstructed phylogenetic (LGT) network has a small SRF distance to the "true" phylogenetic (LGT) network, this reconstructed network is likely to be a good model.

(3) What were potential LGT events?

The straightforward way to quantify LGTs is to detect instances of LGT events, namely inferring individual LGT events related to a gene or a gene family. This problem of detecting LGTs is a general problem that covers the first two problems whose solutions can be easily adapted to find potential LGT events.

Over the years, various methods have been developed to detect LGTs. However, their predictions differ a lot in different studies. The estimates of the proportion of genes affected by LGT ranged from about 2% to 90% (Dagan and Martin, 2007; Kloesges et al., 2011). The conclusions on the transferability of genes were also different. According to the *complexity hypothesis* (Jain et al., 1999; Wellner et al., 2007; Cohen et al., 2011), informational genes seem to be more resistant to LGT than operational genes. Despite that several studies supported the complexity hypothesis to some extent (Beiko et al., 2005; Shi and Falkowski, 2008; Abby et al., 2012; Sjöstrand et al., 2014), a few studies generated different results (Zhaxybayeva et al., 2006; Matzke et al., 2014).

In short, there is little agreement on how to quantify LGTs and thereby their ramifications (Doolittle and Brunet, 2016). Some possible reasons for the discrepancies include biased sampling and uncertainties in the methods (Dagan and Martin, 2007). It was suggested that the most reasonable way is to find which genes were strictly vertically

inherited from the last universal common ancestor (Doolittle and Brunet, 2016). But this is also a difficult problem. Therefore, it remains an open challenging problem to obtain more accurate detection of LGTs.

1.2 Computational methods for analyzing LGT

To solve the above three problems related to the analysis of LGT (Section 1.1.3), different kinds of computational methods have been proposed. We briefly discuss them in this section, respectively, to indicate potential gaps for each problem. In the following chapters, we will give more detailed literature reviews.

1.2.1 Computational methods for detecting genomic islands

Langille et al. (2010) gave a comprehensive review of GI-related features and different computational approaches for detecting GIs. Later, Che et al. (2014a) presented a similar review for detecting PAIs. We will provide a detailed up-to-date review in Section 2.2.

The methods for GI prediction usually use two most indicative features of the horizontal origin of GIs: biased sequence composition and sporadic phylogenetic distribution. Based on the two features, these methods roughly fall into two categories: *composition-based methods* and *comparative genomics-based methods* (Langille et al., 2010). For ease of discussion, we categorize GI prediction methods into two large groups based on the number of input genomes: *methods based on one genome* and *methods based on multiple genomes*. Methods in the former group are often composition-based, whereas methods in the latter group are usually based on comparative genomics. We also discuss *ensemble methods* which combine different kinds of methods and *methods for incomplete genomes* which predict GIs in draft genomes.

Most methods based on one genome utilize sequence composition to identify GIs. Composition-based methods identify GIs by utilizing compositional differences between alien regions and native regions within a single genome. According to the units for measuring genome composition, composition-based methods can be divided into methods at the gene level and methods at the DNA level. These methods are usually straightforward and easily applicable. But they may report many false positives and false negatives.

Some methods based on one genome utilize GI structural characteristics, mainly

including direct integration methods and machine learning methods. These methods can obtain very accurate predictions. But direct integration methods may discard GIs not atypical in terms of certain features and machine learning methods have limited applications due to the requirement of high-quality training data.

Methods based on several genomes compare multiple related genomes to find regions present in a subset but not all of the genomes. Unlike methods based on one genome, this kind of methods can obtain more accurate results by comparing a set of well-chosen genomes. But their predictions are largely dependent on the selected genomes.

The predictions from different kinds of methods may be non-overlapping and complementary to each other (Langille et al., 2010). To achieve the good of both worlds, ensemble methods that integrate different methods have been developed. One common way of integration is to combine the predictions from multiple programs. Another way is to gradually pick up more reliable predictions from the results of one method by other methods. Most of these methods have user-friendly interfaces, but the integration procedures may discard some interesting predictions.

Since many newly sequenced genomes are in draft status, a few methods for incomplete genomes were developed. These methods first require genome assembly to obtain a single genome. After that, they actually use similar methodologies as those for GI detection in complete genomes. Generally speaking, they are still simplistic and have limited applications.

1.2.2 Computational methods related to phylogenetic networks

In recent years, phylogenetic networks have been the subject of intensive theoretic studies (Huson and Bryant, 2006; Huson et al., 2010; Morrison, 2011; Gusfield, 2014). Given the big challenges in reconstructing a phylogenetic network (including a LGT network) from biological data, we mainly focus on the two basic problems arising in the reconstruction and validation of phylogenetic networks: the TCP and CCP. We also consider the extension of the TCP and CCP in computing distances between phylogenetic networks. In Section 3.2, we will provide a more detailed review.

LGT is naturally modeled in a tree-based network (Francis and Steel, 2015), a kind of phylogenetic networks that can be obtained by adding edges to a phylogenetic tree. But currently there are few network-based methods for LGT modeling due to the difficulties

in network reconstruction and the asymmetricity of LGT (Cardona et al., 2015).

Both the TCP and CCP are NP-complete even for binary networks (Kanj et al., 2008; Huson et al., 2010; van Iersel et al., 2010b). As a result, polynomial-time algorithms are only known for several types of restricted networks. For the TCP, polynomial-time algorithms were published for phylogenetic networks with the reticulation-visible property (van Iersel et al., 2010b; Gambette et al., 2015b,a; Gunawan et al., 2016a, 2017). For the CCP on reticulation-visible networks, a polynomial-time algorithm was given in (Huson et al., 2010), and a linear-time algorithm was recently presented in (Gunawan et al., 2017).

Previous methods for computing distances based on the number of trees (soft clusters) displayed in a network firstly enumerate all the trees (soft clusters) displayed in a network, and then check whether the trees (soft clusters) displayed in one network are displayed in another network or not. These methods are not efficient when dealing with networks with many reticulate nodes. Due to the NP-completeness of the TCP and CCP, there are unlikely polynomial-time algorithms for computing the distance based on displayed trees and the distance based on soft clusters (the SRF distance).

1.2.3 Computational methods for detecting LGT events

Numerous methods using various criteria have been proposed to detect LGTs. Some of them were comprehensively reviewed in (Zhaxybayeva, 2009; Azad and Lawrence, 2011; Ravenhall et al., 2015). We will provide a latest review in Section 4.2.

The methods for LGT detection are mainly classified into two categories based on the criteria used. One kind is *phylogenetic method*, which is based on evolutionary history or phylogenetic relationship among multiple different organisms. The other kind is *compositional method*, which is based on properties of sequence composition in a single genome. According to whether a phylogenetic tree is required or not, phylogenetic methods can be classified as implicit methods (without phylogeny reconstruction), tree-based methods, and network-based methods. The methods for GI prediction belong to compositional methods, whereas modeling LGT with phylogenetic networks are network-based methods.

Most LGT detection methods are complementary to each other, as they may detect LGTs of different properties (Podell and Gaasterland, 2007; Zhaxybayeva, 2009). Com-

positional methods are generally good at detecting recent LGTs. Phylogenetic tree-based methods can infer ancient LGTs and they are considered as the most reliable methods. Although tree-based methods were frequently used for LGT detection, network-based methods are equally effective for detecting LGTs in theory (Morrison, 2011). However, network-based methods for LGT detection are still in early development.

It is suggested that the combination of different kinds of methods can predict LGTs better than a single kind of methods (Lawrence and Ochman, 2002; Zhaxybayeva and Doolittle, 2011; Azad and Lawrence, 2011). In practice, the results of compositional methods are often validated by implicit methods (blasting against sequence databases). To ascertain that a LGT event really occurred, tree-based methods are often required to check the incongruence between gene and genome evolutionary histories if a set of orthologs can be identified from the selected organisms.

1.3 The focus and significance of the thesis

In this thesis, we aim to develop better computational methods to solve the three problems described in Section 1.1.3. In this section, we describe the research gaps revealed by the brief review in Section 1.2 and our work towards filling the gaps for each problem, respectively.

- Despite that a large number of GI detection methods of various categories are available, there is still a pressing need for better methods that can work on a single newly sequenced genome. Newly sequenced genomes may have no or incomplete annotations owing to limited time in detailed analysis. Moreover, they may not have enough closely related organisms to compare with. Thus, new methods are required to quickly detect and prioritize reliable GI candidates for further investigations. Given this need, we developed two machine learning methods to detect GIs: GI-SVM (Lu and Leong, 2016b) and GI-Cluster (Lu and Leong, 2017). As a by-product of our work on GI detection, we also published a review on previous GI detection methods (Lu and Leong, 2016a).

  GI-SVM utilizes one-class SVM along with spectrum kernels to identify GIs based on k-mer frequencies. It was shown to have higher recall than programs taking the same input on real biological datasets. GI-Cluster takes advantage of consensus

clustering to detect GIs by integrating multiple GI-related features. It is widely applicable, either to complete and incomplete genomes or to initial GI predictions from other programs (e.g. GI-SVM). GI-Cluster does not require training datasets or existing genome annotations, but it can still achieve comparable or better performance than supervised machine learning methods in comprehensive evaluations. In summary, GI-SVM and GI-Cluster provide researchers with better alternative tools to detect GIs on newly sequenced genomes.

In Chapter 2, we will describe our work regarding GI-SVM and GI-Cluster.

- The current algorithms for the TCP and CCP are only for restricted classes of phylogenetic networks. Given that a large fraction of phylogenetic networks is not reticulation-visible (Zhang, 2016), it is necessary to develop exact algorithms for the TCP and CCP on arbitrary phylogenetic networks. Therefore, we developed two programs for solving TCP (Gunawan et al., 2016b) and CCP (Lu et al., 2017) on arbitrary phylogenetic networks, respectively. We also extended the CCP program to a program for fast computing of the SRF distance between two phylogenetic networks (Lu et al., 2017).

  Although phylogenetic networks are promising tools for modeling LGT, lots of theoretical challenges still exist. The solutions to the TCP and CCP for arbitrary phylogenetic networks may make it more practical to apply network-based methods in LGT analysis. In addition, the methods for comparing phylogenetic networks based on displayed soft clusters can facilitate the validation of reconstructed networks with more reticulations.

  In Chapter 3, we will describe our methods to solve the TCP and CCP and to compare the phylogenetic networks.

- Although it seems more reliable to use multiple methods for LGT detection in practice, very few systematic studies were conducted to investigate the complementary performances of various methods on real biological datasets. Since cynobacteria have been extensively studied in terms of LGT, we performed a case study on a set of cynobacterial genomes by applying multiple LGT detection methods from different categories.

  We investigated the consistencies among predictions from both compositional

and phylogenetic methods on selected cynobacterial genomes. Consistent with previous studies, our comparison results exhibited very low overlap among predictions from various methods, especially methods of different kinds, in spite of certain agreements. This further indicates the necessity of carefully examining predictions from multiple methods before reaching solid conclusions regarding LGTs.

In Chapter 4, we will describe this case study in more detail.

Chapter 2

Identifying genomic islands by machine learning

2.1 Introduction

In this chapter, we first provide a brief description of challenges in GI prediction, followed by a detailed review of GI prediction methods. Then we describe GI-SVM which predicts GIs based solely on k-mer frequencies and GI-Cluster which further improves GI prediction by integrating multiple information.

It is a non-trivial task to find laterally transferred regions of relatively small size in a long genome sequence. Although there are several well-characterized features to distinguish GIs from other genomic regions (Langille et al., 2010), two prominent challenges still exist: the extreme variation of GIs and the lack of benchmark GI datasets.

The mosaic nature and extreme variety of GIs increase the complexity of GI prediction (Vernikos, 2008). The elements within a GI may have been acquired by several LGT events (probably from different origins) and are likely to have undergone subsequent evolutions, such as gene loss and genomic rearrangement (Dobrindt et al., 2004). Consequently, the composition, function and structure of GIs can show various patterns. This can be illustrated by GIs in the same species (Marcus et al., 2000), GIs in Gram-negative bacteria (Hacker et al., 1997), and GIs in both Gram-positive and Gram-negative bacteria (Hacker et al., 1997; Vernikos and Parkhill, 2008). The diversity of GIs prevents an effective way of integrating multiple features for prediction. Choosing only a few features as predictors may discard lots of GIs without those features. Even if

Table 2.1: The available datasets related to genomic islands.

Name	Feature	Availability
Database		
PAIDB (Yoon et al., 2007, 2015)	The only database including most reported PAIs and REIs	http://www.paidb.re.kr/about_paidb.php
Islander (Mantri and Williams, 2004; Hudson et al., 2015)	Intended to be gold standard dataset for accurately mapped GIs	http://bioinformatics.sandia.gov/islander
ICEberg (Bi et al., 2012)	Providing comprehensive information about ICEs	http://db-mml.sjtu.edu.cn/ICEberg/
Constructed dataset		
RVM datasets (Vernikos and Parkhill, 2008)	331 GIs and 337 non-GIs from 37 bacteria of 3 genera	Not available
IslandPick datasets (Langille et al., 2008)	771 GIs and 3770 non-GIs from 118 bacteria of 12 orders	http://www.pathogenomics.sfu.ca/islandpick_GI_datasets/

the fundamental property of GIs, the lateral origin, can be used for prediction, it is still challenging since LGT itself is difficult to ascertain (Ravenhall et al., 2015).

There are still no reliable benchmark GI datasets for validating prediction methods or supervised prediction. With more GIs being predicted and verified, several GI-related databases have been deployed and regularly updated, such as Islander (Mantri and Williams, 2004), PAIDB (Yoon et al., 2007), and ICEberg (Bi et al., 2012) (Table 2.1). However, these databases are mainly for *specific kinds* of GIs, such as tDNA-borne GIs (GIs inserted at tRNA or tmRNA gene sites), PAIs, and ICEs. There are also two constructed GI datasets based on whole-genome comparison (Vernikos and Parkhill, 2008; Langille et al., 2008) (Table 2.1), which were used as training datasets for machine learning methods. But the scale of these datasets is still not large enough, and their reliability has not been verified by convincing biological evidence.

In spite of the above challenges, previous methods have made considerable progress in GI prediction. Figure 2.1 shows an overview of these methods.

2.2 Literature review

In this section, we describe GI detection methods by category in more detail. For reference, we list available programs discussed under each category in Table 2.2.

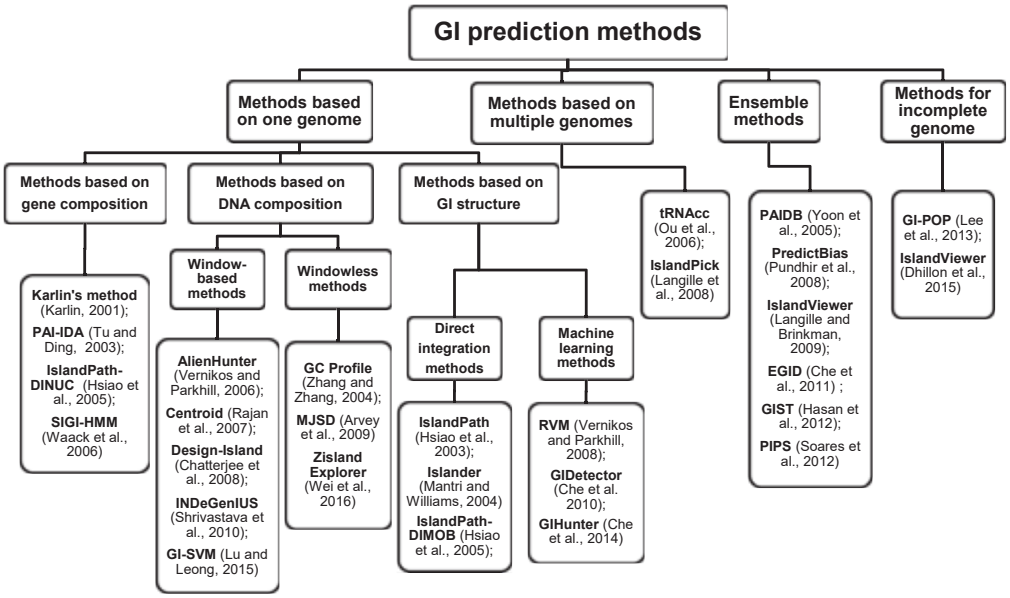

Figure 2.1: The hierarchical overview of computational methods for predicting genomic islands which are discussed in Section 2.2.

2.2.1 GI detection methods based on one genome

In this subsection, we first present the basic idea of composition-based methods and then discuss methods at the gene and DNA level separately.

The major assumption of composition-based methods is that mutational pressures and selection forces acting on the microbial genomes may result in species-specific nucleotide composition (Lawrence and Ochman, 1997). Thus, a laterally transferred region may show *atypical composition* which is distinguishable from the average of the recipient genome. Under this assumption, most compositional methods try to choose certain sequence characteristics as discrimination criteria to measure the compositional differences. Several features have been shown to be good criteria, including *GC content*, *codon usage*, *amino acid usage*, and *oligonucleotide (k-mer) frequencies* (Tsirigos and Rigoutsos, 2005b). Based on these criteria, single-threshold methods are often adopted. The atypicality of each gene or genomic region is measured by a score derived from the comparison with the average of the whole genome via similarity measures. The genes or genomic regions with scores below or above a certain threshold (either predefined or dynamically computed) are supposed to be atypical. The consecutive atypical genes or genomic regions are finally merged to get candidate GIs.

Table 2.2: The summary of selected programs for predicting genomic islands.

Program	Form	Availability
Methods based on gene composition of one genome		
PAI-IDA (Tu and Ding, 2003)	Command line	Upon request
SIGI-HMM (Waack et al., 2006)	Graphical interface	https://www.uni-goettingen.de/en/research/185810.html
Methods based on DNA composition of one genome		
Window-based methods		
AlienHunter (Vernikos and Parkhill, 2006)	Command line	http://www.sanger.ac.uk/resources/software/alien_hunter
Centroid (Rajan et al., 2007)	Command line	Upon request
Design-Island (Chatterjee et al., 2008)	Command line	http://www.isical.ac.in/~rchatterjee/Design-Island.html
INDeGenIUS (Shrivastava et al., 2010)	Command line	Upon request
Windowless methods		
GC Profile (Zhang and Zhang, 2004; Gao and Zhang, 2006; Zhang et al., 2014)	Web-based	http://tubic.tju.edu.cn/GC-Profile
Zisland Explorer (Wei et al., 2017)	Web-based	http://tubic.tju.edu.cn/Zisland_Explorer/
MJSD (Arvey et al., 2009)	Command line	http://cbio.mskcc.org/~aarvey/mjsd
Methods based on GI structure of one genome		
Direct integration methods		
IslandPath (Hsiao et al., 2003)	Web-based	http://www.pathogenomics.sfu.ca/islandpath/
Machine learning methods		
GIDetector (Che et al., 2010)	Command line	http://www5.esu.edu/cpsc/bioinfo/software/GIDetector
GIHunter (Che et al., 2014b)	Command line	http://www5.esu.edu/cpsc/bioinfo/software/GIHunter
Methods base on multiple genomes		
tRNAcc (Ou et al., 2006)	Web-based	http://db-mml.sjtu.edu.cn/MobilomeFINDER/
IslandPick (Langille et al., 2008)	Command line	http://www.pathogenomics.sfu.ca/islandviewer/download/
Ensemble methods		
IslandViewer (Langille and Brinkman, 2009; Dhillon et al., 2013, 2015; Bertelli et al., 2017)	Web-based	http://www.pathogenomics.sfu.ca/islandviewer
EGID (Che et al., 2011)	Command line	http://www5.esu.edu/cpsc/bioinfo/software/EGID
GIST (Hasan et al., 2012)	Graphical interface	http://www5.esu.edu/cpsc/bioinfo/software/GIST
PredictBias (Pundhir et al., 2008)	Web-based	http://www.bioinformatics.org/sachbinfo/predictbias.html
PIPS (Soares et al., 2012)	Command line	http://www.genoma.ufpa.br/lgcm/pips
Methods for incomplete genome		
GI-POP (Lee et al., 2013)	Web-based	http://gipop.life.nthu.edu.tw

GI detection methods based on gene sequence composition

Methods based on gene sequence composition are often designed to detect laterally transferred genes (Azad and Lawrence, 2011), and only a few methods are specifically developed to detect GIs. The methods for LGT detection can be utilized to identify GIs by combing clusters of laterally transferred genes, but they are supposed to be less sensitive, since some genes inside a GI may not show atypicality to allow the whole GI being captured. Here we mainly discuss specific methods for GI detection.

Some methods combine multiple discrimination criteria, such as Karlin's method

(Karlin, 2001) and PAI-IDA (Tu and Ding, 2003). Karlin's method and PAI-IDA predict GIs and PAIs by evaluating multiple compositional features (*GC content*, *dinucleotide frequencies*, *codon usage*, and *amino acid usage*). Karlin's method is a single-threshold method, whereas PAI-IDA uses iterative discriminant analysis. Both methods use a sliding window to scan the genome, and sequences or genes inside each window are used for computation.

Other methods use only a single discrimination criterion, such as IslandPath-DINUC (Hsiao et al., 2003, 2005) and SIGI-HMM (Waack et al., 2006). IslandPath-DINUC uses a single-threshold method to predict GIs as multiple consecutive genes with only *dinucleotide bias*. SIGI-HMM predicts GIs and putative donors of laterally transferred genes based solely on the *codon usage bias* of individual genes. As an extension of SIGI (Merkl, 2004), an earlier method based on scores derived from codon frequencies, SIGI-HMM substitutes the previous heuristic method with Hidden Markov Model (HMM) to model the laterally transferred genes and native genes as different states.

Pros and Cons Methods based on gene sequence composition are generally easy to implement and apply. But what they indeed find are compositionally atypical genomic regions in terms of certain criteria. So there are many false positives and false negatives. Native regions may easily be detected as false positives owing to their atypical composition for reasons other than LGT, such as highly expressed genes (Garcia-Vallve et al., 2003). At the same time, ameliorated GIs (Lawrence and Ochman, 1997) or GIs originated from genomes with similar composition may not be detected. But the false positives can be reduced by eliminating well-known non-GIs. For example, by filtering out putative highly expressed genes based on codon usage, SIGI-HMM was reported to have the highest precision in previous evaluations (Langille et al., 2008).

For methods performing comparisons with the genomic average, laterally transferred regions may contaminate the genome and reduce the accuracy of predictions (Elhai et al., 2012). Furthermore, the predicted boundaries of GIs are not precise, since the boundaries between laterally transferred genes and native genes can be compositionally ambiguous (Azad and Lawrence, 2011). Additionally, these methods at the gene level require reliable gene annotations. Thus, they may not be applied to newly sequenced genomes, which have no or incomplete annotations.

GI detection methods based on DNA sequence composition

The increase of newly sequenced genomes without complete annotations necessitates GI prediction based on DNA sequences alone. Without the aid of gene boundaries, the large genome has to be segmented by other measures. According to genome segmentation approaches, methods based on DNA sequence composition can be classified into two major kinds: *window-based methods* and *windowless methods*.

Window-based methods Window-based single-threshold methods are commonly used for GI detection. These methods use a sliding window to segment the whole genome sequence into a set of smaller regions. There are several representative programs, including AlienHunter (Vernikos and Parkhill, 2006), Centroid (Rajan et al., 2007), INDeGenIUS (Shrivastava et al., 2010), Design-Island (Chatterjee et al., 2008). The major differences among them are in: the size of the sliding window, the choice of the discrimination criterion and similarity measure, and the determination of the threshold.

AlienHunter uses a fixed-size overlapping window of fixed step size. AlienHunter is the first program for GI detection on raw genomic sequences. It measures segment atypicality via relative entropy based on interpolated variable order motifs (IVOM). The threshold can be obtained by either k-means clustering or standard deviation (when there are fewer samples).

Centroid partitions the genome by a non-overlapping window of fixed size. The average of k-mer frequency vectors for all the windows is seen as the centroid. Based on the Manhattan distances from each frequency vector to the centroid, outlier windows are selected by a threshold derived from standard deviation. INDeGenIUS is a method similar to Centroid. But it uses overlapping windows of fixed size and computes the centroid via hierarchical clustering.

Design-Island is a two-phase method utilizing k-mer frequencies. It incorporates statistical tests based on different distance measures to determine the atypicality of a segment via pre-specified thresholds. In the first phase a variable-size window is used to obtain initial GIs, whereas in the refinement phase a smaller window of fixed size is used to scan over these putative GIs for getting final GI predictions.

Some of these methods are designed to alleviate the problem of genome contamination. Design-Island excludes the initially obtained putative GIs when computing

parameters for the entire genome in the second phase.

To deal with the imprecise GI boundaries that result from a large step size, AlienHunter uses HMM to further localize the boundaries between predicted GIs and non-GIs. But most other programs do not consider this issue.

Windowless methods The few windowless methods mainly include GC Profile (Zhang and Zhang, 2004; Zhang et al., 2014) as well as its extensions and MJSD (Arvey et al., 2009).

GC Profile is an intuitive method to calculate global GC content distribution of a genome with high resolution. The abrupt drop in the profile indicates the sharp decrease of GC content and thus the potential presence of a GI. This method was later developed into a web-based tool which is used for analyzing GC content in genome sequences (Gao and Zhang, 2006). However, other features have to be used together with GC Profile for GI prediction due to the poor discrimination power of GC content. More recently, Wei et al. (2017) developed a webserver called Zisland Explorer, which combines GC Profile with codon usage bias and amino acid bias.

MJSD is a recursive segmentation method based on Markov Jensen-Shannon divergence (MJSD) measure. The genome is recursively cut into two segments by finding a position where the sequences to its left and to its right have statistically significant compositional differences. Subsequently, each segment is compared against the whole genome to check its atypicality via a predefined threshold. MJSD can also be used together with clustering to predict GIs in a bacterial genome (Jani et al., 2016).

Pros and Cons Methods based on DNA sequence composition have the similar advantages and disadvantages as methods based on gene sequence composition.

Specifically, window-based methods can be highly sensitive with appropriate implementations. For example, AlienHunter was reported to have the highest recall in previous evaluations (Langille et al., 2008). But their precisions are quite low due to the limited input information. They are also inherently incapable of identifying the precise boundaries between regions with compositional differences(Arvey et al., 2009).

In contrast, windowless methods can delineate the boundaries between GIs and non-GIs more accurately (Arvey et al., 2009). GC Profile and Zisland Explorer are capable

of discovering a few reliable GIs (Zhang et al., 2014; Wei et al., 2017). But it seems subjective to access the abruptness of jump in the GC profile, and only GIs with low GC content can be detected. MJSD is better at predicting GIs of size larger than 10 kb (Arvey et al., 2009), but the procedure to determine segment atypicality still suffers from the contamination of the whole genome.

GI detection methods based on GI structure

The presence of compositional bias is usually not sufficient to assure the foreign origin of putative GIs. Thus, it is necessary to develop methods based on multiple GI-related structural features. According to the approaches of integrating different features, methods based on GI structure can be divided into *direct integration methods* and *machine learning methods*.

Direct integration methods The direct integration methods adopt a series of filters to get more reliable GIs. But some integrated features are only used for validation, since it is difficult to systematically use them for prediction given the extreme GI structural variation. There are mainly two representative programs: IslandPath (Hsiao et al., 2003) and Islander (Mantri and Williams, 2004).

IslandPath is the first program integrating multiple features (*GC bias*, *dinucleotide bias*, *the presence of tDNAs* and *mobility-related genes*) to aid GI detection. But IslandPath only annotates and displays these features in the whole genome, leaving it to the user to decide whether a region is a GI or not. Based on these computed features, a GI can be identified as multiple consecutive genes with both *dinucleotide bias* and *the presence of mobility-related genes* (IslandPath-DIMOB) (Hsiao et al., 2005).

Islander incorporates a method to accurately detect tDNA-borne GIs. Islander seeks specific tDNA signature to find candidate GIs. Several filters are used to exclude potential false positives, such as regions without integrase genes. Recently, the filtering algorithms are refined via incorporating more precise annotations available now (Hudson et al., 2015).

Machine learning methods Several machine learning approaches based on constructed GI datasets have been proposed, including Relevance Vector Machine (RVM)

(Vernikos and Parkhill, 2008), GIDetector (Che et al., 2010), and GIHunter (Che et al., 2014b). The major differences among them are in the choices of training datasets, GI-related features, and learning algorithms.

RVM is the first machine learning method to study structural models of GIs. It is based on the datasets constructed from comparative genomics methods. Eight features of each genomic region were used to train GI models: *IVOM score*, *insertion point*, *GI size*, *gene density*, *repeats*, *phage-related protein domains*, *integrase protein domains* and *non-coding RNAs*.

GIDetector utilizes the same features and training datasets as RVM, but it implements decision tree based ensemble learning algorithm. GIHunter uses the similar algorithm as GIDetector, but adopts slightly different features and datasets. *GI size* and *repeats* are replaced by *highly expressed genes* and *average intergenic distance*. The training datasets are replaced by IslandPick datasets. The predictions of GIHunter for thousands of microbial genomes are available online at `http://www5.esu.edu/cpsc/bioinfo/dgi/index.php`.

Pros and Cons Methods utilizing GI structure can generate more robust predictions. For example, the high reliability of GIs inserted at tDNA sites leads to very few false positives in the predictions from Islander (Hudson et al., 2015). But these methods depend on accurate identification of multiple related features, such as tRNA genes, mobility-related genes, and virulence factors.

Direct integration methods are straightforward, but many GIs may be filtered out due to the lack of certain features. For example, IslandPath-DIMOB was shown to have very low recall in spite of high accuracy and precision (Langille et al., 2008).

Conversely, machine learning approaches can systematically integrate multiple GI features to improve GI prediction. This can be partly reflected by the high recall and precision of GIHunter (Che et al., 2014b). However, the performance of supervised methods is closely related to the quality of training datasets.

2.2.2 GI detection methods based on several genomes

Methods based on several genomes detect GIs from their sporadic phylogenetic distribution. The comparison procedure often involves analyzing results from sequence

alignment tools (Langille et al., 2010), such as local alignment tool BLAST (Altschul et al., 1997), and whole-genome alignment tool MAUVE (Darling et al., 2004).

BLAST and MAUVE can be used to find unique strain-specific regions (GI candidates), whereas MAUVE can also be used to find conserved regions. For example, Vernikos and Parkhill performed genome-wide comparisons via all-against-all BLAST, and then applied manual inspection to find reliable GIs for training GI structural models (Vernikos and Parkhill, 2008). They also differentiated gene gain from gene loss via a maximum parsimony model obtained from MAUVE alignments. Despite the tediousness of manual analysis, there are only two automatic methods based on several genomes: tRNAcc (Ou et al., 2006) and IslandPick (Langille et al., 2008).

The tRNAcc method utilizes alignments from MAUVE to find GIs between a conserved tRNA gene and a conserved downstream flanking region across the selected genomes. It was later integrated into MobilomeFINDER (Ou et al., 2007), an integrative web-based application to predict GIs with both computational and experimental methods. Complementary analysis is also incorporated in tRNAcc to provide additional support, including GC Profile, strain-specific coding sequences derived from BLAST analysis, and dinucleotide differences. But appropriate genomes to compare have to be selected manually.

To facilitate genome selection, IslandPick builds an all-against-all genome distance matrix and utilizes several cut-offs to select suitable genomes to compare with the query genome, making it the first completely automatic comparative genomics method. The pairwise whole-genome alignments are done by MAUVE to get large unique regions in the query genome. After being filtered by BLAST to eliminate genome duplications, these regions are considered as putative GIs.

Pros and Cons Due to the inaccuracies of composition-based methods, methods based on several genomes are preferred if there are appropriate genomes for comparison (Langille et al., 2008). But uncertainties still exist in their predictions. Firstly the results are dependent on the genomes compared with the query genome (Langille et al., 2008). Secondly, it is hard to distinguish between gene gain via LGT and gene loss (Ravenhall et al., 2015). Thirdly, genomic rearrangements can cause difficulties in accurate sequence alignments (Darling et al., 2004). In addition, the applications of

methods based on several genomes are limited, since the genome sequences of related organisms may not be available for some query genomes.

2.2.3 Ensemble methods for GI detection

Different kinds of methods often predict non-overlapping GIs (Langille et al., 2010) and complement each other (Arvey et al., 2009). To make the best of available methods, ensemble methods have been proposed to combine different methods.

One way of combination is to merge the predictions from multiple programs. This approach is implemented in IslandViewer (Langille and Brinkman, 2009) and EGID (Che et al., 2011). IslandViewer is a web-based application combining three programs: SIGI-HMM, IslandPath-DIMOB, and IslandPick. It provides the first user-friendly integrated interface for visualizing and downloading predicted GIs. Newer versions of IslandViewer include further improvements (Dhillon et al., 2013, 2015), such as improving efficiency and flexibility, incorporating additional gene annotations, and adding interactive visualizations. But the underlying integration method is mainly a union of predictions from individual programs. Unlike IslandViewer, EGID uses a voting approach to combine predictions from five programs: Alienhunter, IslandPath, SIGI-HMM, INDeGenIUS, and PAI-IDA. A user-friendly interface for EGID is provided in the program GIST (Hasan et al., 2012).

Another way of combination is to filter the predictions from one method by other methods. This approach is common for PAI prediction, since it is critical to utilize multiple features to discern PAIs from other GIs. Several PAI detection programs adopt this approach, including PAIDB (Yoon et al., 2005), PredictBias (Pundhir et al., 2008) and PIPS (Soares et al., 2012). These programs often combine composition-based methods, comparative genomics methods, and homology-based methods.

Both PAIDB and PredictBias firstly identify *putative GIs* based on compositional bias. For PAIDB, the *putative GIs* homologous to published PAIs (overlapping with PAI-like regions obtained from homology searches) are seen as candidate PAIs. SIGI-HMM and IslandPath-DIMOB are later integrated into PAIDB for GI predictions (Yoon et al., 2015). To overcome the dependency on known PAIs, PredictBias constructed a profile database of virulence factors (VFPD). If the *putative GIs* (or eight contiguous genes) have a pre-specified number of significant hits to VFPD, they are seen as *potential*

PAIs. PredictBias also integrates comparative analysis to validate the *potential PAIs*.

PIPS integrates multiple available tools for computing PAI-associated features. It filters out the initial predictions from comparative genomics analysis via empirical logic rules on selected features (*GC content*, *codon usage*, *virulence factors* and *hypothetical proteins*).

Pros and Cons Combining the predictions of several programs is supposed to perform better than individual programs. Actually, IslandViewer was shown to increase the recall and accuracy without much sacrifice of precision (Langille et al., 2010), and EGID was reported to yield balanced recall and precision (Che et al., 2011).

The available ensemble methods are mostly characterized by user-friendly interfaces, but the combination procedures do not seem to be sophisticated enough. Some valuable predictions made by one method may be discarded in the ensemble method. For example, PredictBias was shown to have lower sensitivity and accuracy than PIPS on two bacterial strains (Soares et al., 2012), which reflects the effects of different integration strategies on the performances to some extent.

2.2.4 GI detection methods for incomplete genomes

Thanks to low-cost high-throughput sequencing, an increasing number of microbial genomes are being sequenced. However, many of these genomes are in draft status. So there is a need to predict GIs in incomplete genomes. Currently, there are only two programs for this purpose: GI-GPS (Lee et al., 2013) and IslandViewer 3 (Dhillon et al., 2015). Both programs firstly assemble the sequence contigs into a draft genome, and then use methods similar to those for predicting GIs in complete genomes.

GI-GPS is a component of GI-POP, a web-based application integrating annotations and GI predictions for ongoing microbial genome projects. GI-GPS uses an assembler within GI-POP for genome assembly. Then an SVM classifier with radial basis function kernel is applied to segments obtained from a sliding window of fixed size along the genome. The classifier is trained on IslandPick datasets and selected GIs from PAIDB. GI-GPS utilizes compositional features in model training to tolerate potential errors in the assembled genome. The predictions from the classifier are filtered by homologous searches to keep only sequences with MGE evidence. Then the boundaries of filtered

sequences are refined by repeats and tRNA genes.

IslandViewer 3 maps the annotated contigs to a completed reference genome to generate a concatenated genome. Then it uses this single genome as input to the normal IslandViewer pipeline.

Pros and Cons GI-GPS and IslandViewer 3 make it feasible to predict GIs for draft genomes. But they are still simplistic and limited. For example, IslandViewer 3 is restricted to the genome which has very few contigs and reference genomes of closely related strains of the same species (Dhillon et al., 2015). Furthermore, it seems inappropriate to apply methods similar to those developed for complete genomes, since draft genome sequences do not have as high quality as whole genome sequences.

2.2.5 Summary

Since the discovery in microbial genomes, the importance of GIs has been gradually appreciated. Extensive research has demonstrated multiple GI-associated signatures, but these features show great variation in different genomes. Nevertheless, several of these features have been revealed to be effective in GI detection and applied in many computational methods, including compositional bias, structural markers and phylogenetically restricted distribution. Based on the input data, we classify these methods into four large groups, which are further divided into subgroups based on the features utilized or the methodology adopted. It should be noted that some methods may belong to multiple categories. For example, tRNAcc and GI-GPS can also be classified as ensemble methods.

In short, distinct kinds of methods detect GIs based on diverse features and assumptions, and thus generate predictions of different reliabilities. Methods based on gene or DNA composition of a single genome provide only rough estimates, since they usually take advantage of very limited information. Methods based on GI structure utilize multiple lines of evidence, and hence are supposed to be more reliable. But compositional or structural features in a single genome can only provide static information for GI prediction. Instead, methods based on several genomes can reveal genetic flux among closely related genomes and provide dynamic information (Vernikos, 2008). Therefore, they can be more accurate.

It is worth noting that most of the GI prediction methods rely on either annotations or comparisons with other closely related genomes. Hence, these methods cannot be easily applied to new genomes with only DNA sequences. With the rapid development of next-generation sequencing, more and more bacterial genomes are sequenced and less time is allocated for detailed analysis. As a result, many genomes may have no or incomplete annotations. Some genomes may still not have enough closely related organisms to compare with. Therefore, it is necessary to develop better methods for a single newly sequenced genome without relying on available annotations. This is helpful in quickly prioritizing GI candidates for further investigations.

To address this need, we developed two new machine learning methods, GI-SVM and GI-Cluster, which will be presented in Section 2.3 and Section 2.4, respectively.

2.3 Detecting GIs based on k-mer frequency

In this section, we present GI-SVM, which detects GIs using only unannotated sequence from a single genome. GI-SVM is based on one-class support vector machine (SVM), utilizing composition bias in terms of k-mer content. There are composition-based methods like Wn-SVM (Tsirigos and Rigoutsos, 2005a) and svm-agp (Metzler and Kalinina, 2014) that use a one-class SVM to detect *atypical genes* in annotated genomes. Our GI-SVM is different from these methods by detecting GIs in unannotated genomes. From our evaluations on three real genomes, GI-SVM can achieve higher recall compared with current methods, without much loss of precision. Besides, GI-SVM allows flexible parameter tuning to get optimal results for each genome. In short, GI-SVM provides a more sensitive method for researchers interested in a first-pass detection of GI in newly sequenced genomes.

2.3.1 Methods

In this subsection, we first describe the basic idea of GI-SVM method. Then we give an overview of this method, followed by detailed description of major steps for predicting GIs. Finally, we show how to tune parameters in practice.

Basic idea behind GI-SVM

GI-SVM is designed to only utilizes a single unannotated genome sequence. This makes it difficult to achieve high precision due to limited input information. Hence, GI-SVM seeks to improve the recall and F-measure of GI prediction (namely, without too much loss of precision). To achieve this goal, we propose to use *one-class SVM* based on k-mer content of genomic sequence. We believe this is effective because of two reasons: Firstly, one-class SVM classifier is generally used to detect novelty or outlier (Schölkopf et al., 2000) in unlabelled dataset. In our context, one-class SVM is well suited for detecting laterally transferred regions, since these regions can be considered as *outliers* in the unannotated genome in terms of its sequence composition. Secondly, several effective string kernels have been used for biological sequence analysis (Ben-Hur et al., 2008). Hence, we believe that the GI-SVM method using a one-class SVM combined with string kernel describing k-mer spectrum will be effective for GI prediction.

Overview of GI-SVM

Given the unannotated whole genome sequence of a bacterium (FASTA file), GI-SVM will generate a list of genomic intervals as GI candidates. The overall procedure in GI-SVM is illustrated in Figure 2.2. Firstly a sliding window of fixed size is used to scan the genome sequence, moving forward with fixed step size. Then the extracted overlapping windows are passed to one-class SVM and get ranked by the score induced from the decision function. Finally a threshold is used to select windows with lower scores, which are further merged to get GI candidates.

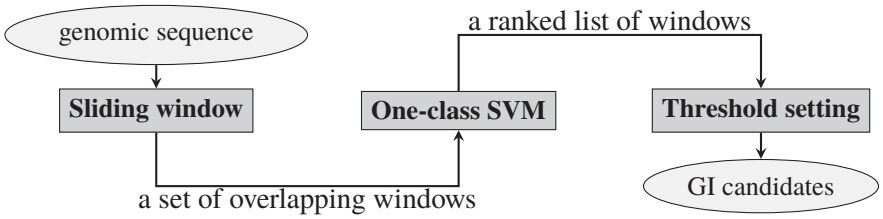

Figure 2.2: The overall procedure in GI-SVM program.

Steps of GI-Cluster for predicting GIs

Since sliding window method is straight-forward, we mainly discuss one-class SVM and threshold setting steps in this subsection.

One-class SVM Given a set of data points, one-class SVM learns decision function f to capture regions containing most of the data points(Schölkopf et al., 2000). The data points outside the boundary are assigned negative values by f, while points within have positive values. The smaller the value, the more *atypical* the data point is. The original data points are often mapped to a feature space by a *kernel function*, which defines similarity of the feature vectors of two data points. Two string kernels are commonly used for biological sequence analysis: spectrum kernel and mixed spectrum kernel.

The spectrum kernel is defined as the inner product of the number of k-mer occurrences in two sequences x, y (Leslie et al., 2002):

$$K_k^{spectrum}(x, y) =< \phi_k(x), \phi_k(y) > \tag{2.1}$$

where $\phi_k(x) = (\phi_a(x))_{a \in B^k}, B = \{A, T, G, C\}$ and $\phi_a(x)$ = number of times k-mer a occurs in sequence x. The mixed spectrum kernel combines the spectrum kernel for k-mers of different lengths with specific weights (Ben-Hur et al., 2008):

$$K_l^{mixedspectrum}(x, y) = \sum_{k=1}^{l} \beta_k K_k^{spectrum}(x, y) \tag{2.2}$$

where β_k is a weight factor for spectrum kernel of k-mers.

In GI detection, the decision function can be seen as a generalized genomic signature (Tsirigos and Rigoutsos, 2005a). Given a set of windows (data points), one-class SVM computes a typicality score for each window based on the kernel used. Hence, we have two variants: GI-SVM$_{\mathrm{S}}$ that uses spectrum kernel, and GI-SVM$_{\mathrm{M}}$ that uses mixed-spectrum kernel. The lower the score, the more likely that the window is alien. The score is scaled into [0, 1], which can be interpreted as the probability of a window being part of a GI. All the windows are then ranked by the score in ascending order.

Threshold setting Given the ranked list of windows from the one-class SVM, a manual or automatic threshold is adopted to select *most atypical* ones as candidate windows belonging to GIs. Then we iteratively merge overlapping candidate windows to generate the list of GI candidates. We also allow *very close* but not overlapping GI candidates to be merged provided that the typicality score in the non-overlapping region is close to the selected threshold.

The setting of automatic threshold is based on the ideal distribution of ranked scores, which is illustrated on *S. typhi* CT18 genome (Figure 2.3a). The ranked scores can be roughly divided into three clusters: small scores with high variance (cluster1, steep part of the curve); scores with immediate variance (cluster2, transition part of the curve); high scores with small variance (cluster3, smooth part of the curve). It is assumed that the majority of sequences in a genome are typical, so cluster3 should correspond to native sequences while cluster1 should correspond to alien sequences. When we use only windows that overlap with the reference GIs (Figure 2.3b), this assumption is further strengthened. Hence, we use a one-dimensional k-means clustering to group the data into three clusters as defined above. The automatic threshold is then set between cluster2 and cluster3 to get higher recall.

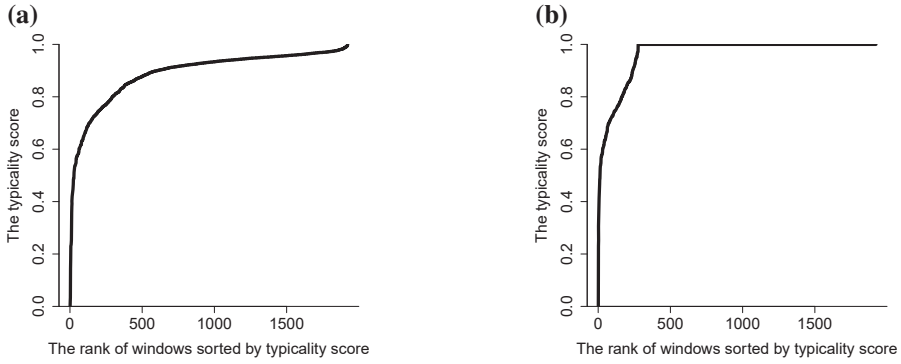

Figure 2.3: The distribution of ranked scores for sliding windows over the *S. typhi* CT18 genome obtained by GI-SVM. (a) The ideal distribution of scores for all the windows. (b) The distribution of scores for windows overlapping with reference GIs.

Parameter selection

In this subsection, we discuss the choices of several important parameters in GI-SVM: one-class SVM parameter ν, k-mer length, window size, and step size. In general, good choices of these parameters rely on the tradeoff between prediction accuracy and computational cost.

Selection of ν The parameter ν represents an upper bound on the fraction of outliers and a lower bound on the fraction of support vectors (Schölkopf et al., 2000). Thus, ν controls the fraction of windows contributing to the generalized genomic signature. The range of ν is (0,1) by definition, which means only a subset of windows will contribute to genomic signature. This is different from the normal genomic signature derived by

counting k-mers in the whole genome, which may be weakened by very atypical regions. The value of ν was explored at ten equidistant values in (0, 1).

Selection of k For GI-SVM$_{\mathrm{S}}$, the value of k for the spectrum kernel determines the mapping of data points into feature space. The typical value of k for LGT detection is 2 to 9, thus k was explored in (2, 9) with a step of one (Langille et al., 2010).

For GI-SVM$_{\mathrm{M}}$, we used k from 1 to 8 with uniform weights in performance evaluation. We note that AlienHunter also integrates k-mer of size 1 to 8. Hence, this can make a nice direct comparison between GI-SVM$_{\mathrm{M}}$ and AlienHunter.

Selection of window size and step size The sliding window method is well known to be sensitive to window size and step size. Large window size decreases resolution while small size is less predictive (Arvey et al., 2009). In evaluations the GI-SVM window size was fixed as 5000 to detect GIs of size closer to 5000 bp. In general, a small step size will generate more overlapping sequences, and thus increase both noise and computational cost. Step size larger than half the window size may cause inaccurate GI localization. Therefore, step size was explored in (1000, 2500) with a step of 500.

2.3.2 Results

In this subsection, we first describe the datasets and metrics used for performance evaluation, followed by the choice of methods to compare with GI-SVM. Then we report the choice of parameters for GI-SVM. Finally, we show the evaluation results.

Evaluation approach

Reference datasets One widely-acknowledged difficulty in GI validation is the lack of gold standard datasets. Fortunately, GIs has been well studied in only a small number of representative organisms, including *S. typhi* CT18 which is a gram-negative pathogen that can cause gastroenteritis and typhoid fever (Marcus et al., 2000), *C. diphtheria* NCTC13129 which is a gram-positive pathogen that causes diphtheria (Cerdeño-Tárraga et al., 2003), and *P. aeruginosa* LESB58 which is a gram-negative pathogen that can cause opportunistic infections in human (Winstanley et al., 2009). The DNA sequence and annotations of each genome were downloaded from NCBI FTP server around

Table 2.3: The general information of the three genomes.

Genome	Accession No.	Genome Size(bp)	GC content(%)	GI size(bp)
CT18	NC_003198.1	4,809,037	52.09	600,077
NCTC13129	NC_002935.1	2,488,635	53.48	473,944
LESB58	NC_011770.1	6,601,757	66.30	451,583

January 2015. Their overall information is summarized in Table 2.3.

For these genomes, the GIs predicted via diverse sources were checked to derive relatively reliable reference GI datasets. For *S. typhi* CT18, 19 GIs were obtained based on those used in a more recent study (Arvey et al., 2009), excluding two GIs of size smaller than 5000 bp. In addition, the GI boundaries for SPI7 were refined based on the annotations in GFF file. For *C. diphtheria* NCTC13129, the GI list was obtained by combining 13 GIs and 10 GIs reported in two studies (Cerdeño-Tárraga et al., 2003; Trost et al., 2012). For *P. aeruginosa* LESB58, 11 GIs reported in the initial comprehensive study (Winstanley et al., 2009) were adopted.

Evaluation metrics The commonly used precision, recall and F-measure (F1 score) were chosen to evaluate GI prediction programs (Langille et al., 2008, 2010; Che et al., 2014b). Their definitions are as follows:

$$Precision = \frac{TP}{TP + FP};\ Recall = \frac{TP}{TP + FN};\ F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

The evaluation is based on the number of protein-coding genes overlapping with GI. A gene is said to be *predicted* if a given fraction of its sequence overlaps with a GI. Here, 50% is used, which can ensure relatively fair comparison. FP refers to the number of *predicted* genes not overlapping with reference GIs. TP refers to the number of *predicted* genes overlapping with the reference GIs. FN refers to the number of genes which overlap with the reference GIs but not the predicted GIs. The accuracy of predicted GI boundary can be evaluated by absolute error $\delta x = |x - x_0|$, where x is the reference boundary and x_0 is the predicted boundary (Vernikos and Parkhill, 2006).

Methods to compare with GI-SVM The results of GI-SVM with optimal parameters were compared with those of four other GI detection programs: AlienHunter, MJSD, Wn-SVM and GIHunter. AlienHunter and MJSD use the same data as our methods and

provide the most direct comparison, whereas Wn-SVM and GIHunter use additional information that our methods do not. These methods are described below.

AlienHunter: AlienHunter is the most popular GI detection program based solely on sequence of a single genome, and was reported to have highest recall in a previous evaluation (Langille et al., 2008). AlienHunter requires no parameters, except one option to optimize predicted boundaries and another option for visualization. Here, AlienHunter without boundary optimization is used for comparison so that both AlienHunter and GI-SVM uses the original sliding window boundary as GI boundary. Besides, the slight refinement of boundary from AlienHunter does not affect much the evaluation.

MJSD: MJSD is another program utilizing only the single genome sequence, which excels at predicting large GIs (Arvey et al., 2009). To compare with MJSD, we ran GI-SVM on the 20 genuine genomes used in their paper (Arvey et al., 2009), and compared them with the GI predictions they reported. As references for these 20 genomes, we used the GIs predicted by both IslandViewer 2 and IslandViewer 3 (September 2015).

Wn-SVM: Wn-SVM (Tsirigos and Rigoutsos, 2005a) is a method that uses a single genome and its gene annotations as input. It uses one-class SVM to predict clusters of transferred genes (GI). To find its good parameter settings, multiple combination of values (window size: 5 to 6, step size: 2 to 3, ν: 0.8 to 0.9, k: 5 to 8) were tested and the best results were adopted. The output of Wn-SVM is in terms of gene, thus we transformed its output into GI predictions by combining adjacent atypical genes in each predicted cluster.

GIHunter: GIHunter (Che et al., 2014b) uses a single genome and gene functional annotation information to perform GI predictions. Hence, it is expected to perform better than GI-SVM. Therefore, it is interesting to do a comparison. The results of GIHunter for the three genomes used were downloaded online.

Finally, to make the comparisons fair we only consider individual GI predictors that use only one genome. Hence, we have excluded the popular IslandViewer method for GI prediction, since it is an aggregate method that uses a combination of two composition-based methods and one comparative genomic method (Dhillon et al., 2015).

Choice of parameters

In general, the optimal range of parameters for GI-SVM are: step size 1500-2500bp; k 5-7 for spectrum kernel; $\nu >= 0.5$. However, the best values for k and ν in the three genomes are different, implying that the optimal parameters of GI-SVM should not be fixed for different genomes. But in most cases the parameters within the suggested optimal range can generate quite good results.

The choice of important parameter ν, k and threshold are discussed below based on the *S. typhi* CT18 dataset. For convenience, step size is fixed to be 2500 bp. Automatic threshold is approximately represented by corresponding fraction of the ranked score list, e.g. threshold 15 means top 15% of the list.

Choice of ν To get reliable GI predictions, the ranking of windows in the ranked list should not vary too much. The rank stability can be measured by Spearman's rank correlation coefficient (Metzler and Kalinina, 2014). If the coefficient is high, the change of ranking relative to adjacent parameters is slight. The results are shown in Figure 2.4a. When ν is very low, the predictions are relatively more unstable. When $\nu \geq 0.5$, the coefficients for k-mers of all sizes become very close to 1. In theory, when ν is larger, the decision boundary is determined by more windows and thus more representative of genomic signature. Therefore $\nu \geq 0.5$ is preferable.

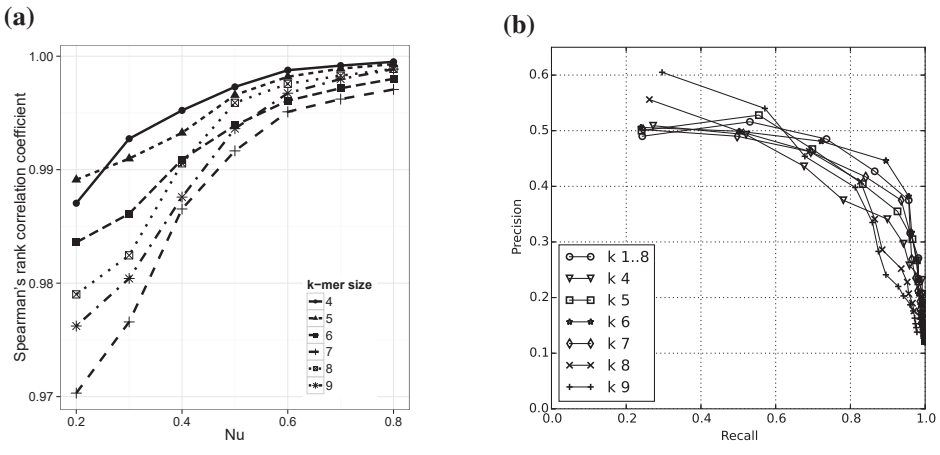

Figure 2.4: The effect of different parameters in GI-SVM on prediction results based on the *S. typhi CT18* dataset. (a) The rank stability for spectrum kernel with k-mers of different size. (b) PRC for k-mers of different size when $\nu = 0.9$.

Choice of k While ν affects the order of window typicality scores, k affects the variances within these scores. Since the ideal score distribution for automatic threshold setting has tail extending to the right, the skewness of score distribution should be less than zero and neither too large nor too small. The experiment results suggest that the intermediate k-mer size 5-7 can ensure desirable skewness value.

To further illustrate the effect of k on GI prediction, the Precision-Recall Curve (PRC) for k-mers of different sizes via selecting different thresholds when $\nu = 0.9$ is shown in Figure 2.4b. For spectrum kernel with $k = 4, 8, 9$, precision reduces a lot when recall becomes higher than 0.8. This is reasonable as lower order k-mers may not provide sufficient discriminative power, while the count of higher order k-mers is not enough to accurately capture genomic signature. To get recall higher than 0.8 with as large precision as possible, it is appropriate to use k-mer size 5-7. The areas under curve (AUC) of combing multiple k-mers is larger than that of most single k-mers. But k-mers of size 5-6 for spectrum kernel have comparable performance as mixed spectrum at multiple thresholds. Thus mixed spectrum kernels (GI-SVM$_{\mathrm{M}}$) is not necessarily better than spectrum kernel (GI-SVM$_{\mathrm{S}}$).

Choice of threshold As suggested by Figure 2.4b, a smaller threshold can lead to higher precision, and precision declines as the threshold increases. At the same time, recall increases to one at higher threshold since more predictions are included. As a result, F-measure firstly keeps increasing and then begins to decrease at certain threshold, which is around top 15% to 20% of the ranked score list. The thresholds chosen automatically for different parameter combinations are very close to the optimal threshold when $\nu > 0.4$, suggesting the automatic threshold setting method can provide reasonably good performance.

Performance evaluation and comparison

In this section, we firstly discuss the overall performance of different GI prediction programs. Then we compare GI-SVM with AlienHunter in detail. Finally we present the comparisons of GI-SVM with Wn-SVM, GIHunter and MJSD.

In general, GI-SVM is effective in identifying previously detected GIs with higher sensitivity (Table 2.4 and 2.5). The optimal results of GI-SVM$_{\mathrm{S}}$ and GI-SVM$_{\mathrm{M}}$ are

both shown. GI-SVM$_S$ and GI-SVM$_M$ achieved highest recall and second highest F-measure for *S. typhi* CT18 and *P. aeruginosa* LESB58, respectively. For *C. diphtheria* NCTC13129, GI-SVM even achieved best F-measure and precision.

The differences of sensitivity among GI-SVM$_S$, AlienHunter, Wn-SVM, and GI-Hunter are also illustrated by SPI1 on *S. typhi* CT18 genome (Figure 2.5), a well-characterized GI of size 39,773 bp, encoding tens of genes for penetrating the intestinal epithelium (Marcus et al., 2000). As can be seen, GI-SVM$_S$ predicted two large intervals, covering most of SPI1 (about 86%), while the two predictions from AlienHunter only cover about 67%. Wn-SVM predicted multiple short intervals due to reliance on gene boundaries, covering about 51% of SPI1. GIHunter only predicted about 24% of SPI1, probably due to lack of classical GI features in SPI1.

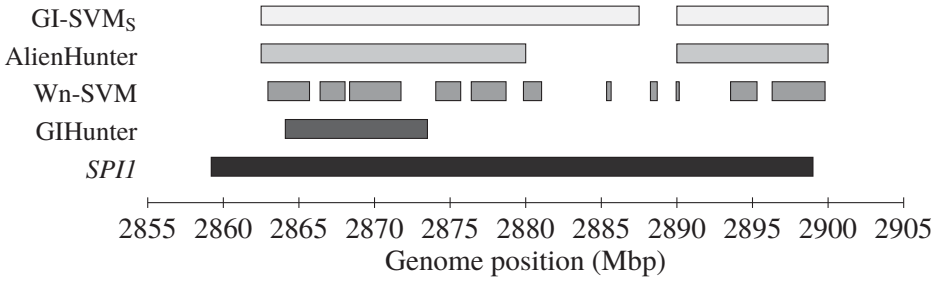

Figure 2.5: The prediction results for SPI1 on *S. typhi* CT18 genome from multiple GI prediction programs.

Comparison with AlienHunter We now compare GI-SVM with AlienHunter comprehensively, as they both use only unannotated genome sequence as input.

In terms of recall and F-measure, GI-SVM achieved much higher recall and slightly higher F-measure than AlienHunter on all the three genomes (Table 2.4). For example, the F-measure of GI-SVM when using spectrum kernel was 7.6% higher than that of AlienHunter on *S. typhi* CT18 dataset, and the recall of GI-SVM$_S$ was 27.5% higher. Using mixed spectrum kernel with the same v, GI-SVM had both higher recall and precision than AlienHunter. In addition, the boundary errors of predictions from GI-SVM on P. aeruginosa LESB58 genome were much lower than those of AlienHunter, suggesting the effectiveness of GI-SVM to some extent.

In consistence with higher recall, the predictions from GI-SVM were less fragmented for large GIs. Take results on *S. typhi* CT18 dataset for example. In total, there are 8 reference GIs with predicted gap (regions of reference GIs not covered by predicted

Table 2.4: The comparison of GI-SVM$_S$, GI-SVM$_M$ and AlienHunter on three genomes. The improvement of GI-SVM over AlienHunter in terms of relative percentage is shown in brackets. #Gene/#GI represents the predicted number of genes and GIs respectively. ABE represents Average boundary error. The parameters for GI-SVM$_S$ on the three dataset are: $\nu = 0.9, k = 6$; $\nu = 0.8, k = 5$; $\nu = 0.5, k = 6$, respectively. The parameters for GI-SVM$_M$ are: $\nu = 0.9$; $\nu = 0.7$; $\nu = 0.5$, respectively.

Data	Tools	#Gene /#GI	Recall	Precision	F1	ABE (bp)
CT18	GI-SVM$_S$	997/96	0.895 (27.5%)	0.446 (-2.4%)	0.596 (7.6%)	3,657
	GI-SVM$_M$	869/91	0.783 (11.5%)	0.448 (-2.0%)	0.565 (2.9%)	3,210
	AlienHunter	764/86	0.702	0.457	0.554	3,297
NCTC 13129	GI-SVM$_S$	997/86	0.639 (8.3%)	0.277 (18.4%)	0.386 (15.2%)	7,583
	GI-SVM$_M$	822/73	0.539 (-8.6%)	0.282 (20.9%)	0.372 (11.0%)	6,842
	AlienHunter	756/69	0.590	0.234	0.335	5,683
LESB 58	GI-SVM$_S$	891/96	0.796 (22.3%)	0.394 (-6.2%)	0.527 (3.3%)	3,338
	GI-SVM$_M$	849/86	0.803 (23.3%)	0.416 (-0.7%)	0.549 (7.6%)	2,834
	AlienHunter	684/71	0.651	0.420	0.510	5,931

Table 2.5: The comparison of GI-SVM$_S$, Wn-SVM and GIHunter on three genomes. The improvement of GI-SVM$_S$ over Wn-SVM and GIHunter in terms of relative percentage is shown in brackets. The parameters for Wn-SVM on the three dataset are: $\nu = 0.9, k = 8, w = 6, s = 3$; $\nu = 0.8, k = 7, w = 6, s = 3$; $\nu = 0.8, k = 8, w = 6, s = 3$, respectively.

Data	Tools	#Gene /#GI	Recall	Precision	F1	ABE (bp)
CT18	GI-SVM$_S$	997/96	0.895	0.446	0.596	3,657
	Wn-SVM	695/361	0.515 (73.8%)	0.368 (21.2%)	0.430 (38.6%)	2,333
	GIHunter	608/23	0.826 (8.2%)	0.676 (-34.0%)	0.743 (-19.9%)	10,396
NCTC 13129	GI-SVM$_S$	997/86	0.639	0.277	0.386	7,583
	Wn-SVM	427/246	0.266 (140.0%)	0.269 (3.0%)	0.268 (44.0%)	3,883
	GIHunter	1248/14	0.900 (-29.0%)	0.216 (28.2%)	0.348 (10.6%)	106,561
LESB 58	GI-SVM$_S$	891/96	0.796	0.394	0.527	3,338
	Wn-SVM	922/432	0.721 (10.4%)	0.345 (14.2%)	0.467 (12.8%)	1,374
	GIHunter	511/16	0.740 (7.4%)	0.640 (-38.4%)	0.687 (-23.3%)	22,117

Table 2.6: The size of predicted gap (regions of reference GIs not covered by predicted GIs) from AlienHunter and GI-SVM$_S$ on *S. typhi* CT 18 genome.

GI	GI Start/End	Gap from AlienHunter(bp)	Gap from GI-SVM$_S$(bp)
SPI6	302,172/361,067	12,499	12,500
Prophage10	1,008,747/1,051,266	29,999	2500; 5000
Bacteriophage	1,887,450/1,933,558	17,499	10,000
Bacteriophage27	2,759,733/2,782,364	9999	2500
SPI1	2,859,262/2,899,034	9999	2500
Bacteriophage	3,515,397/3,549,055	2499; 4999	2500
SPI7	4,409,511/4,543,148	2499; 9999; 2499	2500
SPI10	4,683,690/4,716,539	2499	0

GIs) for AlienHunter and 7 for GI-SVM$_S$ (Table 2.6). The gaps from GI-SVM$_S$ were much smaller than those from AlienHunter. For SPI7, GI-SVM$_S$ predicted two regions covering about 98% of this GI, whereas AlienHunter predicted four regions covering only 85%. Besides, the gap from GI-SVM$_S$ was only 2500 bp, which composes of several genes of bacteriophage origin whose GC content (51.57%) is quite similar to that of the genome (52.09%). The similarity in GC content confirms the typicality of gap region. Thus it is hard to find these typical regions inside GI via a single threshold. By merging adjacent windows with distance not larger than 2500 bp, the sensitivity of GI-SVM can be further improved.

The sensitivity of AlienHunter and GI-SVM$_S$ versus the inferred relative time of insertion of laterally transferred genes present in the reference GIs is also compared (Figure 2.6a), as composition-based methods are known to be biased toward finding recently transferred regions. The relative time of insertion of these genes was obtained from a whole-genome comparative study of 15 closely related genomes with *S. typhi* CT18 as the query genome (Vernikos et al., 2007). On the x-axis, the time of insertion increases from left to right. Namely, 'node1' represents the earliest time of insertion while 'CT18' represents the most recent time. Obviously, GI-SVM$_S$ detected more alien genes at the same time of insertion, especially for more recent insertions. In addition, GI-SVM$_S$ was more sensitive in detecting certain kinds of genes over-represented in GI, such as phage-related, mobile elements (transposase, integrase), hypothetical proteins (Langille et al., 2010) (Figure 2.6b). This again shows the higher sensitivity of GI-SVM over AlienHunter.

The advantage of GI-SVM over AlienHunter is mostly owing to the use of one-class SVM, which is superior than relative entropy used in AlienHunter. One-class SVM was shown to be more sensitive in measuring sequence similarity between alien and native sequences (Tsirigos and Rigoutsos, 2005a). One merit of one-class SVM is that not all sliding windows contribute to the genomic signature. This is more reasonable, because very atypical regions may weaken the signature of genomic background. Furthermore, spectrum kernel is combined with one-class SVM to obtain more sensitive measure to judge the typicality of sequences. Besides, GI-SVM allows users to tune parameters for the genomes at hand, whereas AlienHunter utilizes fixed parameters which may not be applicable for all genomes since different genomes have different features.

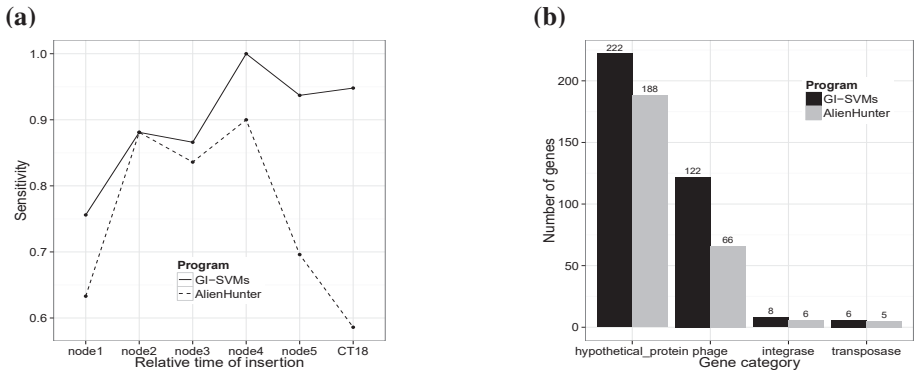

Figure 2.6: The comparisons on the sensitivity of GI-SVM$_S$ and AlienHunter in detecting genes of certain time of insertion and specific function. The numbers of genes within reference GIs for each time point from left to right are 90, 42, 67, 10, 270, and 58, respectively. (a) The sensitivity of GI prediction versus the relative time of insertion of alien genes. (b) The number of certain kinds of genes detected by GI-SVM$_S$ and AlienHunter.

Comparison with Wn-SVM Then we compare GI-SVM with Wn-SVM which uses gene anotation of one genome. The performances of Wn-SVM were no better than GI-SVM$_S$ on all the three genomes (Table 2.5). Overall Wn-SVM predicted much more shorter intervals and has lower F-measure. This is probably because that some genes inside GI may not be atypical enough to be identified as laterally transferred. But the boundary errors of Wn-SVM were much lower with the help of accurate atypical gene predictions.

One most significant difference of GI-SVM from Wn-SVM is that GI-SVM is designed specifically to work on DNA segments rather than genes. Since GI is a cluster of genes, it seems better to detect them as a whole instead of detecting individual genes. Besides, window-based methods are more sensitive in detecting LGTs than gene-based methods according to previous benchmark study (Becq et al., 2010). Another major difference between GI-SVM and Wn-SVM is that GI-SVM utilizes string kernel, which is more efficent than liner kernel on extracted k-mer feature vectors.

Comparison with GIHunter Next, GI-SVM is compared with GIHunter that utilizes multiple kinds of functional annotations in one genome. GIHunter had highest F-measure on two genomes than GI-SVM$_S$ (Table 2.5). Thus the overall performance of GI-SVM was still inferior to GIHunter, since GIHunter utilizes additional annotations as input. But GI-SVM$_S$ had higher recall than GIHunter on two genomes and higher F-measure on one genome. Besides, GIHunter performed poorly in terms of boundary

error, which is mostly caused by large predictions covering several adjacent reference GIs and short predictions covering a small part of GI. Without relying on annotations, the boundary errors of GI-SVM$_S$ were much lower than GIHunter, due in part to higher recall. This indicates that gene-based method is not necessarily better than window-based method in predicting GIs, as the boundary error will still be very high if gene-based method makes wrong predictions.

Comparison with MJSD Lastly, the predictions of GI-SVM$_S$ ($\nu = 0.9, k = 6$) is compared with previous results of MJSD. MJSD predicted 477kb out of 605kb of DNA contained in 21 GIs in *S. typhi* CT18 (Arvey et al., 2009). GI-SVM$_S$ predicted about 530.5 kb out of 600 kb of DNA encoded by 19 GIs. Thus more of each GI is identified by GI-SVM$_S$. For the 20 genomes used in MJSD paper, evaluations of GI-SVM$_S$ predictions on IslandViewer 2 and IslandViewer 3 dataset had similar results, so we adopted evaluations based on IslandViewer 2 which are expected to be closer to the dataset MJSD used. MJSD predicted 97% of the bases detected by all the three components of IslandViewer, while GI-SVM$_S$ predicted 99%. For the regions identified by two components, MJSD identified 74% of bases on average, while GI-SVM$_S$ detected 97% on average. For regions detected by only one component, MJSD predicted around 45% of bases, whereas GI-SVM$_S$ predicted about 74%. These results clearly suggest the higher sensitivity of GI-SVM. For the genomic regions not predicted by any component of IslandViewer, GI-SVM$_S$ only predicted about 1% of the bases. Therefore, the precision of GI-SVM is still reasonable given the much higher recall.

2.4 Detecting GIs by integrating multiple evidence

In this section, we present GI-Cluster, which provides a novel way of integrating multiple GI-related features via consensus clustering. This is different from previous methods which use established thresholds to separate GIs from non-GIs based on specific features, and hence reduces the opportunity of missing real GIs. The comprehensive evaluations on several real genomes show that GI-Cluster can achieve a good balance of recall and precision. Although GI-Cluster does not require training datasets as supervised learning methods, it still has comparable or better performance. GI-Cluster applies to not only a complete genome but also an incomplete genome without the requirement of contig

assembly. GI-Cluster can also be used to improve GI predictions from programs with high recall but low precision. In addition, GI-Cluster incorporates visualization to show the predicted GIs along with related features or GIs obtained from different methods.

2.4.1 Methods

In this subsection, we first describe the basic idea of GI-Cluster method. Then we give an overview of this method, followed by detailed description of major steps for predicting GIs and visualization utility. Finally, we show revisions required to apply GI-Cluster to incomplete genomes and initial GI predictions from other programs.

Basic idea behind GI-Cluster

The challenge in integrating multiple GI-related features is to effectively integrate the predictive power of these features. GIs are genomic regions with several well-known features, but few methods have succeeded to take full power of these features to locate GIs accurately in a genome. To addresses this issue, GI-Cluster utilizes consensus clustering to detect GIs from a genome sequence by combining separate clusterings of genomic segments obtained on multiple GI-related features.

Clustering is commonly used to group similar objects. Since GIs and non-GIs have different features, it is very likely that they form different groups in the feature space. Thus, it seems natural to apply clustering to detect GIs. The unsupervised nature of clustering also makes it widely applicable in the absence of reliable GI training datasets for now. Clustering for GI detection has been applied on sequence compositional features (Azad and Lawrence, 2011; Jani et al., 2016). However, no clustering methods have been performed on multiple kinds of features.

Consensus clustering simply refers to obtaining a single clustering for a particular dataset by integrating various clusterings on the same dataset. It is often used to combine multiple runs of the same clustering algorithm or different clustering algorithms. Bu combining multiple weak learning methods into a strong learning method, consensus clustering can bring benefits that a single clustering algorithm cannot achieve (Vega-Pons and Ruiz-Shulcloper, 2011). The consensus clustering methods based on objects co-occurrence can be naturally applied to GI prediction. The reason that this kind of methods work for GI prediction is as follows. In general, for a single feature, a higher

or lower metric value of a region indicates this region is more atypical. Given a set of segments in a genome sequence, different kinds of features delineating a segment may lead to different partitions of these segments. If several subsets of features suggest that a segment is closer to other segments that show evidence of GIs, this segment is more likely to be a real GI. By identifying segments present in the same group in most partitions, we can get more stable pairwise relationships between segments and hence more robust clustering of these segments.

Overview of GI-Cluster

The overall procedure followed by GI-Cluster is shown in Figure 2.7. Briefly speaking, there are four major steps in predicting GIs:

1. Splitting the genome sequence into a set of non-overlapping segments.
2. Extracting multiple GI-related features for each segment.
3. Performing consensus clustering to partition all segments into two groups, by firstly performing separate clustering on each feature.
4. Postprocessing the clustering result to find potential GI candidates.

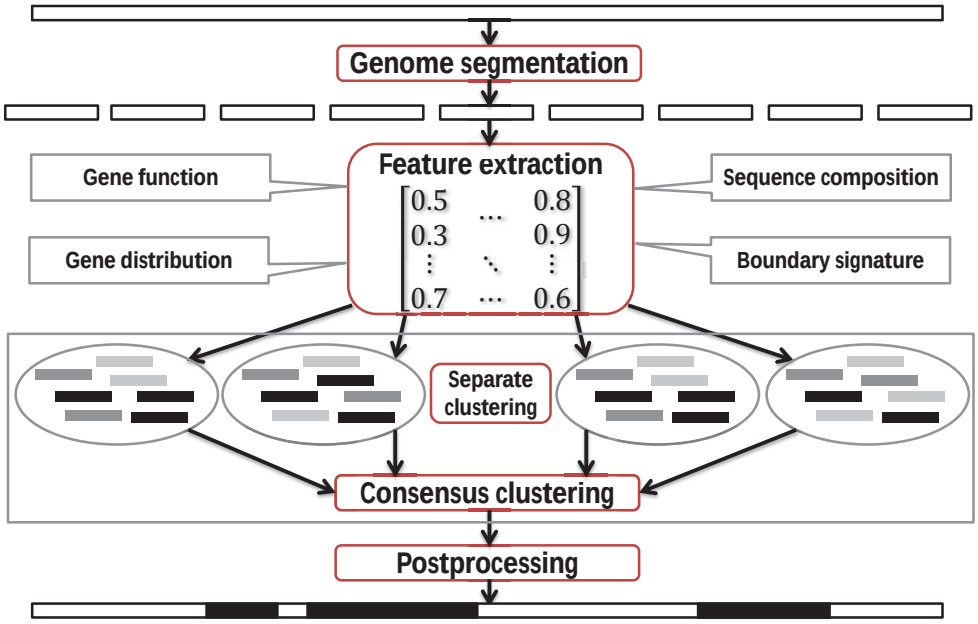

Figure 2.7: The overall procedure adopted in GI-Cluster for identifying genomic islands in a genome sequence.

Steps of GI-Cluster for predicting GIs

In this subsection, we describe the four major steps utilized in GI-Cluster to predict GIs. Due to space limit, the details for each step are described in Appendix A.

Genome segmentation step Firstly, we split the complete genome sequence into non-overlapping short segments to facilitate feature extraction, since GI-related features are often related to a short genomic region. The size of genome segments affects the measurement of GI-related features and thereby downstream analysis. The boundaries of GI candidates are also limited by the initial segmentation, since the subsequent procedures are mainly used to select segments likely to be GIs and refine boundaries locally.

There are two genome segmentation methods related to GI detection: GC profile (Zhang and Zhang, 2004) and MJSD (Arvey et al., 2009). For these two methods, different sets of parameters may lead to distinct segments. Moreover, some segments may be very large and bury the real segmentation points between normal regions and atypical regions. Therefore, for simplicity we use non-overlapping sliding windows of fixed size to segment the genome. Nevertheless, GI-Cluster is still applicable to the segments output by GC profile and MJSD. To find GIs of size larger than 5000 bp, the default window size in GI-Cluster is set to be 5000.

Feature extraction step After obtaining a set of segments, we extract GI-related features for each segment. Previous studies have figured out multiple features useful for discriminating GIs from other regions (Tsirigos and Rigoutsos, 2005b; Langille et al., 2010). Since it is hard to perform feature selection without benchmark datasets, we choose features to compute mainly based on known results in literature.

There are four groups of features computed for a genomic segment: (1) sequence composition; (2) gene function; (3) boundary signature; and (4) gene distribution. For each group, there are several criteria measured by different metrics, which are computed via custom scripts or homologous search against related databases (Table S1).

Since the computation of some features is in the unit of genes, we predict genes from genome sequence via available gene prediction tool. Known gene predictions from NCBI can also be used. However, the gene prediction step is optional. When there are

no gene annotations, GC content and k-mer frequency can be used for predicting GIs.

We compute compositional bias and predict function for each gene and segment. For each gene, we compute several metrics measuring its GC content, codon usage, and k-mer frequency. We also find whether a gene encodes specific functions related to GIs, including mobility, phage, virulence factor, antibiotic resistance, novel function (with no detectable homologs in public databases) and non-coding RNA (ncRNA). For each segment, we compute several metrics measuring its k-mer frequency and GC content. When gene annotations are available, for each segment we also compute several metrics measuring its codon usage and the percentage of each kind of genes within it.

We compute boundary signature and gene distribution only for each segment. We identify the presence of flanking tRNA genes or short repeats around the boundaries of each segment. We also compute gene density (the number of genes per kilo base in a region) and intergenic distance (the average gene distance in a region) for each segment.

After feature extraction, we compute a data matrix with n items and q features to serve as input for clustering. In the matrix, each item represents a genomic segment. By default, we use only features related to sequence composition and gene function for clustering. The remaining features are mainly used to postprocess the results of consensus clustering.

Consensus clustering step After getting the $n * q$ feature matrix, we run consensus clustering on it to obtain two clusters: GIs and non-GIs. There are two steps in consensus clustering: (1) generation step, in which different clusterings are generated; (2) consensus step, in which these diverse clusterings are combined into a unique solution.

In the first step, we use clustering to partition the segments into different groups based on each feature. We compute a $n * n$ connectivity matrix for the clustering result on each feature, in which an entry has value 1 if the corresponding items belong to the same cluster and has value 0 otherwise (Monti et al., 2003).

In the second step, we compute a consensus matrix CM to denote the agreement among the different clusterings on different features (Monti et al., 2003; Wilkerson and Hayes, 2010). Each entry in the consensus matrix represents the proportion of times that two items are clustered together in different runs. The higher the number, the more

likely two items are grouped together. The consensus matrix is obtained by averaging over the connectivity matrices of different clusterings.

The matrix $1 - CM$ can be seen as a distance matrix, on which different clustering methods can be applied to get two clusters. For the interpretation of clustering results, we assign the cluster of smaller size as the GI group, since GIs are usually a small part of the whole genome.

There are several parameters that affect the clustering results, including the choice of GI-related features, the clustering algorithms applied on each feature, and the clustering algorithm applied on the final distance matrix. The choices of these parameters are discussed in Appendix A.

Postprocessing step Given the two groups of segments obtained from consensus clustering step, we use empirical rules based on non-compositional features to pick potential true positives from initial non-GIs and exclude false negatives from initial GIs. For example, an initial non-GI segment with the presence of flanking tRNA gene(s) and mobility-related genes is reclassified as a GI segment, and an initial GI segment showing no evidence in terms of gene function is redesignated as a non-GI segment. These empirical rules affect the accuracy of final output. Strict rules may reduce recall and improve precision, whereas relaxed rules may improve recall and reduce precision.

For a segment belonging to the GI group, we refine its boundaries according to the positions of flanking tRNA genes or short repeats and the closest genes. Then we obtain final GI candidates by merging adjacent segments whose pairwise distance is less than a threshold.

Visualization of predicted genomic islands

GI-Cluster provides two kinds of plots to visualize the predicted GIs. One is feature plot, showing the distribution of predicted GIs and related features along a complete genome on multiple tracks of a circular plot. The other is comparison plot, showing GIs predicted in a complete genome by different methods on multiple tracks of a circular plot. GI-Cluster generates bitmap and vector images for both plots.

These plots help to obtain a more comprehensive picture of potential GIs and provide useful information for further analysis. The feature plot can be used to show

predicted GIs not only from GI-Cluster but also from other methods. It clearly indicates evidence supporting each GI, which may help to remove false positives and recover false negatives. The comparison plot demonstrates the similarities and differences of different predictions, which may help to explore the complementary properties of different methods and find more reliable GI candidates.

Adaptations for incomplete genomes and initial GI predictions

GI-Cluster can be readily adapted to incomplete genomes, since it takes a set of genomic segments as input and does not require the ordering of these segments. Given an incomplete genome, we just use non-overlapping sliding windows of fixed size on different contigs to get a set of segments. One major difference is that we also record the contig from which a segment is obtained from. The subsequent steps are similar to those for a complete genome sequence.

The input genomic segments for GI-Cluster can also be GI candidates detected by methods with high recall but low precision. In this case, GI-Cluster can serve as a filtering tool of the initial GI candidates. We assume most of these GI candidates are real GIs, and hence assign the cluster of larger size as the GI group in the consensus clustering step. The rules for postprocessing are also slightly different from those when the input is the genome sequence.

2.4.2 Results

In this subsection, we first describe the datasets and metrics used for performance evaluation, followed by the choice of methods to compare with GI-Cluster. Then we report the evaluation results on complete genomes, incomplete genomes and initial GI predictions, respectively. Finally, we show the visualizations of GI-Cluster for a selected genome.

Evaluation approach

Reference datasets To get a rough estimation of the performance of GI-Cluster, we used evaluation datasets adopted by Wei et al. (2017), which include GIs collected from literature (L-data set) and GIs predicted by comparative genomics (C-data set) in 11 bacterial genomes. The GIs collected from literature are usually more reliable because

their structure and functions may have been studied in more detail. But there may be some real GIs not reported in previous literature.

Among the 11 genomes, we excluded two genomes with fewer than six GIs in L-data set. For L-data set, we added additional GIs or refined island boundaries according to GIs from Islander, or from literature, or from NCBI annotations. We also included GIs reported in the genome of *Pseudomonas aeruginosa* LESB58 in L-data set (Winstanley et al., 2009). As a result, 10 genomes from six orders were used, including *Burkholderiales* (*Burkholderia cenocepacia* J2315, *Bordetella petrii* DSM 12804), *Corynebacteriales* (*Corynebacterium diphtheriae* NCTC 13129), *Enterobacteriales* (*Cronobacter sakazakii* ATCC BAA-894, *Escherichia coli* CFT073, *Proteus mirabilis* HI4320, *Salmonella typhi* CT18), *Rhizobiales* (*Bartonella tribocorum* CIP 105476), *Lactobacillales* (*Streptococcus equi* 4047), and *Pseudomonadales* (*Pseudomonas aeruginosa* LESB58).

ICEs are a specific group of GIs that encode their own self-conjugative transfer and integration (Bellanger et al., 2014). The structure and function of many ICEs have been well-studied. To further evaluate the performance of selected GI prediction methods, we also checked how they recover seven known ICEs from the ICEberg database (Bi et al., 2012).

There are very few known GIs from incomplete genomes in literature. We collected 17 known GIs (312,686 bp) from *Vibrio cholerae* RC9, which were predicted as five or more consecutive ORFs sporadically distributed among related strains (Chun et al., 2009). The incomplete genome of *V. cholerae* RC9 (Accession: ACHX00000000) contains 11 contigs and 4,211,011 bp. *V. cholerae* RC9 has two chromosomes, with 11 GIs locating at the large chromosome and 6 GIs at the small chromosome. But it is unknown which contigs are from which chromosomes. According to the reference GIs, contig 7 and 9 are on the small chromosome, whereas contig 6, 10 and 11 are on the large chromosome.

More details of the evaluation datasets are provided in Appendix A. The complete lists of GIs used for evaluation are available on `https://github.com/icelu/GI_Cluster/tree/master/evaluation`.

Evaluation metrics Several commonly used metrics were chosen to get comprehensive evaluations, including recall (TPR, sensitivity), precision, F-measure (F1 score), true negative rate (TNR, specificity), overall accuracy (OACC), accuracy (ACC), Matthews correlation coefficient (MCC) and the average of absolute error (ABE) in predicted boundaries (Langille et al., 2008; Lu and Leong, 2016b; Wei et al., 2017). Recall measures the proportions of predicted GIs that are in the reference GIs. Precision measures the proportions of reference GIs that are in the predicted GIs. F-measure is the harmonic mean of recall and precision. TNR measures the proportions of predicted non-GIs that are not in the reference GIs. OACC measures the proportions of correctly predicted GIs and non-GIs. ACC is the arithmetic mean of recall and TNR. MCC is a correlation coefficient between predictions and references. ABE measures the difference between reference boundary and predicted boundary. Their definitions are provided in Appendix A.

Methods to compare with GI-Cluster To make fair comparisons, we mainly compared GI-Cluster with GIHunter, a supervised machine learning method based on similar GI structural features. GIHunter utilizes eight features, including tRNA genes, integrase, transposase, phage-related genes, highly expressed genes, gene density and average intergenic distance. These features are computed from NCBI annotation files. GIHunter can give very accurate predictions and outperformed several commonly used methods in previous evaluations (Che et al., 2014b). The predictions of GIHunter for the selected genomes were downloaded online.

As a reference, we also compared GI-Cluster with IslandViewer. The most recent version of IslandViewer (Bertelli et al., 2017) also integrates the predictions from Islander. The integration makes IslandViewer has better accuracy than each of the four individual programs. Since we used GIs from Islander as references in L-data set, we mainly show the comparisons with predictions by at least one of the other three programs. Because the predictions from Islander are very few, excluding the predictions from Islander does not affect the performance of IslandViewer too much. We downloaded the integrated predictions for each selected genome from IslandViewer 4 (May 2017). Some intervals predicted by different programs may be overlapping, so we merged them before evaluation.

Performance evaluation on real biological datasets

Evaluations on L-data set and C-data set We first compare GI-Cluster (with default parameters) to GIHunter and IslandViewer on L-data set and C-data set (Figure 2.8). In general, IslandViewer had lower recall and higher precision, whereas the recall and precision of GI-Cluster and GIHunter were comparable. Moreover, GI-Cluster had more balanced recall and precision, as suggested by the less variation in the values of F1 and MCC. In addition, GI-Cluster had better performance than GIHunter on some datasets (Table S3).

IslandViewer had lowest average recall (65.7% on L-data set, 54% on C-data set) but highest precision (80% on L-data set, 89.4% on C-data set). Because of varying recall on different genomes, the F1, ACC and MCC of IslandViewer varied a lot despite that the average values were very high. The TNR and OACC of IslandViewer were stable and highest. The possible reason is that the programs integrated in IslandViewer are very specific but less sensitive (Langille and Brinkman, 2009).

GIHunter had highest average recall (83.3% on L-data set and 71.2% on C-data set). The average recall of GI-Cluster (77.5% on L-data set and 67.5% on C-data set) was slightly lower than GIHunter. But the precisions of GIHunter on different genomes varied more than GI-Cluster. Owing to many false positives in the predictions of GIHunter, the TNR, OACC and MCC of GIHunter had wider ranges than those of GI-Cluster.

Regarding the boundary errors, the average ABE of both right and left boundaries of predicted GIs was very high for GIHunter (60,088 bp), followed by that of IslandViewer (8,973 bp) and GI-Cluster (6,491 bp). This suggests that GI-Cluster can generate more accurate GI boundaries.

Evaluations on known ICEs Then we show the performances of GI-Cluster (with default parameters), GIHunter and IslandViewer on predicting seven known ICEs. As shown in Table 2.7, GI-Cluster predicted most of these ICEs with high precision.

Although GIHunter recovered almost all of these ICEs, it often predicted very large intervals and had low precision. For example, GIHunter often predicted a very large region covering one GI or several adjacent GIs. Therefore, it is still challenging to accurately locate each ICE from the predictions of GIHunter owing to the large

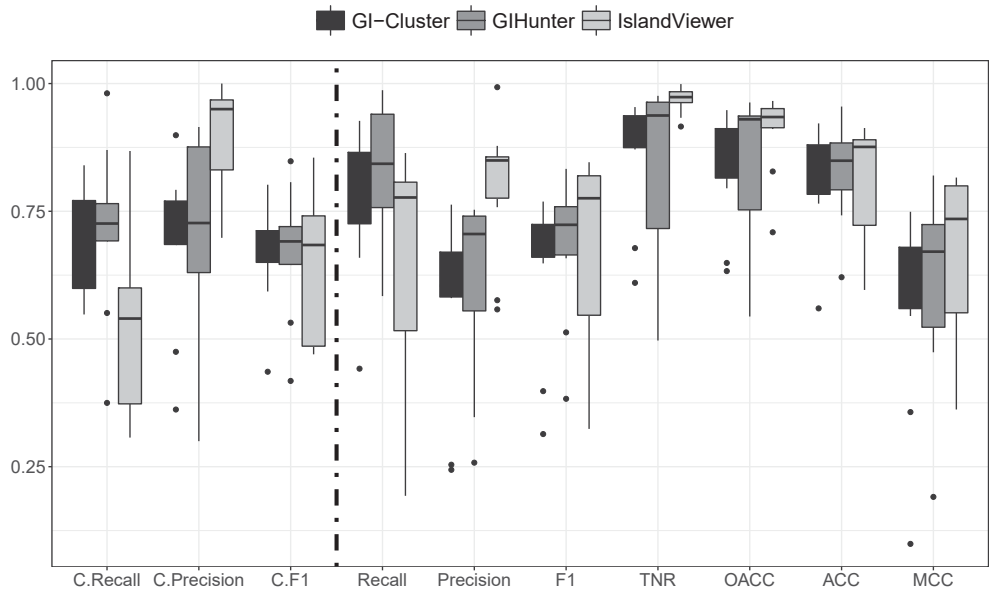

Figure 2.8: Performance comparison of GI-Cluster with GIHunter and IslandViewer on L-data set and C-data set. C.Recall, C.Precision, and C.F1 represent the recall, precision and F1 computed on C-data set. The other metrics were computed on L-data set.

distances between them. In contrast, GI-Cluster and IslandViewer localized GIs and their boundaries more accurately.

Overall, the performance of GI-Cluster is comparable with IslandViewer. In addition, the percentages of bases predicted by GI-Cluster for six ICEs were higher than those predicted by IslandViewer, except for ICEPm1. ICEPm1, a well-studied GI carrying many virulence factors, was firstly detected by a strain-specific comparative genomic hybridization array (Flannery et al., 2009). Therefore, some regions within ICEPm1 may not have atypical composition. By relaxing rules in postprocessing (reclassifying regions with a smaller proportion of phage-related genes or virulence factors or novel genes as GIs), GI-Cluster predicted 98.3% of ICEPm1 with 86.85% precision.

Evaluations on an incomplete genome Next we show the comparisons of GI-Cluster and IslandViewer on the incomplete genome of *V. cholerae* RC9. Generally speaking, GI-Cluster was more sensitive than IslandViewer. Given the 11 contigs from *V. cholerae* RC9, GI-Cluster (hierarchical clustering with ward linkage for consensus clustering) predicted 34 intervals (584,586 bp), covering 11 reference GIs (182,001 bp), whereas IslandViewer predicted 17 large intervals (493,608 bp), covering 8 reference GIs (175,045 bp).

Table 2.7: The percentage of bases of known ICEs predicted (recall, REC for short) and the percentage of predicted bases overlapping with known ICEs (precision, PEC for short) by GI-Cluster, GIHunter and IslandViewer.

ICE	Size (bp)	REC (PEC)		
		GI-Cluster	GIHunter	IslandViewer
ICEPmiUSA1	80,631	99.84 (100)	100 (33.47)	96.76 (98.02)
ICEPm1	92,462	68.35 (82.12)	100 (38.38)	85.92 (82.84)
ICESe2	63,054	100 (96.72)	100 (44.09)	100 (97.86)
ICE-GI1	255,513	82.84 (98.15)	100 (34.94)	70.95 (95.76)
ICE-GI2(3)	245,522	85.92 (97.92)	100 (33.58)	82.36 (99.65)
ICE-GI6	159,096	81.16 (85.91)	100 (32.49)	79.47 (100)
SPI-7	133,499	100 (89.00)	87.93 (97.91)	96.14 (96.61)
Average	147,111	88.30 (92.83)	98.28 (44.98)	87.37 (95.82)

GI-Cluster does not require the assembly of incomplete genomes and can be directly applied to microbes with more than one chromosomes. Additionally, GI-Cluster output GIs in terms of contig positions, which are easy to interpret and analyze. In contrast, for IslandViewer, one and only one reference genome has to be chosen so that the contigs are assembled into a complete genome, and the output are the positions of GIs on the assembled genome. These characteristics of IslandViewer cause issues for incomplete genomes with more than one chromosomes. When applying IslandViewer to *V. cholerae* RC9, we used the large chromosome (chromosome 1) of *V. cholerae* N16961 as reference. After mapping the output of IslandViewer to contig positions, we found these predictions were located at contig 2, 4, 5, 6, 7, 9, 10, and 11, with one interval containing regions from contig 4, 5, and 9. Because contig 7 and 9 are on the small chromosome, the predictions of IslandViewer seem a bit confusing.

We mainly compare the predictions of GI-Cluster and IslandViewer in contig 11 of the incomplete genome, because contig 11 is located in the large chromosome and has more known GIs (Figure 2.9). Clearly, GI-Cluster predicted more known GIs. One region located around position 1,000,000 was not reported in literature, but predicted by both GI-Cluster and IslandViewer. GI-Cluster also yielded three intervals which were not reported by IslandViewer or in literature. We denote them by NG1 (from 160,001 to 165,258), NG2 (from 424,863 to 430,000), and NG3 (from 565,001 to 570,000). According to gene predictions and homology search, these four regions enriched with phage-related genes, virulence factors or novel genes. Therefore, these new predictions

are probably novel GIs and worthy of further investigation.

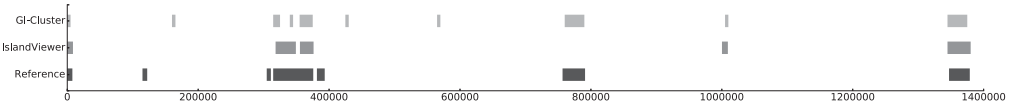

Figure 2.9: Performance comparison of GI-Cluster and IslandViewer in contig 11 of the incomplete genome of *V. cholerae* RC9.

Improvement on initial predictions of genomic islands Finally, we show the application of GI-Cluster on initial predictions from GI detection programs with high recall and low precision. A sensitive GI detection program may output many potential GI candidates, but there may be a large portion of false positives. So it is helpful to automatically filter out potential false positives from these initial predictions. Taking initial GI predictions as input, GI-Cluster can remove intervals with weak evidence of lateral origin, and thereby narrow down the search space for true positives.

We take the predictions of GI-SVM in the genome of *S. typhi* CT18 as an example. GI-SVM (parameters: $\nu = 0.9, k = 6$) predicted 96 GI candidates (997 genes) in this genome (Lu and Leong, 2016b). By running GI-Cluster on these initial predictions, we reduced the number of GI candidates to 51 (798 genes). The precision and F1 on C-data set were improved by around 28% and 16%, respectively. The precision and F1 on L-data set were improved by around 32% and 22%, respectively. As shown in Figure 2.10, the other measures were also slightly improved.

Visualization of predicted GIs in *S. typhi* CT18 genome

In this subsection, we show the feature plot and comparison plot of GI-Cluster for predicted GIs in the genome of *S. typhi* CT18.

Figure 2.11 shows a feature plot. Some regions with less GI-related evidence may be false positives. For example, the region between position 95 and 100 is likely to be a false positive because of the following reasons: it contains no mobility-related genes, antibiotic resistance genes or novel genes; the percentages of phage-related genes and virulence factors are not very high; some part of this region has normal codon usage.

Figure 2.12 shows a comparison plot. From the visualization, it is easy to get an overall picture about the performances of different methods. For example, GI-Cluster effectively removed many false positives predicted by GI-SVM. Moreover, it is helpful

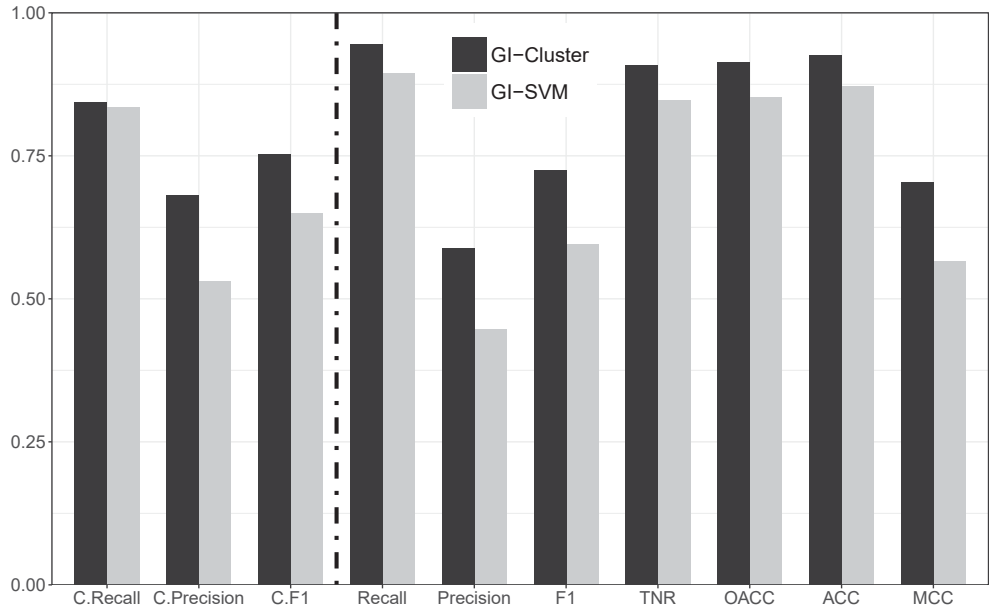

Figure 2.10: Improvement of GI-Cluster over the initial predictions of genomic islands generated by GI-SVM in the genome of *S. typhi* CT18. C.Recall, C.Precision, and C.F1 represent the recall, precision and F1 computed on C-data set. The other metrics were computed on L-data set.

to find more reliable predictions and novel GIs. For instance, several regions between position 210 and 235 were predicted by more than four methods but not reported in literature previously, suggesting that they are probably real GIs.

2.5 Summary

In chapter 2, we focus on the detection of genomic islands in microbial genomes. To better detect GIs for newly sequenced genomes which may have no complete annotations and enough closely related organisms, we proposed two machine learning methods, GI-SVM and GI-Cluster. GI-SVM utilizes one-class SVM to identify GIs based on k-mer frequencies in a single genome sequence. GI-Cluster takes advantage of consensus clustering to detect GIs by integrating multiple GI-related features. Due to the unsupervised nature of one-class SVM and clustering, the two programs do not rely on reliable training datasets which may not be available. As stand-alone tools, they can be conveniently rerun when the genome sequences are updated.

GI-SVM is implemented in Python, integrating the implementation of SVM and spectrum kernels provided by the SHOGUN Machine Learning Toolbox 4.0.0 (Sonnenburg et al., 2010). The source code is available on `https://github.com/icelu/GI_`

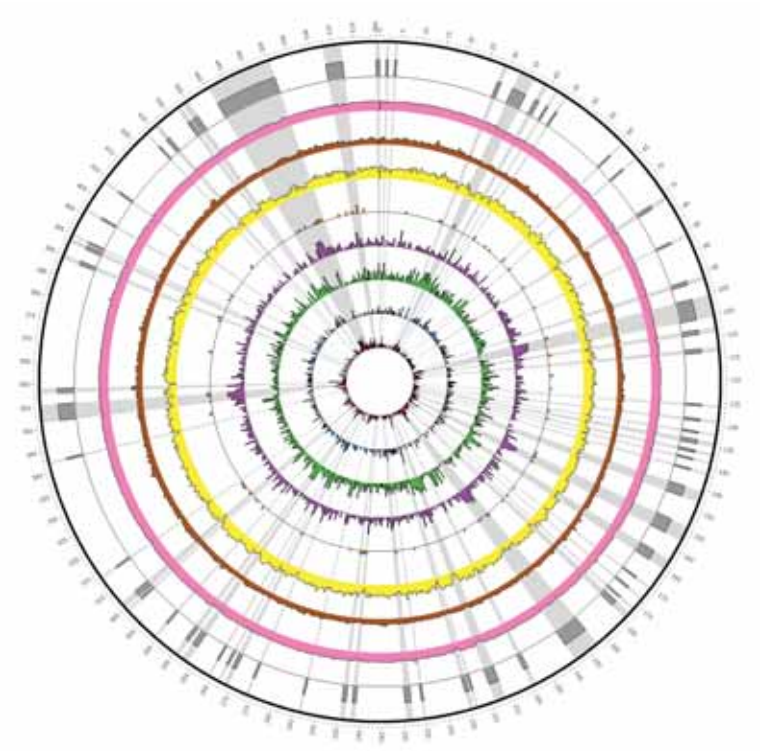

Figure 2.11: Genomic islands (GIs) and related features generated by GI-Cluster in the genome of *S. typhi* CT18. The unit of chromosome position is 10,000, namely 5 representing 50,000. The colour tracks from outermost to innermost are: predicted GIs, GC content, codon adaptation index, covariance for 4-mers, the percentage of mobility-related genes, the percentage of phage-related genes, the percentage of virulence factors, the percentage of antibiotic resistance genes, and the percentage of novel genes.

`Prediction/tree/master/GI_SVM`. GI-Cluster is implemented in Python, R, and Bash. The source code is available on `https://github.com/icelu/GI_Cluster`. Both programs can take the genome sequence file (FASTA format) as the sole input and are easy to use.

GI-SVM is able to identify GIs with higher recall than programs utilizing the same input. GI-SVM also performs better than programs detecting LGTs based on genes, and predicts more accurate boundaries than programs incorporating functional annotations. In brief, GI-SVM provides an alternative choice for researchers interested in a first-pass detection of GIs in newly sequenced genomes. The high sensitivity of GI-SVM can help to locate more GI candidates of certain reliability for further study.

GI-Cluster can separate GIs from non-GIs with a good balance between sensitivity and precision, sometimes outperforming a supervised machine learning method that utilizes similar features. The input for GI-Cluster can be from multiple sources, including incomplete genomes and initial output from other GI prediction tools. For incomplete

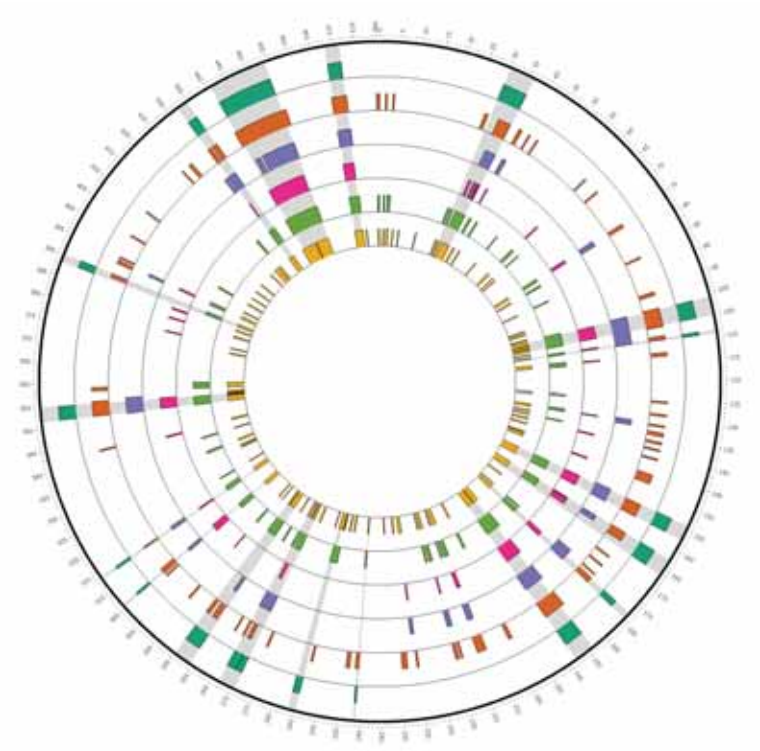

Figure 2.12: Predicted genomic islands (GIs) in the genome of *S. typhi* CT18 by multiple methods. The unit of chromosome position is 10,000, namely 5 representing 50,000. The colour tracks from outermost to innermost are: reference GIs in L-data set, GIs predicted by GI-Cluster, GIs predicted by GIHunter, GIs predicted by IslandViewer, GIs predicted by GI-Cluster from initial prediction of GI-SVM, and GIs predicted by GI-SVM.

genomes, GI-Cluster can generate accurate predictions without the assembly of contigs. For initial candidates from a GI prediction method with very high recall, GI-Cluster can reduce the number of false positives and improve the precision by more than 25%. GI-Cluster also provides feature plot and comparison plot, which may help to reveal insights from an overall picture of predicted GIs along the genome.

Moreover, GI-Cluster provides a flexible framework for analyzing GIs given a set of genomic segments. GI-Cluster computes multiple GI-related features from these segments, which can be then used for consensus clustering. These computed features can also facilitate manual analysis, which is still necessary to ascertain the foreign origin of GI candidates and localize them more accurately. The computation of each feature can be easily replaced by better methods. The functional annotations of genes can be improved by adopting latest database releases and integrating more related databases. New features indicative of a region's lateral origin can also be incorporated into this framework, whereas features turning out to be not discriminative enough can be removed.

Chapter 3

Towards modeling LGT with phylogenetic networks

3.1 Introduction

In this chapter, we first provide a brief description of phylogenetic networks. Next we give a review of the studies on network-based methods for LGT modeling, TCP, CCP, and network comparison. Then, we introduce basic concepts and notation as well as an important decomposition technique. Finally, we describe fast exponential time algorithms to solve the TCP and CCP on arbitrary phylogenetic networks, respectively, followed by a program for fast computation of the SRF distance.

Let X be a set of taxa. A *phylogenetic network* (network for short) over X is an acyclic digraph in which the leaves (i.e., nodes of outdegree zero) are bijectively mapped to X. A taxon typically represents some extant organism or species. A network has a unique root (of indegree zero). A non-leaf node is also called an internal node. There can be two types of internal nodes in a network: *tree nodes*, which include the root and nodes with outdegree of at least one, and *reticulate nodes*, which have indegree of at least two. The tree nodes represent speciation events, and the reticulate nodes represent reticulation events. Normally, all edges entering a tree node are called tree edges, and all edges entering a reticulate node are called reticulate edges.

A *binary* network is a network in which tree nodes have an outdegree of two and reticulate nodes have an indegree of two. If each reticulate node in a network has exactly two parents, the network is *bicombining*. A bicombining network is *binary* if each tree

node is of outdegree two. A phylogenetic tree is a binary network without reticulate nodes. If the unique child of each reticulate node in a network is a tree node or a leaf, this network is called *reduced*.

Many basic computational problems become hard when phylogenetic networks are used to model evolutionary events. To simplify the problems, different biologically-motivated topological constraints have been imposed and thus phylogenetic networks are further divided into different classes. These classes are briefly defined below.

A *cluster network* is a network for which each tree edge represents one cluster. It is easy to compute.

A biconnected component B *properly contains* a reticulate node r if all the edges entering r are contained in B. A *tree cycle* in a network is an undirected cycle that comprises two disjoint path meeting at a reticulate node. If a tree cycle shares no nodes with any other tree cycle, it is called a *gall*. A gall can also be seen as a biconnected component that properly contains exactly one reticulate node.

A *galled tree* is a network for which every biconnected component properly contains exactly one reticulate node. In other words, the individual loops in a galled tree are not interwoven or the reticulate nodes are isolated. Galled tree is the simplest classes of phylogenetic networks. It provides a good trade-off between computational tractability and generality.

A *level-k network* is a bicombining network for which every biconnected component properly contains at most k reticulate nodes. Here, the maximum number of reticulate nodes in any biconnected component of a network is called the *level* of this network. Galled tree is also level-1 network.

A *galled network* is a bicombining network for which every tree cycle is a gall. Galled networks are generalization of galled trees. Galled networks were introduced because a set of clusters may not be represented by a galled tree or a level-k network. For any set of clusters, there always is a galled network representing them. Galled network is suitable for dataset than can be represented by a backbone tree with some attached reticulations.

A *tree-child network* is a network for which every internal node has at least one child with indegree one. In a tree-child network, every reticulate node can reach a leaf by a path comprising only tree nodes. Biologically, this ensures that every nonextant

species has some descendant by speciation events. Galled trees are tree-child networks, but galled networks are not.

A *tree-sibling network* is a network for which every internal node has at least one sibling with indegree one. Biologically, this ensures that at least one of the nonextant species involved in reticulation events has some descendants resulted from speciation events. Tree-child networks are a subset of tree-sibling networks.

A *time-consistent network* is a network for with each node was assigned a time and the time increases from parent node to child node. The time constraint ensures that the organisms involved in the reticulation events must be coexist.

A network is *normal* if it is a tree-child network with no nodes of outdegree one, and if there is a directed path of size larger than one from node u to v then there is no edge (u, v). A *regular* network is a network for which all the descendants of a node are distinct. Normal networks are also regular.

In a network, a node u is a *stable ancestor* of another node v if every path from the root to v passes through u. A node u is stable if u is a stable ancestor of some leaf.

A network is *nearly stable* if every node either is stable or has stable parents. In a nearly stable network, if a node is not stable, all its children are stable. Nearly stable networks include normal and tree-child networks.

A network is *genetically stable* if every reticulate node has a stable parent. Intuitively, this means that each reticulate node obtains stability from one parent. Genetically stable networks are a subset of stable tree-sibling networks. The simulation results suggest that genetically stable networks take up a significantly larger proportion than tree-child networks (Gambette et al., 2015b).

A network is *reticulation-visible (stable)* if each reticulate node is a stable ancestor of some leaf. Namely, every reticulate node separates the network root from at least some leaves. The visibility property was originally introduced to capture the *separation property* of galled networks. Every reticulate node r in the galled network separates the network into two set of node: nodes reachable from the root bypassing r, and nodes that can only be reachable by a path passing through r (Huson et al., 2010). Galled trees, galled networks, and tree-child networks all belong to reticulation-visible networks.

A *tree-based network* is a special kind of phylogenetic network used to quantify the notion that a network is a tree with additional edges (Francis and Steel, 2015). A

tree-based network can be obtained from a tree by adding edges from a point on one tree edge to a point on another tree edge without incurring directed cycles. Moreover, any non-tree-based rooted binary phylogenetic network can be expanded to be tree-based by adding extra edges and leaves which can be seen as 'unseen' taxa in the past. The class of tree-based networks includes tree-child networks, tree-sibling networks, and reticulation-visible networks (Francis and Steel, 2015).

The relationships among several different types of networks are shown in Figure 3.1.

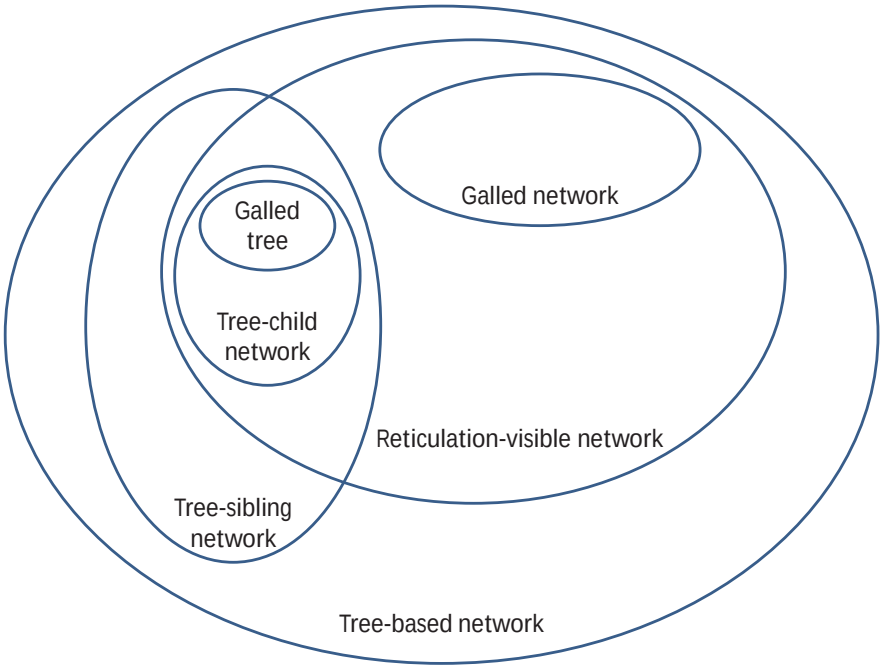

Figure 3.1: The relationships among several types of phylogenetic networks.

There are two different ways to interpret a phylogenetic network: hardwired network in which each tree edge represents one cluster, and softwired network in which each tree edge may represent more than one cluster (Huson et al., 2010). The reticulate edge helps to represent incompatible clusters that cannot be represented by a tree. So the reticulations in these networks do not represent evolutionary events. Given a set of clusters, softwired networks usually use fewer edges to represent them. Cluster network is a kind of hardwired network, which is easy to compute. Galled network and level-k network are softwired networks, on which many basic computational problems are hard to solve. Compared with galled network, level-k network is often simper (van Iersel et al., 2010a). These networks help to show which clades are supported by most trees.

3.2 Literature review

In this section, we first review network-based methods for LGT modeling. Then we discuss the previous work on solving the TCP and CCP. Finally, we briefly introduce some known distance metrics used for network comparison.

3.2.1 Network-based methods for modeling LGT

Network-based methods to reconstruct reticulate evolutionary history were suggested to bring potentially new ways to detect and assess LGTs (Zhaxybayeva, 2009). Unfortunately, there are still no practical network-based methods for modeling LGT. Nevertheless, some researchers have done preliminary work towards the modeling of LGT with phylogenetic networks. Among them, tree-based network and LGT network seems quite promising.

Tree-based networks are subclasses of phylogenetic networks that are particularly appropriate to model LGTs (Francis and Steel, 2015; Zhang, 2016). Tree-based means that the evolution can be represented by a rooted tree and linking edges. The rooted tree represents the evolution of speciation and the linking edges naturally represent LGTs. This representation is also relevant to the long-standing debate of whether the evolution is tree-like with reticulations or the tree should be dispensed.

But tree-based itself does not confer any particular evolution mechanism for the taxa under study. For example, other reticulation events, such as hybridization, can also be used to explain the linking edges in the network. Different from hybridization, LGT is an asymmetric evolutionary event, so all parents of a reticulate node should not be treated equally in a network model.

Based on this observation, Cardona et al. (2015) introduced LGT network, a tree-based network with LGTs which embeds a principal tree and a set of secondary edges between nodes in this tree. The principal tree represents the primary line of evolution, and the secondary edges represent LGTs. A LGT network can generate a principal subtree and a set of secondary subtrees. The principal subtree can be obtained by contracting nodes with outdegree one in the principal tree. Each secondary subtree can be obtained by keeping one secondary edge to a reticulate node in the network. They also developed an algorithm to reconstruct a restricted LGT network given a principle subtree

and a set of secondary subtrees. Figure 3.2 shows such a network reconstructed from four phylogenetic trees which were obtained from real biological data. However, rigorous conditions are required to assure that the principle subtree and secondary subtrees are pairwise different and they can determine a LGT network. These conditions on the input are too strict to find many examples for application of this algorithm.

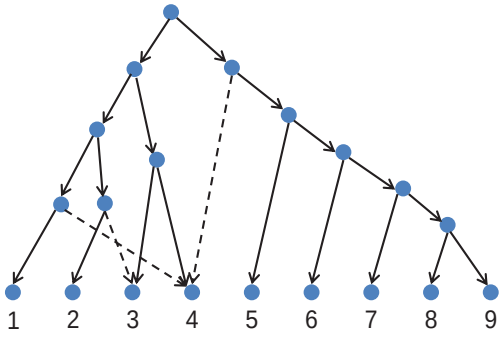

Figure 3.2: A restricted LGT network obtained from real biological data, redrawn from Figure 15 in (Cardona et al., 2015). The 9 leaves in the network represent *Roseobacte denitrificans* OCh 114, *Ruegeria pomeroyi* DSS-3, *Ruegeria sp.* TM1040, *Dinoroseobacter shibae* DFL 12, *Paracoccus denitrificans* PD122, *Rhodobacter sphaeroides* ATCC 17025, *Rhodobacter sphaeroides* KD131, *Rhodobacter sphaeroides* ATCC 17029, and *Rhodobacter sphaeroides* 2.4.1, respectively. The specific LGT events in this figure (the dashed arrows) are still not reported in literature yet, but several LGT events among *Rhodobacter* sp., *Ruegeria pom.* and *Ruegeria* sp. are known.

3.2.2 Methods to solve the TCP and CCP

The TCP and CCP are two fundamental questions in the study of phylogenetic networks. The characterization of a network as its displayed trees and (soft) clusters serves as the basis for network reconstruction and evaluation. To assess a reconstructed species network, it is required to ensure that the network is consistent with known biological knowledge regarding the species. One way to check this is to determine whether existing genes trees are displayed in the network. Besides, by finding whether a gene tree is embedded in a species network, one can detect the presence of reticulation events, such as LGT. It is also of biological interest to know whether a set of taxa forms a clade (is a cluster) in a given phylogenetic network. However, it is not easy to solve the TCP and CCP for general phylogenetic networks. In the following subsections, we will briefly review previous efforts on solving these two problems on restricted classes of phylogenetic networks in chronological order.

Methods to solve the TCP

It is non-trivial to solve the TCP, because the number of trees displayed in a network grows exponentially with the number of reticulate nodes. Actually, the TCP is similar to a hard problem, the subgraph isomorphism problem which asks whether a graph contains a subgraph that is isomorphic to another graph (Gunawan et al., 2016a). The difficulty to solve the TCP lies at how to examine all the reticulate nodes. A naïve algorithm is to randomly delete one edge entering a reticulate node. It will take $O(2^{|R(N)|})$ time to examine them simultaneously in a network N, where $|R(N)|$ is the number of reticulate nodes in N. The key to an efficient solution is to identify the set of reticulate nodes that can be dissolved simultaneously in polynomial time. To simplify the problem, solutions are often pursued for restricted classes of networks.

Nakhleh and Wang (2005) first developed a quadratic-time algorithm for the TCP on the binary tree-child networks. They utilized a network decomposition technique. The runtime is bounded by $O(|V(N)||L(N)|)$ for a network N, where $|V(N)|$ is the number of nodes in N and $|L(N)|$ is the number of leaves in N. Then, Kanj et al. (2008) proved that the TCP is NP-complete for general networks by reducing the TCP to the problem of Node-disjoint Paths. By reducing from the TCP and CCP on general networks, van Iersel et al. (2010b) showed the TCP and CCP are NP-complete for tree-sibling, time-consistent and regular networks. They also developed polynomial-time algorithms for solving the TCP on normal networks, binary tree-child networks and level-k networks.

Later, Gambette et al. (2015a) developed a quadratic-time algorithm for binary nearly-stable networks. This algorithm utilizes so-called subtree-free property. Namely, if a network and a tree share a common subtree, this subtree will be replaced by a new leaf. Then the network will not contain a subtree with more than two leaves. By considering the longest root-to-leaf path with at least four nodes and different types of specific nodes along the path, the algorithm iteratively deletes a reticulation edge at the end of the longest root-to-leaf path until it becomes a tree. Then, it is easy to check whether two trees are isomorphic and find the solution to the TCP. The time complexity of this algorithm for a network N is $O(|L(N)|^2)$ time, where $|L(N)|$ is the number of leaves in N.

Then, Gambette et al. (2015b) developed a quadratic-time algorithm for genetically

stable networks. The algorithm depends on an observation that a network displays a tree if and only if a network modified by removing two disjoints paths from a tree node to two sibling leaves displays the tree modified by removing the two sibling leaves. The subtree induced by two sibling leaves and their parent in a tree is called a *cherry*. The algorithm is recursive and a cherry is selected in the tree and resolved at each step.

Extending on previous work, Gunawan et al. (2016a) developed a cubic-time algorithm for reticulation-visible networks that can be binary or non-binary. They observed two useful properties of reticulation-visible networks: reticulation separability and visibility inheritability. These properties lead to a powerful decomposition theorem. The decomposition theorem states that a reticulation-visible network can be decomposed into a set of disjoint connected tree-node components such that each component contains either all the parents of a reticulate node or a leaf. This theorem makes it feasible to solve the TCP and CCP by the divide-and-conquer algorithm in which the tree-node components can be resolved in succession from the bottom up. For the TCP, a dynamic programming algorithm was used to resolve all the reticulate nodes right below a tree-node component. The time complexity for an arbitrary reticulation-visible network N and a phylogenetic tree T is $O(|V(T)| * |E(N)| * |R(N)|$, where $|V(T)|$ is the number of nodes in T, $|E(N)|$ is the number of edges in N, and $|R(N)|$ is the number of reticulate nodes in N.

This algorithm was later simplified into a quadratic-time algorithm by using proper data structures in (Gunawan et al., 2017). The time complexity for an arbitrary reticulation-visible network N and a phylogenetic tree T is $O(|E(N)| * |L(T)|$, where $|E(N)|$ is the number of edges in N and $|L(T)|$ is the number of leaves in T.

Methods to solve the CCP

Given a phylogenetic tree, it is easy to determine whether a cluster is represented by the tree in linear time in the number of taxa. Similarly, one can determine whether a cluster is represented by a phylogenetic network in polynomial time. However, similar as the TCP, it is challenging to determine whether a soft cluster is represented by a phylogenetic network, because the number of soft clusters displayed in a network also grows exponentially with the number of reticulate nodes (Huson et al., 2010). As a result, the solutions to the CCP are also pursued for restricted classes of networks.

Nakhleh and Wang (2005) developed a quadratic-time algorithm for the binary tree-child networks. The runtime is bounded by $O(|V(N)|^2)$ for a network N, where $|V(N)|$ is the number of nodes in N.

Then, Kanj et al. (2008) proved that the CCP is NP-complete for general networks by reducing the CCP to the problem of Infinite Site on Phylogenetic Networks. They also showed that the CCP is solvable in time $O(2^{k/2}n^2)$ when parameterized by the number of reticulate nodes k in the network, where n is the number of reticulate nodes in the network.

By reducing the CCP to 3-SAT problem, Huson et al. (2010) proved that the CCP is NP-complete for general networks as well. They also provided a polynomial-time algorithm for the CCP on reticulation-visible networks (Huson et al., 2010). The algorithm proceeds by considering whether a cluster is represented by an edge in the network.

Later, van Iersel et al. (2010b) proved that the CCP can be solved in polynomial time for binary level-k networks. The algorithm is based on their TCP algorithm for tree-child networks.

Taking advantage of the decomposition theorem, Gunawan et al. (2016a) developed linear-time algorithms for the CCP on reticulation-visible networks. Similar as their solution to the TCP, after decomposing the network into a set of tree-node components, they apply a dynamic programming algorithm on each tree-node component in a bottom-up manner. They firstly proposed a $O(|L(N)|)$ algorithm for a binary reticulation-visible network N, where $|L(N)|$ is the number of leaves in N. Then they extended it for non-binary reticulation-visible networks Gunawan et al. (2017). The CCP algorithm for a non-binary reticulation-visible network N was proved to have $O(|E(N)|)$ runtime, where $|E(N)|$ is the number of edges in N.

3.2.3 Distance measures to compare phylogenetic networks

To assess the robustness of network reconstruction methods, one has to be able to compare the reconstructed network with simulated networks or true networks. Besides, for the same input, different reconstruction methods may return different networks and a single reconstruction method may return multiple networks. Therefore, it is necessary to compare them to evaluate the performance of reconstruction.

The differences between networks can be quantified by distance metrics. But it is difficult to define a proper metric that is easy to compute, since zero distance means graph isomorphism which is computationally hard. Two kinds of practical approaches have been proposed to solve this problem. One approach is to use rough estimates of similarity. The other approach is to focus on restricted class of phylogenetic networks. Using these two approaches, many distance metrics have been defined (Cardona et al., 2008a; Huson et al., 2010; Than et al., 2008).

Most distance metrics are designed by comparing the represented data in networks (Huson et al., 2010), such as displayed clusters or trees. The distances in this category include: RF (hardwired cluster) distance (Huson et al., 2010), SRF (softwired cluster) distance (Huson et al., 2010), tripartition distance (Moret et al., 2004; Cardona et al., 2009a), cluster-based measure (Than et al., 2008), displayed tree distance (Huson et al., 2010), and tree-based measure (Than et al., 2008).

Some distance metrics compare the topological invariants between networks (Huson et al., 2010). The distances in this category include: subnetwork distance (triplet distance) (Cardona et al., 2009b), path-multiplicity distance (μ-distance) (Cardona et al., 2009c), and nested-labels distance (nodal distance) (Cardona et al., 2009b; Nakhleh, 2010).

Among these distance metrics, only subnetwork distance is a proper metric on all phylogenetic networks. The other distances are only proper metric on subsets of phylogenetic networks. The nested-labels distance is a proper metric on tree-child networks. The RF distance, tripartition distance and path-multiplicity distance were proved to be proper metrics on time-consistent tree-sibling networks (Cardona et al., 2009a).

Now we briefly review the work on computing these distances. Suppose n is the number of leaves, m is the number of nodes and e is the number of edges in each network. The RF distance can be computed in $O(m(m+e)))$ time (Cardona et al., 2009a). Asano et al. (2010) proposed a faster algorithm for computing the RF distance on general networks. The algorithm makes use of word-level parallelism and has $O(ne/logn)$ runtime. The tripartition distance can be computed in $O(m(logm+n))$ time (Cardona et al., 2009a). The path-multiplicity distance can be computed in $O(mn)$ time (Cardona et al., 2009a). The runtimes for the SRF distance, the displayed tree distance, cluster-

and tree-based measures are exponential in the number of reticulate nodes in the two compared networks (Huson et al., 2010; Than et al., 2008).

Dendroscope 3 (Huson and Scornavacca, 2012) implemented the calculations of six distance metrics for networks, including the RF distance, SRF distance, tripartition distance, displayed tree distance, path-multiplicity distance, and nested-labels distance. The Perl package Bio::PhyloNetwork (Cardona et al., 2008b), included in the BioPerl bundle, implemented the computation of the path-multiplicity distance. Than et al. (2008) implemented three distance metrics in PhyloNet, including tree-based measure, cluster-based measure, and tripartition-based measure.

The computation of the SRF distance implemented in Dendroscope is very straightforward. This method exhaustively searches the clusters that are in a phylogenetic tree displayed in one network but are not in any phylogenetic tree displayed in another. Hence, more efficient methods to compute the SRF distance are still in need.

3.2.4 Summary

The prevalence of reticulation events, including LGTs, makes it promising to use phylogenetic networks to model evolution of life. Among the different kinds of networks, tree-based network is a natural model for LGT. Particularly, LGT network, a kind of tree-based network which accounts for the asymmetricity of LGT, has been proposed to model LGT. But it is computationally challenging to reconstruct a phylogenetic network from biological data. The current reconstruction algorithm for a restricted class of LGT network is imposed with strict conditions and not widely applicable. Besides, some fundamental problems arising in the reconstruction of phylogenetic networks are still not solved, such as the TCP and CCP. The efficient solutions to these basic problems may facilitate the development of network reconstruction methods.

In recent years, several efficient algorithms have been developed for the TCP and CCP. But these algorithms work on restricted classes of networks, such as reticulation-visible networks. According to simulation results on the distribution of reticulation-visible networks, reduced networks, and tree-based networks, a large fraction of phylogenetic networks are not reticulation-visible (Zhang, 2016). Therefore, it is necessary to develop algorithms for the TCP and CCP on arbitrary phylogenetic networks that are not necessarily reticulation-visible or binary.

In view of this need, we developed fast algorithms for solving the TCP and CCP on *arbitrary* phylogenetic networks, which will be presented in Section 3.5 and Section 3.6, respectively. The solution to the CCP on arbitrary networks makes it straightforward to efficiently compute the SRF distance. We will present our program for computing the SRF distance in Section 3.6.

3.3 Basic concepts and notation

In this section, we define basic concepts and notation used later. We define *tree nodes* to be the root and nodes of indegree one and outdegree of at least one, and *reticulate nodes* to be nodes of outdegree one and indegree of at least two. We allow degree-two nodes in a network.

Here, we use the following notation for a network N:

- $T(N)$: the set of tree nodes in N.
- $L(N)$: the set of leaves in N.
- $R(N)$: the set of reticulate nodes in N.
- $V(N)$: the set of all nodes in N, namely $T(N) \cup L(N) \cup R(N)$.
- $E(N)$: the set of edges in N.
- $\rho(N)$: the root of N.
- $N - E$: the subnetwork $(V(N), E(N) \backslash E)$ for a subset $E \subseteq E(N)$.
- $N - S$: the subnetwork $(V(N) \backslash V(S), E')$, where $E' = \{(x, y) \in E(N) \mid \{x, y\} \subseteq V(N) \backslash V(S)\}$ for a subnetwork S of N.

For $u, v \in V(N)$, u is a *parent* of v and v is a *child* of u if $(u, v) \in E(N)$. We use $c(r)$ to denote the unique child of $r \in R(N)$. If there is a direct path from u to v, v is called a *descendant* of u.

We use $[r]_N$ to denote the subnetwork below $r \in V(N)$, which consists of all the descendants of r and the edges between them in N. For a leaf ℓ below r, we use $N - [r]_N + \ell$ to denote the subnetwork obtained by replacing $[r]_N$ with ℓ so that ℓ becomes the child of r.

Consider a phylogenetic network M over a set of taxa X and a phylogenetic tree G over a set of taxa Y such that $Y \subseteq X$. After all but one incoming edges are removed for every reticulate node, M becomes a tree M' (Figure 3.3). Note that some internal nodes of M may have become leaves in M'. After the nodes that are not in any path from $\rho(M)$ to a leaf in Y and all the edges incident to them are removed from M', M' becomes a tree M'' over Y. M'' is called a *sub-tree history* of the network M for Y. The tree G is displayed in M if G can be obtained from a sub-tree history of M for Y by contraction (that is, contracting all the nodes with both indegree and outdegree one) (Figure 3.3). If M is a tree without nodes of indegree and outdegree one over the same set of taxa as G, by definition, $M = M' = M''$ and so G will not be displayed in M unless they are identical.

We allow a network to have dummy nodes (i.e., unlabeled nodes of outdegree zero) because such a network may be generated in a recursive step of our algorithms. A dummy reticulate node is a node that has indegree greater than one and outdegree zero, and a dummy leaf is an unlabeled leaf. As an evolutionary model, a phylogenetic network is assumed not to have such nodes, as they are not meaningful. However, the algorithm to be developed works in a recursive manner. Occasionally, the input network will be simplified into a network with such kind of nodes when the algorithm runs. Therefore, in this work, for convenience, we allow a phylogenetic network to have dummy reticulate nodes and dummy leaves. Such a network is called a *general phylogenetic network*.

For a phylogenetic network N over X and $Y \subset X$, we use N_Y to denote the general phylogenetic network obtained from N by removing all the leaves not labeled by a taxa in Y. It is not hard to see that N_Y is over Y. It is also true that a phylogenetic tree G over Y is displayed in N if and only if G is displayed in N_Y.

Given the set of taxa X, a *cluster* is any proper subset of X (excluding the empty set and the full set). A cluster is *trivial* if it contains only one element.

In a phylogenetic tree T over X, each non-root node induces a unique set of taxa that consists of the labels of the leaves below the node, which is called the cluster of the node. A cluster is displayed in T if it is the cluster of some node in T.

A cluster $B \subset X$ is a *soft cluster* displayed in a phylogenetic network N if there is a tree T displayed in N such that B is a cluster displayed in T. A tree node in a network may represent multiple soft clusters, which could be obtained from different trees displayed

in the network. We use $SC(N)$ to denote the set of soft clusters displayed in N.

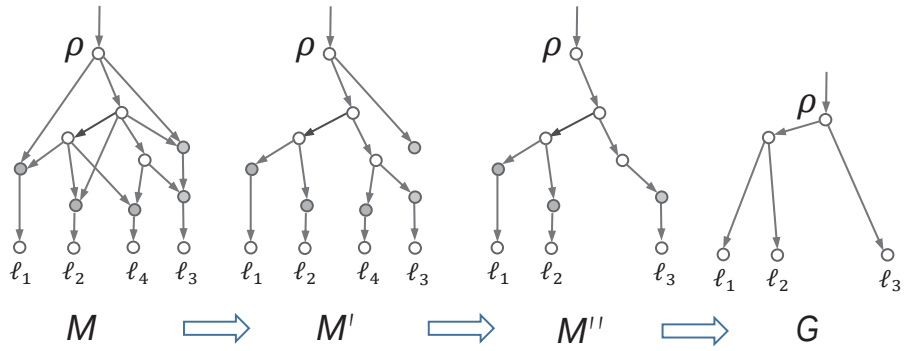

Figure 3.3: Illustration of tree containment. M is a phylogenetic network over taxa $\{\ell_1, \ell_2, \ell_3, \ell_4\}$. Reticulate nodes are represented by filled circles. An incoming edge for the root is added for visualization. G is a phylogenetic tree over $\{\ell_1, \ell_2, \ell_3\}$ displayed in M.

3.4 Network decomposition technique

The key to solving the TCP and CCP is the network decomposition theorem, which was first proposed by Gunawan et al. (2017) for reticulation-visible networks.

Let N be a network over a set of taxa X. For $u, v \in V(N)$, v is a *vertical descendant* of u if a direct path exists from u to v that contains either a single edge from u or multiple edges, each of which enters a tree node. Let $\mathrm{VD}(u)$ denote the set of all vertical descendants of u. If u is a dummy reticulate node, $\mathrm{VD}(u)$ is empty.

Let $r \in R(N)$ have a unique child $c(r)$. If $c(r)$ is also in $R(N)$, $\mathrm{VD}(r) = \{c(r)\}$. If $c(r) \notin R(N)$, then $\mathrm{VD}(r) \subseteq T(N) \cup L(N)$. Additionally, $\mathrm{VD}(r)$ induces a subtree of N with the vertex set $\mathrm{VD}(r)$ and the edges set $\{(v', v'') \in E(N) \mid v', v'' \in \mathrm{VD}(r)\}$. This subtree is denoted by $C(r)$ and is called a *tree component* of N. The subtree induced by $\{\rho\} \cup [\mathrm{VD}(\rho) \cap (T(N) \cup L(N))]$ is also called a tree component, where ρ is the root of N.

In summary, the decomposition theorem says that an arbitrary network N can be decomposed into a set of connected tree components which are separated by reticulate nodes. Specifically, there is a tree component C_r for each $r \in R(N) \cup \{\rho(N)\}$, which is either $\{c(r)\}$ if $r \in R(N)$ and $c(r) \in R(N)$, or a subtree induced by all the tree nodes and leaves that are reachable from r.

A tree component is *trivial* if it contains only one leaf or if it is empty (for a dummy

reticulate node). A node is *visible* on a leaf ℓ if it lies on all the paths from $\rho(N)$ to ℓ. If a node $r \in R(N) \cup \{\rho(N)\}$ is visible on a leaf ℓ, its tree component C_r is *visible* on ℓ as well. Given two tree components $C_{r'}$ and $C_{r''}$, r' and $C_{r'}$ are *right below* $C_{r''}$ if a parent of r' is in $C_{r''}$. A tree component is *exposed* if it contains only one leaf or if all the tree components right below it are trivial.

Obviously, N contains at least one exposed non-trivial tree component. In addition, we have the following lemma (see Appendix B for the proof).

Lemma 3.4.1. *Let C be an exposed tree component. C is visible if and only if C contains a leaf or if a reticulate node r exists right below C such that the parents of r are all in C.*

The above concepts are briefly illustrated in Figure 3.4. See (Gunawan et al., 2017) for more details of the decomposition theorem.

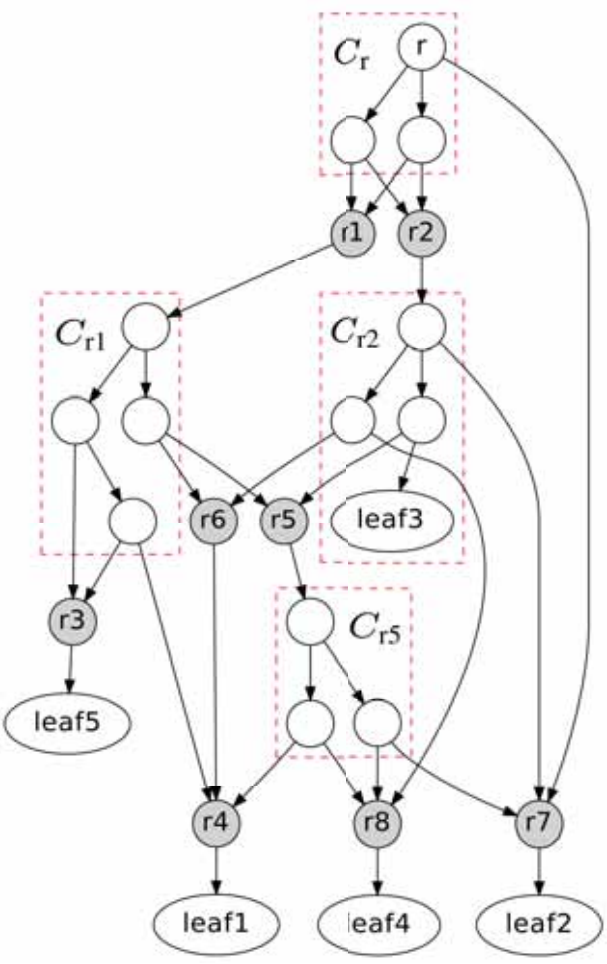

Figure 3.4: A network N and its tree components. There are nine tree components in N. Five of these components are non-trivial: C_r, C_{r1}, C_{r2}, C_{r5}, and C_{r6}, where $C_{r6} = \{r4\}$. C_{r7} and r7 are right below C_{r5}, C_{r2}, and C_r. C_r is visible on all the leaves. C_{r1} and C_{r2} are visible, but neither of them is exposed. C_{r5} is exposed but not visible.

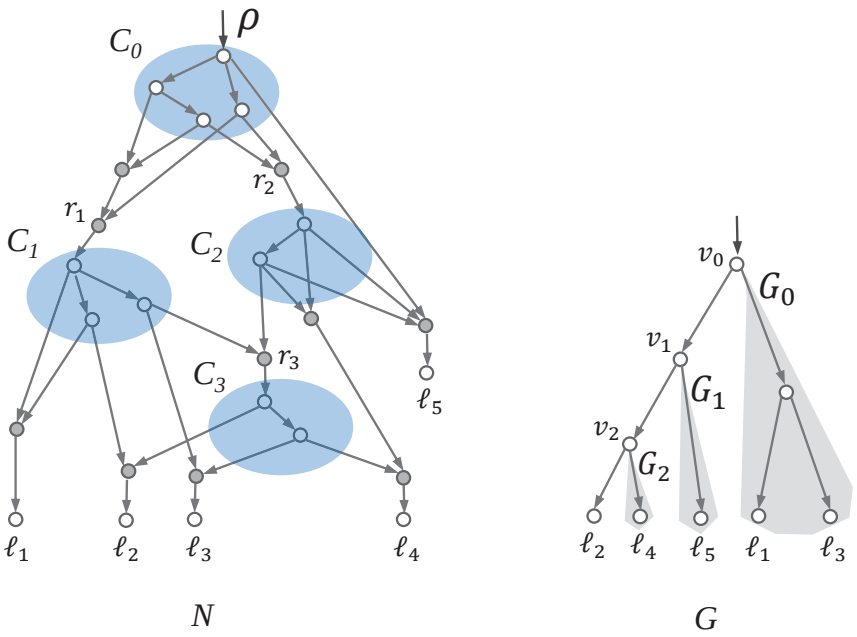

Figure 3.5: A network N and a phylogenetic tree G displayed in N. Left panel: C_0 is the tree component below the root; $C_i = C(r_i)$ for $1 \leq i \leq 3$. For each of the other reticulate nodes, the tree component contains only its unique child. Right panel: G_0, G_1 and G_2 are subtrees branching off the path from the root v_0 to ℓ_2.

Based on the network decomposition technique, our TCP and CCP algorithm work as follows:

> *To determine whether or not a tree G (cluster C) is displayed in a phylogenetic network N, the algorithm selects a non-trivial exposed component M of N. If M is visible, we either find the negative answer to the problem by working on M or we obtain an instance of the problem that is simpler than the input instance (G, N) ((C, N)) in polynomial time proportional to the size of M. In the latter, we reduce the original instance of the TCP (CCP) to a simpler instance.*
>
> *If M is not visible, there is then a reticulate node which has a unique leaf child and does not have all parents in M. In this case, two phylogenetic networks N_1 and N_2 are derived from N, which contain fewer nodes than N. The algorithm is then called on both instances (C, N_1) ((G, N_1)) and (C, N_2) ((G, N_2)) recursively.*

3.5 Locating a tree in an arbitrary phylogenetic network

In this section, we present a fast algorithm for the TCP on an arbitrary phylogenetic network, which was developed by using the decomposition technique reported in Section 3.4. Although this algorithm is of exponential time in the worst case, the theoretical analysis shows that it is much faster than the naïve program for binary phylogenetic networks. The evaluations on both random and real networks also demonstrate that it is very fast in practice.

Formally speaking, the TCP is formulated as below:

Instance: A phylogenetic network N over X and a phylogenetic tree G over Y such that $Y \subseteq X$.

Question: Is G displayed in N?

Here, we would like to point out that the TCP has been studied only for a phylogenetic network and a phylogenetic tree over the same set of taxa in literature (Huson et al., 2010).

3.5.1 Methods

In this subsection, we describe the main idea of the TCP algorithm, followed by some important issues in its implementation.

Description of the TCP algorithm

Let N and G be a network and a phylogenetic tree over the same set of taxa X, respectively (Figure 3.5). In the rest, for simplicity, we use the same symbol to denote the leaf that represents the same taxon in G and N.

Consider $r \in R(N)$ such that $C(r)$ is non-trivial, exposed, and visible. Let L_r be the set of network leaves with respect to which $C(r)$ is visible and let $\ell \in L_r$. A unique path then exists from the tree root $\rho(G)$ to ℓ in G:

$$P_\ell : v_0 = \rho(G), v_1, \cdots, v_t, v_{t+1} = \ell, \tag{3.1}$$

where $t \geq 0$. P_{ℓ_2} is shown in the tree G in Figure 3.5, where $t = 2$. It is not hard to see that $G - P_\ell$ is a union of $t + 1$ disjoint trees G_i, each of which branches off from P_ℓ at

v_i for $i = 0, \cdots, t$. For example, in Figure 3.5, G_0 consists of ℓ_1, ℓ_3 and their parent; G_1 and G_2 are simply ℓ_5 and ℓ_4, respectively. For convenience, we set $G_{t+1} = \{\ell\}$.

Define

$$s_G(r) = \min\{\, i \mid 0 \leq i \leq t+1, L(G_i) \cap L_r \neq \emptyset \,\}, \tag{3.2}$$

where $L(G_i)$ denotes the set of labeled leaves in G_i. Since $\ell \in L(G_i) \cap L_r$, the index $s_G(r)$ is well defined. For G in Figure 3.5, if $L_r = \{\ell_2, \ell_5\}$, we have $s_G(r) = 1$, as G_1 contains ℓ_5 in L_r but G_0 contains neither ℓ_2 nor ℓ_5; if $L_r = \{\ell_2\}$, then $s_G(r) = 3$.

The index $s_G(r)$ can be computed by a simple dynamic programming algorithm on G and L_r, which takes linear time $O(|L(G)|)$ (see (Gunawan et al., 2017)).

For each v_i, we use $G'(v_i)$ to denote the subtree rooted at v_i in G. Formally, for each $s_G(r) \leq i \leq t+1$, $G'(v_i)$ is said to be displayed in the subnetwork below r, if it is displayed in the following subnetwork
$[r]_N - \{c(r'), r' \mid r' \in R(N)$ such that $c(r') \notin L(G'(v_i))\}$, where $c(r')$ is the unique leaf child for $r' \in R(N)$.

If G is displayed by N, the subtree $G'(v_{s_G(r)})$ must be displayed in the subnetwork of N below r. The reason is that r and $C(r)$ are visible with respect to a leaf ℓ', which is in $G_{s_G(r)}$, as well as ℓ, forcing v_i to correspond to a node in $C(r)$ in any display of G in N.

However, a super subtree containing $G'(v_{s_G(r)})$ may possibly be displayed below r in N. We define:

$$d_G(r) = \min\{j \mid\ G'(v_j) \text{ is displayed below } r \text{ in } N\,\}. \tag{3.3}$$

The index $d_G(r)$ can be computed in quadratic time (Gunawan et al., 2017).

For network N' in Figure 3.6, $C(r_3) = C_3$. It is visible with respect to ℓ_2 and exposed. For tree G in Figure 3.5, since $L_{r_3} = \{\ell_2\}$, $s_G(r_3) = 3$. However, $d_G(r_3) = 2$, as $G'(v_2)$ is displayed in the subnetwork below r_3.

Let $N - C(r) + \ell$ denote the network obtained from N as follows:

- for each $x \in R(N)$ right below $C(r)$, removing the edges to x from its parents in $C(r)$ if the child of x is not in the subtree $G'(v_{d_G(r)})$, and
- replacing the resultant subnetwork below r with ℓ so that ℓ becomes the child of

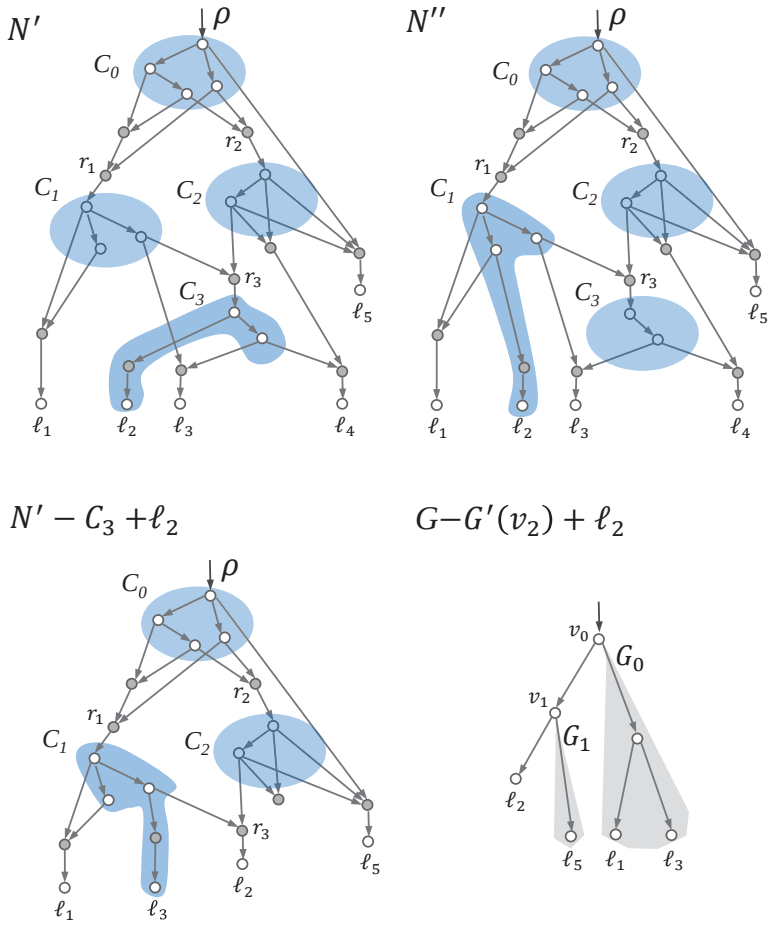

Figure 3.6: Illustration of the TCP algorithm. Top left panel: The network N' obtained from N (Figure 3.5) by deleting the edge entering the parent of ℓ_2 from a node in C_1. Note that C_3 is visible with respect to ℓ_2 in N'. Top right panel: The network N'' obtained from N (Figure 3.5) by deleting the edge entering the parent of ℓ_2 from a node in C_3. Bottom left panel: The network derived from N' by replacing C_3 with ℓ_2 after the subtree $G'(v_2)$ is found to be displayed below r_3. Bottom right panel: The tree obtained from G (Figure 3.5) by replacing $G'(v_2)$ with ℓ_2. $G'(v_2)$ is the subtree of G consisting of ℓ_2, ℓ_4 and its parent v_2 (Figure 3.5).

r.

Similarly, we use $G - G'(v_{d_G(r)}) + \ell$ to denote the tree obtained from G by replacing $G'(v_{d_G(r)})$ with ℓ.

For $C(r_3) = C_3$ and ℓ_2 in the network N' in Figure 3.6 and G in Figure 3.5, since $d_G(r_3) = 2$, $N' - C_3 + \ell_2$ and $G - G'(v_2) + \ell_2$ are those shown in the second row in Figure 3.6.

The following theorem was first proved for reticulation-visible networks by Gunawan et al. (2017). Its correctness for arbitrary networks is proved in Appendix B.

Theorem 3.5.1. *Assume $C(r)$ is non-trivial, exposed and visible with respect to ℓ in N.*

Let $s_G(r), d_G(r), N - C(r) + \ell, G - G'(v_{d_G(r)}) + \ell$ be defined as above.

(i) *If $d_G(r) > s_G(r)$, N does not display G.*

(ii) *If $d_G(r) \leq s_G(r)$, N displays G if and only if $N - C(r) + \ell$ displays $G - G'(v_{d(r)}) + \ell$.*

Consider $r \in R(N)$ such that $C(r)$ is exposed and non-trivial. Assume that $C(r)$ is not visible but a reticulate node r' exists right below $C(r)$ such that $C(r')$ consists of only a leaf. Since $C(r)$ is not visible, r' has at least one parent not in $C(r)$.

We define:

$$N' = N - \{(x, r') \in E(N) \mid x \notin C(r)\} \tag{3.4}$$

and

$$N'' = N - \{(x, r') \in E(N) \mid x \in C(r)\}. \tag{3.5}$$

If r' has only one parent in $C(r)$ in N, r' is of indegree and outdegree one in N'. In this case, r' becomes a tree node and is merged into $C(r)$ in N' (Figure 3.6). If r' has multiple parents in $C(r)$, $C(r')$ is only right below $C(r)$ in N'. Hence, $C(r)$ has become visible with respect to the unique leaf child of r' in N'. For example, in Figure 3.5, C_3 in the network N is exposed, but not visible. If the parent of ℓ_2 is chosen, N' and N'' are shown in the top row in Figure 3.6.

By the definition of tree containment, a tree is in N if and only if it is in either N' or N''.

By combining the two possible cases discussed above, we obtain an algorithm for solving the TCP summarized in Figure 3.7.

Implementation of the TCP algorithm

A couple of issues arising during implementation of our algorithm are worth mentioning here.

Firstly, since the algorithm decodes tree components one by one, the input network and any additional networks created in Step 3 of the algorithm are each represented as a sorted list of non-trivial tree components. The non-trivial tree components are sorted in post order so that after the tree components listed before a tree component are dissolved, the latter becomes exposed. In this way, the non-trivial components are dissolved from the first to the last. When a tree component is dissolved, each tree component listed behind it will be updated if a reticulate node below the former is also below the latter.

TCP Algorithm

Input: An arbitrary network N and a phylogenetic tree G.
Output: TRUE if N displays G and FALSE otherwise.

0. If N is NULL, output FALSE;
1. Select an exposed and non-trivial tree component $C(r)$;
2. If $C(r)$ is visible with respect to a leaf ℓ
   2.1. Compute $s_G(r)$ and $d_G(r)$ as defined in Eqn.(3.2) and (3.3);
   2.2. If $d_G(r) > s_G(r)$, output FALSE;
   2.3. If $d_G(r) = 0$, output TRUE;
   2.4. If $0 < d_G(r) \leq s_G(r)$ do {
      (i) Compute $N_r = N - C(r) + \ell$ and
         $G_r = G - G'(v_{d(r)}) + \ell$ as defined above;
      (ii) Output TRUE if N_r displays G_r and FALSE if not;
   }
3. If $C(r)$ is not visible but a leaf exists below $C(r)$, do {
   3.1. Compute N' and N'' as defined in Eqn. (3.4) and (3.5);
   3.2. If N' displays G, output TRUE;
   3.3. If N'' displays G, output TRUE;
   3.4. Output FALSE;
   }
4. If no leaf exists below $C(r)$, do {
   4.1. If $N - r - C(r)$ displays G, output TRUE;
   4.2. Output FALSE;
   }

Figure 3.7: An algorithm for solving the TCP on an arbitrary network.

Our evaluation tests showed that the runtime of our algorithm was very sensitive to the order in which the tree components were listed. Ideally, visible tree components with more nodes should be listed first whenever possible. We used an array to save the leaf below each reticulate node for which the tree component is a single leaf. The list is updated at the end of Step 2.4.(i) when an exposed and visible tree component is replaced by a network leaf.

Secondly, we used a two-dimensional table to implement a dynamic programming method for computing $d_G(r)$, defined in Equation (3.3). The details of the dynamic programming method can be found in (Gunawan et al., 2017).

3.5.2 Results

In this subsection, we first analyze the worst-case time complexity of the TCP algorithm. We then report its performance on random networks and on one of the largest networks

in the literature.

Theoretical analysis of the time complexity

Our algorithm is recursive by nature. There are mainly three useful observations on its efficiency.

Firstly, the runtime of the algorithm is proportional to the number of non-trivial tree components. If the input network N has only one non-trivial tree component, this tree component is rooted at the network root and hence must be visible with respect to each leaf, implying that the network is visible. In this case, both Step 3 and Step 4 are not executed; Step 2 of the algorithm is executed only once, taking $O(|E(N)||L(G|)$ time (Gunawan et al., 2017), where $|\cdot|$ denotes the cardinality of a finite set.

Secondly, when Step 3 is executed, the algorithm will run on two simplified copies of the current network in sequential order. Since the time spent on each exposed and visible tree component is quadratic to the number of nodes in the component (Gunawan et al., 2017), the runtime of the algorithm is bounded above by $O((m+1)|E(N)||L(G|)$, where m is the number of times Step 3 is executed on the input network N. Since the TCP is NP-complete, m is likely an exponential function of the number of reticulate nodes in N, $|R(N)|$, unless NP = P.

Thirdly, the input network N and tree G satisfy $L(G) \in L(N)$. Hence, we simply use $|L(N)|$ to replace $|L(G)|$ in the time complexity analysis. In this way, the time complexity of our algorithm is written as $O((m+1)|E(N)||L(N|)$.

Given a reduced network N and a phylogenetic tree G, the naïve algorithm will consider all the reticulate nodes one by one. For each reticulate node r, there will be k_r possibilities if the indegree of r is k_r. Therefore, the naïve algorithm will create $\prod_{r\in R(N)} k_r$ trees and then take linear time to determine whether or not G is displayed in each of them.

Assuming the indegrees of reticulations nodes of N are constant-bounded, we have $\prod_{r\in R(N)} k_r = \Theta(2^{|R(N)|})$. Combining this fact and the third point made above, we propose to measure the efficiency of our algorithm by comparing $\log_2 m$ against $|R(N)|$. The former is denoted by $b(N,G)$ and is called the *effective reticulation number* of the algorithm for N and G. Additionally, $b(N) = \max_G b(N,G)$.

A theoretic bound for bicombining networks The analysis of our algorithm is difficult in general. However, we are able to establish the following theorem for bicombining reduced networks (see Appendix B for the proof).

Theorem 3.5.2. *For any bicombining reduced network N,* $b(N) \leq \log_2\left(\frac{1+\sqrt{5}}{2}\right) \times |R(N)| \approx 0.694|R(N)|$.

The above theorem indicates that Step 3 of our algorithm is executed $2^{0.694|R(N)|}$ times at most on an input network N and a phylogenetic tree. Hence, the algorithm has time complexity $O\left(2^{0.694|R(N)|}|E(N)||L(N)|\right)$.

Performance for random networks

For performance evaluations, we ran the program over thousands of phylogenetic trees and networks with 5 to 30 leaves on a cluster with 32 GB RAM and 8 cores. The simulated networks were generated by using a network generator reported by Zhang (2016).

Since the spaces of trees and networks with 10 or more leaves are both huge, it was very hard to have unbiased evaluation. For instance, for many random networks over 20 to 30 leaves generated by a computer, it is impossible to run our program on every tree against each of them. On the other hand, since there were only a small fraction of random trees displayed in each generated random network, our program finished very quickly for most trees. Additionally, it is impossible to ran the naïve method on a network with 30 reticulate nodes on any computer currently. These facts prevented us to have a clear picture of our program's performance on the entire spaces of trees and networks.

Here, we report the performance of our program on five groups of random networks with seven leaves. Each group contained 18 random networks. The networks in the k-th group had $5(1 + k)$ reticulate nodes, for each k from one to five. The percentages of the trees in the entire tree space with the same effective reticulation numbers were calculated and summarized in Figure 3.8. There are 10,395 phylogenetic trees with seven leaves.

We obtained several facts about our test. For all but three networks, over 90% of phylogenetic trees had an effective reticulation number of seven or less. There was a network with 20 reticulate nodes for which about 95% of phylogenetic trees had the effective reticulation number of 9 (represented by the orange bar in the middle). For the

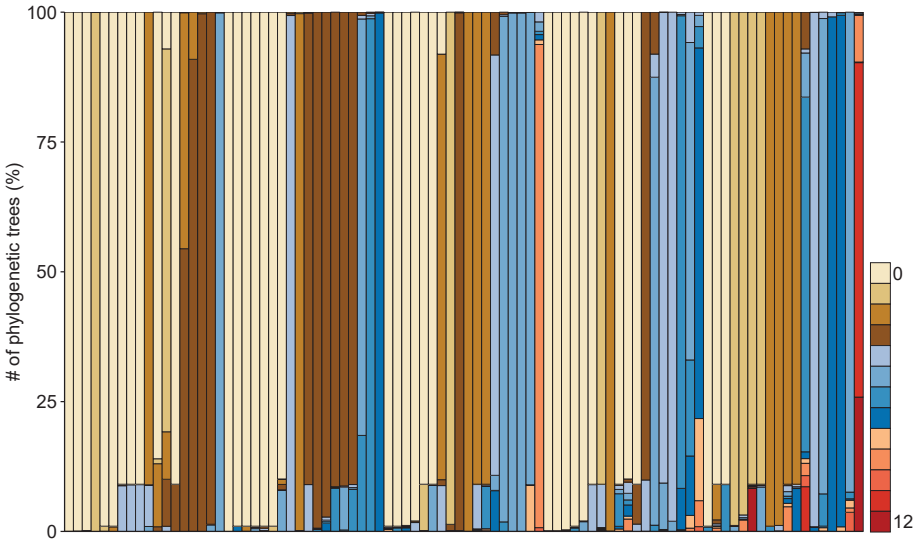

Figure 3.8: Summary of the performance of the TCP program on simulated random networks. The data were collected from our test on 90 random networks with seven leaves, which were divided into five groups. Each group contained 18 networks with the same number of reticulate nodes, arranged roughly in increasing order of the smallest effective reticulation number in a row along the x-axis. The five groups were arranged from left to right in increasing order of the number of reticulate nodes. Each of the stacked bars in a column represents the percentage of trees that had the same effective reticulation number when the program ran on them against the corresponding network.

rightmost network with 30 reticulate nodes, almost every tree had an effective reticulation number of 9 or more. These facts strongly suggest that the effective reticulation number is at most half the number of reticulate nodes in the network for each tree and each network.

In terms of CPU time, the whole test on $5 \times 18 \times 10,395$ network-tree pairs took 18 hours and 38 minutes, implying the program took 7.2 centiseconds on average for each network-tree pair.

Application to a network in the literature

We selected one of the largest networks in the literature to validate the performance of our TCP algorithm. This is a bicombining network over seven taxa (denoted A, Figure B.1) from (Charlton et al., 2008) that has as many as 32 reticulate nodes. This network is an ancestral recombination graph reconstructed to study the phylogenetic relationships among the M2 double-stranded RNA in the *Rhizoctonia* species complex. Here, we report $b(A, G)$ for each of the 10,395 possible phylogenetic trees over the seven taxa.

The distribution of $b(A, G)$ is presented in Table 3.1. There are 4,255 trees displayed

Table 3.1: The distribution of $b(A, G)$ in the space of phylogenetic trees over the same set of taxa as A.

	No. of trees displayed in A	**No. of inconsistent trees**
4	63	
5	561	
6	278	
7	544	
8	411	
9	478	
10	659	
11	766	40
12	433	416
13	62	5,352
14		322

Each entry is the number of trees with $b(A, G)$ being equal to the corresponding number in the first column

in the network model A. The effective reticulation numbers $b(A, G)$ for these trees vary from 4 to 14. However, $b(A, G)$ for the 6,140 trees not displayed in A ranges narrowly from 10 to 14.

Network A has 77 nodes (32 reticulate nodes, 38 tree nodes, and 7 leaves). Table 3.1 shows that for each tree, the effective reticulation number is less than $32/2$. Hence, based on our theoretical analysis, the program is roughly 851 ($= 2^{16}/77$) times as fast as the naïve approach.

3.6 Comparing arbitrary phylogenetic networks via soft clusters

In this section, we present an algorithm for the CCP on an arbitrary phylogenetic network, which was developed by using the decomposition technique reported in Section 3.4 We then extend it into an algorithm for computing the SRF distance. Similar to the TCP algorithm in Section 3.5, these two algorithms have exponential time complexity in the worst case, but they were shown to be very fast in evaluations on both random and real networks. As an application of the programs, we examined the differences of networks reconstructed for two datasets in the literature. We also conducted a preliminary study of the distributions of the RF and SRF distances in the phylogenetic network space.

Formally speaking, the CCP is formulated as below:

Instance: A phylogenetic network N over a set of taxa X and $B \subset X$.

Question: Is $B \in SC(N)$?

Let N_1 and N_2 be two networks over the same set of taxa X. The *SRF distance* between them is defined to be $(|SC(N_1) \backslash SC(N_2)| + |SC(N_1) \backslash SC(N_2)|)/2$, denoted by $d_{SRF}(N_1, N_2)$.

It is worth noting that the SRF distance is not a strict metric, since two distinct networks may represent the same set of soft clusters and hence the SRF distance between them will be zero (Huson et al., 2010). Nevertheless, the SRF distance provides a useful measure of network dissimilarity.

3.6.1 Methods

In this subsection, we describe the main idea of the CCP algorithm and the algorithm to compute the SRF distance.

Description of the CCP algorithm

With the aid of the generalized decomposition theorem, we extend the linear-time CCP algorithm for reticulation-visible networks in (Gunawan et al., 2017) to arbitrary networks. Although this algorithm seems simple, it has significantly less time complexity when the input network is binary.

Let N be a network over X and $B \subset X$, respectively. We examine a non-trivial exposed tree component C_r of N.

The reticulate nodes below C_r are divided into inner-reticulate nodes for which the parents are all in C_r, and cross-reticulate nodes for which some parents are not in C_r. We use $IR(C_r)$ and $CR(C_r)$ to denote the sets of inner- and cross- reticulate nodes, respectively. For example, in Figure 3.4, $IR(C_{\mathrm{r5}}) = \emptyset$ and $CR(C_{\mathrm{r5}}) = \{\mathrm{r4}, \mathrm{r7}, \mathrm{r8}\}$.

We use L_r to denote the set of leaves on which C_r is visible:

$$L_r = \{c(r') \mid r' \in IR(C_r)\} \cup L(C_r).$$

We use $\check{L}_r$ to denote the set of leaves below C_r which are in B and on which C_r is not visible:

$$\check{L}_r = \{c(r') \mid r' \in CR(C_r)\ s.t.\ c(r') \in B)\}.$$

For example, in Figure 3.4, $L_{r5} = \emptyset$ and we can get $\check{L}_{r5} = \{\text{leaf1}, \text{leaf2}\}$ when assuming $B = \{\text{leaf1}, \text{leaf2}, \text{leaf5}\}$.

Suppose that L_r is non-empty. C_r is then visible with respect to a leaf $\ell \in L_r$. We first check whether B is a soft cluster in C_r. This can be solved by a linear-time algorithm (Gunawan et al., 2017). If not, we then solve the CCP according to the relationship between L_r and B.

Let $\bar{B} = X \backslash B$. If $L_r \cap B \neq \emptyset$ and $L_r \cap \bar{B} \neq \emptyset$, B must be a soft cluster of a node in C_r if B is a soft cluster in N (Gunawan et al., 2017).

If $L_r \cap \bar{B} = \emptyset$, B may be a soft cluster of $\rho(C_r)$ or a node in a larger subnetwork containing C_r. Assuming that $r' \in CR(C_r)$, we then define:

$$N_a = N - \{(u, r') \in E(N) \mid (c(r') \notin B \wedge u \in V(C_r)\}$$
$$-\{(u, r') \in E(N) \mid (c(r') \in B \wedge u \notin V(C_r))\}.$$

The leaves below the root of C_r in N_a (i.e., $L([\rho(C_r)]_{N_a})$) are then $L_r \cup \check{L}_r$. We denote $L([\rho(C_r)]_{N_a})$ as $\hat{B}$ for convenience.

Since $L_r \subseteq B$ and $\check{L}_r \subseteq B$, $\hat{B} \subseteq B$. If $\hat{B} = B$, B is a soft cluster of $\rho(C_r)$ in N_a. Otherwise, if $\hat{B} \subset B$, we set:

$$\begin{cases} B' = (B \cup \{\ell\}) \backslash \hat{B}, \\ N'_a = N - [\rho(C_r)]_{N_a} + \ell. \end{cases} \tag{3.6}$$

If $L_r \cap B = \emptyset$, B may be a soft cluster of a node in C_r if $\check{L}_r \neq \emptyset$. Otherwise, when B is not a soft cluster of a node in C_r and $r' \in CR(C_r)$, we define:

$$N_b = N - \{(u, r') \in E(N) \mid (c(r') \notin B \wedge u \notin V(C_r))\}$$
$$-\{(u, r') \in E(N) \mid (c(r') \in B \wedge u \in V(C_r))\}.$$

We can then set:

$$N'_b = N - [\rho(C_r)]_{N_b} + \ell. \tag{3.7}$$

With this notation, we can get Theorem 3.6.1 for arbitrary networks, which is similar to a theorem proved for reticulation-visible networks in (Gunawan et al., 2017). Theorem

3.6.1 is proved in Appendix B.

Theorem 3.6.1. *Assume that C_r is a non-trivial, exposed and visible tree component in a network N over the taxa set X, and that $B \subset X$. Let L_r, $\hat{B}$, B', N'_a, and N'_b be defined above.*

(i) *If $\hat{B} \subset B$, B is a soft cluster in N if and only if B' is a soft cluster in N'_a.*

(ii) *If B is not a soft cluster of a node in C_r and $L_r \cap B = \emptyset$, B is a soft cluster in N if and only if B is a soft cluster in N'_b.*

Suppose that C_r is not visible. If $C_r \neq \{c(r)\}$, there is at least one reticulate node r' right below C_r such that $C_{r'}$ is trivial and at least one parent of r' is not in C_r. If $C_r = \{c(r)\}$ and $c(r) = r'$, then at least one parent of r' is not r. We can now define:

$$N' = \begin{cases} N - \{(u, r') \in E(N) \mid u \notin C_r\} & \text{if } C_r \neq \{c(r)\} \\ N - \{(u, r') \in E(N) \mid u \neq r\} & \text{if } C_r = \{c(r)\} \end{cases} \tag{3.8}$$

and

$$N'' = \begin{cases} N - \{(u, r') \in E(N) \mid u \in C_r\} & \text{if } C_r \neq \{c(r)\} \\ N - \{(u, r') \in E(N) \mid u = r\}. & \text{if } C_r = \{c(r)\} \end{cases} \tag{3.9}$$

Clearly, B is a soft cluster in N if and only if B is a soft cluster in either N' or N''.

In consideration of all the cases above, we have come up with an algorithm for solving the CCP on an arbitrary network, which is given in Figure 3.9.

Description of the SRF distance algorithm

We now use the CCP algorithm to compute the SRF distance between two arbitrary networks on the same taxa set X.

For X, we define a *k-cluster* as a cluster having k taxa. We enumerate all the possible clusters over X by generating all the k-clusters of X for each k ranging from 1 to $|X| - 1$. We then call the CCP algorithm on each cluster to see whether it is a soft cluster in only one network.

The time complexity of this SRF distance algorithm is $O(2^{|L(N)|}T(N))$, where $T(N)$ is the time complexity of the CCP algorithm.

In contrast, the program for computing the SRF distance in Dendroscope first finds trees displayed in each network, then extracts clusters from these trees to get the soft

THE CCP ALGORITHM

Input: An arbitrary network N and a cluster $B \subset L(N)$.
Output: TRUE if $B \in SC(N)$ and FALSE otherwise.

0. If $|V(N)| \leq 1$, output FALSE;
1. If $|B| = 1$, output TRUE;
2. Select an exposed and non-trivial tree component C_r;
3. If C_r is visible with respect to a leaf ℓ, do {
   3.1 If $B \in SC(C_r)$, output TRUE;
   3.2 If $B \notin SC(C_r)$, do {
      3.2.1. If $L_r \cap B \neq \emptyset$ and $L_r \cap \bar{B} \neq \emptyset$, output FALSE;
      3.2.2. If $L_r \cap \bar{B} = \emptyset$, do {
         3.2.2.1. If $\hat{B} = B$, output TRUE;
         3.2.2.2. If $\hat{B} \subset B$, set $B = B'$ and $N = N'_a$
               as defined in Eqn. (3.6);
       }
      3.2.3. If $L_r \cap B = \emptyset$, set $N = N'_b$ as defined in Eqn. (3.7);
      3.2.4. Output TRUE if $B \in SC(N)$ and FALSE otherwise;
    }
   }
4. If C_r is not visible but a leaf exists below C_r, do {
   4.1. Compute N' and N'' as defined in Eqn. (3.8) and (3.9);
   4.2. If $B \in SC(N')$, output TRUE;
   4.3. If $B \in SC(N'')$, output TRUE;
   4.4. Output FALSE;
   }
5. If no leaf exists below C_r, do {
   5.1. If $B \in SC(N - r - C_r)$, output TRUE;
   5.2. Output FALSE;
   }

Figure 3.9: An algorithm for solving the CCP on an arbitrary network.

clusters displayed in each network, and finally traverses the two sets of soft clusters to compute their symmetric difference. If the networks are bicombining, the time complexity for this method is $O(2|L(N)| * 2^{|R(N)|} + 2q)$, where q is the number of the soft clusters displayed in a network. We will compare this approach and our SRF distance program in next section.

3.6.2 Results

In this subsection, we first report the performance of the CCP algorithm, including its time complexity as well as performance on both simulated and empirical networks. Next we show the comparison of the program in Dendroscope and our program for the SRF

distance. As an application, we then examine the SRF distances between phylogenetic networks reconstructed from two datasets in the literature. Finally, we present our preliminary comparison of the RF and SRF distances.

Performance of the CCP program

Theoretical analysis of the time complexity According to the analysis in (Gunawan et al., 2017), the runtime of Step 4 of the CCP algorithm is $O(|E(C_r)|)$, where $E(C_r)$ is the set of edges in the tree component C_r. Thus the time complexity of the CCP algorithm is $O((m+1)|E(N)|)$ for a general network N, where m is the number of times Step 4 is executed. Note that m should be an exponential function of $|R(N)|$ because of the NP-completeness of the CCP. If N is a bicombining reduced network, the time complexity of the CCP algorithm is $\left(2^{0.694|R(N)|}|E(N)|\right)$.

We denote $\log_2(m)$ as $b(N,B)$ and call it the *effective reticulation number* of the CCP algorithm for the network N and the cluster B. We use $b(N) = \max_B b(N,B)$ to represent the effective reticulation number of the CCP algorithm for the network N.

To the best of our knowledge, the only previously known algorithm for solving the CCP on an arbitrary network is the naïve algorithm which considers all the trees displayed in a network and checks whether the input cluster is displayed in one of them. The number of possible trees displayed in a network N can be as large as $\prod_{r\in R(N)} deg^-(r)$, where $deg^-(r)$ is the indegree of r. This number equals $2^{|R(N)|}$ when N is bicombining. It takes $O(|L(T)|)$ time to check whether a cluster is displayed in a tree T (Huson et al., 2010). Thus the effective reticulation number seems to be a good indicator of the efficiency of the CCP algorithm. If $\log_2(m)$ is smaller than $|R(N)|$, the CCP algorithm will be faster than the naïve algorithm in theory.

Performance on random networks We examined the performance of the CCP program on random networks in term of the effective reticulation number. The tests were done on computers each with 32 GB RAM and a 2.1 GHz AMD Opteron 32-core CPU.

We tested the CCP program on random networks with 10 to 30 leaves and 10 to 80 reticulate nodes. Given that the number of clusters over 15 leaves is huge, it was hard to conduct the evaluation on the whole space of clusters. We therefore generated random clusters for testing on networks with more than 15 leaves. According to the results, the

effective reticulation number for each network–cluster pair was frequently smaller than half the number of reticulate nodes in the network.

Here, we report the performance of the CCP program on five groups of networks with 10 leaves and all the possible 1022 (= $2^{10} - 2$) clusters. Each group contained 20 networks, and the networks in the k^{th} group had $5(1 + k)$ reticulate nodes for each k from 1 to 5. The wall clock time on 102,200 (= $5 \times 20 \times 1022$) network–cluster pairs was 15 minutes and 15 seconds, implying that on average, the program took about one centisecond for each network–cluster pair.

Figure 3.10 shows the percentages of the clusters displayed in the entire cluster space with the same effective reticulation numbers for each network. Several facts were observed from the test. Firstly, the effective reticulation numbers for the networks in each group increase with the number of reticulate nodes. For example, the effective reticulation numbers for most networks are <5 for the first group, whereas the effective reticulation numbers for more than half of the networks are >9 for the last group. Secondly, there are at least three distinct values of effective reticulation numbers for each network and all the clusters, except for five networks. The effective reticulation number of value one appears for all the networks, since it is easy to determine whether the trivial clusters are soft clusters displayed in a network. Thirdly, the highest effective reticulation number 12 only appears for the 12^{th} network in the last group and one cluster, which is barely seen in Figure 3.10 because of the extremely low percentage.

Application to a network in the literature We selected the same network as in Section 3.5.2 to validate the performance of the CCP algorithm. Our test showed that all the clusters on the seven taxa appear as soft clusters displayed in the network A. We calculated $b(A, B)$ for each cluster B on the seven taxa. The distribution of $b(A, B)$ is shown in Table 3.2. The effective reticulation number $b(A)$ is 8, 1/4 of the number of reticulate nodes in A. This suggests that the CCP program is about thousands of times as fast as the naïve method for this real network.

Performance of the program for the SRF distance on random networks

The comparison tests for the program in Dendroscope and our program were performed on computers each with 128 GB RAM and a 2.6 GHz Intel Xeon E5-2690 24-core CPU.

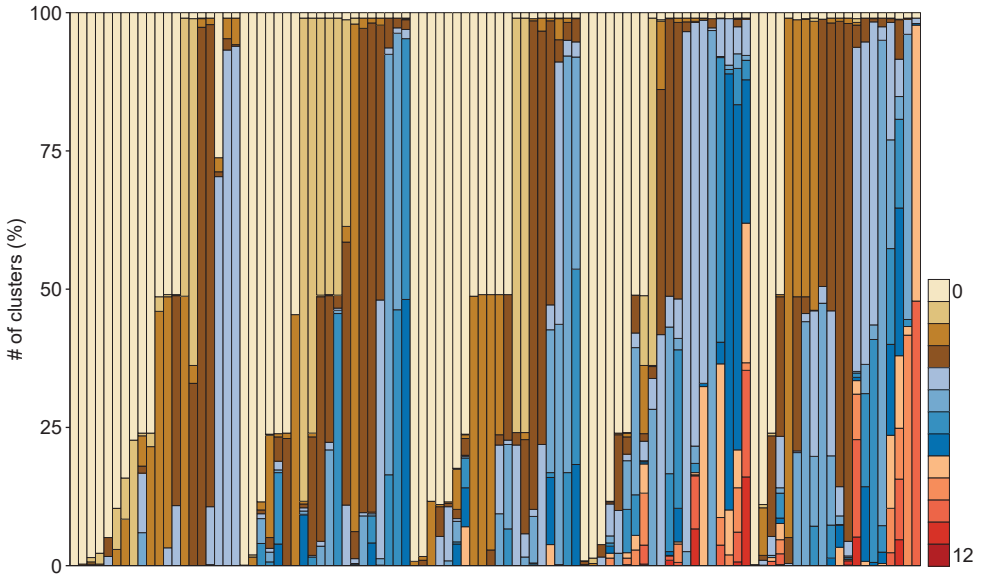

Figure 3.10: Summary of the performance of the CCP program on five groups of random networks with 10 leaves. Along the x-axis, the five groups were arranged from left to right in increasing order of the number of reticulate nodes. The 20 networks in each group were arranged roughly in increasing order of the smallest effective reticulation number. Each stacked bar in a column represents the percentage of clusters that had the same effective reticulation number when the program ran them against the corresponding network.

Table 3.2: The distribution of $b(A, B)$ in the space of clusters over the same set of taxa as the network A.

$b(A, B)$	0	2	4	6	8
#Cluster	8	3	45	49	21

#Cluster refers to the number of soft clusters with the same value of $b(A, B)$.

For the generation of random networks, we considered six cases. In the k^{th} case, we generated six groups of network pairs. The j^{th} group consists of 3000 pairs of networks with $4k$ leaves and $kj/4$ reticulate nodes, where k was from 1 to 6 and $j = 1, 2, 4, 5, 6$.

In the comparison test, we computed the SRF distance for each pair of networks in every group. The results are summarized in Figure 3.11.

Our program ran faster than the program in Dendroscope for networks with up to 16 leaves. However, our program became slower than the latter when there were more than 16 leaves. This is reasonable, since the number of clusters displayed increases exponentially with the number of taxa and it takes even long time for our program to merely list all the possible clusters when there were more than 16 leaves.

Additionally, the memory usage of our program was extremely low compared with

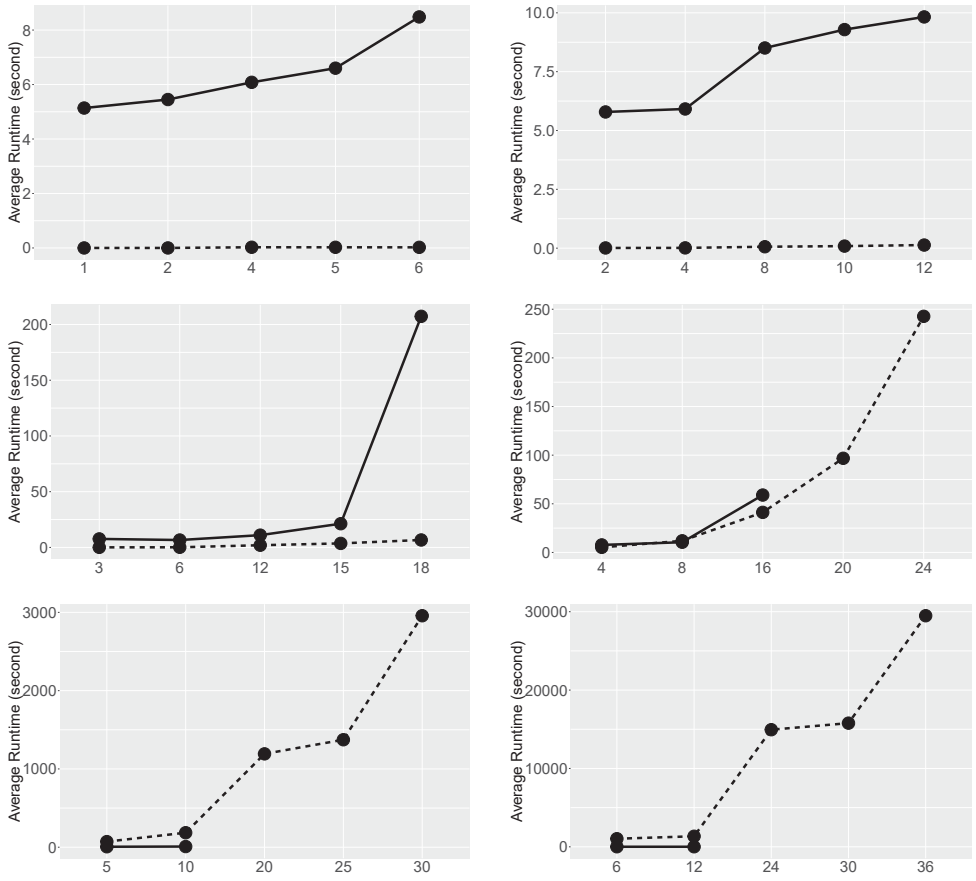

Figure 3.11: Performance of our program (dashed line) and the program in Dendroscope (solid line) on random networks. The x-axis represents the number of reticulate nodes in a network.The random networks examined had 4 (top left), 8 (top right), 12 (middle left), 16 (middle right), 20 (bottom left), and 24 leaves (bottom right).

the program in Dendroscope. The memory usage of the Dendroscope program increased rapidly with the number of reticulate nodes in a network. For example, the average maximum resident memory for networks with 12 leaves and 18 reticulate nodes was around 95 GB, which is approximately six times that for networks with 12 leaves and 15 reticulate nodes. Because of this, the average runtime of the Dendroscope program for networks with 12 leaves and 18 reticulate nodes sharply increased. During test, the Dendroscope program failed to get results for networks with more than 12 leaves and 20 reticulate nodes. Hence, some data points are missing for the Dendroscope program in the two panels at the bottom in Figure 3.11. In contrast, our program can run on networks with more than 30 reticulate nodes. Even for networks with 24 leaves and 36 reticulate nodes, the average maximum resident memory of our program was less than 32 MB. Thus the test shows that our program is computationally efficient when the number of reticulate nodes in the input network is large.

Table 3.3: The average pairwise SRF distances between the output networks from Hybroscale on three sets of gene trees reported by van Iersel et al. (2010a).

Gene trees	#Taxa	#Ret	#Networks	Avg pairwise SRF distance
rbcL, *waxy*, *ITS*	11	6	63	12.2
ndhF, *rbcL*, *waxy*	12	5	123	8.0
phyB, *rbcL*, *rpoC*	15	6	40	1.4

#Ret refers to the number of reticulate nodes in the reconstructed networks.

Although our program runs slow for networks with many leaves, it can be easily parallelized for speeding up. We used OpenMP to implement a parallel version of it by parallelizing the enumerations of clusters. This parallel version ran at least 20 times faster than the original program with slightly extra memory. For 3,000 pairs of networks each with 20 leaves and 25 reculation nodes, the parallel version finished in about 36 seconds with less than 40 MB memory on average.

Computing the SRF distances on real biological data We first computed the SRF distance between networks over a set of grass species. The Proaceae dataset, originally from Grass Phylogeny Working Group (2001), has often been used for validating network reconstruction methods. The dataset contains sequences for six loci: *ITS* (internal transcribed spacer of ribosomal DNA), *ndhF* (NADH dehydrogenase, subunit F), *phyB* (phytochrome B), *rbcL* (ribulose 1,5-biphosphate carboxylase/oxygenase, large subunit), *rpoC* (RNA polymerase II, subunit β''), and *waxy* (granule bound starch synthase I). Rooted binary gene trees were built for these loci previously by (Schmidt, 2003). From the six trees, van Iersel et al. (2010a) constructed 57 subsets of gene trees for comparisons of network reconstruction methods.

A recent method called Hybroscale (Albrecht, 2015) can compute all the representative networks with the minimum number of reticulate nodes from a set of multiple binary phylogenetic trees. We ran Hybroscale on three subsets of gene trees from the grass dataset, which are on 11, 12, and 15 taxa, respectively (Table 3.3). The reconstructed networks have less than seven reticulate nodes. Since there are tens of output networks for each input dataset, we computed their pairwise SRF distances to examine their dissimilarity. As shown in Table 3.3, the average SRF distances between the networks for all the datasets are relatively small, which implies that the computed networks are rather similar.

On the other hand, different network reconstruction methods on the same data could produce very different networks. Using five gene trees (*ITS*, *ndhF*, *phyB*, *rbcL*, *rpoC2*), we constructed three networks: a cluster network (Figure B.2) obtained from a program in (Huson and Rupp, 2008), a galled network (Figure B.3) obtained from a program in (Huson et al., 2009), and a reticulate network (Figure B.4) obtained from PIRN (Wu, 2010). Since the original network reconstructed by PIRN had reticulate nodes with more than one child and leaves with more than one parent, it was transformed into an equivalent one satisfying our definition in this work. The three networks have 18, 7, and 13 reticulate nodes and contain 445, 261, and 209 soft clusters, respectively. The SRF distance between the cluster network and the galled network is 199. The SRF distance between the galled network and the reticulate network is 118. The SRF distance between the reticulate network and the cluster network is 185. This suggests that the galled network is more similar to the reticulate network than to the cluster network. This also reflects that the SRF distance is sensitive to the structural properties of phylogenetic networks.

We also computed the SRF distance between networks over six mosquito species. To study phylogenetic relationships and introgression among six mosquito species in the *Anopheles gambiae* species complex, Fontaine et al. (Fontaine et al., 2015) constructed a network (denoted $M1$) by employing tree-based methods on the whole-genome sequences. Later, Wen et al. (Wen et al., 2016) rebuilt a similar network (denoted $M2$) for the six species by directly applying a network inference method on the gene trees. The two networks are shown in Figure B.5. $M1$ has three reticulate nodes and $M2$ has four reticulate nodes. There are 18 and 24 soft clusters displayed in $M1$ and $M2$, respectively. The SRF distance between $M1$ and $M2$ is 7, implying that the two networks are still quite different in the embedded soft clusters.

Comparison of the RF distance and the SRF distance

Although the RF and SRF distances were proposed to measure the dissimilarity of networks, their relationship is unclear (Huson et al., 2010). Here, we performed a preliminary comparison of these two measures.

Given a fixed number of leaves and reticulate nodes, we generated 100,000 random network pairs and computed their RF and SRF distances. Figure 3.12 shows the distri-

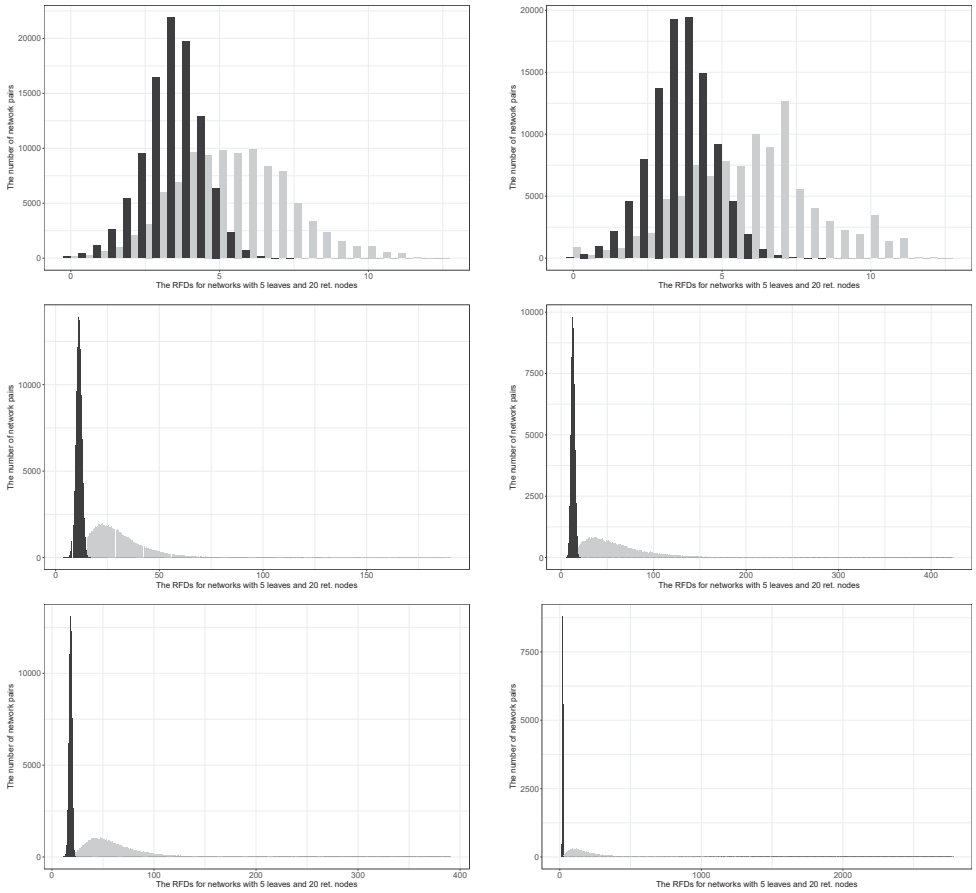

Figure 3.12: The distribution of the RF (black) and SRF (grey) distances between random networks. Histograms of the number of network pairs with k leaves and m reticulate nodes, where (k, m) = (5, 10) (top left), (5, 20) (top right), (10, 10) (middle left), (10, 20) (middle right), (15, 10) (bottom left), and (15, 20) (bottom right).

butions of these two measures in the space of networks with different numbers of leaves and reticulate nodes. The results suggest the following three facts:

(i.) There are at least as many soft clusters as clusters displayed in a network. Therefore, as expected, the SRF distance has larger range than the RF distance.

(ii.) The RF distance seems to have a normal distribution of small mean and small variance.

(iii.) The distribution of the SRF distances seems not to be normal. It is skewed towards small distances (especially for networks with more leaves) and a small fraction of network pairs had much larger SRF distances than the average SRF distance.

Taken altogether, these three facts indicate that the SRF distance is a fine metric for networks and hence more suitable than the RF distance for measuring the dissimilarity of networks.

3.7 Summary

In chapter 3, we study the modeling of LGTs by phylogenetic networks. Although tree-based network and more specifically LGT network provide a natural representation of LGT, challenges in efficiently reconstructing a network from biological data still prevent the wide application of LGT modeling in networks. Some basic problems arising in network reconstruction and verification are still to be solved, including the NP-complete TCP and CCP.

Based on a network decomposition technique, we implemented fast programs for solving the TCP and CCP for arbitrary phylogenetic networks in which nodes are not necessarily binary. The resulting CCP program is further extended into a computer program for fast computation of the SRF distance between phylogenetic networks. These programs were implemented in C. The TCP program is available on `http://www.math.nus.edu.sg/~matzlx/tcp_package`. The CCP program and the SRF program are available on `https://github.com/icelu/PlyloNetwork`.

For the TCP program and the CCP program, the theoretical analysis shows that they are much faster than the naïve programs. The evaluations on random and empirical networks demonstrate that they are fast enough to solve real instances arising in evolutionary genomics. For the SRF distance program, our simulation experiments show that it ran fast for networks with an intermediate number of leaves and reticulate nodes. The comparisons between the RF and SRF distance suggest that it is more appropriate to use the SRF distance to measure networks dissimilarity of network. Therefore, the SRF distance program is ready for assessing a network reconstructed by a new method via comparing it with other networks.

In short, although these programs have exponential time complexity in the worst case, they may facilitate reconstructing and validating network models in evolutionary and comparative genomics. The programs for solving the TCP and CCP may serve as a stepping stone towards the development of methods for reconstructing a LGT network. The program for computing the SRF distance can then help to compare the reconstructed LGT networks.

Chapter 4

Detecting LGTs via multiple methods - a case study

4.1 Introduction

In this chapter, we first provide a brief description of challenges in LGT detection. Then, we give a detailed review about known methods for detecting LGTs. Finally, we report a case study which investigates the performances of different LGT detection methods on cyanobacterial genomes.

The major evidence to detect LGTs is the indirect information reflected in the DNA or protein sequences, as these sequences provide the best clues about their origin and ancestry (Ochman et al., 2000). However, The compound factors in evolutionary history left intricate puzzles. For example, the available sequences may have undergone important signal loss due to reticulate evolution and several frequently occurring biological processes that dim the evolutionary history, such as amelioration and obliteration by subsequent events (Ragan and Beiko, 2009). Therefore, it is computationally challenging to infer LGT events from a collection of extant DNA or protein sequences.

Three major challenges in LGT detection are listed below:

1. The laterally transferred sequences may evolve in the host genome via amelioration (Lawrence and Ochman, 1997). Amelioration means that the transferred sequences become adjusted to the composition properties of the resident genome. Because the transferred sequences may subject to the same mutational processes as the recipient after being integrated, they may have evolved and adopted the

signature of native sequences. As a result, it is challenging to distinguish between foreign sequences and native sequences.

2. The scenarios of LGTs may be quite complicated. The genes transferred into one organism may be further transferred to another organism. In case of multiple LGT events, it is hard to detect the exact order and direction of each event. LGT, together with other evolutionary mechanisms such as gene duplication and loss, can generate gene histories which deviated from species history. It is non-trivial to distinguish these different events. To validate the LGT candidates, other possibilities have to be eliminated.

3. It is difficult to detect LGTs occurring between closely related organisms, as the donor and recipient organism share high sequence similarity.

It is still not clear which measure can best capture the clues left by LGTs. In practice two measures are commonly used: anomalous sequence composition and unexpected evolutionary history (Koonin et al., 2001). The unexpectedness of evolutionary history can be measured by several criteria, such as unexpected tree topologies between gene tree and species tree, unanticipated ranking of sequence similarity among homologs, and unusual phyletic patterns (or sporadic phylogenetic distribution).

Different kinds of LGT detection methods have been developed by utilizing different criteria. Ravenhall et al. (2015) provided a comprehensive review of these methods. Here, we classify these methods from a slightly different perspective. As shown in Figure 4.1, the available LGT detection methods can be roughly classified into two categories, which are further divided into several subcategories.

4.2 Literature review

In this section, we first briefly review compositional methods for LGT detection. Then we discuss implicit phylogenetic methods and tree-based methods in more detail. The network-based methods for LGT modeling have been reviewed in Section 3.2.1. After obtaining a LGT network, it is easy to figure out LGT events. for this reason, we do not review network-based methods here.

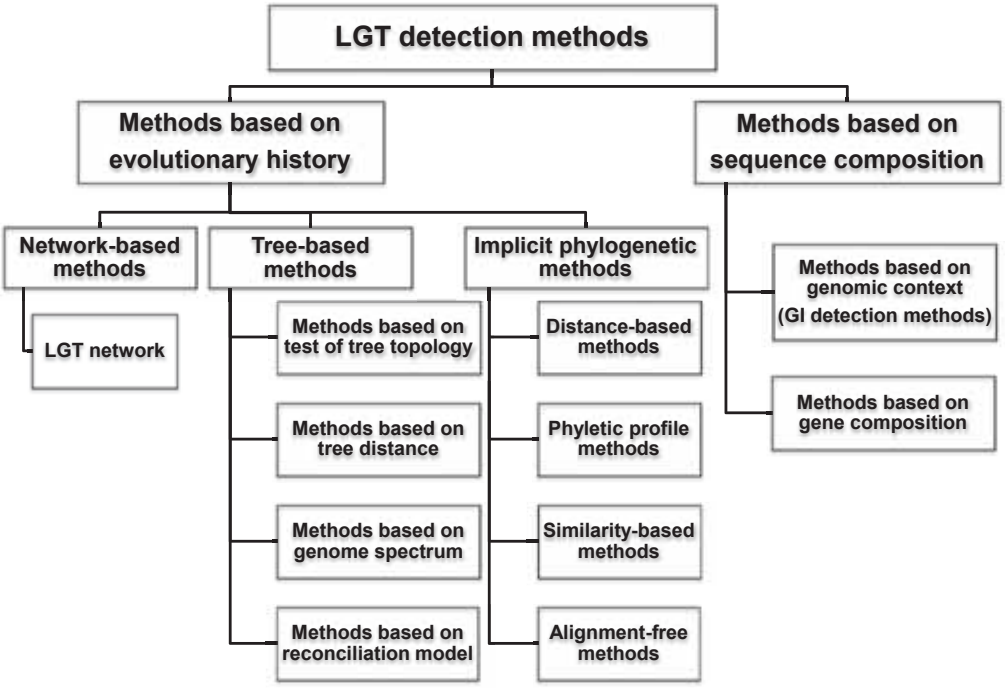

Figure 4.1: The hierarchical overview of computational methods for detecting LGTs.

4.2.1 Compositional methods for LGT detection

Compositional methods detect sequences acquired by LGT based on the discrepancies of compositional properties between laterally transferred regions and native regions in a single genome.

Many compositional methods were developed for detecting GIs. GIs provide the genomic context for locating individual genes probably acquired by LGT. Even if the species-specific features of a transferred gene have become weak, the location of this gene in a well-specified GI may still suggest its non-native origin (Ravenhall et al., 2015). GI prediction methods have been reviewed in Section 2.2.

Methods based on gene sequence composition are often designed for LGT detection. Azad and Lawrence (2011) gave a comprehensive review of the available methods developed from 1998 to 2009. They also proposed a multiple-threshold method to improve the performance of most compositional methods via two steps Azad and Lawrence (2011), which firstly identify alien and native genes with strong features by using conservative thresholds and then classify genes of ambiguous features by incorporating additional annotations. To reduce the interference of potential foreign genes on computing the average measures for the whole genome, Elhai et al. (2012) proposed a method called Core

Gene Similarity (CGS), which utilizes the differences of octamer frequencies between a query gene and a selected set of conserved core genes.

Pros and Cons

Methods based on sequence composition can efficiently uncover all the putative genes of foreign origin given the genome sequence and/or gene annotations of an organism. Because they do not rely on comparisons with other genomes, they can identify orphan genes of alien origin that do not have orthologs.

However, most foreign genes detected by compositional methods are recently transferred, because anciently transferred sequences may have adapted to the host genome via amelioration. There are several other limitations of compositional methods as well. These methods may not provide information of the potential donors of foreign genes based on just one genome. They are also well known to be error-prone, since codon bias and base composition were discovered as poor indicators of LGT (Koski et al., 2001). Sequences with ambiguous compositional features may be poorly classified. Some native regions with atypical composition for reasons other than LGT (such as stochastic factors and selection for unusual composition) may easily be detected as false positives (Cortez et al., 2009).

4.2.2 Implicit phylogenetic methods for LGT detection

Implicit phylogenetic methods utilize evolutionary relationships of different organisms without building phylogenetic trees. They often use three indicators to detect LGTs: unexpected evolutionary distances, unusual taxonomy distribution, and anomalous sequence similarity scores. Most of these methods use sequence alignments to compute the pairwise evolutionary distances between organisms. In contrast, a few methods detect LGTs without relying on alignment. Based on using alignment or not, implicit phylogenetic methods can be classified into alignment-based methods and alignment-free methods.

Alignment-based methods

According to the criteria used, alignment-based methods can be further classified into three subcategories: distance-based methods, phyletic profile methods, and similarity-

based methods.

Distance-based methods Distance-based methods aim to detect unexpected evolutionary distances between genes in a cluster of orthologous group (COG) and corresponding genomes, assuming that each relevant organism is represented by only one protein in a set of COGs. They are often based on the molecular clock hypothesis which states that the evolution rate of homologous genes is approximately constant in different species.

The evolutionary distance can be measured in different ways. Clarke's phylogenetically discordant test method (Clarke et al., 2002) measures the distance by the ranking of reciprocal best blast match for each gene. Some other methods measure the differences based on substitution rates, as genes inherited by descent should often accumulate substitutions at a constant rate (Novichkov et al., 2004; Dessimoz et al., 2008). The significant deviation of gene distance from genome distance can be detected by statistical techniques. For example, DLIGHT uses a likelihood test to check the hypothesis of a LGT event and no LGT event (Dessimoz et al., 2008).

Phyletic profile methods Taxonomy distribution among organisms is often depicted by phyletic profile, the presence and absence of genes in multiple organisms. An unexpected phyletic profile may suggest the occurrence of LGT. For instance, if the orthologs of a gene are present in all or almost all sequenced bacterial genomes but only in one archaeon genome, it is highly likely that this gene was transferred from a bacterium to the archaeon (Novichkov et al., 2004).

Phyletic profile methods try to use the most parsimonious explanation (via LGT or gene duplication or gene loss) to infer the patchy distribution of genes. Genes present only in distantly related organisms are strong indications of LGT, because the alternative explanation by differential gene loss would require all the closely related organisms carried this gene previously, which seems unfavourable. For example, GeneTrace (Kunin and Ouzounis, 2003) assumes that: if a gene is present in multiple members of a clade, it was probably vertically inherited; if a gene is absent in some members of a clade, there was probably gene loss; if a gene shows patchy distribution in distantly related clades, it was likely to be laterally transferred. This kind of methods usually first finds a set of

homologs among selected organisms and then investigates the taxonomy distribution of these homologs. Similarity search methods or clustering algorithms are often used to find homologs.

Similarity-based methods Similarity-based methods assume that laterally transferred genes have anomalous high sequence similarity with genes from evolutionarily distantly related organisms. The rational is that more similar sequences are often more closely related. In practice, BLAST-based best match methods are often used for LGT detection. The presence of top BLAST hit in an unrelated organism for a gene is also a straightforward use of phyletic profile.

Some sophisticated techniques based on BLAST were also designed, because the top BLAST hit may not be most phylogenetically relevant. One earlier method Darkhorse (Podell and Gaasterland, 2007) integrates the lineage frequency of BLAST matches over the whole query genome to rank all the laterally transferred candidates. A recent method HGTector (Zhu et al., 2014) systematically analyzes the BLAST hit distribution patterns based on a predefined evolutionary classification.

Alignment-free methods

There are only a few available alignment-free methods for LGT detection. Three earlier methods are based on frequency profile of k-mers (FFP) (Sandberg et al., 2001; Dalevi et al., 2006) and Chaos Game Representation (CGR) (Dufraigne et al., 2005), respectively. These three methods are also classified as compositional methods (Azad and Lawrence, 2011), as they use k-mer frequencies in the genome sequences to detect LGTs. The two methods based on FFP use naïve Bayesian classifier to predict the most probable origin of a query sequence from all the input genomes. Their major differences lie on the underlying assumption in computing the probability $P(M|G)$ of a k-mer M within genome G. The method in (Sandberg et al., 2001) simply assumes the probabilities $P(M|G)$ for all k-mers are independent, whereas the method in (Dalevi et al., 2006) utilizes Markov models to handle dependencies among k-mers. The method based on CGR assumes that a DNA fragment whose CGR deviates from that of the whole genome was laterally transferred. So this method can also be used with only one genome. When applied to multiple genomes, this method assumes that genomes whose CGRs

are similar to the CGR of the putative laterally transferred DNA fragment are potential donors.

One recent method ALFY (Domazet-Lošo and Haubold, 2011) makes use of exact word matching to search local homologs of a query genome from a set of other genomes, assuming that the sudden changes of local homology along the query genome indicate the occurrence of LGT. A latest method developed by Cong et al. (2016) adopts TF–IDF, a measure frequently used in information retrieval, to identify LGTs. This method requires to first divide the input genomes into taxonomic groups. Then, TF represents frequencies of k-mers from a DNA segment in genomes of its own group and IDF represents frequencies of k-mers from a genome in all the groups.

Pros and Cons

Distance-based methods are fast as they do not rely on multiple sequence alignments or inference of gene trees. In spite of the absence of phylogenetic trees, this kind of methods can still provide details of LGTs in a robust way, including the donor and recipient, and the distance to past LGT events (Dessimoz et al., 2008). But the clock assumption used in distance-based methods is often violated in practice.

Phyletic profile methods are also very efficient, as the only time-consuming part is the homology search. However, they may not detect orthologous replacement, which cannot be depicted by patchy distribution of genes in a gene family.

Similarity-based methods can systematically identify all putative laterally transferred genes in an entire genome, based on all-against-all search of each protein against the database. But blasting each protein in a large genome against a huge database takes a considerable amount of time. Besides, errors in database may affect the predictions. The similarity may also be due to conservation or convergent evolution.

Alignment-free methods are usually much faster. They can quickly identify a candidate list of putative laterally transferred regions, serving as effective complements of alignment-based methods (Domazet-Lošo and Haubold, 2011). They also avoid issues in aligning large amounts of data which may be too divergent to get meaningful alignments. Besides, they do not take genes as analysis unit and thus can detect any genomic regions of potential lateral origin. But the predictions from alignment-free methods still need to be further investigated by other more reliable methods.

4.2.3 Phylogenetic tree-based methods for LGT detection

Phylogenetic tree-based methods detect LGTs based on the assumption that a gene with significantly different evolutionary history may be laterally transferred, since all native genes in the genome are supposed to have similar evolutionary histories.

The application of tree-based methods for LGT detection in practice is quite cumbersome, mainly including four steps: (1) selecting gene families; (2) aligning sequences of gene families; (3) reconstructing (unrooted) gene trees and (rooted) reference tree; (4) identifying incongruences between gene trees and reference tree. However, most tree-based methods are specifically designed to solve the last step.

Ideally, the gene tree for a set of orthologous genes related through speciation events should be consistent with the species tree. But the occurrence of xenologs arisen by LGTs may cause inconsistencies between the two trees. To detect LGTs, different gene trees are compared with the reference species tree to assess the significance of any incongruities. The significant incongruities between a gene tree and the species tree may be caused by LGT.

Based on ways to discriminate topology incongruences between trees, tree-based methods can be roughly divided into four categories: methods based on test of tree topology, methods based on tree distance, methods based on genome spectrum, and methods based on reconciliation model. Some earlier tree-based methods were assessed and reviewed in (Poptsova and Gogarten, 2007; Poptsova, 2009).

Methods based on test of tree topology

Methods based on test of tree topology mainly use likelihood-based statistical tests to assess the likelihood of the topology of a gene tree given the species tree as the null hypothesis (Ravenhall et al., 2015). The commonly used tests include: Kishino-Hasegawa (KH) test, Shimodaira-Hasegawa (SH) test, Approximately Unbiased (AU) test (Poptsova, 2009), and expected likelihood weights (ELW) test (Abby et al., 2010). The greater the p-value, the more likely the tested gene tree is consistent with the species tree. The gene tree with p-value lower than a certain threshold is considered to be incongruent with a species tree, which is probably due to LGT. However, this kind of methods cannot detect specific LGT events.

Methods based on tree distance

Methods based on tree distance detect LGTs by comparing gene trees and species tree with different distance measures. If the distance between a gene tree and the species tree is significantly larger than the mean of the distances between each gene tree and the species tree, LGT might have occurred. The commonly used distance measures include: Robinson–Foulds (RF) distances, Subtree Pruning and Regrafting (SPR) distances, Maximum Agreement Forest (MAF). The RF distance only gives the number of different bipartitions between trees (Poptsova, 2009). SPR and MAF are said to be the most appropriate models to explain LGTs (Abby et al., 2010), aiming to identify minimal series of steps via which the gene tree and species tree can be rendered congruent.

SPR refers to pruning a subtree from a tree by cutting one edge and then regrafting the subtree by the same cut edge to another edge of this tree (Beiko and Hamilton, 2006). In particular, the edge regrafted onto represents the donor taxon and the cut edge corresponds to the recipient taxon. A series of SPR operations consist of an edit path which can be applied to the species tree to generate a topology consistent with the gene tree. SPR has been utilized by several programs to infer LGTs, such as EEEP (Beiko and Hamilton, 2006), RIATA-HGT (Nakhleh et al., 2005; Than and Nakhleh, 2008), and T-Rex (Boc et al., 2010).

MAF refers to finding the smallest number of edges to cut in both gene tree and species tree in order to obtain two identical “forests" of rooted subtrees (Abby et al., 2010). Prunier utilizes MAF to detect LGTs by statistical reconciliation of phylogenetic forests (Abby et al., 2010).

Methods based on genome spectrum

Genome spectral approaches first decompose a gene tree into small subtrees, mainly bipartitions or quartets, and then check whether the statistically supported substructures are congruent with those represented by the majority gene trees or not (Zhaxybayeva, 2009). Hence, they do not require a completely resolved reference tree.

Bipartition is a decomposition of a phylogenetic tree into two parts connected by a single edge. In bipartition analysis, a gene tree which gives rise to statistically

supported bipartitions conflicting with those represented by a plurality of gene trees can be considered to have experienced LGT events (Poptsova and Gogarten, 2007).

Since the support for bipartitions tends to be smaller when the number of organisms in the gene tree increases, quartet decomposition method was proposed. Quartet is a subtree with four leaves. The rational is similar as that of bipartition methods. Specifically, if the statistically supported quartets of a gene tree are incompatible with those represented by most gene trees, LGT events might occur to this gene family (Zhaxybayeva et al., 2006).

Methods based on reconciliation model

The reconciliation of gene tree and species tree is a typical tree-based method. Given a species tree and a gene tree, this kind of method uses a mathematical model to map a range of evolutionary events onto the gene tree in a way consistent with the species tree (Ravenhall et al., 2015). Besides LGTs, other possible evolutionary events considered in the model include gene duplication, gene loss, incomplete lineage sorting or homologous recombination. The reconciliation problem is NP-hard under most formulations (Bansal et al., 2013). Various models, algorithms, and programs have been developed to reconcile gene tree and species tree, which were comprehensively reviewed in (Doyon et al., 2011).

Generally speaking, there are two frameworks for reconciliation (Doyon et al., 2011). One is parsimony framework which seeks an optimal solution given the costs of considered evolutionary events. The other is probabilistic framework which aims to find a solution with maximum posterior probability or maximum likelihood. Other than differences in the framework, different methods may also differ in the kind of events considered. For example, the parsimony method PHYLTR (Tofigh et al., 2011) only accounts for gene duplications and LGTs, whereas the parsimony algorithm in (Stolzer et al., 2012) takes into account gene duplication, transfer, loss, and incomplete lineage sorting. The types of input gene trees or species trees that can be handled by different methods are also different. Most methods require binary trees as input. Only a few programs can deal with non-binary trees (Stolzer et al., 2012). Some programs require either rooted or unrooted gene trees, whereas others can accept both rooted and unrooted gene trees (Jacox et al., 2016).

Pros and Cons

The phylogenetic tree-based methods are generally considered to be more reliable in inferring LGTs (Gogarten and Townsend, 2005; Keeling and Palmer, 2008; Daubin and Szöllősi, 2016). They are not sensitive to amelioration and hence can detect very ancient transfers. The details of LGT events can be predicted, including the donor and recipient organisms, the direction of transfers, and the order of different transfers.

But tree-based methods are computationally expensive. Moreover, their predictions are still not definitive and suffer from diverse sources of uncertainties or errors. The accuracy of these methods strongly depends on several prerequisites, such as the selection of gene families, the reliability of species tree, and the inference of gene trees (Poptsova, 2009). For example, it is non-trivial to distinguish LGTs from potential artefacts generated in tree building. Besides, tree-based methods only consider genes with orthologs or single copy genes and thus may miss many laterally transferred genes (Poptsova, 2009; Wellner et al., 2007).

4.2.4 Summary

Owing to the difficulty in obtaining a reliable set of LGTs, it is hard to evaluate different kinds of methods. Despite that simulation can be used for validation, the simulated data may not reflect the true extent of LGTs (Ravenhall et al., 2015). Nevertheless, it is believed that available methods can still not provide a very precise picture of LGTs in the whole genome of one organism, suffering from false negatives and false positives to an unknown extent (Zhaxybayeva, 2009). For instance, they may easily detect some native genes as transferred, because these native genes may have atypical features to support their biological functions.

In general, each kind of methods recognizes LGTs of different features based on different criteria and assumptions (Lawrence and Ochman, 2002). For example, according to the benchmark work of several compositional methods on simulated data, the performance of each method varied with different characteristics of LGTs, including the origin, quantity, size, host genome, and recipient genome (Becq et al., 2010). Besides, different methods may detect LGTs at discrepant ages, with low overlap sometimes smaller than expected by chance, especially for methods from different categories

(Ragan, 2001; Ragan et al., 2006; Zhu et al., 2014).

As each method may predict only a subset of true LGT events, different methods are often complementary to each other. Taken altogether, they form a tool box for LGT detection. Thus, the utilization of multiple methods seems necessary in practice. It was suggested that applying multiple methods can yield better predictions of LGT (Lawrence and Ochman, 2002). For example, predicted LGTs from phylogenetic methods can be verified by compositional methods, and vice versa (Elhai et al., 2012; Sjöstrand et al., 2014).

However, it is still unclear how the different combination of predictions from diverse methods affect the accuracy of final results (Zhaxybayeva, 2009). There were few studies that systematically investigate how the predictions from a variety of methods complement with each other in practice. The analysis of LGTs detected by various methods on well-studied organisms may provide a good starting point. In consideration of this need, we conducted a case study on cyanobacteria, which will be presented in Section 4.3.

4.3 Case study

In this section, we present our case study on cyanobacterial genomes, which was aimed to investigate the complementary performances of different kinds of LGT detection methods on real biological datasets.

Cyanobacteria belong to a phylum of bacteria that obtain their energy through photosynthesis. They are the only known bacteria with the function of oxygenic photosynthesis. A large number of cyanobacterial genomes have been completely sequenced. With the available genomes, numerous studies have been done to analyze LGTs in cyanobacteria. Extensive LGTs among cyanobacteria were reported, especially between marine *Synechococcus* and *Prochlorococcus marinus* (Beiko et al., 2005; Zhaxybayeva et al., 2006, 2009; Abby et al., 2012). Hence, cyanobacteria serve as ideal organisms for a case study on performances of LGT detection methods.

After selecting a set of cyanobacterial genomes, we applied several representative compositional and phylogenetic methods to predict LGTs. We also included our new methods present in Chapter 2 and 3, which illustrates the application of these methods

in practice. Finally, we analyzed the results to find similarities and differences among the predictions from different methods.

4.3.1 Methods

In this subsection, we describe our methods to detect LGTs with different compositional and phylogenetic methods.

Given that there are numerous LGT detection methods, we only selected a few representatives in different categories based on their availability and known performances. The overall procedure and adopted tools are shown in Figure 4.2. The details are provided in the following subsections.

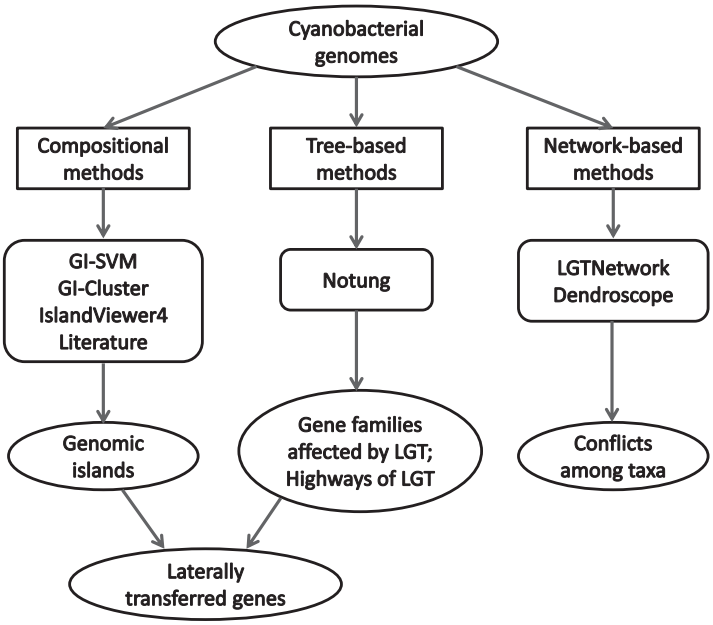

Figure 4.2: The pipeline of utilizing both compositional and phylogenetic methods to detect LGTs in selected cyanobacterial genomes.

Selection of cyanobacterial genomes

For phylogenetic analysis, we mainly used 40 cyanobacterial genomes, of which 37 genomes are included in database HOGENOM (Penel et al., 2009) and 3 genomes were studied in (Zhaxybayeva et al., 2006), which are shown in Table 4.1. The abbreviations for these genomes are the same as those in database or literature, respectively. We chose these genomes because there are known gene trees for gene families selected from them

in literature (Zhaxybayeva et al., 2006, 2009; Abby et al., 2012; Szöllősi et al., 2012, 2013a,b). Since it is difficult to build accurate gene trees from real data, we did not build gene trees from scratch but used the available gene trees which are supposed to be of high reliability.

For GI prediction, we selected genome *Synechococcus* sp. WH8102 (SYNPX) as a representative, since a high portion of genes in this genome were reported to be affected by LGT (Dufresne et al., 2008). We also predicted GIs on genome *Synechococcus* sp. CC9605 (SYNSC) because this genome was shown to have extensive gene exchanges with SYNPX.

LGT detection via compositional methods

Among various compositional methods, we mainly used GI prediction methods, including IslandViewer 4 and the two methods developed in Chapter 2 (GI-SVM and GI-Cluster). Given the DNA sequence of a selected genome, we applied GI-SVM and GI-Cluster to detect GI candidates. For GI-Cluster, we also took as input the gene predictions and coding sequences downloaded from new NCBI ftp site (`ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/`, June 2017). We downloaded GI predictions from IslandViewer 4 (June 2017).

We collected GIs predicted on the selected cyanobacterial genomes from literature as reference (Dufresne et al., 2008). These GIs were obtained via methods based on sequence composition, mobility evidence, and the presence of core gene blocks.

Given the GI predictions, we extracted genes within each GI, which are candidates of laterally transferred genes. The extracted genes were represented by their RefSeq IDs.

LGT detection via phylogenetic tree-based methods

Among various phylogenetic tree-based methods, we chose methods based on reconciliation model, since they are more realistic by considering reticulation events other than LGT when explaining the conflicts between gene tree and species tree. Among the available tools for gene tree-species tree reconciliation, Notung (`http://www.cs.cmu.edu/~durand/Notung/`) is a well-maintained package that includes sophisticated reconciliation algorithms. Different from other similar methods, Notung

Table 4.1: The 40 cyanobacterial genomes used in the case study.

Genome	**Abbreviation**
Synechococcus elongatus elongatus	SYELO1
Synechococcus elongatus PCC 6301	SYNP6
Synechococcus elongatus PCC 7942	SYNE7
Synechococcus sp. CC9311	SYNS3
Synechococcus sp. CC9605	SYNSC
Synechococcus sp. CC9902	SYNS9
Synechococcus sp. JA-2-3B'a(2-13)	SYNJA
Synechococcus sp. JA-3-3Ab	SYNJB
Synechococcus sp. PCC 7002	SYNP2
Synechococcus sp. RCC307	SYNR3
Synechococcus sp. WH 7803	SYNPW
Synechococcus sp. WH 8102	SYNPX
Synechocystis sp. PCC 6803	SYNY3
Prochlorococcus marinus str. AS9601	PROMS
Prochlorococcus marinus str. MIT 9211	PROM4
Prochlorococcus marinus str. MIT 9215	PROM2
Prochlorococcus marinus str. MIT 9301	PROM0
Prochlorococcus marinus str. MIT 9303	PROMM
Prochlorococcus marinus str. MIT 9312	PROM9
Prochlorococcus marinus str. MIT 9313	PROM3
Prochlorococcus marinus str. MIT 9515	PROM5
Prochlorococcus marinus str. NATl1a	PROM1
Prochlorococcus marinus str. NATl2a	PROMT
Prochlorococcus marinus subsp. marinus str. CCMP1375	PROMS
Prochlorococcus marinus subsp. pastoris str. CCMP1986	PROMP
Cyanothece sp. ATCC 51142	CYAA5
Cyanothece sp. PCC 7424	CYAP7
Cyanothece sp. PCC 7425	CYAP4
Cyanothece sp. PCC 8801	CYAP8
Nostoc punctiforme PCC 73102	NOSP7
Nostoc sp. PCC 7120	ANASP
Anabaena variabilis ATCC 29413	ANAVT
Trichodesmium erythraeum IMS101	TRIEI
Thermosynechococcus elongatus BP-1	THEEB
Microcystis aeruginosa NIES-843	MICAN
Acaryochloris marina MBIC11017	ACAM1
Gloeobacter violaceus PCC 7421	GLVIO1
Prochlorococcus marinus MED4	2Prochloro
Nostoc punctiforme ATCC 29133	Nostoc
Crocosphaera watsonii WH 8501	Crocosphae

The first 37 genomes are included in database HOGENOM, of which the first one is included in version 4 and the others are included in version 5. The last 3 genomes are from (Zhaxybayeva et al., 2006).

captures incomplete lineage sorting, which can reduce overestimation of other events (Stolzer et al., 2012). Moreover, Notung is easy to use and yields comprehensive output

regarding LGTs. Due to the disparities among methods based on reconciliation model, we only used Notung 2.9 to infer LGTs from different sets of gene trees and species trees, aiming to investigate the performance of the same tree-based method on slightly different input.

We applied Notung on seven sets of gene trees reported in literature (Table 4.2). Except for dataset T1809 and T1127, the gene families used for tree building were extracted from database HOGENOM. For unrooted gene trees, we used Notung to root them. Because some trees could not be rooted, slightly fewer gene trees were used in reconciliation analysis. The species tree for dataset T465, T469, T474, and T1099 was from (Szöllősi et al., 2012). The species trees for the other three datasets are from the same sources as the gene trees.

Table 4.2: The seven sets of gene trees used in this case study.

Dataset	#Taxa	#Original trees	Original root	#Rooted trees	Reference
T465	36	474	unrooted	465	(Szöllősi et al., 2012)
T469	36	474	unrooted	469	(Szöllősi et al., 2012)
T473	36	473	rooted	473	(Szöllősi et al., 2013b)
T1099	36	1099	rooted	1099	(Szöllősi et al., 2013a)
T1809	18	1812	unrooted	1809	(Zhaxybayeva et al., 2009)
T977	14	977	unrooted	977	(Abby et al., 2012)
T1127	11	1128	unrooted	1127	(Zhaxybayeva et al., 2006)

The genomes represented by these trees are all included in Table 4.1. T465, T469, T473, and T1099 are for the same 36 genomes from HOGENOM version 5. All the 14 genomes represented by trees in T977 are included in HOGENOM, except that one of them is only included in version 4. For the 18 genomes represented by trees in T1809, one of them is 2Prochloro and the others are included in HOGENOM version 5. For the 11 genomes represented by trees T1127, 8 of them are included in HOGENOM version 5. There were two datasets of unrooted trees reported in (Szöllősi et al., 2012). One dataset, denoted by T465, includes gene trees inferred by PhyML. The other dataset, denoted by T469, includes gene trees inferred by Treefinder.

To find laterally transferred genes in a genome, predictions of Notung on only five datasets were used, because it is hard to find the corresponding gene families from dataset T1809 and T1127. From these predictions, we extracted gene families with incoming transfers to the genome. Then we obtained their RefSeq IDs by first mapping HOGENOM ID to UniProt ID and then utilizing the ID mapping service provided by UnitProt (The UniProt Consortium, 2017).

For reference, we also included the sets of 323 core genes and 359 shell genes (laterally transferred genes) predicted by a tree-based method on 682 gene families from 13 cyanobacterial genomes (Shi and Falkowski, 2008). Of the 13 genomes, 12 of them

are included in Table 4.1 and the last one is *Prochlorococcus marinus* SS120. We found RefSeq IDs for these genes via previously reported locus. Due to annotation update, some of these genes have been removed. As a result, 319 core genes and 358 shell genes were obtained.

LGT detection via network-based methods

There are very few available programs for building a rooted phylogenetic network from a large set of gene trees. We tried the only algorithm for building a (restricted) LGT network developed by Cardona et al. (2015). To show the conflicts among gene trees, we used Dendroscope (Huson and Scornavacca, 2012) to build several kinds of rooted phylogenetic networks that are easy to compute, including cluster network, galled network, and level-k network. These networks cannot model LGT explicitly, but they show alternative placements of taxa represented by different gene trees.

Because these methods require as input rooted gene trees for single-copy gene families, we only used the 473 rooted gene trees (dataset T473) from (Szöllősi et al., 2013b) for network reconstruction.

As a validation step, we checked whether some well-known clades are represented by the reconstructed networks. Previous results show that high-light adapted *Prochlorococcus* spp. are a monophyletic clade and low-light adapted *Prochlorococcus* spp. are a paraphyletic group (Zhaxybayeva et al., 2009). Thus, among the 36 cyanobacterial genomes represented by the input trees, we checked whether the high-light adapted *Prochlorococcus* spp. (including PROMS, PROM2, PROM0, PROM9, PROM5, and PROMP) and low-light adapted *Prochlorococcus* spp. (including PROM4, PROMM, PROM3, PROM1, PROMT, and PROMS) are soft clusters displayed in the reconstructed networks. This was performed by the CCP program developed in Section 3.6.

Although all the input clusters are displayed in the cluster network, galled network, and level-k network, some input trees may not be displayed. To get a basic idea of how the input trees are represented by these networks, we checked whether each input tree is displayed in each of the three networks, respectively. This was performed by the TCP program developed in Section 3.5.

4.3.2 Results

In this subsection, we report the results from three aspects: the overlap of predictions from different methods; the highways of LGT and their correlation to GIs; the properties of reconstructed phylogenetic networks.

The overlap of predictions from different methods

We computed putative laterally transferred genes predicted by different methods for genome SYNPX. Then we measured the overlap of predictions from methods in the same category, from the same method on different input, and from methods across different categories. To quantify the agreements among different methods, the overlap factor (OF) between two datasets was also calculated as described in (Zhu et al., 2014). The larger OF means that the two sets of predictions are more consistent and thus the two methods are more likely to predict the same sets of laterally transferred genes.

Generally speaking, the predictions from GI detection methods were more consistent, as indicated by the relatively high overlap in Figure 4.3 and large OFs in Table 4.3 . This is probably because that these methods rely on similar assumptions on GI-related features, such as atypical composition and evidence of specific functions. However, 188 laterally transferred genes reported in literature were still not predicted by the other four methods. GI-Cluster also yielded 281 uniquely predicted alien genes.

Surprisingly, extremely low overlap occurs among the predictions from the same tree-based method (Notung) on different sets of input trees (Figure 4.4 and Table 4.3). Except for predictions on dataset T465 and T469, which had more overlaps (39 genes) between each other, the predictions on the other datasets had very few overlaps. One possible reason is that tree-based methods usually generate only a few predictions. For example, the numbers of laterally transferred genes predicted on dataset T1099, T977, T473, T469, and T465 were 11, 41, 3, 58, and 62, respectively. However, the large discrepancies among results from the same method on different input trees built for similar genomes suggest that tree-based methods are easily affected by multiple factors such as taxa sampling and tree topology.

Across different method categories, the overlap was also very low (Figure 4.5 and Table 4.3). In other words, the alien genes predicted by compositional methods and

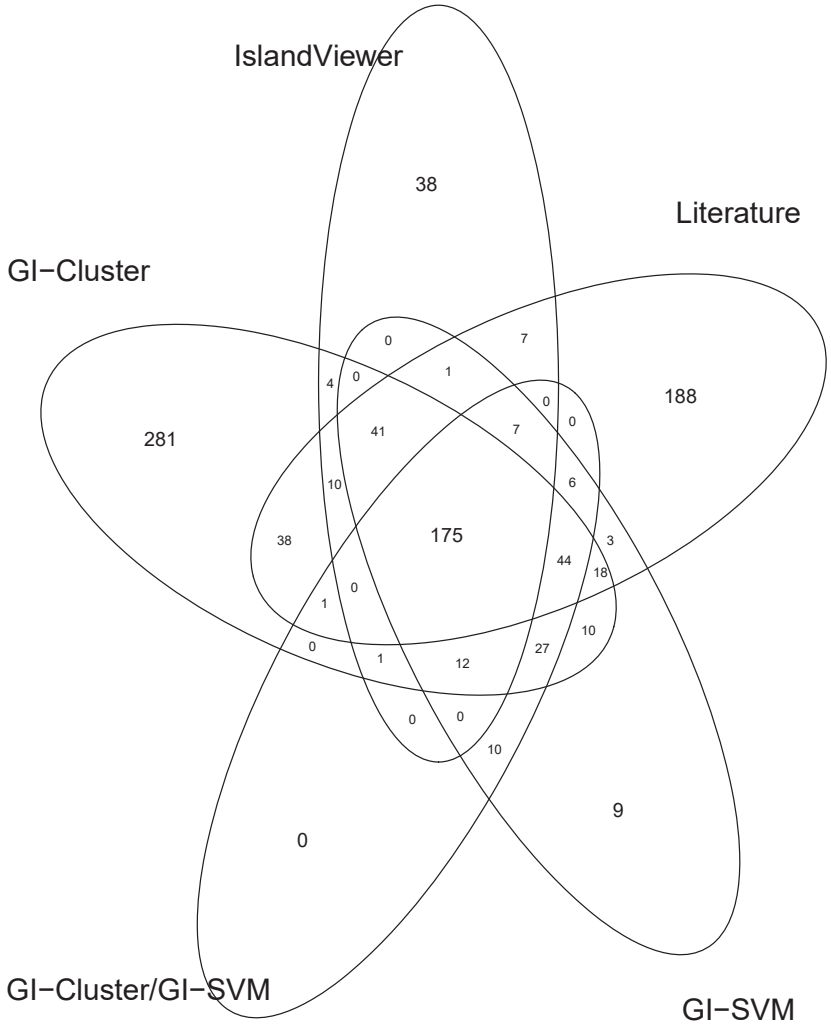

Figure 4.3: The overlap of predicted laterally transferred genes by different GI prediction methods. GI-Cluster/GI-SVM represents the predictions of GI-Cluster based on the output of GI-SVM. Literature represents GIs collected from literature.

tree-based methods were mostly not overlapping. This is not unexpected given the large discrepancies between the two kinds of methods, which is also consistent with previous results on *Escherichia coli* K12 MG1655 genome (Ragan, 2001; Ragan et al., 2006) and *Rickettsia felis* genome (Zhu et al., 2014). Interestingly, around a quarter of the core genes were predicted by other methods as laterally transferred, and a majority of the shell genes were not predicted by other methods. This implies that the sets of core and shell genes are largely determined by the adopted methods and conclusions derived from these genes should be treated with caution.

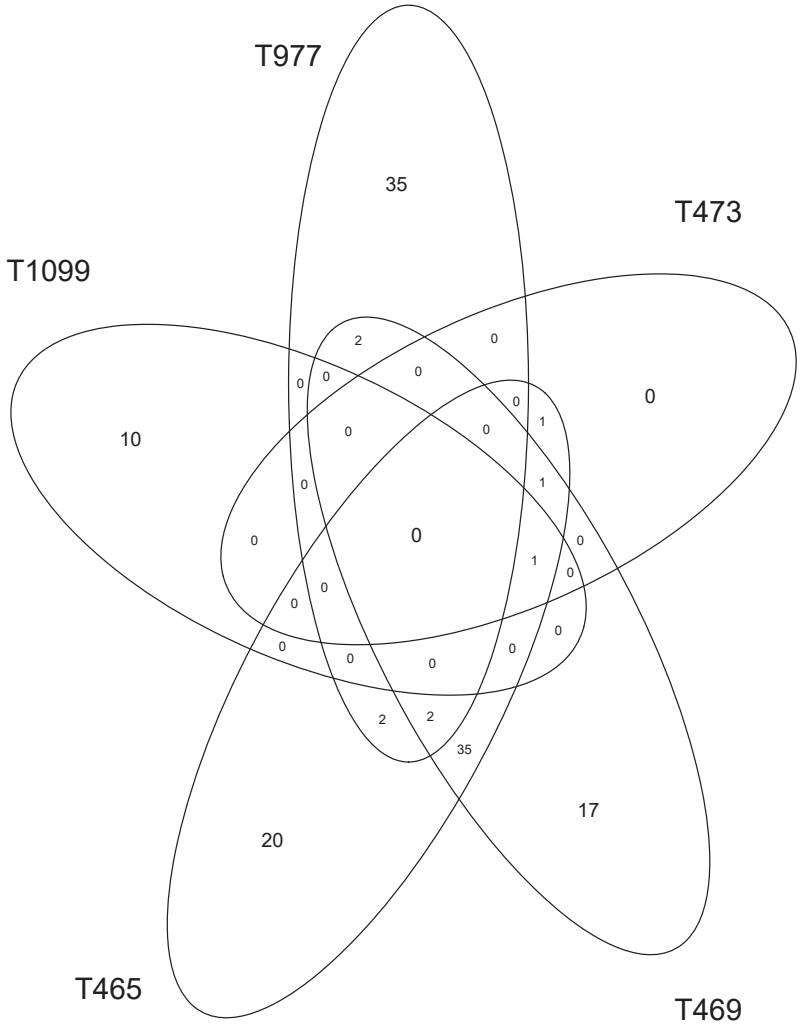

Figure 4.4: The overlap of predicted laterally transferred genes by Notung from different sets of gene trees and species trees.

Table 4.3: Agreement among different LGT detection methods measured by overlap factor (OF).

GI prediction methods			Tree-based methods			Methods of different kinds		
Dataset 1	**Dataset 2**	**OF**	**Dataset 1**	**Dataset 2**	**OF**	**Dataset 1**	**Dataset 2**	**OF**
IslandViewer	GI-Cluster	237.64	T1099	T977	0	GI-Cluster	Literature	191.62
IslandViewer	GI-SVM	404.25	T1099	T473	4.34	GI-Cluster	T465	4.53
IslandViewer	GI-Cluster/GI-SVM	326.61	T1099	T469	1.60	GI-Cluster	shell	16.35
IslandViewer	Literature	291.59	T1099	T465	1.55	GI-Cluster	core	13.84
GI-Cluster	GI-SVM	403.58	T977	T473	0	Literature	T465	0
GI-Cluster	GI-Cluster/GI-SVM	319.19	T977	T469	4.46	Literature	shell	51.03
GI-Cluster	Literature	191.62	T977	T465	4.25	Literature	core	71.07
GI-SVM	GI-Cluster/GI-SVM	674.93	T473	T469	6.48	T465	shell	18.33
GI-SVM	Literature	373.06	T473	T465	11.15	T465	core	22.71
Literature	GI-Cluster/GI-SVM	285.00	T469	T465	125.45	core	shell	0

OF equals 0 when there are no overlapping genes between two datasets.

Highways of gene sharing

A highway of gene sharing between two taxa represents that many individual LGT events occurred between these two taxa (Beiko et al., 2005). Highways may represent

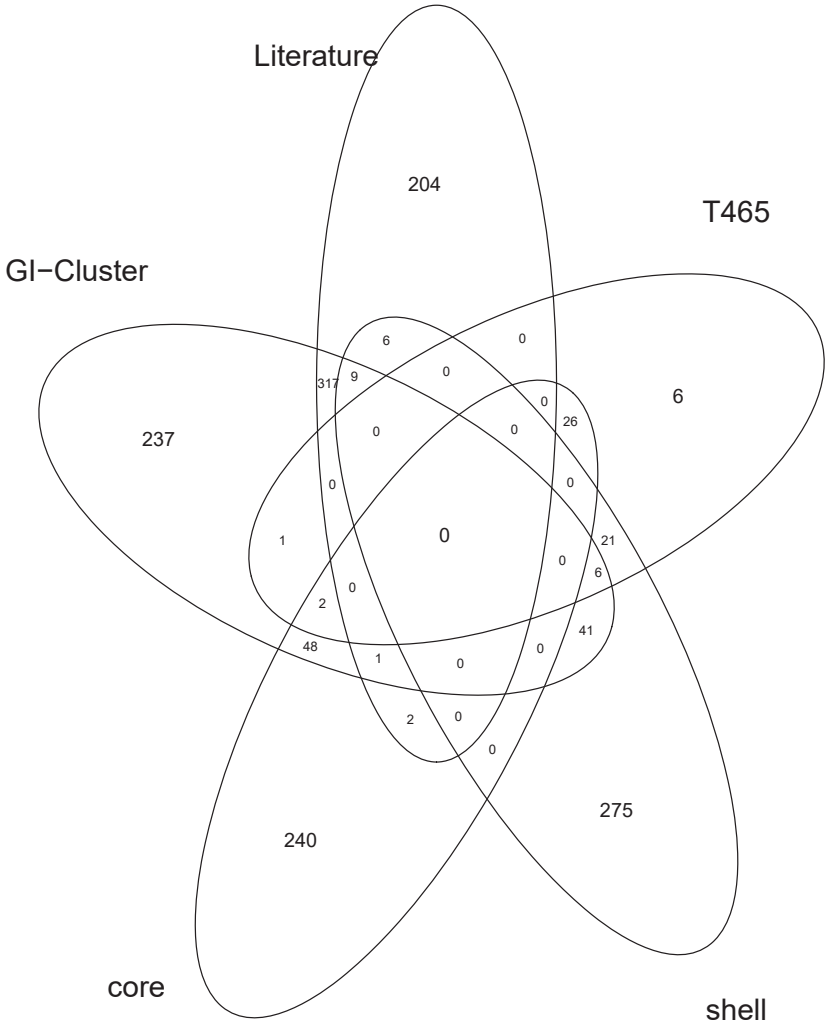

Figure 4.5: The overlap of predicted laterally transferred genes by methods across different categories (GI detection methods and phylogenetic tree-based methods). Here, "core" and "shell" represent the set of core genes and shell genes (laterally transferred genes) predicted in (Shi and Falkowski, 2008), respectively. For predictions from Notung, only those on dataset T465 are included in this plot because there are too few predictions on other datasets.

major evolutionary events and have important biological implications (Bansal et al., 2011). Here, we say a path exists between two taxa in a dataset if LGTs occurred between these two taxa in either direction, and a highway is then roughly defined as a path on which the number of LGTs is relatively large in the dataset. For each dataset in Table 4.2, we computed the top highway and counted the numbers of LGTs involved in the highways. Particularly, we computed the highways involving SYNPX as well as the numbers of outgoing or incoming LGTs for SYNPX. Finally, we investigated the relationship between highways and GIs.

In general, extensive gene flow was detected between SYNPX and other cyanobacteria. SYNPX was involved in the top first highway for five datasets (T465, T469, T977, T1127, and T1809). For dataset T473 and T1099, SYNPX was not involved in the top 20 highways. This is probably because that gene trees in these two datasets were built by significantly different methods (Szöllősi et al., 2013b,a).

The predicted frequent gene exchange parters of SYNPX in highways were also consistent on most datasets, in spite of numerical differences in the predicted LGTs (Table 4.4). The most frequent gene exchange parter of SYNPX was detected to be SYNSC on five datasets. On dataset T473, SYNSC was the second most frequent gene exchange parter of SYNPX. On dataset T1127, the most frequent partner of SYNPX was PROM3, which was probably because that SYNSC was not included in the sampled taxa. Additionally, SYNPX transferred more genes to others than genes it received, except for dataset T1127.

Table 4.4: The numbers of LGTs involving SYNPX and the common gene exchange parters of SYNPX in highways, detected by Notung on different datasets.

	T465	**T469**	**T473**	**T1099**	**T1809**	**T977**	**T1127**
#LGT_in	74	66	3	9	132	44	520
#LGT_out	165	175	14	87	533	359	95
Top 1 partner	SYNSC	SYNSC	SYNPW	SYNSC	SYNSC	SYNSC	PROM3
Top 2 partner	SYNS9	SYNS9	SYNSC,SYNR3	SYNS9	GLVIO1_SYNJA(B)	SYNS9	PROMS

LGT_in represents incoming LGTs to SYNPX, and LGT_out represents outgoing LGTs from SYNPX. GLVIO1_SYNJA(B) represents the ancestor of GLVIO1, SYNJA, and SYNJB.

However, the numbers of LGTs in highways predicted on different datasets had a wide range, from 6 LGTs in the top first highway on dataset T473 to 403 LGTs in the top first highway on dataset T1127. These differences are probably caused by taxa sampling and the inference of gene trees. More LGTs were likely to be detected among certain set of taxa. For instance, 309 LGTs were involved in the top first highway predicted on dataset T1809, which are for 18 genomes from marine *Synechococcus* and *Prochlorococcus marinus*. The methods of inferring gene trees also affect the results a lot. For example, gene trees in dataset T473, T469, and T465 were built for almost the same set of gene families in the same genomes, but much less LGTs were detected on dataset T473 whose gene trees were built by a method that takes into account the unrepresented species.

On one hand, a large number of genes were exchanged between two taxa on highways.

On the other hand, GIs are reservoirs of laterally transferred genes. Therefore, it seems interesting to understand the relationship of genes inside GIs with genes involved in highways. To study this relationship, we computed the overlap between genes within GIs and genes participated in highways for genome SYNSC, which was predicted to receive many genes from SYNPX. We obtained GI candidates on genome SYNSC from three sources (predictions in literature, predictions from GI-SVM and GI-Cluster), and computed the top first highway from SYNPX to SYNSC on five datasets (T469, T465, T473, T977, and T1099) with Notung. For illustration purposes, we only selected predictions from Notung on two datasets (T469 and T977) which had more genes involved in highways. As shown in Figure 4.6, there was no overlap between genes involved in highways and genes in GIs, expect one gene predicted by both GI-Cluster and Notung on dataset T977. This extremely low overlap is consistent with previous results that highways predicted by phylogenetic methods cannot be detected by compositional methods (Sjöstrand et al., 2014).

Phylogenetic networks showing conflicts among taxa

The power of network-based methods in inferring LGTs is still very limited. Here, we mainly show conflicts among taxa that were detected by network-based methods and examine the consistencies of these conflicts with predictions from other sources.

When applying the program for reconstructing a restricted LGT network on dataset T473, we only found three subtrees with three taxa (CYAP4, THEEB, ACAM1) on which the program can be applied. This is not surprising in consideration of the strict restrictions of this program on input. When using all clusters appearing in more than 20% of the 473 input gene trees, the cluster network, galled network, and level-k network built by Dendroscope all showed only one reticulation which was also among these three taxa. Notung also reported tens of LGTs among these three taxa on dataset T465 and T469, despite that it reported no LGTs among these taxa on dataset T473. Hence, there might be gene exchanges among these species, which is worth further analysis.

In Figure 4.7, we show the cluster network, galled network, and level-k network built by Dendroscope for all clusters appearing in more than 10% of the 473 input gene trees. Since these networks have 36 leaves and only a few reticulate nodes, we used Dendroscope to quickly compute distances between each pair of networks. The

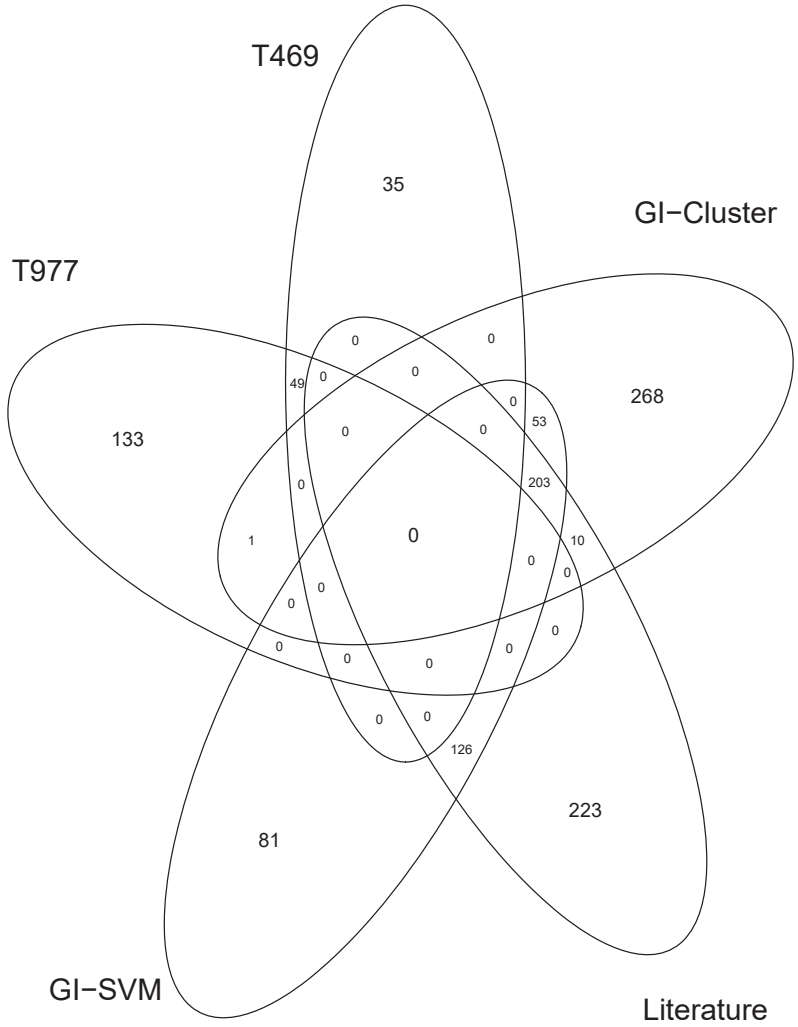

Figure 4.6: The overlap of genes within predicted GIs by different programs on SYNSC and genes involved in the top first highway from SYNPX to SYNSC predicted by Notung.

SRF and RF distance between the cluster network and the galled network are 0.5 and 1, respectively. The SRF and RF distance between the galled network and the level-k network are 0 and 1, respectively. The SRF and RF distance between the cluster network and the level-k network are 0.5 and 2, respectively. Because of the few topological differences among these three networks, it is reasonable that the SRF distances and RF distances are small. As galled network and level-k network are built to represent the same set of soft clusters, their SRF distance is 0 despite their network topologies are slightly different.

There are mainly three groups of taxa with conflicts (reticulations) in the three networks (Figure 4.7). Except one group containing ACAM1, CYAP4, and THEEB, the other two groups are: (a) the clade containing MICAN, CYAP7, CYAA5, and CYAP8;

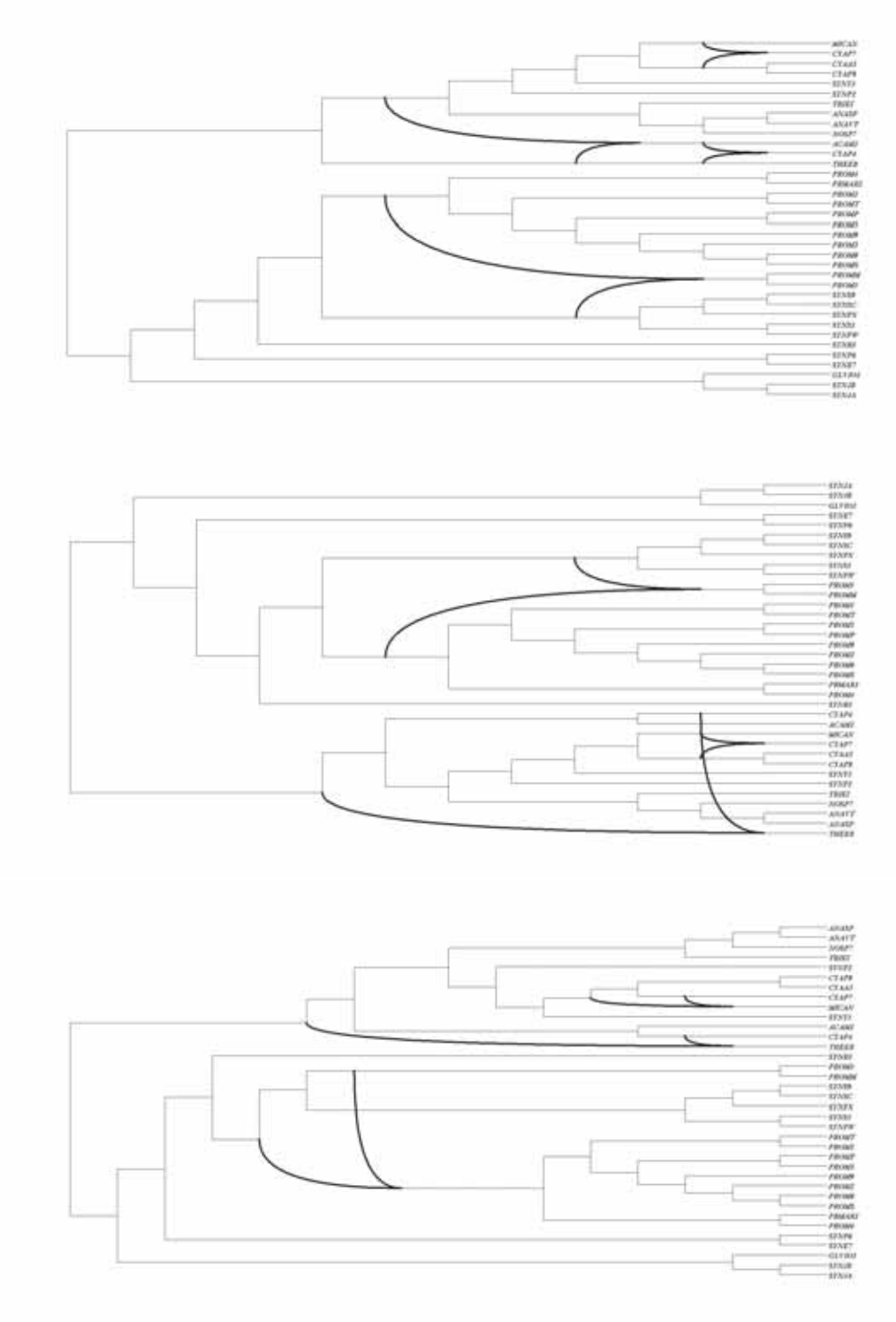

Figure 4.7: The cluster network (top), the galled network (middle), and the level-k network (bottom), built by Dendroscope from 473 rooted gene trees for 36 cyanobacterial genomes.

(b) the clade containing *Prochlorococcus* spp. and marine *Synechococcus*. For these two groups, there were also tens of LGTs detected by Notung on dataset T473, T465, and T469, respectively. For species in group (b), frequent LGTs among them were also

reported in (Zhaxybayeva et al., 2006, 2009). Therefore, network-based methods can effectively detect conflicts that are probably caused by LGT.

Consistent with previous results, the high-light adapted *Prochlorococcus* spp. are a soft cluster of some node in each of the three networks in Figure 4.7, whereas the low-light adapted *Prochlorococcus* spp. are not soft clusters of any node in the three networks.

It is interesting that the cluster network, galled network, and level-k network display the same set of 53 gene trees. This is probably because that these 53 gene trees can represent all the input clusters. The fact that a majority of gene trees are not displayed in these networks suggests that these networks may fail to represent the evolutionary history of a large number of gene families.

4.4 Summary

In chapter 4, we focus on the general detection of LGT events. The accurate estimates of LGT can provide insights on the impact of LGT and deepen our understanding on evolution of life. Unfortunately, like other problems in computational evolutionary biology which infer past events from the extant data, the clues left for LGT detection were dimmed and perplexed owing to various evolutionary and stochastic factors. In spite of the challenges, numerous methods have been proposed to unravel the extent of LGT, including GI detection methods discussed in Chapter 2 and network-based methods discussed in Chapter 3. However, current methods often give discrepant predictions, especially for methods in different categories. Hence, most of these methods are believed to be complementary to each other. But the agreements among predictions from multiple methods in practice have not been fully investigated.

In view of this gap, we conducted a case study by analyzing the predictions of multiple LGT detection methods on a set of cyanobacterial genomes. Our results suggest that different kinds of methods generated diverse predictions with low agreement, which is consistent with previous results in literature. For compositional methods, their predictions had relatively higher overlap in spite of certain inconsistencies. For phylogenetic tree-based methods, the predictions from the same program applied on different sets of gene and species trees for similar genomes yielded extremely low

overlap despite broad agreements on general outcomes, suggesting that this kind of method is easily affected by the choice of taxa and the building of input trees. Although highways of LGT detected by tree-based methods involved tens to hundreds of genes, most of these genes did not overlap with genes in predicted GIs, implying the disparities between the two kinds of methodology. Network-based methods could show several LGT-related conflicts among taxa, but much work is still to be done to model LGTs explicitly with networks.

To sum up, the results from this case study indicate the huge limitations of a single kind of LGT detection methods in practice. To get a more comprehensive picture of LGTs in a genome or several genomes, it is indeed necessary to utilize multiple methods. Despite that the great disparities among different kinds of methods make it hard to reconcile the differences, predictions obtained from multiple methods should be carefully examined before reliable conclusions can be reached, whenever possible. For phylogenetic methods, taxa sampling and tree inferences should also be prudently performed to avoid potential bias.

Chapter 5

Conclusion

5.1 Summary

LGT is an important evolutionary process. It can play a significant role in genome innovation and biological diversity, particularly for prokaryotes. LGT is also related to the spread of antibiotic resistance and pathogenicity. However, debates regarding the impact and extent of LGT still exist. To further understand the role of LGT, it is critical to quantify the prevalence of LGT. In this thesis, we tried to approach three problems arising in the quantitative characterization of LGT.

The first problem is about how to detect GIs in a genome. We address this problem in Chapter 2. GI is an important tool of LGT and genome evolution for bacteria. GIs often contain genes involved in important adaptive functions. Therefore, GI prediction in bacterial genomes is of great importance. The rapid increase of newly sequenced genomes requires better GI detection methods to quickly locate important regions for further analysis. In view of this need, we developed two machine learning methods to detect GIs in a single genome: GI-SVM and GI-Cluster. GI-SVM is designed for first-pass GI predictions without relying on annotations. GI-Cluster is designed to detect GIs by integrating multiple GI-related features, relying on pre-built databases to obtain annotations. According to the evaluations on real biological datasets, GI-SVM is highly sensitive and GI-Cluster can achieve a good balance of recall and precision. Additionally, GI-Cluster can further refine the predictions from GI-SVM or other sensitive programs. In short, GI-SVM and GI-Cluster provide researchers two alternative tools for better GI detection.

The second problem is how to model LGT with rooted phylogenetic networks. We address this problem in Chapter 3. Tree-based network, especially LGT network, can represent LGTs in a natural way. But it is quite challenging to build such rooted phylogenetic networks from biological data. So, we mainly aimed to solve two fundamental problems arising in the reconstruction and verification of phylogenetic networks: the TCP and CCP. Even the TCP and CCP are NP-complete in general. By utilizing a powerful decomposition technique, we implemented fast programs for solving them on arbitrary phylogenetic networks. The resulting CCP program is further extended into a program for quickly computing the SRF distance between phylogenetic networks. The time complexities of these programs are exponential due to their inherent difficulties, but they were shown to be fast enough to solve real biological instances in evolutionary studies. These programs may help to develop more practical programs for building and comparing phylogenetic networks in general, and hence promote the reconstruction of LGT networks.

The third problem is how to infer LGT events. This problem is more general than the first two problems, and actually the solutions to the first two problems may help to solve this third problem. We address this problem in Chapter 4. Based on different criteria, numerous methods have been developed to detect LGTs. These methods exhibit a wide range of diversity, and they are suggested to be used together to provide better predictions. However, very few studies systematically investigated the complementary performances of various methods in practice. To further understand how individual methods complement with each other, we performed a case study on cyanobacterial genomes which have been extensively studied in terms of LGT. We applied both compositional methods and phylogenetic methods to the selected genomes, including the methods for GI detection and network reconstruction that are discussed in Chapter 2 and 3, respectively. Consistent with previous outcomes, our results reveal very low overlap among predictions of different methods, particularly methods from different categories. Our study also shows extremely low overlap among predictions of the same method on different input. Therefore, for accurate LGT detection in practice, it is necessary to carefully apply multiple methods and treat conclusions with caution.

5.2 Future work

Our work on the three problems related to LGT analysis can be further improved or extended to other applications. In this section, we briefly discuss the future work for each problem, respectively.

Several work can be done with regard to GI detection. Firstly, our new methods for GI detection can be further enhanced. For instance, more intelligent spectrum kernels may improve GI-SVM by making full advantage of informative k-mers of fixed-order or variable orders. It also seems promising to improve the methods to find possible donors of GIs. Secondly, our programs may be applied to metagenomics datasets to find potentially interesting discoveries. Currently we only tested GI-SVM and GI-Cluster on genome sequences. Since huge amounts of metagenomics dataset have been available, it seem possible to investigate whether there are extensive gene exchanges among environmental bacteria from these data. Lastly, the methodology used in developing GI-SVM and GI-Cluster may also be extended to other problems. For example, one-classed SVM based on k-mer frequencies may be used for detecting outliers in different contexts, such as identifying contamination in next-generation sequencing data. In addition, the consensus clustering approach may also be adapted in other scenarios that require effective integrations of heterogeneous features.

For our work on phylogenetic networks, our current programs for phylogenetic networks still have relatively limited applications. To make phylogenetic networks more useful in practice, more efficient algorithms have to be developed. For example, the time complexity of algorithms for solving TCP and CCP can be further reduced. We are trying to improve the CCP algorithm by resolving the invisible tree components in a more smart way. This new CCP algorithm will then help to improve the efficiency of our SRF program. The improvement of the TCP algorithm may help to develop new methods for reconstructing a tree-based network from multiple gene trees. Moreover, the TCP and CCP may help to solve other problems in a more general context. The most related context is when it is required to check whether a tree is represented by a graph, or whether a set of leaves is represented by some node in a graph. The ideas adopted in solving the TCP and CCP may also be helpful for other similar problems.

Finally, regarding our case study, only a small set of genomes and LGT detection

methods were included. The workflow applied in this case study could also be used in investigating more genomes and methods to get more general conclusions. In addition, it may be helpful to collect a relatively reliable set of LGTs at different time scales for cyanobacteria which have a relative abundance of fossil record (Szöllősi et al., 2012). These reliable LGTs may provide a benchmark set for evaluating LGT detection methods in future. The evaluations may in turn provide insights on the development of better methods for LGT detection.

Bibliography

Abby SS, Tannier E, Gouy M, et al. (2010). *Detecting lateral gene transfers by statistical reconciliation of phylogenetic forests*. *BMC Bioinformatics*, 11:324.

Abby SS, Tannier E, Gouy M, et al. (2012). *Lateral gene transfer as a support for the tree of life*. *Proc. Natl. Acad. Sci. U.S.A.*, 109(13):4962–4967.

Achaz G, Boyer F, Rocha EP, et al. (2007). *Repseek, a tool to retrieve approximate repeats from large dna sequences*. *Bioinformatics*, 23(1):119–121.

Albrecht B (2015). *Computing all hybridization networks for multiple binary phylogenetic input trees*. *BMC Bioinformatics*, 16:236.

Altschul SF, Madden TL, Schäffer AA, et al. (1997). *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs*. *Nucleic Acids Res.*, 25(17):3389–3402.

Arvey AJ, Azad RK, Raval A, et al. (2009). *Detection of genomic islands via segmental genome heterogeneity*. *Nucleic Acids Res.*, 37(16):5255–5266.

Asano T, Jansson J, Sadakane K, et al. (2010). *Faster computation of the Robinson-Foulds distance between phylogenetic networks*. In *CPM*, pages 190–201.

Avery OT, MacLeod CM, and McCarty M (1944). *Studies on the chemical nature of the substance inducing transformation of pneumococcal types induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type III*. *J. Exp. Med.*, 79(2):137–158.

Awadalla P (2003). *The evolutionary genomics of pathogen recombination*. *Nat. Rev. Genet.*, 4(1):50–60.

Azad RK and Lawrence JG (2011). *Towards more robust methods of alien gene detection*. *Nucleic Acids Res.*, 39(9):e56–e56.

Bansal MS, Banay G, Gogarten JP, et al. (2011). *Detecting highways of horizontal gene transfer*. *J. Comp. Biol.*, 18(9):1087–1114.

Bansal MS, Banay G, Harlow TJ, et al. (2013). *Systematic inference of highways of horizontal gene transfer in prokaryotes*. *Bioinformatics*, 29(5):571–579.

Bapteste E, van Iersel L, Janke A, et al. (2013). *Networks: expanding evolutionary thinking*. *Trends Genet.*, 29(8):439–441.

Becq J, Churlaud C, and Deschavanne P (2010). *A benchmark of parametric methods for horizontal transfers detection*. *PLOS ONE*, 5(4):e9989.

Beiko RG and Hamilton N (2006). *Phylogenetic identification of lateral genetic transfer events*. *BMC Evol. Biol.*, 6(1):15.

Beiko RG, Harlow TJ, and Ragan MA (2005). *Highways of gene sharing in prokaryotes*. *Proc. Natl. Acad. Sci. U.S.A.*, 102(40):14332–14337.

Bellanger X, Payot S, Leblond-Bourget N, et al. (2014). *Conjugative and mobilizable genomic islands in bacteria: Evolution and diversity*. *FEMS Microbiol. Rev.*, 38:720–760.

Ben-Hur A, Ong CS, Sonnenburg S, et al. (2008). *Support vector machines and kernels for computational biology*. *PLOS Comput. Biol.*, 4(10):e1000173.

Benveniste RE and Todaro GJ (1974). *Evolution of c-type viral genes: inheritance of exogenously acquired viral genes*. *Nature*, 252:456–459.

Bertelli C, Laird MR, Williams KP, et al. (2017). *Islandviewer 4: expanded prediction of genomic islands for larger-scale datasets*. *Nucleic Acids Res.*, 45(W1):W30–W35.

Bi D, Xu Z, Harrison EM, et al. (2012). *ICEberg: a web-based resource for integrative and conjugative elements found in Bacteria*. *Nucleic Acids Res.*, 40(D1):D621–D626.

Boc A, Philippe H, and Makarenkov V (2010). *Inferring and validating horizontal gene transfer events using bipartition dissimilarity*. *Syst. Biol.*, 59(2):195–211.

Boyd EF, Almagro-Moreno S, and Parent Ma (2009). *Genomic islands are dynamic, ancient integrative elements in bacterial evolution. Trends Microbiol.*, 17(2):47–53.

Brzuszkiewicz E, Thürmer A, Schuldes J, et al. (2011). *Genome sequence analyses of two isolates from the recent Escherichia coli outbreak in Germany reveal the emergence of a new pathotype: Entero-Aggregative-Haemorrhagic Escherichia coli (EAHEC). Arch. Microbiol.*, 193(12):883–891.

Cardona G, Llabrés M, Rosselló F, et al. (2008a). *A distance metric for a class of tree-sibling phylogenetic networks*. *Bioinformatics*, 24(13):1481–1488.

Cardona G, Llabrés M, Rosselló F, et al. (2009a). *Metrics for phylogenetic networks I: Generalizations of the Robinson-Foulds metric*. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 6(1):46–61.

Cardona G, Llabres M, Rossello F, et al. (2009b). *Metrics for phylogenetic networks II: Nodal and triplets metrics*. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 6(3):454–469.

Cardona G, Pons JC, and Rosselló F (2015). *A reconstruction problem for a class of phylogenetic networks with lateral gene transfers*. *Algorithms Mol. Biol.*, 10:28.

Cardona G, Rosselló F, and Valiente G (2008b). *A perl package and an alignment tool for phylogenetic networks*. *BMC Bioinformatics*, 9:175.

Cardona G, Rossello F, and Valiente G (2009c). *Comparison of tree-child phylogenetic networks*. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 6(4):552–569.

Cerdeño-Tárraga A, Efstratiou A, Dover L, et al. (2003). *The complete genome sequence and analysis of Corynebacterium diphtheriae NCTC13129*. *Nucleic Acids Res.*, 31(22):6516–6523.

Charlton ND, Carbone I, Tavantzis SM, et al. (2008). *Phylogenetic relatedness of the M2 double-stranded RNA in Rhizoctonia fungi*. *Mycologia*, 100(4):555–564.

Chatterjee R, Chaudhuri K, and Chaudhuri P (2008). *On detection and assessment of statistical significance of Genomic Islands*. *BMC Genomics*, 9:150.

Che D, Hasan MS, and Chen B (2014a). *Identifying pathogenicity islands in bacterial pathogenomics using computational approaches*. *Pathogens*, 3(1):36–56.

Che D, Hasan MS, Wang H, et al. (2011). *EGID: an ensemble algorithm for improved genomic island detection in genomic sequences*. *Bioinformation*, 7(6):311–314.

Che D, Hockenbury C, Marmelstein R, et al. (2010). *Classification of genomic islands using decision trees and their ensemble algorithms*. *BMC Genomics*, 11(Suppl 2):S1.

Che D, Wang H, Fazekas J, et al. (2014b). *An accurate genomic island prediction method for sequenced bacterial and archaeal genomes*. *J. Proteomics Bioinform.*, 7(8):214.

Chen L, Zheng D, Liu B, et al. (2016). *VFDB 2016: hierarchical and refined dataset for big data analysisâĂŤ10 years on*. *Nucleic Acids Res.*, 44(D1):D694–D697.

Chun J, Grim CJ, Hasan NA, et al. (2009). *Comparative genomics reveals mechanism for short-term and long-term clonal transitions in pandemic vibrio cholerae*. *Proc. Natl. Acad. Sci. U.S.A.*, 106(36):15442–15447.

Clarke GP, Beiko RG, Ragan MA, et al. (2002). *Inferring genome trees by using a filter to eliminate phylogenetically discordant sequences and a distance matrix based on mean normalized blastp scores*. *J. Bacteriol.*, 184(8):2072–2080.

Cohen O, Gophna U, and Pupko T (2011). *The complexity hypothesis revisited: connectivity rather than function constitutes a barrier to horizontal gene transfer*. *Mol. Biol. Evol.*, 28(4):1481–1489.

Cong Y, Chan Yb, and Ragan MA (2016). *A novel alignment-free method for detection of lateral genetic transfer based on TF-IDF*. *Sci. Rep.*, 6:30308.

Cortez D, Delaye L, Lazcano A, et al. (2009). *Composition-based methods to identify horizontal gene transfer*. In *Horizontal Gene Transfer: Genomes in Flux*, pages 215–225. Humana Press.

Dagan T and Martin W (2007). *Ancestral genome sizes specify the minimum rate of lateral gene transfer during prokaryote evolution*. *Proc. Natl. Acad. Sci. U.S.A.*, 104(3):870–875.

Dalevi D, Dubhashi D, and Hermansson M (2006). *Bayesian classifiers for detecting HGT using fixed and variable order markov models of genomic signatures*. *Bioinformatics*, 22(5):517–522.

Darling ACE, Mau B, Blattner FR, et al. (2004). *Mauve: multiple alignment of conserved genomic sequence with rearrangements. Genome Res.*, 14(7):1394–403.

Daubin V and Szöllősi GJ (2016). *Horizontal gene transfer and the history of life*. *Cold Spring Harb. Perspect. Biol.*, 8(4):a018036.

Dessimoz C, Margadant D, and Gonnet GH (2008). *DLIGHT–lateral gene transfer detection using pairwise evolutionary distances in a statistical framework*. In *RECOMB*, pages 315–330.

Dhillon BK, Chiu TA, Laird MR, et al. (2013). *IslandViewer update: improved genomic island discovery and visualization*. *Nucleic Acids Res.*, 41(W1):W129–W132.

Dhillon BK, Laird MR, Shay Ja, et al. (2015). *IslandViewer 3: more flexible, interactive genomic island discovery, visualization and analysis*. *Nucleic Acids Res.*, 43(W1):W104–W108.

Dobrindt U, Hochhut B, Hentschel U, et al. (2004). *Genomic islands in pathogenic and environmental microorganisms*. *Nature Rev. Microbiol.*, 2(5):414–424.

Domazet-Lošo M and Haubold B (2011). *Alignment-free detection of horizontal gene transfer between closely related bacterial genomes*. *Mob. Genet. Elements.*, 1(3):230–235.

Doolittle WF and Bapteste E (2007). *Pattern pluralism and the tree of life hypothesis*. *Proc. Natl. Acad. Sci. U.S.A.*, 104(7):2043–2049.

Doolittle WF, Boucher Y, NesbØ CL, et al. (2003). *How big is the iceberg of which organellar genes in nuclear genomes are but the tip? Philos. Trans. R. Soc. Lond., B, Biol. Sci.*, 358(1429):39–58.

Doolittle WF and Brunet TDP (2016). *What is the tree of life? PLOS Genet.*, 12(4):e1005912.

Doyon JP, Ranwez V, Daubin V, et al. (2011). *Models, algorithms and programs for phylogeny reconciliation*. *Brief. Bioinform.*, 12(5):392–400.

Dufraigne C, Fertil B, Lespinats S, et al. (2005). *Detection and characterization of horizontal transfers in prokaryotes using genomic signature*. *Nucleic Acids Res.*, 33(1):e6.

Dufresne A, Ostrowski M, Scanlan DJ, et al. (2008). *Unraveling the genomic mosaic of a ubiquitous genus of marine cyanobacteria*. *Genome Biol.*, 9(5):R90.

Elhai J, Liu H, and Taton A (2012). *Detection of horizontal transfer of individual genes by anomalous oligomer frequencies*. *BMC Genomics*, 13:245.

Enright MC, Robinson DA, Randle G, et al. (2002). *The evolutionary history of methicillin-resistant Staphylococcus aureus (MRSA)*. *Proc. Natl. Acad. Sci. U.S.A.*, 99(11):7687–7692.

Finn RD, Bateman A, Clements J, et al. (2014). *Pfam: the protein families database*. *Nucleic Acids Res.*, 42(D1):D222–D230.

Flannery EL, Mody L, and Mobley HL (2009). *Identification of a modular pathogenicity island that is widespread among urease-producing uropathogens and shares features with a diverse group of mobile elements*. *Infect. Immun.*, 77(11):4887–4894.

Fontaine MC, Pease JB, Steele A, et al. (2015). *Extensive introgression in a malaria vector species complex revealed by phylogenomics*. *Science*, 347(6217):27–28.

Francis AR and Steel M (2015). *Which phylogenetic networks are merely trees with additional arcs?* *Syst. Biol.*, 64(5):768–777.

Frank C, Werber D, Cramer JP, et al. (2011). *Epidemic profile of shiga-toxin–producing escherichia coli o104: H4 outbreak in germany*. *N. Engl. J. Med.*, 365(19):1771–1780.

Galperin MY, Makarova KS, Wolf YI, et al. (2015). *Expanded microbial genome coverage and improved protein family annotation in the COG database*. *Nucleic Acids Res.*, 43(D1):D261–D269.

Gambette P, Gunawan AD, Labarre A, et al. (2015a). *Locating a tree in a phylogenetic network in quadratic time*. In *RECOMB*, pages 96–107.

Gambette P, Gunawan AD, Labarre A, et al. (2015b). *Solving the tree containment problem for genetically stable networks in quadratic time*. In *IWOCA*, pages 197–208.

Gao F and Zhang CT (2006). *GC-Profile: a web-based tool for visualizing and analyzing the variation of GC content in genomic sequences*. *Nucleic Acids Res.*, 34(suppl_2):W686–W691.

Garcia-Vallve S, Guzman E, Montero MA, et al. (2003). *HGT-DB: a database of putative horizontally transferred genes in prokaryotic complete genomes*. *Nucleic Acids Res.*, 31(1):187–189.

Gogarten JP, Doolittle WF, and Lawrence JG (2002). *Prokaryotic evolution in light of gene transfer*. *Mol. Biol. Evol.*, 19(12):2226–2238.

Gogarten JP and Townsend JP (2005). *Horizontal gene transfer, genome innovation and evolution*. *Nat. Rev. Microbiol.*, 3(9):679–687.

Grass Phylogeny Working Group (2001). *Phylogeny and subfamilial classification of the grasses (Poaceae)*. *Ann. Mo. Bot. Gard.*, 88(3):373–457.

Griffith F (1928). *The significance of pneumococcal types*. *J. Hyg.*, 27(02):113–159.

Gunawan AD, DasGupta B, and Zhang L (2016a). *Locating a tree in a reticulation-visible network in cubic time*. In *RECOMB*, page 266.

Gunawan AD, DasGupta B, and Zhang L (2017). *A decomposition theorem and two algorithms for reticulation-visible networks*. *Inform. Comput.*, 252:161–175.

Gunawan AD, Lu B, and Zhang L (2016b). *A program for verification of phylogenetic network models*. *Bioinformatics*, 32(17):i503–i510.

Gusfield D (2014). *ReCombinatorics: The Algorithmics of Ancestral Recombination Graphs and Explicit Phylogenetic Networks*. MIT Press, Cambridge, USA.

Hacker J, Bender L, Ott M, et al. (1990). *Deletions of chromosomal regions coding for fimbriae and hemolysins occur in vitro and in vivo in various extra intestinal escherichia coli isolates*. *Microb. Pathog.*, 8(3):213–225.

Hacker J, Blum-Oehler G, Mühldorfer I, et al. (1997). *Pathogenicity islands of virulent bacteria: structure, function and impact on microbial evolution*. *Mol. Microbiol.*, 23(6):1089–1097.

Hacker J and Carniel E (2001). *Ecological fitness, genomic islands and bacterial pathogenicity*. *EMBO Rep.*, 2(5):376–381.

Hacker J and Kaper JB (2000). *Pathogenicity islands and the evolution of microbes*. *Annu. Rev. Microbiol.*, 54:641–679.

Hasan MS, Liu Q, Wang H, et al. (2012). *GIST: Genomic island suite of tools for predicting genomic islands in genomic sequences*. *Bioinformation*, 8(4):203.

Ho Sui SJ, Fedynak A, Hsiao WWL, et al. (2009). *The association of virulence factors with genomic islands*. *PLOS ONE*, 4(12):e8094.

Hsiao W, Wan I, Jones SJ, et al. (2003). *IslandPath: aiding detection of genomic islands in prokaryotes*. *Bioinformatics*, 19(3):418–420.

Hsiao WWL, Ung K, Aeschliman D, et al. (2005). *Evidence of a large novel gene pool associated with prokaryotic genomic islands*. *PLOS Genet.*, 1(5):1–11.

Huber KT, van Iersel L, Moulton V, et al. (2015). *How much information is needed to infer reticulate evolutionary histories?* *Syst. Biol.*, 64(1):102–111.

Hudson CM, Lau BY, and Williams KP (2015). *Islander: a database of precisely mapped genomic islands in tRNA and tmRNA genes*. *Nucleic Acids Res.*, 43(D1):D48–D53.

Huson DH and Bryant D (2006). *Application of phylogenetic networks in evolutionary studies*. *Mol. Biol. Evol.*, 23(2):254–267.

Huson DH and Rupp R (2008). *Summarizing multiple gene trees using cluster networks*. In *WABI*, pages 296–305.

Huson DH, Rupp R, Berry V, et al. (2009). *Computing galled networks from real data*. *Bioinformatics*, 25(12):i85–i93.

Huson DH, Rupp R, and Scornavacca C (2010). *Phylogenetic networks: concepts, algorithms and applications*. Cambridge University Press, Cambridge, UK.

Huson DH and Scornavacca C (2012). *Dendroscope 3: An interactive tool for rooted phylogenetic trees and networks*. *Syst. Biol.*, 61(6):1061–1067.

Hyatt D, Chen GL, LoCascio PF, et al. (2010). *Prodigal: prokaryotic gene recognition and translation initiation site identification*. *BMC Bioinformatics*, 11:119.

Jacox E, Chauve C, Szöllősi GJ, et al. (2016). *eccetera: comprehensive gene tree-species tree reconciliation using parsimony*. *Bioinformatics*, 32(13):2056–2058.

Jain R, Rivera MC, and Lake JA (1999). *Horizontal gene transfer among genomes: the complexity hypothesis*. *Proc. Natl. Acad. Sci. U.S.A.*, 96(7):3801–3806.

Jani M, Mathee K, and Azad RK (2016). *Identification of novel genomic islands in liverpool epidemic strain of pseudomonas aeruginosa using segmentation and clustering*. *Front. Microbiol.*, 7:1210.

Juhas M, van der Meer JR, Gaillard M, et al. (2009). *Genomic islands: tools of bacterial horizontal gene transfer and evolution*. *FEMS Microbiol. Rev.*, 33(2):376–393.

Kanj IA, Nakhleh L, Than C, et al. (2008). *Seeing the trees and their branches in the network is hard*. *Theor. Comput. Sci.*, 401(1-3):153–164.

Karlin S (2001). *Detecting anomalous gene clusters and pathogenicity islands in diverse bacterial genomes*. *Trends Microbiol.*, 9(7):335–343.

Keeling PJ and Palmer JD (2008). *Horizontal gene transfer in eukaryotic evolution*. *Nat. Rev. Genet.*, 9(8):605–618.

Kloesges T, Popa O, Martin W, et al. (2011). *Networks of gene sharing among 329 proteobacterial genomes reveal differences in lateral gene transfer frequency at different phylogenetic depths*. *Mol. Biol. Evol.*, 28(2):1057–1074.

Koonin EV, Makarova KS, and Aravind L (2001). *Horizontal gene transfer in prokaryotes: quantification and classification*. *Annu. Rev. Microbiol.*, 55(1):709–742.

Koski LB, Morton RA, and Golding GB (2001). *Codon bias and base composition are poor indicators of horizontally transferred genes*. *Mol. Biol. Evol.*, 18(3):404–412.

Krzywinski M, Schein J, Birol I, et al. (2009). *Circos: an information aesthetic for comparative genomics*. *Genome Res.*, 19(9):1639–1645.

Kunin V, Goldovsky L, Darzentas N, et al. (2005). *The net of life: reconstructing the microbial phylogenetic network*. *Genome Res.*, 15(7):954–959.

Kunin V and Ouzounis CA (2003). *Genetrace–reconstruction of gene content of ancestral species*. *Bioinformatics*, 19(11):1412–1416.

Kyle JL, Cummings CA, Parker CT, et al. (2012). *Escherichia coli serotype o55: H7 diversity supports parallel acquisition of bacteriophage at shiga toxin phage insertion sites during evolution of the o157: H7 lineage*. *J. Bacteriol.*, 194(8):1885–1896.

Langille MG and Brinkman FS (2009). *IslandViewer: an integrated interface for computational identification and visualization of genomic islands*. *Bioinformatics*, 25(5):664–665.

Langille MGI, Hsiao WWL, and Brinkman FSL (2008). *Evaluation of genomic island predictors using a comparative genomics approach*. *BMC Bioinformatics*, 9:329.

Langille MGI, Hsiao WWL, and Brinkman FSL (2010). *Detecting genomic islands using bioinformatics approaches*. *Nature Rev. Microbiol.*, 8(5):373–382.

Lawrence JG and Ochman H (1997). *Amelioration of bacterial genomes: rates of change and exchange*. *J. Mol. Evol.*, 44(4):383–397.

Lawrence JG and Ochman H (2002). *Reconciling the many faces of lateral gene transfer*. *Trends Microbiol.*, 10(1):1–4.

Lee CC, Chen YPP, Yao TJ, et al. (2013). *GI-POP: A combinational annotation and genomic island prediction pipeline for ongoing microbial genome projects*. *Gene*, 518(1):114–123.

Leslie C, Eskin E, and Noble WS (2002). *The spectrum kernel: a string kernel for SVM protein classification*. *PSB*, pages 564–575.

Lowe TM and Eddy SR (1997). *tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence*. *Nucleic Acids Res.*, 25(5):955–964.

Lu B and Leong HW (2016a). *Computational methods for predicting genomic islands in microbial genomes*. *Comput. Struct. Biotechnol. J.*, 14:200–206.

Lu B and Leong HW (2016b). *GI-SVM: A sensitive method for predicting genomic islands based on unannotated sequence of a single genome*. *J. Bioinform. Comput. Biol.*, 14(01):1640003.

Lu B and Leong HW (2017). *GI-Cluster: detecting genomic islands via consensus clustering on multiple features*. Submitted.

Lu B, Zhang L, and Leong HW (2017). *A program to compute the soft Robinson–Foulds distance between phylogenetic networks*. *BMC Genomics*, 18(Suppl 2):111.

Mallet J (2007). *Hybrid speciation*. *Nature*, 446(7133):279–283.

Mantri Y and Williams KP (2004). *Islander: a database of integrative islands in prokaryotic genomes, the associated integrases and their DNA site specificities*. *Nucleic Acids Res.*, 32(suppl_1):D55–D58.

Marcus SL, Brumell JH, Pfeifer CG, et al. (2000). *Salmonella pathogenicity islands: big virulence in small packages*. *Microb. Infect.*, 2(2):145–156.

Martin DP, Lemey P, and Posada D (2011). *Analysing recombination in nucleotide sequences*. *Mol. Ecol. Resour.*, 11(6):943–955.

Matzke NJ, Shih PM, and Kerfeld CA (2014). *Bayesian analysis of congruence of core genes in prochlorococcus and synechococcus and implications on horizontal gene transfer*. *PLOS ONE*, 9(1):e85103.

McArthur AG, Waglechner N, Nizam F, et al. (2013). *The Comprehensive Antibiotic Resistance Database*. *Antimicrob. Agents Chemother.*, 57(7):3348–3357.

Merkl R (2004). *SIGI: score-based identification of genomic islands*. *BMC Bioinformatics*, 5:22.

Metzler S (2014). *Identification of Gene Transfer Events in Viruses*. Ph.D. thesis, Universität des Saarlandes Saarbrücken.

Metzler S and Kalinina OV (2014). *Detection of atypical genes in virus families using a one-class svm*. *BMC Genomics*, 15:913.

Monti S, Tamayo P, Mesirov J, et al. (2003). *Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data*. *Mach. learn.*, 52(1-2):91–118.

Moret BM, Nakhleh L, Warnow T, et al. (2004). *Phylogenetic networks: modeling, reconstructibility, and accuracy*. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 1(1):13–23.

Moriel DG, Bertoldi I, Spagnuolo A, et al. (2010). *Identification of protective and broadly conserved vaccine antigens from the genome of extraintestinal pathogenic Escherichia coli*. *Proc Natl Acad Sci U S A*, 107(20):9072–9077.

Morrison D (2011). *Introduction to Phylogenetic Networks*. RJR Productions, Uppsala, Sweden.

Morrison DA (2014). *The genealogical world of phylogenetic networks: the first HGT network, available at http://phylonetworks.blogspot.com.es/2014/04/the-first-hgt-network.html*.

Nakhleh L (2010). *A metric on the space of reduced phylogenetic networks*. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 7(2):218–222.

Nakhleh L, Ruths D, and Wang LS (2005). *RIATA-HGT: a fast and accurate heuristic for reconstructing horizontal gene transfer*. In *COCOON*, pages 84–93.

Nakhleh L and Wang LS (2005). *Phylogenetic networks: Properties and relationship to trees and clusters*. In *TCSB II*, pages 82–99.

Nawrocki EP, Burge SW, Bateman A, et al. (2015). *Rfam 12.0: updates to the rna families database*. *Nucleic Acids Res.*, 43(D1):D130–D137.

Nawrocki EP and Eddy SR (2013). *Infernal 1.1: 100-fold faster rna homology searches*. *Bioinformatics*, 29(22):2933–2935.

Novichkov PS, Omelchenko MV, Gelfand MS, et al. (2004). *Genome-wide molecular clock and horizontal gene transfer in bacterial evolution*. *J. Bacteriol.*, 186(19):6575–6585.

Ochman H, Lawrence JG, and Groisman EA (2000). *Lateral gene transfer and the nature of bacterial innovation*. *Nature*, 405(6784):299–304.

Ou HY, Chen LL, Lonnen J, et al. (2006). *A novel strategy for the identification of genomic islands by comparative analysis of the contents and contexts of tRNA sites in closely related bacteria*. *Nucleic Acids Res.*, 34(1):1–11.

Ou HY, He X, Harrison EM, et al. (2007). *MobilomeFINDER: web-based tools for in silico and experimental discovery of bacterial genomic islands*. *Nucleic Acids Res.*, 35(suppl_2):W97–W104.

Pearson MM and Mobley HL (2007). *The type iii secretion system of proteus mirabilis hi4320 does not contribute to virulence in the mouse model of ascending urinary tract infection*. *J. Med. Microbiol.*, 56(10):1277–1283.

Peden JF (1999). *Analysis of codon usage*. Ph.D. thesis, University of Nottingham.

Penel S, Arigon AM, Dufayard JF, et al. (2009). *Databases of homologous gene families for comparative genomics*. *BMC Bioinformatics*, 10(Suppl 6):S3.

Podell S and Gaasterland T (2007). *Darkhorse: a method for genome-wide prediction of horizontal gene transfer*. *Genome Biol.*, 8(2):R16.

Popa O and Dagan T (2011). *Trends and barriers to lateral gene transfer in prokaryotes*. *Curr. Opin. Microbiol.*, 14(5):615–623.

Poptsova M (2009). *Testing phylogenetic methods to identify horizontal gene transfer*. In *Horizontal Gene Transfer: Genomes in Flux*, pages 227–240. Humana Press.

Poptsova MS and Gogarten JP (2007). *The power of phylogenetic approaches to detect horizontally transferred genes*. *BMC Evol. Biol.*, 7:45.

Pundhir S, Vijayvargiya H, and Kumar A (2008). *PredictBias: a server for the identification of genomic and pathogenicity islands in prokaryotes*. *In Silico Biol.*, 8(3, 4):223–234.

Ragan MA (2001). *On surrogate methods for detecting lateral gene transfer*. *FEMS Microbiol. Lett.*, 201(2):187–191.

Ragan MA and Beiko RG (2009). *Lateral genetic transfer: open issues*. *Philos. Trans. R. Soc. Lond., B, Biol. Sci.*, 364(1527):2241–2251.

Ragan MA, Harlow TJ, and Beiko RG (2006). *Do different surrogate methods detect lateral genetic transfer events of different relative ages?* *Trends Microbiol.*, 14(1):4–8.

Rajan I, Aravamuthan S, and Mande SS (2007). *Identification of compositionally distinct regions in genomes using the centroid method*. *Bioinformatics*, 23(20):2672–2677.

Ravenhall M, Škunca N, Lassalle F, et al. (2015). *Inferring horizontal gene transfer*. *PLOS Comput. Biol.*, 11(5):e1004095.

Sandberg R, Branden CI, Ernberg I, et al. (2003). *Quantifying the species-specificity in genomic signatures, synonymous codon choice, amino acid usage and G+C content*. *Gene*, 311:35–42.

Sandberg R, Winberg G, Bränden CI, et al. (2001). *Capturing whole-genome characteristics in short sequences using a naive bayesian classifier*. *Genome Res.*, 11(8):1404–1409.

Schmidt H and Hensel M (2004). *Pathogenicity islands in bacterial pathogenesis*. *Clin. Microbiol. Rev.*, 17(1):14–56.

Schmidt HA (2003). *Phylogenetic trees from large datasets*. Ph.D. thesis, University of Düsseldorf.

Schölkopf B, Williamson RC, Smola AJ, et al. (2000). *Support vector method for novelty detection*. In *Adv. Neural. Inf. Process. Syst.*, pages 582–588.

Scrucca L, Fop M, Murphy TB, et al. (2016). *mclust 5: Clustering, classification and density estimation using gaussian finite mixture models*. *R. J.*, 8(1):289–317.

Sharp PM and Li WH (1987). *The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications*. *Nucleic Acids Res.*, 15(3):1281–1295.

Shi T and Falkowski PG (2008). *Genome evolution in cyanobacteria: the stable core and the variable shell*. *Proc. Natl. Acad. Sci. U.S.A.*, 105(7):2510–2515.

Shrivastava S, Siva Kumar Reddy CV, and Mande SS (2010). *INDeGenIUS, a new method for high-throughput identification of specialized functional islands in completely sequenced organisms*. *J Biosciences*, 35(3):351–364.

Sjöstrand J, Tofigh A, Daubin V, et al. (2014). *A Bayesian method for analyzing lateral gene transfer*. *Syst. Biol.*, 63(3):409–420.

Skippington E and Ragan MA (2011). *Lateral genetic transfer and the construction of genetic exchange communities*. *FEMS Microbiol. Rev.*, 35(5):707–735.

Soares SC, Abreu VaC, Ramos RTJ, et al. (2012). *PIPS: Pathogenicity island prediction software*. *PLOS ONE*, 7(2):e30848.

Sonnenburg S, Rätsch G, Henschel S, et al. (2010). *The shogun machine learning toolbox*. *J. Mach. Learn. Res.*, 11:1799–1802.

Soucy SM, Huang J, and Gogarten JP (2015). *Horizontal gene transfer: building the web of life*. *Nat. Rev. Genet.*, 16(8):472–482.

Stolzer M, Lai H, Xu M, et al. (2012). *Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees*. *Bioinformatics*, 28(18):i409–i415.

Szöllősi GJ, Boussau B, Abby SS, et al. (2012). *Phylogenetic modeling of lateral gene transfer reconstructs the pattern and relative timing of speciations*. *Proc. Natl. Acad. Sci. U.S.A.*, 109(43):17513–17518.

Szöllősi GJ, Rosikiewicz W, Boussau B, et al. (2013a). *Efficient exploration of the space of reconciled gene trees*. *Syst. Biol.*, 62(6):901–912.

Szöllősi GJ, Tannier E, Lartillot N, et al. (2013b). *Lateral gene transfer from the dead*. *Syst. Biol.*, 62(3):386–397.

Than C and Nakhleh L (2008). *SPR-based tree reconciliation: Non-binary trees and multiple solutions*. In *APBC*, pages 251–260.

Than C, Ruths D, and Nakhleh L (2008). *PhyloNet: a software package for analyzing and reconstructing reticulate evolutionary relationships*. *BMC Bioinformatics*, 9:322.

The UniProt Consortium (2017). *Uniprot: the universal protein knowledgebase*. *Nucleic Acids Res.*, 45(D1):D158–D169.

Tofigh A, Hallett M, and Lagergren J (2011). *Simultaneous identification of duplications and lateral gene transfers*. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 8(2):517–535.

Trappe K, Marschall T, and Renard BY (2016). *Detecting horizontal gene transfer by mapping sequencing reads across species boundaries*. *Bioinformatics*, 32(17):i595–i604.

Trost E, Blom J, de Castro Soares S, et al. (2012). *Pangenomic study of Corynebacterium diphtheriae that provides insights into the genomic diversity of pathogenic isolates from cases of classical diphtheria, endocarditis, and pneumonia*. *J. Bacteriol.*, 194(12):3199–3215.

Tsirigos A and Rigoutsos I (2005a). *A sensitive, support-vector-machine method for the detection of horizontal gene transfers in viral, archaeal and bacterial genomes*. *Nucleic Acids Res.*, 33(12):3699–3707.

Tsirigos A and Rigoutsos I (2005b). *A new computational method for the detection of horizontal gene transfer events*. *Nucleic Acids Res.*, 33(3):922–933.

Tu Q and Ding D (2003). *Detecting pathogenicity islands and anomalous gene clusters by iterative discriminant analysis*. *FEMS Microbiol. Lett.*, 221(2):269–275.

van Iersel L, Kelk S, Rupp R, et al. (2010a). *Phylogenetic networks do not need to be complex: using fewer reticulations to represent conflicting clusters*. *Bioinformatics*, 26(12):i124–i131.

van Iersel L, Semple C, and Steel M (2010b). *Locating a tree in a phylogenetic network*. *Inform. Process. Letters*, 110(23):1037–1043.

Vega-Pons S and Ruiz-Shulcloper J (2011). *A survey of clustering ensemble algorithms*. *Int. J. Pattern Recogn.*, 25(03):337–372.

Vernikos GS (2008). *In silico prediction of genomic islands in microbial genomes*. Ph.D. thesis, University of Cambridge.

Vernikos GS and Parkhill J (2006). *Interpolated variable order motifs for identification of horizontally acquired DNA: revisiting the Salmonella pathogenicity islands*. *Bioinformatics*, 22(18):2196–2203.

Vernikos GS and Parkhill J (2008). *Resolving the structural features of genomic islands: A machine learning approach*. *Genome Res.*, 18(2):331–342.

Vernikos GS, Thomson NR, and Parkhill J (2007). *Genetic flux over time in the Salmonella lineage*. *Genome Biol.*, 8(6):R100.

Waack S, Keller O, Asper R, et al. (2006). *Score-based prediction of genomic islands in prokaryotic genomes using hidden Markov models*. *BMC Bioinformatics*, 7:142.

Wang H, Fazekas J, Booth M, et al. (2011). *An integrative approach for genomic island prediction in prokaryotic genomes*. In *ISBRA*, pages 404–415.

Wang H and Song M (2011). *Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming*. *R. J.*, 3(2):29.

Wei W, Gao F, Du MZ, et al. (2017). *Zisland explorer: detect genomic islands by combining homogeneity and heterogeneity properties*. *Brief. Bioinform.*, 18(3):357–366.

Wellner A, Lurie MN, and Gophna U (2007). *Complexity, connectivity, and duplicability as barriers to lateral gene transfer*. *Genome Biol.*, 8(8):R156.

Wen D, Yu Y, Hahn MW, et al. (2016). *Reticulate evolutionary history and extensive introgression in mosquito species revealed by phylogenetic network analysis*. *Mol. Ecol.*, 25(11):2361–2372.

Wilkerson MD and Hayes DN (2010). *ConsensusClusterPlus: a class discovery tool with confidence assessments and item tracking*. *Bioinformatics*, 26(12):1572–1573.

Williams KP (2002). *Integration sites for genetic elements in prokaryotic tRNA and tmRNA genes: sublocation preference of integrase subfamilies*. *Nucleic Acids Res.*, 30(4):866–875.

Winstanley C, Langille MGI, Fothergill JL, et al. (2009). *Newly introduced genomic prophage islands are critical determinants of in vivo competitiveness in the liverpool epidemic strain of pseudomonas aeruginosa*. *Genome Res.*, 19(1):12–23.

Wu Y (2010). *Close lower and upper bounds for the minimum reticulate network of multiple phylogenetic trees*. *Bioinformatics*, 26(12):i140–i148.

Yoon SH, Hur CG, Kang HY, et al. (2005). *A computational approach for identifying pathogenicity islands in prokaryotic genomes. BMC Bioinformatics*, 6:184.

Yoon SH, Park YK, and Kim JF (2015). *PAIDB v2.0: exploration and analysis of pathogenicity and resistance islands*. *Nucleic Acids Res.*, 43(D1):D624–D630.

Yoon SH, Park YK, Lee S, et al. (2007). *Towards pathogenomics: A web-based resource for pathogenicity islands*. *Nucleic Acids Res.*, 35(suppl_1):395–400.

Zhang L (2016). *On tree-based phylogenetic networks*. *J. Comp. Biol.*, 23(7):553–565.

Zhang R, Ou HY, Gao F, et al. (2014). *Identification of horizontally-transferred genomic islands and genome segmentation points by using the GC Profile method*. *Curr. Genomics*, 15(2):113–121.

Zhang R and Zhang CT (2004). *A systematic method to identify genomic islands and its applications in analyzing the genomes of Corynebacterium glutamicum and Vibrio vulnificus CMCP6 chromosome I*. *Bioinformatics*, 20(5):612–622.

Zhaxybayeva O (2009). *Detection and quantitative assessment of horizontal gene transfer*. In *Horizontal Gene Transfer: Genomes in Flux*, pages 195–213. Humana Press.

Zhaxybayeva O and Doolittle WF (2011). *Lateral gene transfer*. *Curr. Biol.*, 21(7):R242–R246.

Zhaxybayeva O, Doolittle WF, Papke RT, et al. (2009). *Intertwined evolutionary histories of marine synechococcus and prochlorococcus marinus*. *Genome Biol. Evol.*, 1:325–339.

Zhaxybayeva O, Gogarten JP, Charlebois RL, et al. (2006). *Phylogenetic analyses of cyanobacterial genomes: quantification of horizontal gene transfer events*. *Genome Res.*, 16(9):1099–1108.

Zhou Y, Liang Y, Lynch KH, et al. (2011). *PHAST: A fast phage search tool*. *Nucleic Acids Res.*, 39(suppl_2):W347–W352.

Zhu Q, Kosoy M, and Dittmar K (2014). *HGTector: an automated method facilitating genome-wide discovery of putative horizontal gene transfers*. *BMC Genomics*, 15:717.

Appendix A

Supplementary materials for Section 2.4

In this chapter, we give more details about GI-Cluster method presented in Section 2.4.

A.1 Supplementary details of GI-Cluster method

In this section, we provide more technical details regarding the GI-Cluster method.

A.1.1 Feature extraction step

We compute four groups of features for a genomic segment: (1) sequence composition; (2) gene function; (3) boundary signature; and (4) gene distribution. Gene prediction was performed by Prodigal (version 2.6.3) (Hyatt et al., 2010), a commonly used tool for microbial gene finding. The computation for each feature is via custom scripts or database search (Table A.1). The details for computing each group of features are described in the following subsections.

Computation of sequence composition

For each gene, we compute several metrics measuring its GC content, codon usage, and k-mer frequency. For each segment, we compute several metrics measuring its k-mer frequency, GC content and the average of metrics for codon usage of genes within the segment.

GC content is the simplest and earliest type of features proposed to detect LGT events (Tsirigos and Rigoutsos, 2005b). A genomic region with GC content that deviates significantly from GC content of the genome is likely to be laterally transferred. We compute the GC content of each gene as well as GC(k), with $k = 1, 2, 3$, the GC content determined by considering only the nucleotides occupying the k^{th} position within each codon of a gene. In clustering, we exclude the GC content of second codon positions since it was shown to be very similar across species (Lawrence and Ochman, 1997). The GC content for each segment is measured by the average of the values for all genes within it when gene predictions are available. To measure the GC deviation of each gene (segment) from the genome, we compute χ^2 value between GC content for each gene (segment) and GC content for the genome, which was used in previous studies (Lawrence and Ochman, 1997; Tsirigos and Rigoutsos, 2005b). A higher χ^2 value means that the GC content of the gene or segment deviates more from that of the genome.

Codon usage is another commonly used compositional feature to detect GIs (Waack et al., 2006). A genomic region with atypical codon usage patterns is likely to be laterally transferred. To measure the codon usage deviation of each gene, we compute several metrics that are commonly used for detecting regions with atypical codon usage, including χ^2 of relative synonymous codon usage (RSCU) (Lawrence and Ochman, 1997), codon usage bias (Cub) (Zhang and Zhang, 2004), amino-acid usage bias (AAub) (Zhang and Zhang, 2004), and codon adaptation index (CAI) (Sharp and Li, 1987). Cub and χ^2 value measure the codon usage bias of a gene against the average of all genes in

Table A.1: The methods used for extracting GI-related features in a genome.

Feature	**Discriminant criterion**	**Measure**	**Computation method**
Sequence composition	**GC content**	χ^2	Python scripts
	Codon usage	χ^2, Cub, AAub, CAI	CodonW, Python scripts
	***k*-mer frequency**	Covariance	Python scripts
Gene function	Mobility-related gene	Percentage	HMMer against Pfam
	Phage-related gene	Percentage	BLAST against PHAST
	Virulence factor	Percentage	BLAST against VFDB
	Antibiotic resistance gene	Percentage	BLAST against CARD
	Novel gene	Percentage	BLAST against COG
	Non-coding RNA	Count	Infernal against Rfam
Gene distribution	Gene density	Definition	Python scripts
	Intergenic distance	Definition	Python scripts
Boundary signature	tRNA	Binary (presence-absence)	tRNAscan-SE
	Short repeat	Binary (presence-absence)	Repseek

Discriminant criterion in bold can be applied to genomes without gene predictions.

the genome. A higher value suggests that the gene is more likely to have atypical codon usage. Similarly, AAub is the amino acid usage bias of a gene against the average of all genes in the genome, and a gene with a larger AAub value may be atypical. CAI measures the similarity of the codon usage of a gene to that of highly expressed genes for an organism. A gene with a higher CAI value are more likely to be normal. CAI is computed by CodonW (Peden, 1999) and the other three metrics are computed by custom scripts.

The frequencies of k-mers (k usually ranges from two to nine) in different genomes show species-specificity (Sandberg et al., 2001, 2003). It has been demonstrated that k-mer frequencies are better indicators in detecting LGT events (Becq et al., 2010). The order of k-mer may affect its discrimination power. Generally, high order k-mers ($k >= 4$) can provide more sufficient discrimination of regions with atypical composition than low order k-mers, but there may not be enough data for high order k-mers in a sequence (Vernikos and Parkhill, 2006). Thus we compute the frequencies for k-mers of size two to eight of a gene or segment and the genome. To measure k-mer frequency bias of each gene or segment, we compute the distance of k-mer frequency vectors between each gene and the genome. We use covariance distance measure, which was shown to be effective in detecting laterally transferred genes (Tsirigos and Rigoutsos, 2005b). The final values were scaled to be between zero and one. A genes or segment with a lower covariance value is more likely to be atypical.

Computation of gene function

For each gene, we find whether it encodes specific functions related to GIs, including mobility, phage, virulence factor, antibiotic resistance, novel function and non-coding RNA (ncRNA). For each segment, we compute the percentage of each kind of genes within it. The detection of such kind of genes relies on homologous search against related databases.

The presence of mobility-related genes in a genomic segment may serve as evidence of LGT, so it was used in several tools for GI prediction (Hsiao et al., 2005; Mantri and Williams, 2004). Mobility-related genes mainly include: insertion sequence element, transposase and integrase. We use HMMer (HMMER 3.1b2, http://hmmer.org/) to find these genes by searching against 128 HMM profiles retrieved from the Pfam HMM

library for Pfam-A families (Finn et al., 2014) (version Nov 2016).

Prophage can be seen as one kind of GIs (Langille et al., 2010). To find phage-related genes, we use BLAST (version 2.5.0) (Altschul et al., 1997) against PHAST prophage and virus database (Zhou et al., 2011) (version Nov 2016).

Novel genes are genes with no detectable homologs in public databases. They were reported to be significantly more prevalent in GIs than non-GIs in many organisms (Hsiao et al., 2005). We use COG (clusters of orthologous groups of proteins) database (Galperin et al., 2015) to identify novel genes. Genes are assigned to different COG functional categories and those without COG assignments are considered novel.

Virulence factors were shown to be disproportionately associated with GIs in pathogens (Ho Sui et al., 2009). So we use BLAST to search the presence of virulence factors against protein sequences of full dataset from VFDB (Chen et al., 2016) (version Nov 2016) which contains high-quality virulence factors from various bacterial pathogens and is updated regularly.

GIs are also often related to the dissemination of antibiotic resistance (Juhas et al., 2009). So we detect antibiotic resistance genes by blasting against CARD (McArthur et al., 2013) (version 1.1.1, Nov 2016) which provides rigorously curated antibiotic resistance genes.

The ribosomal RNA (rRNA) operon often has atypical composition for reasons other than LGT (Vernikos and Parkhill, 2008). Besides, a region with a few tRNAs is also not likely be a GI. So we count the number of rRNA and tRNA genes within each segment and exclude segments with more rRNA and tRNA genes than protein-coding genes during clustering. We find ncRNA genes by using Infernal (Nawrocki and Eddy, 2013) to search against Rfam covariance models (Nawrocki et al., 2015) (version Dec 2016).

Computation of boundary signature

For each segment, we identify the presence of flanking tRNA genes or short repeats around its boundaries. The computed boundary signatures are mainly used for refining boundaries of segments that are assigned to the group of GIs after clustering.

The tRNA genes often serve as insertion sites of GIs and have been used for detecting a subset of GIs (Mantri and Williams, 2004; Ou et al., 2006; Hudson et al., 2015). We

use tRNAscan-SE (version 1.3.1) (Lowe and Eddy, 1997) to predict tRNA genes in a genome. Then we use custom scripts to find tRNA genes flanking each segment.

Short repeats of varying sizes are also reported to be around the endpoints of some GIs (Hacker et al., 1997). We use Repseek (version 6.6) (Achaz et al., 2007) to detect all the repeats in a genome. Then we use custom scripts to find short direct repeats (DRs) or inverted repeats (IRs) flanking each segment.

Computation of gene distribution

For each segment, we compute its gene density and intergenic distance according to their definitions. Gene density refers to the number of genes per kilo base in a region. Intergenic distance refers to the average gene distance in a region.

GIs were suggested to have lower gene densities and shorter intergenic distances than non-GIs (Che et al., 2014b). It was shown that the average intergenic distance for non-GIs are generally shorter than that of GIs (Wang et al., 2011). These features may have some differentiation power, but they may not be as effective as compositional and functional features (Che et al., 2014b). Moreover, the size of genomic segments affects the values of gene density and average intergenic distance. Thus, we compute these two features only for postprocessing and manual analysis.

A.1.2 Consensus clustering step

To deal with feature variety, after getting the clusterings of segments on each feature, we may use subsampling to randomly select the clusterings for different subsets of features and combine them into a single clustering. If the results of consensus clustering are more stable and robust to sampling variability, we are more confident that the attained clusters represent the real separation of GIs from non-GIs. In principle, the more samplings are, the more stable results should be. For simplicity, we can also combine the clusterings of all the computed features.

When using feature subsampling, we get a set of consensus matrices on different subsets of features, and then obtain a single consensus matrix as the average of all these consensus matrices. When using the clusterings on all the features, we combine all the connectivity matrices to obtain a single consensus matrix.

We use optimal 1D k-means clustering (Wang and Song, 2011) and model-based

clustering (kernel density estimation) (Scrucca et al., 2016) on each feature for now, but other clustering methods can also be used. Due to the complexity of genome, the demarcation between normal segments and atypical segments may not be very clear and there may be some segments with weak memberships in both normal and atypical groups. So we set the cluster number to be three when using 1D k-means clustering. For model-based clustering, the cluster number can be automatically estimated.

For clustering on the final consensus matrix, hierarchical clustering with average linkage is used by default because it is intuitive and often gives reasonable results. The heat map obtained from hierarchical clustering can also be used to assess the composition of resultant clusters. Other clustering methods can also be used by parameter passing.

Due to the uncertainties of parameter choices in the clustering process, the results of GI-Cluster may be slightly different in different runs. In practice, users can try different parameters to find more reasonable results, with the aid of extensive annotations provided by GI-Cluster.

A.1.3 Postprocessing step

Among the initial non-GI segments, the following postprocessing rules are used by default: a segment with the presence of flanking tRNA gene(s) and mobility-related genes is reclassified as a GI segment; a segment with at least five protein-coding genes and no less than 80% genes being phage-related genes or novel genes is also redesignated as a GI segment.

Among the initial GI segments, a segment with more than five protein-coding genes but no flanking tRNA genes or mobility-related genes is reclassified as a non-GI segment. At the same time, a segment with the following properties is also excluded: there are no flanking tRNA genes or mobility-related genes; the percentages of virulence factors, phage-related genes, antibiotic resistance genes and novel genes are smaller than given thresholds; the gene density is higher than a given threshold; the intergenic distance is smaller than a given threshold. By default, the thresholds are set to be the median value of the values for all segments in the genome or 0.1 if the median value is very low.

If the input is from the GI candidates detected by methods with high recall but low precision, the above rules are changed a bit. An initial non-GI segment with the presence of flanking short repeats and mobility-related genes is also reclassified as a GI

segment. An initial non-GI segment with no less than 60% of the genes being either virulence factors, or phage-related genes, or antibiotic resistance genes or novel genes is considered as a GI segment too. If an initial non-GI segment has phage-related genes and the percentage of phage-related genes and novel genes is no less than 60%, it is included into the GI group as well. Among the initial GI segments, a segment with the following properties is excluded: there are no flanking tRNA genes or mobility-related genes; the percentages of virulence factors, phage-related genes, antibiotic resistance genes and novel genes are all smaller than 20%.

For segments belonging to the GI group, their boundaries are firstly revised according to the positions of flanking tRNA genes or short repeats. By default, we use tRNA genes or short repeats around 1 kb of the endpoints of each segment to refine its boundary. Then, we relocate the obtained segments by the closest genes, requiring that at least half of a gene overlap with a segment. Finally, we merge adjacent segments if the distance between them is less than a threshold (5000 bp by default).

A.1.4 Visualization of predicted GIs

To intuitively display multilayered annotations, we use Circos (Krzywinski et al., 2009), which is effective in showing positional relationships between genomic segments.

For the feature plot, we allocate eight tracks, because there are too many features to display them all in the limited space. The default configuration contains one track for each of the following features: GC content, codon adaptation index, covariance for 4-mers, the percentage of mobility-related genes, the percentage of phage-related genes, the percentage of virulence factors, the percentage of antibiotic resistance genes, and the percentage of novel genes. Users can adjust parameters to see the distribution of other features.

For the comparison plot, users can easily adapt the provided configuration files to show at most nine sources of predictions in a Circos plot.

A.2 Supplementary Results

In this section, we first provide more details about the datasets and metrics used for performance evaluation. Then we provide more evaluation results.

A.2.1 Evaluation approach

Reference datasets

The overall information of 10 selected bacterial complete genomes for evaluation is summarized in Table A.2. For simplicity, we refer to each genome in shorthand, as indicated in the first column.

For CT18, LESB58, and NCTC13129, we used GIs collected in (Lu and Leong, 2016b) as reference. For LESB58, we also included the newly predicted GIs in (Jani et al., 2016). For HI4320, we refined the boundaries of some GIs according to annotations from NCBI and added two GIs predicted by Islander and in literature (Pearson and Mobley, 2007), respectively. For CFT073, BAA894, and DSM12804, we also added one additional GI predicted by Islander, respectively. For Sequi4047, we added two GIs and refined the boundaries of some GIs according to annotations from NCBI.

Table A.2: The general information of 10 complete bacterial genomes for evaluation of GI prediction tools, grouped by taxonomic orders.

Genome	Accession	Size (bp)	GC (%)	#GI (Gene)	GI size (bp)
J2315	NC_011000	3,870,082	66.68	9(339)	358,581
DSM12804	NC_010170	5,287,950	65.48	8(1078)	1,057,881
NCTC13129	NC_002935	2,488,635	53.48	23(432)	473,944
BAA894	NC_009778	4,368,373	56.77	14(429)	355,771
CFT073	NC_004431	5,231,428	50.47	14(934)	819,327
HI4320	NC_010554	4,063,606	38.90	9(452)	420,364
CT18	NC_003198	4,809,037	52.09	19(497)	600,015
CIP105476	NC_010161	2,619,061	38.86	16(745)	729,224
Sequi4047	NC_012471	2,253,793	41.28	8(303)	272,740
LESB58	NC_011770	6,601,757	66.30	23(659)	730,819

In addition, for eight known integrative and conjugative elements (ICEs) (ICEPmiUSA1, ICEPm1, ICESe2, ICE-GI1, ICE-GI2, ICE-GI3, ICE-GI6 and SPI-7), we used boundaries from ICEberg database (Bi et al., 2012), expect for ICE-GI2 and ICE-GI3. Since ICE-GI2 (from 1350129 to 1493558) and ICE-GI3 (from 1493541 to 1595651) are overlapping, we combine them into one, denoted as ICE-GI2(3).

For C-data set, there are positive datasets (GIs) and negative datasets (non-GIs) for nine of these genomes (except *P. aeruginosa* LESB58).

Evaluation metrics

We used eight metrics to evaluate the performance of GI prediction tools, including recall (TPR, sensitivity), precision, F-measure (F1 score), true negative rate (TNR, specificity), overall accuracy (OACC), accuracy (ACC), Matthews correlation coefficient (MCC) and the average of absolute error (ABE) in predicted boundaries (Langille et al., 2008; Lu and Leong, 2016b; Wei et al., 2017). Their definitions are as follows:

$$Precision = \frac{|TP|}{|TP| + |FP|};$$

$$Recall(TPR) = \frac{|TP|}{|TP| + |FN|};$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall};$$

$$TNR = \frac{|TN|}{|TN| + |FP|};$$

$$OACC = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|};$$

$$ACC = \frac{TPR + TNR}{2};$$

$$MCC = \frac{2 * Precision * Recall}{Precision + Recall};$$

$$ABE = |x - x_0|.$$

The evaluations on L-data set are based on the number of protein-coding genes overlapping with a GI candidate. A gene is *predicted* if at least half of its sequence overlaps with a GI. True positives (TPs) are predicted genes overlapping with the reference GIs; false positives (FPs) are predicted genes not overlapping with reference GIs; false negatives (FNs) are genes which overlap with the reference GIs but not the predicted GIs.

The evaluations on C-data set are based on the number of nucleotides overlapping with a GI candidate. TPs are nucleotides included in both the reference GIs and the predicted GIs; FPs are nucleotides in the predicted GIs overlapping with the negative dataset; FNs are nucleotides in the reference GIs but not in the predicted GIs.

For ABE, x is the reference left (right) endpoint of a GI and x_0 is the predicted left (right) endpoint.

A.2.2 Performance evaluation on real biological datasets

Evaluations on L-data set and C-data set

GI-Cluster had better performance than GIHunter on some real biological datasets (Table A.3). For example, GI-Cluster had better precision and accuracy than GIHunter on both L-data set and C-data set of genome CIP105476 and Sequi4047. GI-Cluster also had better recall and accuracy than GIHunter on the L-data set of genome CT18 and LESB58. For genome CT18, although GI-Cluster had lower precision than GIHunter on L-data set, GI-Cluster had higher recall than GIHunter on C-data set.

Table A.3: Comparison of GI-Cluster and GIHunter on several genomes.

Data	Tool	#Gene	#GI	Recall	Precision	F1	ABE(bp)	TNR	OACC	ACC	MCC	C.Recall	C.Precision	C.F1
	GI-Cluster	662	53	0.659	0.742	0.698	6,267	0.871	0.795	0.765	0.545	0.618	0.685	0.650
CIP105476	GIHunter	1351	8	0.962	0.531	0.684	169,053	0.521	0.680	0.742	0.488	0.981	0.482	0.646
	Difference			-0.303	**0.211**	**0.014**	-162,786	**0.350**	**0.141**	**0.023**	**0.057**	-0.363	**0.203**	**0.004**
	GI-Cluster	393	30	0.842	0.649	0.733	4,957	0.919	0.907	0.880	0.686	0.583	0.899	0.707
Sequi4047	GIHunter	862	15	0.987	0.347	0.513	46,801	0.668	0.717	0.828	0.474	0.765	0.630	0.691
	Difference			-0.145	**0.302**	**0.220**	-41,844	**0.251**	**0.082**	**0.052**	**0.212**	-0.182	**0.269**	**0.016**
	GI-Cluster	748	64	0.873	0.580	0.697	4,245	0.913	0.908	0.893	0.664	0.771	0.727	0.748
CT18	GIHunter	608	23	0.827	0.676	0.744	10,394	0.945	0.931	0.886	0.709	0.697	0.727	0.712
	Difference			**0.046**	-0.096	-0.047	-6,149	-0.032	**0.023**	**0.007**	-0.045	**0.074**	0.000	**0.036**
	GI-Cluster	807	82	0.721	0.589	0.648	4,039	0.937	0.913	0.829	0.603			
LESB58	GIHunter	511	16	0.584	0.753	0.658	18,678	0.976	0.932	0.780	0.628			
	Difference			**0.137**	-0.164	-0.010	-14,639	-0.039	**0.020**	**0.049**	-0.025			

C.Recall, C.Precision, and C.F1 represent the recall, precision and F1 computed on C-data set. The other metrics were computed on L-data set.

Appendix B

Supplementary materials for Chapter 3

In this chapter, we provide formal proofs of several theorems and supplementary data for the work presented in Chapter 3.

B.1 Proof of Theorems

In this section, we prove a few theorems used in developing the TCP and CCP algorithm.

B.1.1 Proof of Lema 3.4

Proof. If C contains a leaf x, the root u of C is then a tree node and there is a unique path P from u to x that contains only tree nodes. Consider a path P' from the network root to x. If u does not appear in P', P' and P intersects at a reticulation node in P', as x is in them. This is impossible. Hence P' must contain u, implying that u is visible with respect to x. Similarly, we can prove that u is visible with respect to the child of a reticulation node y if all the parents of y are in C. By definition, C is visible if u is visible.

Conversely, let C be visible with respect to a leaf x and let x be not in C. Since C is exposed, x is the child of a reticulation node r right below C. Assume a parent y of r is not in C. Since C is exposed, y is not below C, as there is a non-trivial component below C otherwise. This implies that any path from the network root to x through y does not contain the root of C, contradicting the visibility assumption on C. This completes the

proof.

B.1.2 Proof of Theorem 3.5.1

Proof. For convenience, we simply set $s = s_G(r)$ and $d = d_G(r)$.

(i). The statement is equivalent to that if N displays G, then $d \leq s$.

Assume N displays G. By definition, a subtree history M of N exists such that that G is obtained from M by contracting nodes of in-degree and out-degree one. Without loss of generality, we assume the root of M is the network root, i.e., $\rho(M) = \rho(N)$.

Since r is visible with respect to ℓ, the unique path $\rho(M)$ to ℓ in M must contain r. $G'(v_{t+1})$ is displayed below r. The proof is complete if $s = t + 1$.

If $s < t + 1$, by the definition of s, $G(s)$ contains a leaf $\bar{\ell} \neq \ell$ such that r is also visible with respect to $\bar{\ell}$. Thus, r is also in the unique path from $\rho(M)$ to $\bar{\ell}$. Thus, as the least common ancestor of ℓ and $\bar{\ell}$, v_s must be mapped to a node u below r in M. Hence, $G'(v_s)$ can be obtained from the substree of M rooted at u by contraction. Hence, $s \leq d$. The proof of the fact (i) is completed.

(ii) Assume $d \leq s$. Note that $G'(v_s)$ is a subtree of $G'(v_d)$.

(**Sufficiency**) Let $N_r = N - C(r) + \ell$ and $G_r = G - G'(v_d) + \ell$.

Assume N_r displays G_r. A subtree history M_r of N_r exists such that G_r can be obtained from M_r by contraction.

On the other hand, since $G'(v_d)$ is displayed below r, a subtree history M'' exists in $[r]_N - \{c(r'), r' \mid r' \in R(N) \text{ s.t. } c'(r) \notin L(G'(v_d))\}$ from which $G'(v_d)$ can be obtained by contraction. Let M' be the tree obtained from M_r by replacing the leaf ℓ with M''.

Clearly, M' is a subtree history of N and G can be obtained from M' by contraction. Hence, N displays G.

(**Necessity**) Assume that a subtree history M' of N exists from which G can be obtained by contraction. If $\rho(M') \neq \rho(N)$, there is a path from $\rho(N)$ to $\rho(M')$ consisting of nodes of in-degree and out-degree one in M'. Without loss of generality, we assume that $\rho(M') = \rho(N)$.

Let v_d correspond to a node $u \in V(M')$. Since v_d is an ancestor of ℓ, u is in the path P from $\rho(N)$ to ℓ. Since r is visible with respect to ℓ, r and its child $c(r)$ are both in the path P. Hence u is either below $c(r)$ or above r, the root of the tree component $C(r)$.

If u is below $c(r)$, by the definition of d, v_{d+1} must correspond to node above $c(r)$ in M'. Let M'_r be the subtree obtained from M' by replacing $M'(c(r))$ with ℓ, where $M'(c(r))$ is the subtree rooted at the unique child $c(r)$ of r. Clearly, M'_r is a subtree history of N_r. In addition, G_r can be obtained from M'_r by contraction. The proof is completed for the case that u is below $c(r)$.

If u is above r and hence strictly above $c(r)$ in M'. Let M'' be the subtree history of $[r]_N - \{c(r'), r' \mid r' \in R(N)$ such that $c'(r) \notin L(G'(v_d))\}$, a subnetwork below r, from which $G'(v_d)$ can be obtained by contraction. Let $P(u, c(r))$ be the path from u to $c(r)$ in M'. We consider:

$$M = M' - M'(u) + P(u, c(r)) + M'',$$

where $M'(u)$ is the subtree of M' rooted at u. Clearly G can be obtained from M by contraction. Since M'' corresponds to $G'(v_d)$, the subtree obtained from $M' - M(u) + P(u, c(r))$ by replacing $c(r)$ with a leaf labeled with ℓ is a subtree history of N_r from which G_r can be obtained by contraction. This finishes the proof.

B.1.3 Proof of Theorem 3.5.2

Proof. Note that $m(N)$ is equal to the largest possible number of times Step 3 is executed on N and any phylogenetic tree. When Step 3 is first executed on N, the two networks N' and N'' are created. For these three networks,

$$m(N) \leq 1 + m(N') + m(N''). \tag{B.1}$$

Let us assume that N' and N'' are created by examining an exposed non-trivial tree component C and a reticulation node r right below C. Since the tree component C is exposed, the child of r is a network leaf, say ℓ. Because C is neither trivial nor visible, another reticulation node r' exists right below C. By the construction of N', C is visible with respect to ℓ in N'. Hence, when the algorithm is called on N', Step 2 should first be executed, eliminating all the reticulations below C including r' simultaneously. Additionally, r has become a tree node in N' and N''.

Let $M(k)$ denote the largest possible number of times Step 3 is executed on any bi-combining reduced network with k reticulation nodes and any phylogenetic tree,

namely:

$$M(k) = \max_{N \in S} \{m(N) \mid |R(N)| = k\},$$

where S is the set of bi-combining reduced networks. Here, clearly, $M(1) \leq 0$ and $M(2) \leq 1$. By Eqn. (5.1), this discussion implies that:

$$M(k) + 1 \leq [M(k-2) + 1] + [M(k-1) + 1].$$

Since the Fibonacci numbers F_k satisfy $F_0 = 1$ and $F_1 = 1$ and $F_k = F_{k-1} + F_{k-2}$ ($k \geq 2$), we have the following inequality:

$$M(k) + 1 \leq F_k \leq \frac{1}{\sqrt{5}} \left(\frac{1}{2}(1 + \sqrt{5}) \right)^k + \frac{1}{2},$$

implying that:

$$b(N) \leq \lceil \log_2 M(|R(N)|) \rceil \leq \log_2 \left(\frac{1}{2}(1 + \sqrt{5}) \right) \times |R(N)|.$$

B.1.4 Proof of Theorem 3.6.1

Proof. (i). Assume that B' is a soft cluster in N'_a. Suppose B' is a soft cluster of a node u. Here, ℓ is a leaf below u. If we re-expand ℓ into $[\rho(C_r)]_{N_a}$ to obtain N, the soft cluster of u will become $(B \backslash \hat{B}) \cup \hat{B}$, namely B. Hence, B is a soft cluster in N.

Assume that B is a soft cluster in N. Suppose B is a soft cluster of a node v in a tree T, where $T = N - E$ and $E \subset E(N)$. Because $\hat{B} \subset B$, B contains a leaf $\bar{\ell}$ which is not below $\rho(C_r)$. As $\bar{\ell}$ is below v in T, v must be above $\rho(C_r)$ in T.

Let $r' \in CR(C_r)$ and $c(r') \in B$. Here, r' has at least one parent in C_r, and $c(r')$ is a leaf below v in T since C_r is exposed. Let $(p_{r'}, r') \in E(N)$ such that $p_{r'} \in C_r$, and $(p'_{r'}, r') \in E$.

Let $T' = T - \{(p'_{r'}, r')\} + \{(p_{r'}, r')\}$. $\hat{B}$ is then the cluster of $\rho(C_r)$ in T' and B is the cluster of v. After replacing $[\rho(C_r)]_{N_a}$ with ℓ to get N'_a, the cluster of v is $(B \cup \ell) \backslash \hat{B}$. Hence, B' is a soft cluster in N'_a.

(ii) Assume that B is a soft cluster in N'_b. Since N'_b is a subnetwork of N, B is a soft cluster in N.

Assume that B is a soft cluster in N. Suppose B is a soft cluster of a node v in a tree

T, where $T = N - E$ and $E \subset E(N)$.

Since B is not a soft cluster of a node in C_r, $L_r \cap B = \emptyset$ and $\rho(C_r)$ is visible on leaves in L_r, $\rho(C_r)$ is not below v in T and v is not below $\rho(C_r)$ in T either.

Let $r' \in CR(C_r)$; r' then has at least one parent $p_{r'}$ in C_r.

If $c(r') \in B$, $c(r')$ is a leaf below v in T since C_r is exposed. Then $p'_{r'}$ is not in C_r.

If $c(r') \notin B$, $p'_{r'}$ may or may not be in C_r. Suppose $(p'_{r'}, r') \in E$. If $p'_{r'} \notin C_r$, we can replace $(p'_{r'}, r')$ with $(p_{r'}, r')$ and define $T' = T - \{(p'_{r'}, r')\} + \{(p_{r'}, r')\}$. The cluster of v in T' is then the same as the cluster of v in T. After replacing $[\rho(C_r)]_{N_b}$ with ℓ to get N'_b, the cluster of v is still B. Hence, B is a soft cluster in N'_b.

B.2 Phylogenetic networks on real biological datasets

In this section, we show several phylogenetic network used to show the application of our programs.

B.2.1 A phylogenetic network over seven fungi species

Figure B.1 shows an ancestral recombination graph over seven fungi species, which was reconstructed to study the phylogenetic relationships among the M2 double-stranded RNA in the Rhizoctonia species complex (Charlton et al., 2008).

B.2.2 Three phylogenetic networks reconstructed from a grass dataset

Figure B.2, Figure B.3, and Figure B.4 show three different kinds of phylogenetic networks, which were reconstructed from five gene trees (*ITS*, *ndhF*, *phyB*, *rbcL*, *rpoC2*) of a grass dataset (van Iersel et al., 2010a) by three different algorithms (Huson and Rupp, 2008; Huson et al., 2009; Wu, 2010), respectively.

B.2.3 Two phylogenetic networks over six mosquito species

Figure B.5 shows two phylogenetic networks reconstructed over six mosquito species, which were reported in (Fontaine et al., 2015) and (Wen et al., 2016), respectively.

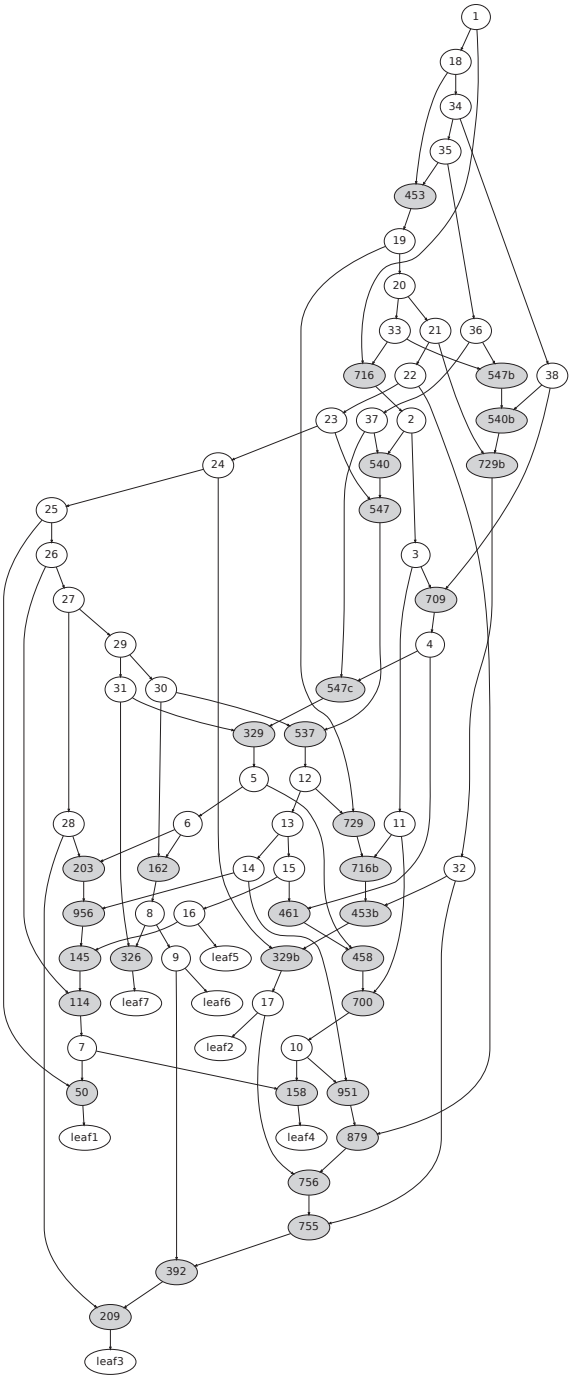

Figure B.1: A bicombining network redrawn from (Charlton et al., 2008).

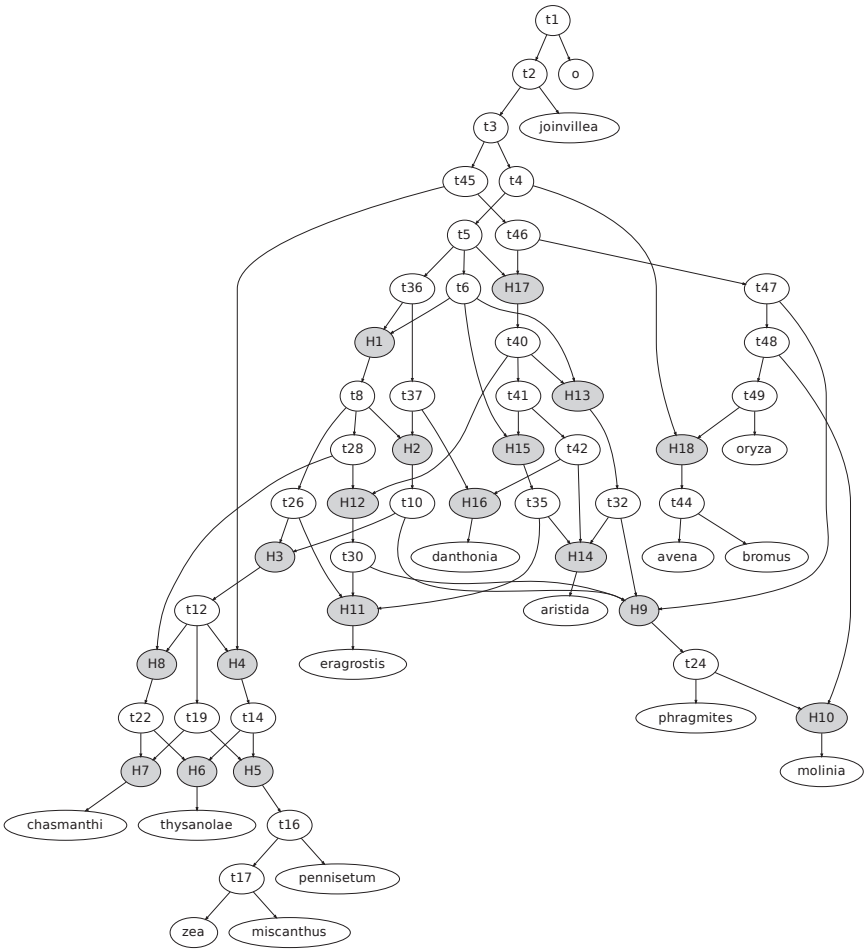

Figure B.2: A cluster network reconstructed by Dendroscope (Huson and Scornavacca, 2012) from five gene trees of a grass dataset.

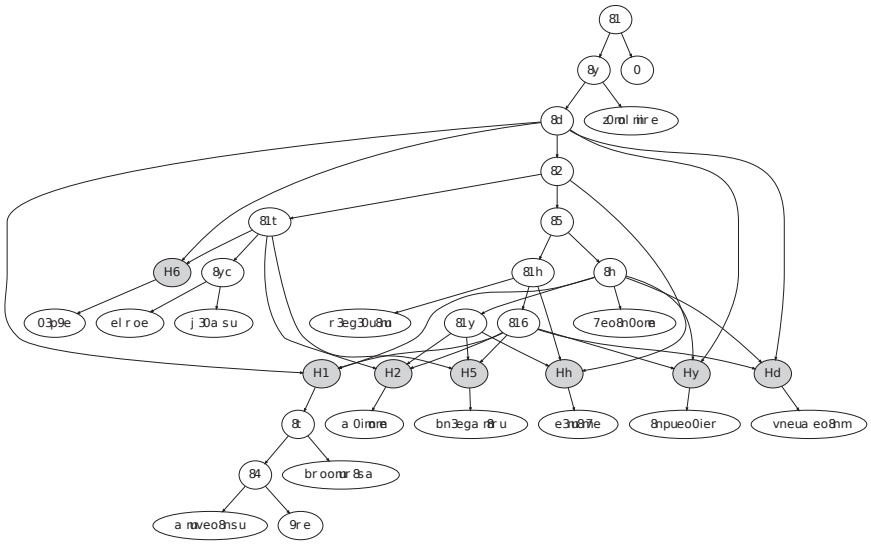

Figure B.3: A galled network reconstructed by Dendroscope (Huson and Scornavacca, 2012) from five gene trees of a grass dataset.

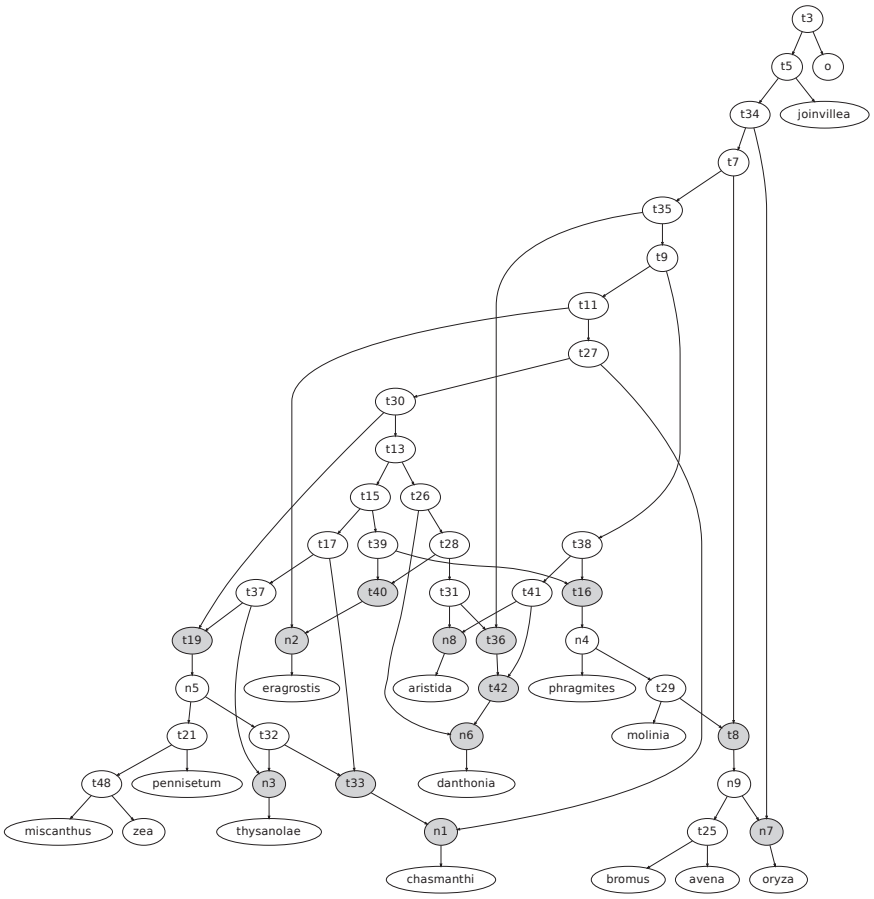

Figure B.4: A reticulate network reconstructed by PIRN (Wu, 2010) from five gene trees of a grass dataset.

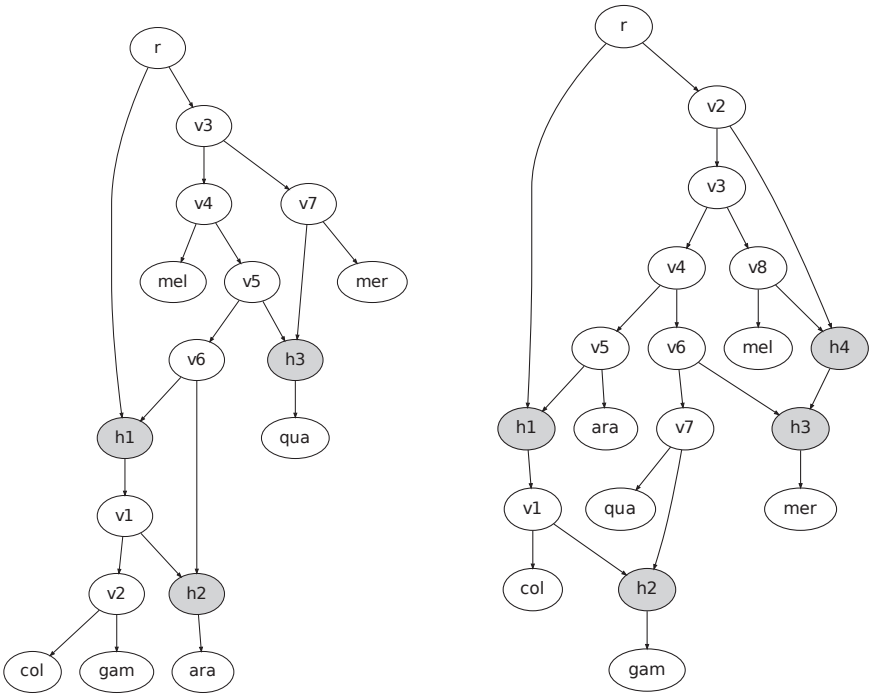

Figure B.5: Left panel: A phylogenetic network redrawn from Figure 1C in (Fontaine et al., 2015). Right panel: A phylogenetic network redrawn from Figure 6 in (Wen et al., 2016).