THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE DE BORDEAUX

SPÉCIALITÉ INFORMATIQUE

Par Thomas HUME

Practical and theoretical approaches for module analysis of protein-protein interaction networks

Sous la direction de Macha NIKOLSKI

Soutenue le XXX

Membres du jury :

Francois PELLEGRINI	Professeur des universités	Président
Macha NIKOLSKI	Directrice de recherche	Directrice
Stéphane VIALETTE	Directeur de recherche	Rapporteur
Jean-Michel COUVREUR	Professeur des universités	Rapporteur

Pour tous ceux qui n'ont cessé de croire en moi.

Résumé

Cette these

Mots clés: Bioinformatique, Optimization combinatoire, Maximum-Weight Connected Subgraph, Recherche de modules biologiques

Abstract

This thesis

Keywords: Bioinformatics, Combinatorial optimization, Maximum-Weight Connected Subgraph, Biological module discovery

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Where first biologists looked at independant *living organisms* in their environments and interacted with them at the human scale, modern biologists have much more complex views of and interactions with the living world. For example modern molecular biologists looks into the invisible and analyses cellular processes. And most –if not all– modern biologist are intersted as much with the subject of their study as with its surrounding that it connect to throught its interactions.

No definition of life is unequivocal, but most agree that in order to be considered a living organism one has be able to grow and adapt to external stimuli in order to maintain its homeostasis (maintaining a stable state by means of one or many internal processes) and be able to replicate. According to this definition the most basic unit of life is the *cell*. Cells can either be organisms in themselves, bacteria being the perfect example, or the basis for complex multicellular organisms, for example animals and land plants.

Cells are made of many biomolecules in constant interactions. These molecules form many structures, including a membrane and every other mechanisms that are required for the cell to function as a living organism. Two fundamental kinds of biomolecule are the nucleic acid chains, such as deoxyribonucleic acid (DNA) and ribonucleic acid (RNA), and amino acid chains, such as proteins.

Nucleic acid molecules are strings of nucleotides that encode and transmit informations.

Proteins are complex macromolecules that constitute the mechanisms through which the cell function. There exist many different types of proteins, each doing one or more specific tasks. Proteins rarely act alone, and in order to do their work, they interact with other types of proteins, and thus form protein-protein interaction networks.

At any given time, a cell contains a large number of proteins for each kind. By the random nature of biomolecules interactions, the relative concentration for each of these protein types guide the functioning of the cell overall.

It is difficult to directly measure the concentrations for each type of proteins. However, modern microarray or sequencing techniques allow the measure of the precursor of the proteins: messenger RNA. Hence, an histogram of these proxies of the proteins concentration can be made: *expression profiles*.

Expression profiles are like the molecular signatures of the cells. Indeed, the proteins that are present in a skin cell are very different from those in a blood cell.

These molecular signatures allow the automated classification of different cells by type.

Differential analysis is a technique where the expression profiles for a control cell line are compared to the one from a condition cell line, and the differences are extracted. It allows the detection of slight variations in cell functioning within the same cell type.

There exist many computational methods for the analysis of these expression profiles. In this thesis we contribute to the *active module discovery* problem, where the expression levels for each proteins are used to extract proteins of interests. These are proteins that are highly differentiated between the control and the condition.

Many statistical approaches have been proposed to detect these sets of proteins that are important in the phenomenon under study. One shortcoming of many of these methods is that they occasionally find sets of proteins that have little in common for the cellular processes, and biological interpretation of these results become difficult, if even significant.

Here, we take the *connected module* approach, where modules are proteins of interest with the added constraint that they have to physically interact. A few techniques have been proposed that use the protein-protein interactions networks as graph structures. Computationally, these techniques are all connected to the *connected subgraphs problems*, such as the PRIZE-COLLECTING STEINER TREE (PCST) problem or the MAXIMUM-WEIGHT CONNECTED SUBGRAPH (MWCS) problem.

The main contribution of this thesis is the introduction of a model for the detection of active connected modules across two species. We are looking for active connected modules that are similar in composition between two species. The similarity is a flexible ratio of similar proteins over all proteins in the solution.

We present a mixed-integer programming formulation of our model, and propose a branch-and-cut algorithm to solve it to optimality in reasonable run time on practical instances.

We then analyse our model from a complexity standpoint. We demonstrate that the problem is APX-hard in the general case. We also show that it can be solvable in polynomial time and fixed parameter tractable polynomial time for some categories of input.

1.1 Biological entities and their analysis

We first briefly overview the types of information that is available to study biological systems. This information can be described following the *central dogma of molecular biology* (Crick 1970): genetic (DNA) sequences, transcribed RNA sequences and finally protein sequences. Raw DNA sequences are strings of the four letters, called *bases*, comprising genes. Different species can have different numbers of genes within the genome of the organism. For prokaryotes, each gene is typically $1,000$ to 2500 bases long (Xu et al. 2006). The GenBank repository of nucleic acid sequences (`http://www.ncbi.nlm.nih.gov/genbank/`) currently holds a total of approximately 202 billion bases in 188 million entries[1]. At the next level are protein sequences that correspond to real gene products within the cells and are strings of 21 amino acid-letters. At present, the UniProtKB protein sequence database (`http://www.uniprot.org/`) contains about 500 thousands reviewed protein sequences[2], a typical bacterial protein being approximately 300 amino acids long.

The central dogma states that DNA makes RNA, and RNA makes proteins. At each step, a cell translates the information between the different alphabets. That is, DNA sequences are translated into RNA sequences at the first step, and RNA sequences are translated into protein sequences at the second step, see fig. 1.1.

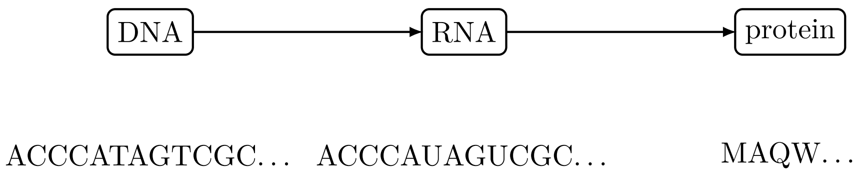

Figure 1.1: **Central Dogma of Molecular Biology.** Below each structure type are examples using 12 dummy DNA characters, the corresponding messenger RNA sequence, and the protein sequence.

[1]Release 210.0, October 15 2015
[2]Release 2015_10

1.1.1 Entities

1.1.1.1 Genes

A genome is typically a set of sequences, corresponding to chromosomes, over the 4-letter alphabet: cytosine ('C'), guanine ('G'), adenine ('A'), and thymine ('T'). The set of genes of an organism encodes the instructions for a cell to perform its functions. In order to study how cells perform their functions, it is first necessary to identify genes. Process of identifying regions within the genome that encode genes is called *gene finding*, see for review (Stormo 2000). The common principle behind gene prediction methods is to distinguish between protein-coding regions (genes) and non-coding regions, based on their statistical properties.

```
CGGGGTTAAATCCACTACCCTCTCCCCACGCACTC
TAGTAATTACTCTATTTCCACGTCATGTTTCCGGG
```

Figure 1.2: 70 nucleobases from the human (hsa) STAT3 gene.

Based on the observation that given two species A and B, there exist gene versions $a \in A$ and $b \in B$ that are similar in sequence and have related functions, the parsimony principle suggests that they must have been inherited from the common ancestor. Genes that have a *similar sequence* are said to be *homologs*. Among homologs there are two main subgoups: orthologs and paralogs. *Orthologs* are homologous genes that evolved from the same ancestral gene in the last common ancestor of the compared species. For example, for a and b to be orthologs, they must have descended from an ancestral gene c in C, last common ancestor of A and B. On the other hand, *paralogs* are homologous genes, that have evolved by duplication. For example, $a' \in A$ having a similar sequence to a (and b), might have descended from the duplication of a that has happened after the ancestors of A and B have taken different evolutionary paths. For further discussion of orthology, see (Makarova et al. 2007; Kuzniar et al. 2008).

```
CCGGGTTATAGCCACTATTCTC-CCCCACGCAATC
TAGTAATTACTCTATTTCCACGTCATATTTCCGGG
```

Figure 1.3: 70 nucleobases from the mouse (mus) Stat3 gene. Bases highlighted in red indicate a variation from the corresponding bases in the human gene, '-' indicates that the base is abscent in the mouse gene.

1.1.1.2 RNA

Like DNA, RNA is a nucleic acid sequence, but with some differences. For example, RNA molecules are single-stranded, while DNA is double-stranded. Also DNA and RNA differ slightly at the nucleotide level, in particular there is a 'U' (uracil) instead of 'T'. Most importantly, RNA sequences - called mRNA for messenger RNA - correspond to the coding regions. The non-coding regions are excised during the transcription process.

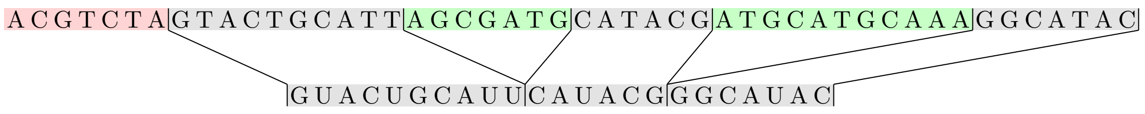

Figure 1.4: **Gene transcription.** After the removal of the *primer* region, which indicates the start of a gene, each *exon* is translated by replacing every Thymine base by an Uracil base.

Different species have their own sets of RNA sequences at any given time. In particular, they respond to environmental stimuli by varying the level of RNA molecules that are present at a given time, process that is called *gene expression*. The set of RNA molecules of an organism is called *transcriptome*.

1.1.1.3 Proteins

Translation, which is the second step in gene expression, "reads" the RNA sequences and translates (according to the genetic code) them into amino acid sequences, that is sequences over the alphabet of 22 amino-acid letters. Each group of three bases in the mRNA constitutes a *codon*, and each codon defines a particular amino acid. This is why it is called a triplet code. Consequently, the RNA sequence is used as a template to build sequences of amino acids that constitute proteins.

1.1.2 Biology as data science

Since the completion of the Human Genome Project in 2003, thousands of species have seen their genome fully sequenced (Reddy et al. 2014). Nowadays, Regalado (2014) estimated that our sequencing capacity exceeds 35 Petabases (or approximately 250,000 human genomes) per year. Looking into the future, with further advancement of high-throughput sequencing technologies, the prospect of seeing more than 1 billion genomes sequenced in the next 20 years is realistic (Schatz 2015).

Genetic information is only one type of data. In the last 20 years, many techniques have been used to look deeper into cellular functions. Microarrays, RNA-sequencing, high precision microscopy, and mass spectrometry, all allow automated collection of observations of the cellular developments and regulation processes.

With the collection of these volumes of data, a number of challenges arise, among which (i) the collection of data into accessible repositories, and (ii) the analysis and interpretation of the underlying knowledge. Analysis requirements of biological data have brought biology within the scope of data science, towards what is called quantitative biology.

1.1.2.1 Biological databases

1.1.3 High throughput experiments

Microarrays is an experimental technique that can analyze the expression of many genes simultaneously and in an efficient manner (Smyth et al. 2005; Sealfon and Chu 2011). Microarrays can be used to perform different measurements, among others are the detection and measurement of gene expression at the mRNA or protein level, detection of mutations and location of chromosomal changes. A microarray is a collection of spots attached to a solid surface. Each spot – corresponding to one gene – on a microarray contains multiple identical strands of DNA and that are unique to this spot. Thousands of spots are arranged in rows and columns on a solid surface. The resulting data is the intensity of fluorescence for each spot that represents the level of expression of the corresponding gene.

Consequently, data produced by microarray technique requires a certain amount of image processing

This ability to quantify many molecular targets in parallel was an important step towards the understanding of complex biological processes. However, microarray data is subject to variability and noise. A number of normalization techniques have been developed to deal with these artifacts (REF, REF). However, normalization can not adjust for the major flaw of microarrays, that is batch effect. Consequently, combining batches of data produced by microarrays can potentially lead to erroneous results (W. E. Johnson et al. 2007).

1.1.3.1 NGS

1.1.3.2 RNA-Seq

Comparison microarray - NGS (A. C. Richard et al. 2014)

1.2 Computation and complexity classes

In this section we introduce the fundamental concepts and terminology that will be used at length through this work.

A *computational problem* is a mathematical question that is susceptible to being solved by a computer. Even though any specific mathematical question is potentialy a computational problem, it is usual for computational problems to regroup a set of similar questions. For exemple, alhough *"Is 42 a prime number?"* is a mathematical question that can be solved by a computer, *"Given* *a number n, is n a prime number?"* is really the computational problem here. Every possible input to the problem form in *instance* of said problem, for which there might very well be an infinite number, and each of which having its own specific *solution*. In the aforementioned exemple there are an infinite number of instances, one for every n, each with its own *yes* or *no* answer. By convention, "computational" is very often omited, hence the previous problem would be called for exemple *"the primality test* problem*"*.

Theoretical computer science is an inclusive field of both mathematics and general computer sciences that studies many of the theoretical aspects of computation. In this field any computation is expressed for a specific *model of computation*. These models of computation are the basis on which one can formally express computations with the use of *algorithms*.

Computational problems are solved with algorithms, series of well-defined sequential or parallel operations that can be executed by the model for which it was designed. Formally, an algorithm unambiguously define an *effective method* in the *Turing-Church* sense, that is a series of finite number of mechanical steps that the model can execute and that always produce a correct answer (for the problems it is intended to solve).

Serial algorithms, containing only sequential operations, form the vast majority of known algorithms and are usually identified simply as "algorithms". On the other hand, algorithms containing parallel sequences are aptly named *parallel algorithms*[3], and although their *fork* points (points of separation) and *join* points (points of junction) are well defined, their analysis comes with its own share of idiosyncrasies. Parallel algorithms are the foundation of modern optimization problem solvers. The authors recommend (Jada 1992) for a thorough introduction to this subject. Furthermore the sequential transitions need not be deterministic: the integration of a random factors is the basis for the *randomized algorithms*. Ramdomized algorithms are also very important in optimization contexts, particularly for approximation schemes with the use of *randomized rounding*. We recommend (Williamson and Shmoys 2011, Chapters 5 and 6) for a clear and comprehensive treatment of the topic.

Even though the Mixed-Integer Programs (MIP) Solvers that we use in this work implement inhenrently parallel and optionally randomized algorithms (see section 1.3.1.2) to solve our instances, the study of said algorithms and the associated models of computation is well beyond the scope of this manuscript since in the general case (including ours) a parallel and/or randomized algorithm does not change the inherent caracteristics (that we investigate) of the problem that it solves.

[3]Not to be confused with *concurrent algorithms*, algorithms for which concurrent steps can execute in undefined order, even though the two are often closely related.

We thus have on one hand computational problems and on the other hand algorithms (for specific models of computations) that solve their problems[4] by computing their solution. For a given problem, different algorithms, running on different models of computation, can exhibit very different caracteristics, when they exists at all. The study of models of computations and their expressiveness in term of algorithms —and hence of the problems that they can solve, and with which caracteristics— is part of the *theory of computation*. For a general introduction to the subject we recommend (Sipser 2012). The problems that we study in this manuscript are deeply driven by their practical applications. Hence from now on, for the sake of simplification and unless specified otherwise, we consider only one theoretical model of computations: *Turing machines*. These Turing machines are an abstract model that allows us to theoretically ground the research of algorithms' properties and are the foundation of the Turing-Church's theory of effective methods.

The fact that there exists a potentially large number of algorithms that solve a given computational problem certainly does not mean that they are all equal, quite the opposite. There are many caracteristics that can be used to compare and classify algorithms, the most important ones being the *complexity measures* that satisfy the *Blum complexity axioms*. Namely, the two most significant measures are the *execution time complexity* (also named *run time*, *runtime*, or simply *time complexity*) and the *memory space complexity* (or simply *space complexity*) measures, informally the minimum number of steps and the minimal amount of storage that an algorithm must use to successfully compute the solution to the problem at hand. These measures provide the means to quantify the necessary resources that an algorithm requires for its computation to complete, which gives us an element to express the *efficiency* of the algorithm. When unspecified "complexity" usually means "execution time complexity".

Since algorithms solve computational problems, they have to accept as many different input as there exist instances of the problem. The *analysis of algorithm* is a field that provides tools that make possible the study of algorithms under any of their valid inputs, enabling the expression of complexity measures as functions of the inputs. In the *"the primality test* problem*"* previously mentioned, the complexity of an algorithm solving this problem would therefore be a function of n, the abstract number that is tested and which is the input of the algorithm. Since the algorithm and its complexity hold for any arbitrary large number n the complexity is generayly measured asymptotically. To denote this asymptotic representation of complexity, we use the *Big-O asymptotic notation* (O). Formally, the notation means

$$f(n) \in O(g(n)) \quad \iff \quad \exists M, n_0 \quad \text{s.t.} \quad |f(n)| \leq |M \times g(n)| \quad \forall\, n \geq n_0.$$

[4]Technically an algorithm cannot solve a problem since it only define a series of step, *executing* (or *running*) the algorithm solves the problems; nevertheless it is extremely common for the expression to be used and this manuscript no different.

Thus the Big-O function designate an asymptotic upper bound for the function and allows for a notation that represents its growth rate, also known as the *order (of growth) of the function.* Informally, it means that the term with highest growth rate in the function $f(n)$, stripped of its factor, is lesser or equal than $g(n)$ asymptotically (usually equal since we want the tightest bound possible). Thereby, constant growth rate is denoted $O(1)$, logarithmic growth rate: $O(lg(n))$, linear growth rate: $O(n)$, quadratic growth rate: $O(n^2)$, etc. When unspecified, "complexity" usually means "worst case asymptotic complexity" in opposition to "exact", "average-case" or "best-case complexity" which use are less common in theoretical contexts, because mostly dependent on the details of the execution model[5] and/or valid only for some restricted set of inputs[6].

Note that the notation $O(g(n))$ actually defines the set of all functions such that the definition holds, that is of all functions $f(n) \in O(g(n))$. And since clearly $O(1) \subset O(lg(n)) \subset O(n) \subset O(n^2) \subset \ldots$, this notation provides a natural classification of algorithms into ever more inclusive classes of increasing complexity. Although in the general case not all such classes hold an inclusion relationship with each other, hierarchical classification of classes is central to complexity analysis of computational problems. Classes for a specific resource *res* (e.g. time) are defined by the set of problems that can be solved in $O(f(n))$ in regard to *res* for any input of size n. Even though not stricly correct, we often denote such a class the *$f(n)$ res class*, and say that an algorithm is *in (the order of) $f(x)$ res*. *Computational complexity theory* (oftentimes simply *complexity theory*) is the field of knowledge that uses the complexity measures to classify problems into such complexity classes.

Where the analysis of algorithms study the fundamental caracteristics of algorithms, complexity theory is concerned with the study of the problems themselves, outlined by all conceivable algorithms that can solve it. In practice when all the knowledge that we have about a problem is a set of algorithms that can solve it, we consider a problem to be at most as difficult (that is: complex) as the "best" known algorithm that solves the problem, in other words the one which complexity is in the most restrictive complexity class (for a given resource). For exemple, the fastest known algorithm that can solve the primality test problem is Lenstra Jr and Pomerance (2002)'s and is in $O\Big((\log n)^6 \times (2 + \log(\log n))^c\Big)$ time, for some real valued constant c; the primality test problem is thereby in this complexity class. Even though some

[5]E.g. $O(1)$ multiplication when the result holds inside machine registers against $O\big(n^{\lg 3}\big)$ for arbitrary large n-digits number with the Karatsuba multiplication algorithm (Karatsuba and Ofman 1963).

[6]E.g. the insertion sort which for an n elements list takes $O(n)$ when the list is already sorted, $O(kn)$ time to sort the list when it contains at most k invertions, and $O(n^2)$ time in the general (worst) case.

conjecture suggest that there may be, it is not known if a faster algorithm actually exists for this problem, in other word it is not know if there exists a more restrictive complexity class that includes the primality test problem.

Oftentimes, the best technique to classify problems is to analyse how one problem compares to a well known other, using a technique called *problem reduction*. Being able to reduce a problem A to a problem B means that there exists a method capable of transforming any instances of A into instances of B, and is noted $A \leq B$. Technically, a reduction from A to B allows any algorithm that solve B to also solve A. Provided that the reduction's algorithm does not take any more resources than the algorithm that solves B, it means that A cannot be harder to solve than B, since any of its instances can be reduced to B. Conversely, if we know that A is a difficult problem (we know that some of its instances are difficult to solve), being able to reduce all of its instances to any of problem B proves that B is at least as difficult as A, since we know that some of its instances are indeed difficult from the reduction of the difficult ones from A.

Most often, problem reductions compare problems of same *problem type*. There exist many types of problems: function problems (which informally are problems that ask $y = f(x)$ given x), counting problems (which are problems for which we count the number of answers), optimization problems (which are problems that ask which is the *best* solution in a set of candidates, to be formally introduced in section 1.3), etc. The most commonly used in complexity theory are *decision problems*, problems that ask a binary "yes/no" question.

Decision problems are often central to complexity theory because the two most prevalent complexity classes, P and NP, are formally defined for these problems. Furthermore most problems types can be reasonably converted to decision problems (and often the other way aroud), see section 1.3.1 for an application of this principle to combinatorial optimization problems. The first class, P, includes all decision problems that can be solved in *polynomial time* by a *deterministic Turing machine*, and the second class, NP, all decision problems that can be solved in *polynomial time* by a *nondeterministic Turing machine*. Since the set of nondeterministic choices includes all deterministic choices, clearly $P \subseteq NP$.

There exist many other complexity classes, sometimes even defined for only one or a very small set of specific problems. For an up to date compendium of complexity classes we recommend the list maintained by Aaronson et al. (2016).

A problem p is said to be *hard* for some complexity class C, or C*-hard*, if every problem in C can be reduced to p using a reasonable volume of resources. Most often, a reasonable volume of resources amount to a polynomial quantity, and since the most restrictive resource is the run time[7], *polynomial-time reduction* are most commonly used. Thus, the set of C*-hard* problems is the set of all problems that are harder or equally hard as the most difficult problems in C.

A problem p is said to be *complete* for some complexity class C, or C*-complete*, if p is in C and p is C*-hard.* Thus, the set of C*-complete* problems is the set of all problems that are all equally the hardests to solve in C. Since it is known that any problem in C can be reduced to any one problem $p \in C$*-complete*, and considering that being able to reduce p to a problem q is equivalent to being able to reduce any problem in C to q, the set of C*-complete* problems is very often used to prove C*-hardness* (of problem q).

1.3 Optimization problems

Mathematical optimization problems are problems where we search for the *optimal element* (maximal or minimal depending on the problem) of a real-valued *objective function*, subject to some constraints over its domain. It is very rare that only the value of the optimal element is required, and most often it is the *valuation of the input* that leads to the optimal element, that is the values of input parameters of the function, that we really require. Formally, an optimization problem have the following form[8]:

> Given an *objective function* $f\colon D_f \to \mathbb{R}$ that we want to minimize,
> given the *choice set* $CS \subseteq D_f$,
> find an element $\mathit{opt} \in CS$ such that $f(\mathit{opt}) \leq f(x) \quad \forall x \in CS$.

Note that by convention this standard form is for minimization problems; maximization problems are strictly equivalent since, for any given maximization problem, negating the objective function efficitively makes it a minimization problem. Depending on the context, the objective function is also named the *cost function*, cost that we usually want to minimize, or the *utility* (or *profit*) *function*, utility (or profit) that we usually want to maximize.

When the choice set is simple enought or clearly defined, the previous problem notation is oftentimes shortened to the smaller (but much less explicit) version $\min\limits_{x\in CS} f(x)$, or $\arg\min\limits_{x\in CS} f(x)$ when the valuation of x is explicitly required. These notations are most often used in statistical optimization fields, such are machine learning, where optimization problems hold a central place; and such use we indeed apply in section 3.2.2 (albeit in a maximization form) where

[7]In comparison to space, since any amount of space that a deterministic algorithm requires first needs to be run through, a deterministic algorithm is always at most as fast as the amount of memory that it requires.

[8]This is the form predominantly used in strictly mathematical situations.

the parameters of a *mixture model* are optimized by a *maximum-likelihood estimation*.

In most practical applications, however, the choice set rarely possess a straightforward definition, and is instead described by a set of *constraints*, that is a set of equalities and inequalities such that $\forall x \in D_f$, $x \in CS$ if and only if x satisfies them all. In regard to the objective function, all points $x \in D_f$ are *potential solutions* and constitute the *search space* of the optimization problem, all $x \in CS$ are *candidate solutions* (or *feasible solutions*) to the problem, and all points $x \in D_f$ s.t. $x \notin CS$, that is points of the search space where at least one constraint is *violated*, are the *infeasible solutions*. The **c**hoice **s**et CS is thus the set of all **c**andidate **s**olutions. Given a set of constraints, an optimization problem is formally defined as such[9]:

$$\begin{array}{rll} \min & f(x) & \\ \text{subject to} & g_i(x) \leq G_i & \forall i \in [1, m] \\ & h_j(x) = H_j & \forall j \in [1, n] \end{array}$$

$f(x)$ is the objective function, $\bigcup_{i\in[1,m]} g_i$ is the set of all inequality constraints, and $\bigcup_{j\in[1,n]} h_j$ the set of all equality constraints, and the *subject to* is often abbreviated as simply *s.t.*

Many optimization problems can give the impression of being relatively simple, particularly the ones with a finite choice set. Actually, methodically enumerating all possible inputs of the function that respect the constraints and computing their associated value, then selecting the optimal one from them all, can appear as a perfectly valid method for most optimization problems. And indeed, such a technique can be the basis of a valid algorithm, although it is not very efficient in most general. These algorithms are an application of the *exhaustive search* (or *brute-force search*) methods. However, the cost of these technique is proportional to the number of candidates solutions to test for. In practice the search space is often huge in regard to the size of the input, with many common search spaces (of common optimization problems) growing respectively at an exponential rate; in some cases the search space is not even enumerable. These special cases where the search space is finite constitute the very specific topic of *combinatorial optimization* and will be treated in details in section 1.3.1. In general, the search space is not necessarily finite nor even enumerable, and unless the objective function and the constraints conform to some specific restrictions, the search space may prove to be very hard to naively explore in practice.

[9]This is the *standard form* predominantly employed in computational contexts and the one we will mostly use from now on.

The effective size of the search space is often compounded by the fact that optimization problems are essentially expressed in high-dimensional spaces. With high-dimensional problems the x that we optimize over becomes a vector, oftentimes noted $\mathbf{x}$, and each component of the vector is said a variable of the problem. In this context, constraints not only describe restrictions to the set of available candidate solutions, but also interdependancies between these variables. These interdependancies define the *Pareto optimality* of the problem, where the variables are valuated such that any one cannot be improved without making any other (and the objective function as a whole) worse. Typically, the Pareto optimality is often applied when the objective function is a combination of multiple criteria.

When multiple objective functions are combined, that is when multiple criteria need to be optimized at the same time simultaneously, the problem is said a *multi-objective optimization*, or *Pareto optimization.* Since in the general case (and very often in practice) a single solution cannot optimize all criteria at the same time, trade-offs have to be made, and some objectives must be optimized in favor of others. Specifically, some ratio $\alpha \in [0, 1]$ have to be selected such that, for some objective functions $g(\mathbf{x})$ and $h(\mathbf{x})$, the combined objective $f(\mathbf{x}) = \alpha g(\mathbf{x}) + (1 - \alpha)h(\mathbf{x})$ is optimized. The generalization to k objectives does not scale well, and in practice it is very difficult to select a good trade-off (or trade-offs for $k > 2$). We thus define the *Pareto front* as the (most often infinite) set of all vectors $\mathbf{x}$ that are optimal for some value of α. Finding the Pareto front of a problem is very similar to the *Skyline query* problem (or *Skyline operator*), and Rekatsinas et al. (2015) gave a recent overview of the difficulties involved. In many cases, multi-objective optimization is problematic either because trade-offs have to be selected beforehand without a well defined models for the consequences, or because computing the Pareto front (or a part) understandably remains a computationally intensive task to this day. In chapter 3 we present a technique that we use to express an inherently multi-criteria problem into a constrained single-objective problem, therefore preventing some of the aforementioned problems and allowing for an easier interpretation of our results.

Linear programs *Linear programs* are optimization problems where the objective function and the inequality and equality constraints are linear. Each inequality constraint can be seen as a hyperplane, that is a d-dimensional generalisation of a 1-d point, of a 2-d line or of a 3-d plane. Hence, each inequality constraint effectively reduce the choice set by requiring that the solution is in the half-space (either on the greater-than or lesser-than side) defined by its geometric equivalence. As a result the combination of all inequality constraints produce a polytope, a d-dimentional generalisation of a 2-d polygon and 3-d polyhedron. Given the nature of the inequalities and since the search space is defined by the intersection of many half-spaces, it is either empty (no point can respect all constraints at the same time) or convex. Furthermore, the objective function effectively defines a continuous scalar field and its gradient defines the direction of the optimization. Finding the optimal value of the problem is then equivalent to finding the points of the polyhedron where the objective function is optimal. Since the objective function is linear, either there is one unique optimum on a vertice of the polyhedron, or there as infinitely many of them that form an edge or face of the polyhedron. See fig. 1.5 for a visual representation in 2-dimensions of a simple linear problem. Equality contraints are not strictly necessary since they can always be replaced by two opposite inequalities ($h(x) = H$ becomes the two constraints $h(x) \leq H$ and $-h(x) \leq -H$). Geometrically, this describe a d'-dimensional hyperplane, with $d' < d$ depending on the variables used in the constraint, effectively reducing the search space dimension. Since a problem with n variables can be represented in a n-dimentional space, the number of variables of a problem and its dimension are often used synonymously, even though formally the dimension of the problem $d \leq n$ since the variables do not define orthogonal vectors in general.

In 1947 Dantzig proposed the first method to solve linear program: the *simplex algorithm*. Geometrically, the algorithm moves from one vertex to another connected vertex until it finds one where it detect that all other connected vertice are suboptimal (or equal); and since the search space is a convex polytope this point is provably optimal. Even though Klee and Minty (1970) showed that there exists pathological instances for which the algorithm run in exponential-time relative to the number of variables[10], it remains used in practice even nowadays. Furthermore, many modern techniques are grounded on the geometrical representations that the algorithm introduced. Khachiyan (1980) proposed the first polynomial-time algorithm to solve linear programming problems, the *ellipsoid method*, which was an important theoretical result, but with very few practical applications due to high exponents. Karmarkar (1984) introduced the most important result in the field yet, a polynomial-time *interior point* algorithm that is faster than the simplex on most problems.

Convex optimization is the generalization of linear programming to convex objective functions, convex inequalities and linear equality constraints. The search space of convex optimization problems is itself convex, as the intersection of half-spaces of convex support. This convexity property provide an important invariant: any local optimum is also the gloval optimum.

Nesterov et al. (1994) showed that most convex problems[11], including lin-

[10]Which is very closely related to the number of constraints due to *duality*. Unfortunately the *primal-dual theory* is beyond the scope of this manuscript, see (Papadimitriou and Steiglitz 1982) for details.

[11]Specifically linear programs, but also *second order cone programs* and *semidefinite programs*.

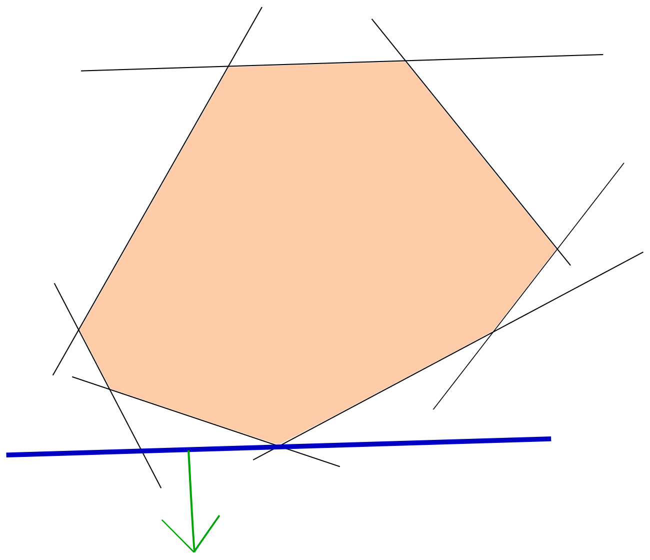

Figure 1.5: **Visual representation of a 2-dimensional linear programming problem.** The problem is constrained by seven inequalities, and the orange section represents the intersection of the half-spaces defined by these inequalities. The blue line represents the optimal isoline of the objective function (that is, all points (x, y) s.t. $f(x, y) = o$ and o is the optimum of the problem) and the green arrow the direction of the gradient (that is, the direction of the optimization). Clearly, the objective function is at its optimum on the vertex of the polygon.

ear programs, can be solved in polynomial-time complexity using variations of the interior-point methods that Karmarkar (1984) introduced for linear programming. In general, however, convex optimization remains a NP-hard problems.

Approximation algorithms a

Optimization complexity a

XXX PO, NPO, decision problems $\leftrightarrow$ optimization problems ($\leftarrow$ $\exists$ solution with at most / at lest bound b ; $\rightarrow$ enumerate (or binary search over) possible values of b) XXX a

XXX APX, FPTAS and PTAS XXX

1.3.1 Combinatorial optimization

Combinatorial optimization is the large field of research interested in optimization problems defined over finite search spaces. That is, where the problems is to search for one optimal element inside a discrete set of objects.

Although there exists some easy combinatorial problems, in general this is a class of problems that are often very difficult, for at least two reasons.

1. For one, even if the search space is finite, the number of feasible solutions is frequently so high that any enumeration is impossible in practice. For exemple in the two-sequences alignment problem. Given two sequences $\vec{a} = a_1a_2 \dots a_m$ and $\vec{b} = b_1b_2 \dots b_n$ $(n \leq m)$ over the same alphabet, find the alignment that maximize the number of similar characters between the two alignments. In this problem, the number of non redundant alignments between $\vec{a}$ and $\vec{b}$ is $N(m,n) = N(m-1,n) + N(m,n-1) = \binom{m+n}{n}$ which is a clear exponential growth.

2. Secondly, in many cases, the equivalent decision problem is itself a NP-hard problem. For exemple in the sequence assembly problem with the *shortest superstring problem.* Given a set of strings $s_1, s_2, \dots, s_n$, construct a superstring S that contains all s_i as substrings, and such that the length of S is minimal. The decision problem equivalence, knowing given s if there exists a superstring S_l of length $l \leq s$, is actually NP-complete. The proof is given by a reduction of the *vertex cover problem* (decision version) to this shortest superstring problem as shown by Maier (1978) for alphabets of size 5, and later by Räihä and Ukkonen (1981) for binary alphabets.

Combinatorial optimization is closely related to various algorithmic research domain both, and in particular to the complexity theory and to the combinatorics field (and any enumerative approaches).

- TSP, MST (min spanning tree)
- subdomain of mathematical optimization
- many applications
- two closely related fields, *combinatorial opt.* (loosely named) and *integer programming*
- distinction between problems for which polynomial-time algorithms exists and NP-complete problems
- for NP problems real-world instances (that do not exhibit difficult behavior) vs. random

- solving subproblem with polynomial-time algorithm (fpt algorithm..., cf)
- approximation
- For problems in P, greedy algorithms, dynamic programming, or linear programming
- expression as decision problem
- lots of categorie and subcategories
- Crescenzi and Kann (1995) published a compendium of NP opt. problems, that they maintain somewhat up to date at (Crescenzi and Kann 2016).

1.3.1.1 Polynomial-time solvable exemple: the min-cut problem

XXX Equivalent to the Max-flow problem, min-cut max-flow theorem, Papadimitriou and Steiglitz (1982, Section 6.1) for proof XXX

1.3.1.2 Branch-and-* algorithms

Branch-and-bound The branch-and-bound technique is a search space exploration technique based on the explicit traversal and pruning of the a state tree. It is based on the technique that Land and Doig (1960) applied to integer programming.

It is very similar to the *star familly* of algorithms (such as A*, B*, ...) and to the *alpha-beta* algorithm in regards to both space exploration and to pruning.

in chapter 3 use the min-cut algorithm introduced in section 1.3.1.1 as an heuristic

Branch-and-cut The branch-and-bound algorithm requires an explicit definition of the search space for its branch operation, in order to construct the list of candidate (sub-)solutions to explore.

Branch-and-cut is equivalent to branch-and-bound with the addition that additional constraints can be added during the execution. In pratice,

XXX Cannot express the whole problem (exponential number of constraints) XXX

XXX Simply faster thanks to the addition of local cuts XXX

For an introduction to optimization problems and the fundamental techniques of the domain (including Branch-and-bound methods) we recommend Papadimitriou and Steiglitz (1982)'s classic. For an *advanced reference* book on difficult optimization problem and their algorithms, their analysis and their precise classification, we suggest Hromkovič (2013)'s book.

Chapter 2

State of the art

2.1 Statistical analysis of gene expression

Nowadays, the differential analysis of transcriptomic profiles is ubiquitous in molecular biology. The ability to broadly assey biomolecular profiles of multiple samples cheaply enabled the wide adoption of techniques based on the differential analysis of multiple whole-genome samples. Indeed, genes for which their expression profiles are correlated over many different conditions are very likely to be involved in the same processes or to exhibit similar functionalities (Ideker et al. 2002). On the other hand, genes with significantly different expression profiles under conditions are candidates for choice as biomarkers for the biological phenomenons under study (Altman and Raychaudhuri 2001).

For example in cancer research, the ability measure transcriptomic state within a cell is a central technique for the analysis of mutated cell biomolecular machinery. Indeed, in order to maximize therapeutic effect of treatment and minimize side effects, it is very important to be able to characterize the genetic profiles of distinct tumor types. For a long time, tumor classification was based on medical expertize and assessment based on the clinical evolution and on the physical appearance of either the cancerous growth or its cells. However, even for tumor of the same type and grade, widely different clinical development and outcome were observed. A technique allowing for the classification and the detection of cancer types and subtypes was dearly needed.

The advances of gene expression profiling techniques allowed Golub et al. (1999) to develop such a classification technique based on simple statistical testing: the final prediction is the class for which the sum of scores computed statistically for each genes[1] is maximal.

XXX

It has been shown that gene profiles can be directly linked to their cell line's

[1]They actually compute the score only for genes that they consider *informative*, even though they acknowledge this selection as ”somewhat arbitrary”.

origin through consistent correspondence between gene expression patterns and origin of the tissue (Ross et al. 2000). Moreover, whole cell expression profiles can be used to classify cancer tumors, and further predict patient clinical outcome (Charles M Perou et al. 2000; Sorlie et al. 2001; Van't Veer et al. 2002; Van De Vijver et al. 2002; Estevez-Garcia et al. 2015)[2]

Even though transcriptome analysis is widespread in cancer research, it allows for many different applications. XXX List other applications: antibiotic treatment, ... XXX

However, most of those studies are based on gene-centric methods, which use univariate statistical testing to call for significantly differentially expressed genes.

2.2 Networks analysis

2.2.1 Biological networks

Increasingly advanced experimental methods are used to provide evidence of existing interactions, and nowadays comprehensive resources provide access to this knowledge.

- Automated: (Szklarczyk et al. 2014)
- Curated: (Orchard et al. 2012)

2.2.2 Orthologous genes

What is homology: (McCune and Schimenti 2012)

Cluster of orthologous genes (COG): (Tatusov et al. 1997).

Inparanoid: (O'Brien et al. 2005).

COG =¿ PPI networks alignment (difficult problem (El-Kebir et al. 2011)).

-¿ Identification of functionally similar with brute force PPI topology analysis across species: (Bandyopadhyay et al. 2006)

Orthology information can effectively be represented as a k-partite graph between k species.

XXX in conclusion, use those functionally similar as bipartite ? XXX

[2]The literature in the domain is quite extensive. We suggest here the first studies, as well as one recently published which still uses DNA chip for genes expression profiling.

2.2.3 Network analysis

2.3 Modules

There exists mostly two approaches to use biological networks to detect interesting biological processes. The first category is composed of *comparative* approaches of the networks structures, that enables to compare networks from different species or of the same species but under different conditions. They are treated in section 2.3.1. The second category is composed of techniques that aim to select sets of genes in specific biological contexts. These techniques usually combine both biological network structures and experimental data, and are called *integrative* approaches since they combine data of different types into a single algorithm. They are treated in section 2.3.2.

2.3.1 Topological modules

This section present methods that use the networks in themselves to extract structures deemed interesting. In order to do so, they use the topological structures of the networks to detect substructures.

Sharan and Ideker (2006) identify three broad types of such approaches: network alignment, network integration, and network querying methods. Even though what they call network integration is an important category of networks comparison problems, which are used for example for protein interaction prediction (Rhodes et al. 2005) or for protein modules detection (Kelley and Ideker 2005; Zhang et al. 2005), it is unrelated from our core problem.

What they name networks alignments and networks querying problems are two closely related categories of problems. In general terms, graph alignments are made of two seemingly similar subproblems: the *local graph alignment* problem, and the *global graph alignment* problem. Precisely, in the local alignment problem we look for a subgraph in the larger input that most closely resemble a query graph or query criteria. In the global alignment problem we try to match every nodes of one graph to every nodes of the other graph.

These two subproblem mirror the similarities that exists between the local and global sequence alignment problems. Indeed, similarly to the global sequence alignment problem which is usually used to compare different species or organisms of the same species, the global network alignment problem correspond to the *network alignment* category describe above. And similarly to the local sequence alignment problem which is usually used to compare or locate subsequences, the local network alignment problem correspond to the *network querying* category.

When looking for matchings networks, vertices are usually mapped on a one-to-one basis, but it can be many-to-one or many-to-many queries. Furthermore, the matching can be weighted, or constrained by a bipartite similarity relationship between the two graphs. They are all optimization problems where some criteria, for example the number of matched nodes or edges, must be maximized.

These problems pertains to the *module discovery* class of problems over biological networks[3].

2.3.2 Active modules

The category of methods use the networks combined with biological experimental data, for example expression profiles, to detect structures of interest in the context of the experiment. These methods are well adapted for an understanding of single-species processes, for which the interpretation of the results follow logically from the inputs. They can also be used to look across multiple species for interesting process.

Already in 2001, Altman and Raychaudhuri (2001) recognized the need to use networks of genetic interactions to tackle the analysis of expression data. One of the key concepts to understand biological processes in those networks is that of *modules*. Modules are considered to be sets of entities[4] that function in a coordinated fashion or physically interact (for a review see Mitra et al. (2013)).

A possible formulation for the problem of finding modules within a network is to look for connected sub-networks that maximize weights on the nodes. These weights typically represent some measure of biological activity, for example the expression level of genes. In their seminal work, Ideker et al. (2002) were the first to solve the problem of finding gene modules within a biological network, using simulated annealing heuristic optimization. They recognize that finding the optimal (with respect to the sum of the nodes' weights) module in a biological network is formally equivalent to the combinatorial MAXIMUM-WEIGHT CONNECTED SUBGRAPH problem. This problem will be treated in section 2.4.

More biology of modules XXX TODO: (Dittrich et al. 2008; Yamamoto et al. 2009; Backes et al. 2012; Mitra et al. 2013).

2.3.2.1 Cross-species analysis

Several authors already identified the benefits of combining cross-species experimental data. At the single gene level, Noort et al. (2003) have demonstrated that conserved co-expression is a strong co-evolutionary signal. More recent studies suggested to identify conserved biological processes.

Talk about cMonkey: (Reiss et al. 2006). And cMonkey2: (Reiss et al. 2015).

[3]It is also named *community structure discovery* when applied over social networks.

[4]Genes, proteins, etc.

Lu et al. (2009) analyzed transcriptomic profiles of human and mouse macrophages and dendritic cells, under two conditions, to derive common response genes involved in innate immunity. They used a probabilistic graphical model that propagates information for each gene about its involvement in the innate immunity response across species or cell types and conditions then seeks pathway enriched with those common responses genes.

Waltman et al. (2010) presented a multi-species integrative method to heuristically identify conserved biclusters. In their setting, a conserved bicluster is a subset of orthologous genes and a subset of conditions that achieve a high score with respect to co-expression, motif co-occurrence and network density.

Kristiansson et al. (2013) proposed a method for the analysis of gene expression data that takes the homology structure between the different species into account. Their method is an extension of the standard Fisher's method for meta-analysis (Hu et al. 2006; Campain and Y. H. Yang 2010; Tseng et al. 2012) that explicitly account for in-paralagous and orthologous genes and is able to call differential expression with increased statistical power when compared to methods ignoring this relationship.

Dede and Oğul (2014) introduced a method that finds triclusters consisting of genes that are coexpressed across a subset of samples and a subset of species.

Modules as connected subgraphs Deshpande et al. (2010) developed the neXus algorithm for finding conserved active subnetworks. neXus is based on simple notions of activity and orthology and uses a heuristic search strategy. The authors use the average fold change of genes in a module as a measure for activity. To deal with conservation, they collapse paralogous genes within a cluster of orthologous genes (COG) (Tatusov et al. 1997)[5] into single nodes in the respective networks. They find modules using a seed-and-extend greedy heuristic that starts from a pair of orthologous seed nodes and then tries to simultaneously grow the two subnetworks by including pairs of neighboring orthologous genes, taking into account their activity as well as the interaction confidences. This strategy enforces a very stringent conservation policy: only modules whose genes are fully conserved are found. In addition, the locality of the greedy search strategy impairs the ability to find larger conserved modules and extending the search space around the seed genes drastically increases the runtime.

In recent work, Zinman et al. (2015) introduce ModuleBlast, a method that, similarly to neXus, represents groups of orthologous proteins as single nodes in a combined network and tries to find connected subnetworks that are differentially expressed. The novelty of the method is the classification of the found modules according to the sign of the log fold change expression values. By doing so, the authors are able to assess whether conserved active modules show

[5]As provided by databases such as Inparanoid (O'Brien et al. 2005).

consistent or inconsistent eession patterns. However ModuleBlast, like neXus, requires strict conservation of module genes. We see that as an important limitation that our present work aim to correct.

2.4 The maximum-weight connected subgraph problem

This section is separated in two parts. In section 2.4.1 we first introduce the STEINER TREE, the PRIZE-COLLECTING STEINER TREE (PCST), and the MAXIMUM-WEIGHT CONNECTED SUBGRAPH (MWCS) problems. We also provide the main complexity results for the MWCS problem, and since a number of those results mostly follow from results for the PCST problem, we provide the most important ones for this problem too. In section 2.4.3 we present the main methods used to solve those problems. There are many techniques to solve these problems, and they can be classified to two caterogies. The theoretically-sound methods, that can provide either a provably optimal (or a proven gap[6]) solution, or an approximation solution to a proven bounded-factor. And the heuristic methods, that approximate the optimal solution to an unknown factor, but which are usually much faster.

2.4.1 Problems introductions

The STEINER TREE problem is an extremely well known combinatorial optimization problem, which is part of Karp's original 21 NP-complete problems (Karp 1972). It has its origin in the geometry of Jakob Steiner's eponym Steiner problem, and pertain to the class of mathematical optimization problems over graphs. It is informally defined as follows. A large number of variations of this problem exists: the *Steiner tree problems*. Hauptmann and Karpinski (2014) maintain an extensive and up-to-date compendium of those variants.

One of the variants of the STEINER TREE problem is the PRIZE-COLLECTING STEINER TREE problem, or PCST problem. Is is informally defined as follows.

pcst: Given a graph, a non-negative weight for each vertex, and a non-negative cost for each edge, the goal is to find the subtree that maximizes the sum of the weights minus the sum of the costs.

The PCST problem is an *utility versus cost optimization* variant (as formally defined by Conrad et al. (2007)) of the STEINER TREE problem. Bienstock et al. (1993) were the firsts to formally introduce the problem, and the first recognized proof of NP-hardness follows from Camerini et al. (1979) (even through the

[6]Proven maximal distance to the optimal solution.

problem was not formally defined at the time). Surprisingly[7], Feigenbaum et al. (2001) were the first to prove that the PCST problem is NP-hard to approximate within any constant ratio $0 < \epsilon < 1$, or APX-hard, using a reduction from SAT. This variant is highly relevant to our context as there exist bidirectional reductions between the PCST and MWCS problems, and for a long time reducing an MWCS instance to a PCST instance was the technique of choice to actually solve the problem (cf. section 2.4.3).

The MAXIMUM-WEIGHT CONNECTED SUBGRAPH problem, or MWCS problem, falls within the same classification as various Steiner tree problems: it is a problem of combinatorial optimization over graphs. It is informally defined as follows.

mwcs: Given a graph and a real-valued weight for each vertex, the goal is to find the connected subset of vertices that maximizes the sum of the weights.

First observe that, since the edges are unweighted, the solution is trivial, full or empty, if the vertices' weights are respectively all positive or all negative. This is in constrast with the PCST problem where the positive weights of the vertices oppose the costs of the edges.

From a computer science point of view, the MWCS problem is simple in its definition. Nevertheless, it is actually a very difficult problem to solve, and in many cases remains intractable. The first text to prove NP-hardness of the problem is the unpublished manuscript from Vergis (1983). Manuscript that D. S. Johnson (1985, Section 5) acknowledge when he looked into a series of graph problems in his famous *NP-Completeness Columns*. The proof is based on the reduction from the STEINER TREE problem, provided by Garey and D. S. Johnson (1979). Johnson made the fundamental observation that the solution to the MWCS problem can always be reduced to a tree, since the additional edges serve no purpose. Karp later provided another proof of NP-hardness for the MWCS problem in (Ideker et al. 2002, Supplementary Material), by providing a reduction from the MINIMUM SET COVER problem, another problem which is one of his 21 first NP-complete problems (Karp 1972).

Indeed, Álvarez-Miranda et al. (2013a) proved that the MWCS problem is actually APX-hard itself. Using the SAT reduction previously mentioned (Feigenbaum et al. 2001), they extended the result to MWCS using a straightforward reduction of PCST to MWCS.

In the same way the STEINER TREE broadly defines a large set of related problems, the MWCS problem defines itself a set of closely related problems. These problems can be applied in many different contexts, of which: social network sciences, operations research, certainly networks design, and system biology, which is our main interest in this manuscript.

[7]They themselves acknowledge that this is an easy result (Feigenbaum et al. 2001, footnote 12).

Based on the basic version of the MWCS problem, the principal variants are the *constrained* versions, with an allocated budget and additional cost assigned to each vertex. The k-CARDINALITY MWCS, the CARDINALITY-CONSTRAINED MWCS, and the BUDGET-CONSTRAINED MWCS serve interesting purposes. The k-CARDINALITY MWCS require that the solution be comprised of exactly k vertices, whereas the CARDINALITY-CONSTRAINED MWCS variant require that the solution includes at most K vertices. The BUDGET-CONSTRAINED MWCS variant assigns an additional positive cost to each vertex, and requires that the sum of the costs be at most a given budget B. Clearly, assigning a positive cost of 1 for each node of the CARDINALITY-CONSTRAINED MWCS problem provides a trivial reduction to the BUDGET-CONSTRAINED MWCS problem.

Note that for all these constrained versions, the problems remain non-trivial even when the nodes' weights are all positive. Furthermore, note that while removing the connectivity constraint in the basic version of the problem makes it trivial, in those variants the problems then become equivalent to the 0-1 KNAPSACK problem (which is another one of Karp's original 21 NP-complete problems (Karp 1972)).

They can all be defined either on graphs or on directed graphs, and there exists rooted variants where one (or multiple) root(s) have to be selected in the solution.

There also exists minimization variants or reformulations for those problems. Most of them are strictly equivalent from an optimality standpoint: minimizing the sum of the opposite of the nodes' weights. However, they differ from approximation standpoint, see (Feigenbaum et al. 2001; D. S. Johnson et al. 2000) for details.

2.4.2 mwcs in practice

The MWCS problem and its cardinality-constrained and budget-constrained variants are used in numerous important practical applications.

Hochbaum and Pathria (1994) first described the fixed cardinality variant of the problem (k-CARDINALITY MWCS). They relate its use in two contexts. First in off-shore oil-drilling where each facility is represented by a node and the weight their costs over benefits. Second in forest harvesting where we need to find the k connected parcel to harvest considering their associated benefits. They call it the CONNECTED k-SUBGRAPH problem. They where the first to observe that, for this variant, since adding a constant to the weight of each node does not change the optimal set of nodes, in the optimality context the nodes' weights can all be non-negative. They also show that the problem is NP-hard even for bipartite or planar input graphs or if the nodes' weights are boolean.

H. F. Lee and Dooly (1998) reintroduced the k-CARDINALITY MWCS problem, and called it simply the MAXIMUM-WEIGHT CONNECTED GRAPH (MCG) problem. They were the firsts to introduce the rooted variant where a single root is provided for the solution, that they called the CONSTRAINED MCG (CMCG). They used this rooted variant to provide a decomposition scheme of the MWCS problem into multiple CMCG subproblems. They acknowledge that the optimal solution is NP-hard to find, and provide heuristics for their incremental roots selection. They use all problems that they introduce in the fiber-optic networks design context.

Gomes' team looked into the budget constrained version of the problem, both rooted and unrooted variants, that they apply in the context of *conservation planning* (Conrad et al. 2007; Gomes et al. 2008; Dilkina and Gomes 2010).

C.-Y. Chen and Grauman (2012) used the MWCS problem to extract signatures from a video stream for the activity detection problem. They represent the video as a three dimentionnal 6-connect graph, i.e. a space-time matrix, where each node's weights represent video features. They then use standard classifier to classify the signatures.

Carvajal et al. (2013) also looked into the forest planning problem to select contiguous regions of forest to maximizes ecosystem protection in nature reserve design, i.e. the number of species and habitats preserved.

Furthermore, as stated in section XXX, Dittrich et al. (2008), Yamamoto et al. (2009), Backes et al. (2012), and Mitra et al. (2013) all used the MWCS problem or its variants in system biology to extract connected sets of genes.

2.4.3 Solving the mwcs problems

Over the years, many methods have been proposed to solve the MWCS problem. They can mostly be categorized into two groups: 1) methods that first reduce the MWCS instance into a PCST instance, or 2) through direct modelization.

In both groups, many techniques have been proposed in order to make instances easier to solve. Instance size reductions are often used, which leads to exact methods if quality guarantees are preserved through reduction, or heuristics (possible approximation) methods when more aggressive simplifications are performed. Size reduction algorithms are fundamental techniques in solving large practical Steiner tree problems (or similar, such as the MWCS problem), and applying known exact reduction is the first step for real world instances resolution (Polzin 2003).

2.4.3.1 Through the Prize-Collecting Steiner Tree problem

In their seminal paper, Goemans and Williamson (1995) were the firsts to provide an approximation algorithm (a 2-approximation) for a number of *constrained forest* problems, such as the PCST problem. They further develop their method in (Goemans and Williamson 1997). This is an important contribution since they reformulate the problem into an easy to approximate formulation (the standard formulation is APX-hard), which became the basis for many other contributions afterward.

Builting on these algorithms, D. S. Johnson et al. (2000) proposed multiple variations of this 2-approximation for PCST, that provide better performance, and application to variants such as the QUOTA PCST where the sum of the weights must be at least a given *quota*, and the BUDGET PCST where the sum of the costs must be at most a given *budget*.

Lucena and Resende (2004) introduces the *generalized subtour elimination constraints* technique, and uses the separation algorithm first described by Fischetti et al. (1994).

In two papers, Ljubić et al. (2005, 2006) solved the MWCS problem by combining previously introduced methods for various Steiner tree problems over directed graphs. Their technique was the first to solve to optimality some of the privous benchmark instances and was order of magnitude faster than previous methods on some other instances.

Dittrich et al. (2008) were the first to describe a reduction from MWCS to PCST with linear conservation of the objective function optimal value. Given this reduction, any method previously proposed to solve the PCST problem could now be used to solve instances of the MWCS problem, application that they demonstrate in system biology for module extraction. They also gave an approximation-preserving reduction from PCST to MWCS.

Chimani et al. (2009) introduces an ILP formulation equivalent to the GENERALIZED SUBTOUR ELIMINATION CONSTRAINTS formulation proposed by Lucena and Resende (2004) that they call the *directed cut* formulation. They then propose a stronger separation algorithm for this new formulation, which is more efficient in pratice than the previous technique introduced by Fischetti et al. (1994).

2.4.3.2 Direct formulation

Direct formulation methods include all methods that can be used to solve the MWCS problem without first explicitly reducing the MWCS instance to a PCST instance.

Quintão et al. (2008) provide multiple reformulation of the k-cardinality variant of the MWCS problem, from which they formulation strong linear relaxations. Quintão et al. (2010) further improves the technique by further integrating the constraints introduced very early by Miller et al. (1960).

Backes et al. (2012) proposed an exact formulation to solve the k-cardinality subtrees and connected subgraphs problems.

Álvarez-Miranda et al. (2013a,b) introduced the first technique which does not explicitly model the graph edges. Instead they introduce a Branch-and-Cut scheme where connectivity violations are detected from candidate solutions and corresponding constraints are recursively added to the model. They showed that their technique outperformed most of the other methods on practical instances at the time.

El-Kebir and Klau (2014) introduce new preprocessing rules that they apply until an stable state is obtained. They introduce a new graph decomposition, into biconnected and triconnected components, and solve each subproblem using a standard branch-and-cut approach similar to the one introduced by Álvarez-Miranda et al. (2013a).

Althaus et al. (2014) describe, for k-induced subgraph, a combination of (Fischetti et al. 1994), (Chimani et al. 2009), and (N. Cohen 2010). In their as of yet unpublished manuscript, Althaus and Blumenstock (n.d.) further improved their previous algorithm by introducing exact and heuristic reductions of the size of the MWCS instances. They solve the newly reduced problems using a combination of two MIP programs formulations: the (N. Cohen 2010) formulation for spanning tree problems, and the addition of the *generalized subtour elimination constraints* to further reduce the polyhedron size. They use Chimani et al. (2009)'s *directed cut* formulation of the constraints since the separation algorithm for it is very efficient. They integrate some of El-Kebir and Klau (2014)'s exact reductions in their algorithm, and propose heuristic reductions if the practical instance is still too large.

Chapter 3

xHeinz: a cross-species module discovery tool

Protein-protein interaction networks play a key role in understanding of cellular processes. Among bioinformatic techniques that rely on these networks, module extraction and network alignment are two majors classes of methods (see chapter *state of the art*). Traditionally, module extraction allows for the discovery of interesting gene sets from single species experiments. Network alignment is often used to discover conserved structures between species, that is similar subnetworks that are assumed to have the same biological functionality.

Molecular profiles are most often measured and validated on well studied model species. This is partly due to the fact that, when differential analysis is involved, the experiments require 1) sufficient replication, and 2) control and condition samples (Trapnell et al. 2013) for the results to be statistically significant. Unfortunately these two requirements are difficult to obtain in human studies since there are large variations between physiological states of humans, making statistical analysis of replicates more difficult. Model species or cellular models are thus the source of choice for experimentation and gene expression analysis, and bioinformatics techniques for gene sets extraction often works with single species experimental data.

Unfortunately, immediate transferability from model organisms to human is rare, when possible (Okyere et al. 2014). In their systematic review of cross-species extrapolation in pharmacokinetic modeling, Thiel et al. (2015) recently estimated that, at best, roughly 83.5% of the model extrapolations[1] are in agreement with known results. As a result, Csermely et al. (2013) attribute the very low phase-II survival rate of potential drug compounds (25%) to the lack of transferability between model organisms and human.

[1] In their study, mouse is the model organism and human the transfer target, which is a very standard coupling in phamacological transfer studies.

In this chapter we present a cross-species module discovery technique that we initially introduced in (El-Kebir et al. 2015). It enables the simultaneous search of interesting gene sets in the two species, and such that those gene set have a high percentage of conservation across the two species.

In section 3.1 we present our technique as a mathematical model that makes it possible to identify conserved active modules across two species. Building upon the single-species modules extraction model described in (Dittrich et al. 2008), our model inherits its notions of modularity and activity: 1) a set of genes forms a module if it induces a connected subnetwork, and 2) the activity of a module is the sum of the activities of its individual genes. The activity of each gene is quantified using a beta-uniform mixture model on the distribution of p-values that characterize the differential behavior. Our model introduces a flexible conservation policy, which allows to specify the minimum fraction of nodes in the solution that must be conserved. A rigorous complexity analysis of our model is the main topic of chapter 4.

We then cast our model as an integer linear programming formulation and present xHeinz, a branch-and-cut algorithm and its implementation. xHeinz is an *exact optimization method* that solves our model to provable optimality (given enough time), or reports a solution with a quality guarantee[2] (if stopped before full convergence).

In section 3.2 we apply xHeinz to understand the mechanisms underlying Th17 T cell differentiation in both mouse and human. As a main biological result, we find that the key regulation factors of Th17 differentiation are conserved between human and mouse and demonstrate that all aspects of our model are needed to obtain this insight. We further demonstrate the robustness of our approach by comparing samples of the differentiation process obtained at different time points, in which we search for optimal, conserved active modules under a wide range of conservation ratios. Using a permutation test, we show that our results are statistically significant. Finally, we discuss the main differences between our results and the results obtained by the neXus tool (see chapter *state of the art*) on the same data set.

3.1 Algorithmic approach: the mwccs problem

3.1.1 Mathematical model

We consider the conserved active modules problem in the context of two species networks, which we denote by $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Nodes in these networks are labeled by their activity – defined by $w \in \mathbb{R}^{V_1 \cup V_2}$ and conserved node pairs are given by the symmetric relation $R \subseteq V_1 \times V_2$. The aim is to

[2]A provable maximum optimization gap.

identify two maximal-scoring connected subnetworks, one in each network, such that a given fraction α of module nodes are conserved. The formal problem statement is as follows:

Problem 1 (Conserved active modules)**.** *Given $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, $w \in \mathbb{R}^{V_1 \cup V_2}$ and $R \subseteq V_1 \times V_2$, the task is to find a subset of nodes $V^* = V_1^* \cup V_2^*$ with $V_1^* \subseteq V_1$ and $V_2^* \subseteq V_2$ such that the following properties hold.*

- ***Activity:*** *Node activity scores are given by $w \in \mathbb{R}^{V_1 \cup V_2}$, where positive scores correspond to significant differential expression. For details see section 3.2.2. We require that the sum $\sum_{v \in V^*} w_v$ is maximal.*
- ***Conservation:*** *Conserved node pairs are given by the relation $R \subseteq V_1 \times V_2$. We require that at least a certain fraction α of the nodes in the solution must be conserved, that is, $|U^*| \geq \alpha \cdot |V^*|$ where $U^* := \{u \in V_1^* \mid \exists v \in V_2^* : uv \in R\} \cup \{v \in V_2^* \mid \exists u \in V_1^* : uv \in R\}$.*
- ***Modularity:*** *We require that the induced subgraphs $G_1[V_1^*]$ and $G_2[V_2^*]$ are* connected*.*

The model allows a trade-off between conservation and activity. If no conservation is enforced ($\alpha = 0$), the solution will correspond to two independent maximum-weight connected subgraphs, thereby achieving maximal overall activity. Conversely, if complete conservation is required ($\alpha = 1$), the solution can only consist of conserved nodes, which results in the lowest overall activity modules. The user controls this trade-off by varying the value of the parameter α from 0 to 1. The activity score monotonically decreases as α increases, see fig. 3.1.

3.1.2 Mixed-Integer Linear programming approach

We formulate the conserved active modules problem as an integer programming (IP) problem in the following way.

$$\max \sum_{v \in V_1 \cup V_2} w_v x_v \tag{3.1}$$

$$\text{s.t. } m_u = \max_{uv \in R} x_u x_v \qquad u \in V_1 \tag{3.2}$$

$$m_v = \max_{vu \in R} x_u x_v \qquad v \in V_2 \tag{3.3}$$

$$\sum_{v \in V_1 \cup V_2} m_v \geq \alpha \sum_{v \in V_1 \cup V_2} x_v \tag{3.4}$$

$$G_1[\mathbf{x}] \text{ and } G_2[\mathbf{x}] \text{ are connected} \tag{3.5}$$

$$x_v, m_v \in \{0, 1\} \qquad v \in V_1 \cup V_2 \tag{3.6}$$

This formulation satisfies the properties of activity, conservation and modularity.

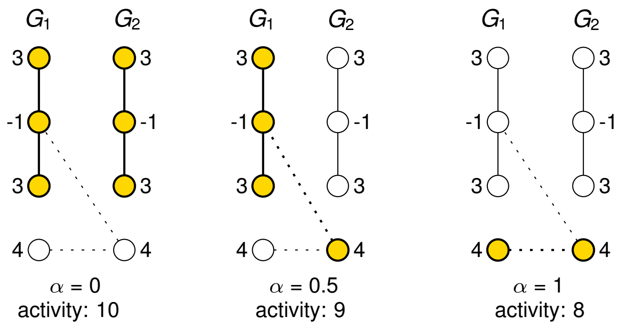

Figure 3.1: **Trade-off between activity and conservation.** Three optimal solutions (indicated in yellow) for varying conservation ratios α in a toy example instance. Node activities are given next to the nodes, conserved node pairs are linked by dotted lines. The activity of a conserved module is the sum of the activities of its comprising nodes. The parameter α denotes the minimum fraction of nodes in a solution that must be conserved, *i.e.* connected by a dotted line.

Activity.
Variables $\mathbf{x} \in \{0,1\}^{V_1 \cup V_2}$ encode the presence of nodes in the solution, *i.e.*, for all $v \in V_1 \cup V_2$ we want $x_v = 1$ if $v \in V^*$ and $x_v = 0$ otherwise. The objective function (3.1) uses these variables to express the activity of the solution, which we aim to maximize.

Conservation.
Variables $\mathbf{m} \in \{0,1\}^{V_1 \cup V_2}$ encode the presence of conserved nodes in the solution. Recall that a node $u \in V_1^*$ ($u \in V_2^*$) that is present in the solution is conserved if there is another node $v \in V_2^*$ ($v \in V_1^*$) in the solution such that the two nodes form a conserved node pair $uv \in R$ ($vu \in R$). This corresponds to constraints (3.2) and (3.3). Indeed, constraints (3.2) encode that a node $u \in V_1$ that is present in the solution ($x_u = 1$) is conserved if there exists a related node $v \in V_2$ ($uv \in R$) that is also present in the solution ($x_v = 1$). Similarly, constraints (3.3) defines conserved nodes in V_2 that are present in the solution. We linearize $x_u x_v$, in a standard way, by introducing binary variables $\mathbf{z} \in \{0,1\}^R$ such that $z_{uv} = x_u x_v$ for all $uv \in R$:

$$z_{uv} \leq x_u \qquad uv \in R \tag{3.7}$$
$$z_{uv} \leq x_v \qquad uv \in R \tag{3.8}$$
$$z_{uv} \geq x_u + x_v - 1 \qquad uv \in R \tag{3.9}$$
$$z_{uv} \in \{0,1\} \qquad uv \in R \tag{3.10}$$

Subsequently, we model the max function in (3.2) and (3.3) as follows.

$$m_u \geq z_{uv} \qquad \forall v \in {V_2}^* \text{ st. } uv \in R \tag{3.11}$$

$$m_v \geq z_{uv} \qquad \forall u \in {V_1}^* \text{ st. } uv \in R \tag{3.12}$$

$$m_u \leq \sum_{uv \in R} z_{uv} \qquad u \in V_1 \tag{3.13}$$

$$m_v \leq \sum_{uv \in R} z_{uv} \qquad v \in V_2 \tag{3.14}$$

This set of constraints encode the two required conditions:

$$m_u = \begin{cases} 1 & \text{if at least one of its counterpart is present,} \\ 0 & \text{otherwise.} \end{cases}$$

On one hand, (3.11) define a set of constraints: one for each node $v \in {V_2}^*$ such that $uv \in R$. This set of constraints effectively instruct the ILP that m_u must be 1 if at least one of the counterparts of u is in the solution. The same reasoning goes for (3.12). On the other hand, (3.13) constraint the variable m_u to be 0 if none of the counterparts of u are in the solution. The same reasoning goes for (3.14).

We model the required degree of conservation by constraint (3.4): the fraction of conserved nodes in the solution is at least α.

Modularity.
In addition, we satisfy the modularity property by requiring in (3.5) that $G_1[\mathbf{x}]$ and $G_2[\mathbf{x}]$ are connected.

Constraint (3.5) states that the nodes encoded in the solution $\mathbf{x}$ induce a connected subgraph in both G_1 and G_2. There are many ways to model connectivity, *e.g.*, using flows or cuts (Magnanti and Wolsey 1995). However, Dilkina and Gomes (2010) showed that cut-based formulations perform better in practice. Recently, Álvarez-Miranda et al. (2013a) have introduced a cut-based formulation that only uses node variables. In an empirical study, the authors show that their formulation outperforms other cut-based formulations. We model connectivity along the same lines. Since the constraints that we will describe are similar for both graphs, we introduce them only for graph $G_1 = (V_1, E_1)$.

$$\sum_{v \in V_1} y_v \leq 1 \tag{3.15}$$

$$y_v \leq x_v \qquad v \in V_1 \tag{3.16}$$

$$x_v \leq \sum_{u \in \delta(S)} x_u + \sum_{u \in S} y_u \qquad v \in V_1, \{v\} \subseteq S \subseteq V_1 \tag{3.17}$$

$$y_v \in \{0, 1\} \qquad v \in V_1 \cup V_2 \tag{3.18}$$

Where $\delta(S) = \{v \in V_1 \setminus S \mid \exists u \in S : uv \in E_1\}$ denotes the *neighbors* of S.

The modularity property states that $\mathbf{x}$ should induce a connected subgraph in G_1. However in our model, we don't explicitely model graph connectevity, and model local connectivity instead. The first sum of (3.17) state that x_v can only be 1 if, for all sets $S \subseteq V$ containing v, it holds that there is a neighbor u of S in the solution. Informally, this constraint is a form of expension where we require the nodes in the solution to be part of the neighborhoud of the other nodes in the solution, hence forming a connected subgraph. However this is not sufficient, because this would require all nodes to be included in the end. We solve this by introducing binary variables $\mathbf{y} \in \{0,1\}^{V_1}$ that determine a root node, which serves as a local expension termination condition. First, constraints (3.15) and (3.16) state that at most one node v part of the solution can also be the root node – in which case $y_v = 1$. Second, the last sum of (3.17) state that x_v can only be 1 if, for all sets $S \subseteq V$ containing v, it holds that the root node is in S. Informally and integrating the first sum, this constraint encode that the graph is locally connected around v if, for all possible sets $S \subseteq V$ containing v, either the root node is part of S or at least one node of the neighborhoud of S is part of the solution.

There is an exponential number of such constraints. Therefore, we cannot add all them to our initial formulation. Instead we use a branch-and-cut approach, that is, at every node of the branch-and-bound tree we identify all violated constraints and add them to the formulation. Finding violated inequalities corresponds to solving a minimum cut problem, which we do using the algorithm by Boykov and Kolmogorov (2004).

To further improve the performance, we also strengthen our model with the following constraints. None of those constraints are necessary, but they help the ILP solver by reducing the search space.

$$y_v = 0 \qquad v \in V, w_v < 0 \tag{3.19}$$

$$\sum_{u \in V} y_u \geq x_v \qquad v \in V, w_v \geq 0 \tag{3.20}$$

$$x_v \leq \sum_{u \in \delta(\{v\})} x_u + y_v \qquad v \in V \tag{3.21}$$

$$y_v \leq 1 - x_u \qquad u, v \in V, u < v, w_u \geq 0, w_v \geq 0 \tag{3.22}$$

Constraints (3.19) states that the root node must be a positively weighted node, which reduces the search space of the root node. Constraints (3.20) explicitly state that the root node variable must be 1 if at lease one of the positive nodes is in the solution. This was already required for the previous set of constraints but never explicitly instructed to the solver. Constraints (3.21) is an optimization for the cases where the set S in (3.17) is a singleton. Finally, constraints (3.22) are symmetry breaking constraints: they encore an ordering for the possible root selections. It effectively requires that among all positively weighted nodes in the solution, the root node is the smallest one – according to some arbitrary order[3]. This last set of constraints also provide determinism: the ordering garantee that two runs of the ILP will choose the same root node.

3.2 Material and methods

We apply our method to the recently discovered interleukin-17 producing helper T cells (Th17), which exposes the problems highlighted in chapter 3.

These cells form a separate subset of helper T cells with a differentiation pathway distinct from those of the established Th1 and Th2 cells (Park et al. 2005). Th17 cells are known to contribute to pathogenesis of inflammatory and autoimmune diseases such as asthma, rheumatoid arthritis, psoriasis and multiple sclerosis and play also a role in cancer immunology (Wilke et al. 2011). They originate from naïve helper cells, responding to environmental stimulus by activating a differentiation and specialization process (Steinman 2007).

Understanding the pathways and regulatory mechanisms that mediate the decision making processes resulting in the formation of Th17 is a critical step in the development of novel therapeutics that will facilitate rational manipulation of the immune response. Unfortunately, the vast majority of data collected so far originates from studies performed on mice (Tuomela et al. 2012) and, most importantly, a comprehensive comparison of the Th17 differentiation process in model organisms and in human is missing. Several studies indicate that the differentiation and phenotype of human and mouse Th17 cells are similar (Annunziato and Romagnani 2009). Both subsets serve similar pro-inflammatory functions and produce the same hallmark cytokines and similar receptors. Furthermore, most of the already identified regulator genes show high sequence conservation.

These findings indicate that the differentiation process seems well conserved between human and mouse and that a cross-species approach is reasonable. Other studies, however, show stimulus requirements for effective differentiation of human cells that differ from those required for mice (McGeachy and Cua 2008; O'Garra et al. 2008; Annunziato et al. 2009).

The simultaneous analysis of both human and mouse expression data allows the identification of conserved candidate regulators, as well as potential drug targets. Most of our current understanding on Th17 cell differentiation relies on studies carried out in mice, whereas the molecular mechanisms controlling human Th17 cell differentiation are less well defined. A characterization of the similarities and differences will not only increase our understanding of this fundamental process, but is also essential for sound translational research.

[3] In our case: the order of apperances of the nodes in the network definition.

3.2.1 Experimental procedure

We summarize here the experimental procedure followed by Tuomela et al. (2012) and Yosef et al. (2013) to generate transcriptomic profiles.

Tuomela et al. (2012) isolated CD4+ T-cells from umbilical cord blood of several healthy neonates, arranged in three different pools, then activated with anti-CD3 and anti-CD28. Cells from each pool were then divided in two batches, one to be polarized toward Th17 direction, and one serving as control (Th0). Th17 differentiating cytokines consisted of IL6 (20 ng/mL), IL1B (10 ng/mL) and TGFB (10 ng/mL), along with neutralizing anti-IFNG (1 µg/mL) and anti-IL4 (1 µg/mL). Three biological replicates of human cells, for both conditions (coming from each pool), were collected between 0.5 – 72 h (0.5 h, 1 h, 2 h, 4 h, 6 h, 12 h, 24 h, 48 h, 72 h time points) and hybridized on Illumina Sentrix HumanHT-12 Expression BeadChip Version 3. The microarray data were analyzed using the beadarray Bioconductor package (Dunning et al. 2007).

Yosef et al. (2013) purified CD4+ T-cells from spleen and lymph nodes from wild type C57BL/6 mice, then activated with anti-CD3 and anti-CD28. For Th17 differentiation, cells were cultured with TGFB (2 ng/mL), IL6 (20 ng/mL), IL23 (20 ng/mL) and IL1B (20 ng/mL) during 0.5 – 72 h (at time points 0.5 h, 1 h, 2 h, 4 h, 6 h, 8 h, 10 h, 12 h, 16 h, 20 h, 24 h, 30 h, 42 h, 48 h, 50 h, 52 h, 60 h, 72 h), and finally hybridized on an Affymetrix HT_MG-430A.

3.2.2 Microarray processing, statistical analysis and node scoring

Preprocessed and quantile normalized data sets were downloaded from GEO under the accession numbers GSE43955 and GSE35103. As downloaded from GEO, both the human and the mouse time-series were already filtered by retaining only the probes with detection p-values < 0.05 in at least one time point and one condition. Following the original studies, we further only retained probes having a standard deviation > 0.15 over all the conditions and time points; as well as being annotated by a single Ensembl gene. Finally, a single probe was selected for each gene by taking, for each Ensembl gene, the probe having the largest variance accross all samples. In total, 12,307 and 18,497 probes passed the filters for the mouse and human data set, respectively.

Differential expression between Th17 and Th0 conditions were estimated using the limma package (Smyth 2005). Human samples were indicated as paired according to the experimental design so as to account for the pooled human samples. For mouse samples, calling was performed on all Th0 vs Th17 samples, regardless of the mouse donor. To determine which genes were differentially expressed at a given time point, we used a linear model to estimate the interaction between the treatment and the time effect. The linear models used for the human and mouse studies include one interaction term for each time point and exclude the intercept (In R, the formula reads: $\sim 0 + \text{treat} : \text{time}$). Differential expression at any time point K of interest were determined by the contrasts $\text{Th17.time}_\text{K} - \text{Th0.time}_\text{K}$. We report in this study results for the following time points: 2 h, 4 h, 24 h, 48 h, 72 h.

Following (Dittrich et al. 2008), we computed positive and negative scores for each gene at each time point by fitting a beta-uniform mixture model using the implementation in the BioNet package (Beisser et al. 2010). The method proceeds as follows:

Similarly to (Pounds and Morris 2003), the distribution of the gene-wise p-values $x = x_1, \dots, x_n$ is described as a beta-uniform mixture (BUM) model, which is a mixture of a $B(a, 1)$ beta distribution (signal) and a uniform distribution (noise): $\lambda + (1 - \lambda)ax^{a-1}$, for $0 < a < 1$, with mixture parameter λ and shape parameter a of the beta distribution. The log likelihood is defined as $\log(\lambda, a; x) = \sum_{i=1}^{n} \log(\lambda + (1 - \lambda)ax_i^{a-1})$, and consequently the maximum-likelihood estimations of the unknown parameters are given by $[\hat{\lambda}, \hat{a}] = \arg\max_{\lambda,a}(\lambda, a; x)$. The parameter estimates have been obtained using numerical optimization. As detailed in (Pounds and Morris 2003), the BUM model allows the estimation of a false discovery rate (FDR) that can be controlled via a p-value threshold $\tau(\text{FDR})$. The adjusted log likelihood ratio score is then defined as

$$s(x, \text{FDR}) = \log \frac{\hat{a}x^{\hat{a}-1}}{\hat{a}\tau(\text{FDR})^{\hat{a}-1}} = (\hat{a} - 1)(\log(x) - \log(\tau(\text{FDR}))) \, .$$

Genes whose differential expression is considered significant given the FDR threshold obtain a positive score while genes showing no differential expression will receive a negative score. The size of the resulting module can be regulated with this FDR parameter. Throughout this study, FDR = 0.1 was used for all samples and species.

However, due to the experimental noise and paired design, the human samples have much higher intra-group variance, resulting in significant calls having p-values orders of magnitude higher than the mouse calls. This results in a range of scores that is much narrower for human than for mouse, possibly imbalancing results towards mouse modules. To correct for this effect, scores of mouse genes were rank normalized to the scores of the human genes as follows: the scores (as defined by the BUM model) were sorted, and for each gene the score of the i-th mouse gene was set to the score of the i-th human gene.

Comparison of the distribution of scores before and after normalization showed that compared to usual Benjamini-Hochberg FDR and log fold change cut-offs ($|\log \text{FC}| \geq 1$), the loss in statistical power was inconsequential and that this procedure ensured that mouse and human genes had comparable score distributions.

3.2.3 Network and orthology databases

The human and mouse background networks were downloaded from STRING v9.1, protein.actions.detailed.v9.1.txt (Franceschini et al. 2013), which is a database that contains experimentally verified direct protein interactions. Note that this network also contains interactions predicted based on orthology, so-called *interologs*. Ideally, we would prefer to use only experimentally predicted interactions, but currently, for mouse, such available data is too incomplete to result in a meaningful background network. Outlier nodes with a degree above 40 times the interquartile range plus the 75th percentile of the distribution of all node degrees were removed (ELAVL1, UBC, Ubb, Ubc). The resulting mouse network has $16,821$ nodes and $483,532$ edges and the human network has $16,255$ nodes and $315,442$ edges.

For any given timepoint, we performed a preprocessing step where we retained the subgraphs of the input networks induced by the genes that meet the microarray filtering criteria. This reduced the number of nodes to 8,453 human nodes, 6,882 mouse nodes and 14,779 nodes in the orthology mapping. Among these, up to 250 nodes (depending on the time point) have positive scores. The rank normalization as described in section 3.2.2 ensured that the number of positive human nodes is in the order of the number of positive mouse nodes.

Orthology information was downloaded from Ensembl release 59 (Flicek et al. 2013) and all human and mouse orthologs were kept, regardless of the identity scores. The orthology mapping corresponds to a bipartite graph involving $67,304$ human proteins and $43,953$ mouse proteins linked by $104,007$ edges, grouped in $16,552$ bicliques with an average size of 6.72 proteins (SD: 5.34).

3.2.4 Results

3.2.4.1 xHeinz identifies statistically significant conserved modules at different levels of conservation

We applied xHeinz on samples from the Th17 human and mouse data sets for time points 2 h, 4 h, 24 h, 48 h and 72 h. We solved these instances for different values of $\alpha \in [0, 1]$ with a step size of 0.1. All computations were done in single-thread mode on a desktop computer (Intel XEON e5 3 Ghz) with 16 Gb of RAM and a time limit of 12,000 CPU seconds. After this timeout, the best feasible solution is returned by the solver.

Figure 3.2 shows for the five time points and eleven values of the α parameter, the human and mouse scores of the found modules as well as the distribution of the module contents. For 26 of the 55 instances we solved the conserved active modules problem to provable optimality within the time and memory limit. The optimality gap of a solution is defined as $(UB - LB)/|LB|$,

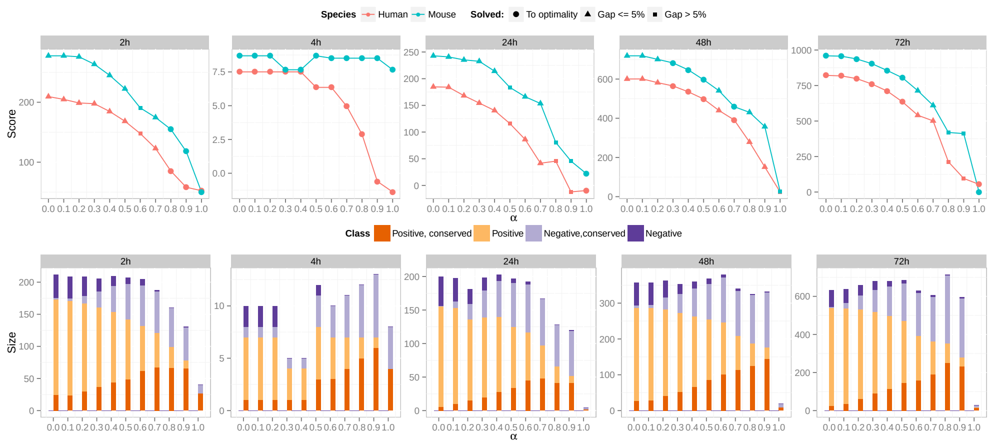

Figure 3.2: **Statistics of xHeinz solutions.** The conserved active module problem was solved for five time points (columns) over a sequence of 11 consecutive values of the α conservation parameter (x-axis). We report in the top row the score of the best solution (y-axis) and whether optimality was proven by our algorithm (circles). All runs were limited to 12,000 CPU seconds on a standard desktop computer. The second row illustrates how module contents vary as α increases. The height of each bar indicates the size of the respective module, colors indicate the fraction of positive and conserved nodes.

where LB and UB are the value of the best solution and the lowest upper bound as identified by the branch-and-cut algorithm, respectively. Of the 29 instances that are not solved to optimality, 22 have a gap smaller than 5%.

Any feasible solution for a conservation ratio of α is also a solution for any $\alpha' \leq \alpha$. We indeed see in fig. 3.2 that this property holds, the solution values decrease monotonically with increasing α. Consequently, if we obtain an optimal solution (*i.e.*, with maximal activity score) for α' then, any solution for α must have its activity score that is greater or equal to the one for α'. When we only account for the optimal solutions of our instances, we indeed see that this property holds.

As an added validation, we observe that the solutions for $\alpha = 0$ (no conservation constraints) are identical to the solutions obtained by running the single species method Heinz, described by Dittrich et al. (2008), separately on the two networks.

There is a sharp decrease in module size for $\alpha = 1$. Indeed, this is the most restrictive setting since it enforces that all the nodes in a module must be conserved. We also observe that as α increases, both positive and negative *conserved* nodes are added, indicating that we manage to retrieve informative nodes in a gradual manner. See also section 3.3.3 for a detailed analysis of module overlap for all combinations of α values.

When we compare solutions across time points, we see that the conserved active modules capture two phases of the differentiation process. We observe high activity at 2 h as well as at the late time points. Several authors reported such biphasic behavior during early Th17 differentiation, for example Ciofani et al. (2012) and Yosef et al. (2013) in mouse, and Tuomela et al. (2012) for human. The low activity score observed at the 4 h time point is in line with Yosef et al. (2013)'s mouse studies, which suggest that after the initial induction sustained by Stat3 and Stat1 in the first four hours, a phase of Rorc induction takes place and lasts until the 20 h time point, after which the effective protein level of Rorc starts to increase and to trigger the cytokine production phase. Our model and the solutions obtained suggest that these dynamics are conserved between the two organisms.

Furthermore, fig. 3.3 illustrates the strong conservation of these overall dynamics, in human and mouse at 2 h. The plots show that in our modules all but two conserved gene pairs change expression in the same direction. It is worth noting that we do not enforce any conservation of directionality in the xHeinz model.

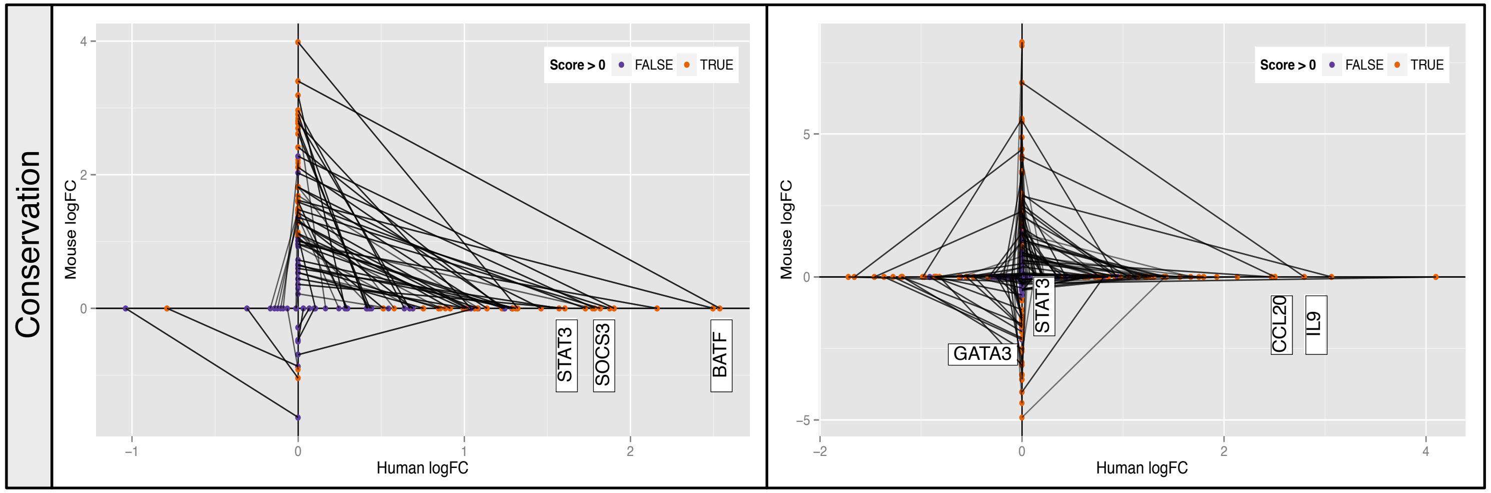

Figure 3.3: **Comparison of log fold change expression in mouse and human for conserved gene pairs at** 2 h **(left) and** 48 h **(right).** Each panel shows the log fold change correlation between conserved gene pairs: For each pair, a line segments connects the human logFC (x-axis) to the mouse logFC (y-axis). Point color indicates whether the human or mouse gene has a positive score. A line segment in the 1st or 3rd quadrant signifies positively correlated logFC values whereas a link in the 2nd and 4th quadrant corresponds to negative correlation. The sign of the activity score is indicated by the coloring. Genes discussed in the main text are indicated with white boxes.

3.2.4.2 Early regulation of Th17 differentiation is conserved between human and mouse

In the following, we study the two phases of the Th17 differentiation process in more detail. We focus on the 2 h and 48 h time points. We selected for this evaluation $\alpha = 0.8$ for both time points, as this value provides a balance between conservation and activity and produces modules of interpretable size. All results at all time points are available on the accompanying website. Figure 3.4 reports a reduced version of the resulting human and mouse Th17 modules for the two time points, and fig. 3.5 and fig. 3.6 show the full, unfiltered module contents of the modules.

We assess statistical significance of the resulting modules by performing 100 runs on randomized networks for each value of α, and additional 400 runs for the selected $\alpha = 0.8$. We do this using two randomization methods: (1) permuting the node weights while keeping the graph fixed, and (2) permuting the network topology while keeping the node weights and the node degrees fixed as described in (Mihail and Zegura 2003). With the exception of a few extreme cases at the 48 h time point, all modules were found to be highly significant. For details see section 3.3.3.

Our model for conserved modules relies on the hypothesis that similar biological processes between two related species are realized by orthologous genes. To evaluate the relevance of the conserved modules returned by xHeinz, we solved the conserved active modules problem between the Th0 and Th17 conditions at 2 h and 48 h.

At the 2 h time point, xHeinz identifies a conserved module consisting of 58 human and 50 mouse proteins. Interestingly, both the human and mouse modules are centered around STAT3/Stat3, even if these genes are not the ones showing the higher fold change in both species. STAT3 is a signal transducer having transcription factor activity and was shown to play a key role in the differentiation process of Th17 (Harris et al. 2007). Once activated by Th17 polarizing cytokines (such as IL6 in our case), it eventually binds to the promoter regions of IL17A/Il17a and IL17F/Il17f cytokines and activates transcription. These cytokines are the hallmark cytokines produced by activated Th17 cells. It is worth noting that IL17/Il17 cytokines and associated receptors are not in the 2 h modules, as these proteins have been shown to be expressed only at later time points (Tuomela et al. 2012). Moreover, STAT1/Stat1, another member of the STAT family, is part of the solution and belongs to the central core of the human and mouse modules, which is consistent with its major role during the early phases of Th17 differentiation (Yosef et al. 2013).

We also applied xHeinz to find a conserved module at a later time point (48 h). Kinetics analysis of Th17 differentiation showed that the effective secretion of Th17 hallmark cytokines only happens after several days of polarization (Tuomela et al. 2012; Yosef et al. 2013) and we do observe in these modules a significant enrichment for interleukin related proteins present in both species, which was absent for the 2 h modules, such as up-regulation of IL9/Il9. Secretion of IL9 by Th17 cells have been demonstrated both in mouse and human cells (Beriou et al. 2010), Il9 is known to be induced by Bcl3 (M. Richard et al. 1999), and Bcl3 inhibition has been recently shown to affect the function of Th17 cells in mouse (Ruan et al. 2010). We also observe the conserved down-regulation of GATA3/Gata3, which is known to be the master regulator of Th2 cells (Zheng and Flavell 1997), and is likely to constrain the Th17 regulation program (Hamburg et al. 2008). Similarly to the modules found at 2 h, the 48 h modules are centered around STAT3, although at the 48 h time point this gene is not differentially expressed anymore neither in human or mouse (resp. logFC of 0.17, score of -4.59 for human, and logFC 0.52, score of -3.21 for mouse). This observation is in line with the major role of STAT3 along the differentiation process at all time points (Yosef et al. 2013). To the contrary, STAT1 has been indicated as an exclusively early regulator (Yosef et al. 2013) in mouse and is indeed not present anymore in the 48 h modules.

We also observe the presence of the RORA/RORC/Rora/Rorc members of the RORs family of intracellular transcription factors, which are considered to be the master regulators of the Th17 lineage (X. O. Yang et al. 2008), and have been implicated in both species (Crome et al. 2009). Interestingly, these regulators are linked to the up-regulation of the vitamin-D receptor (VDR/Vdr), whose role in Th17 differentiation and several human auto-immune related disease have been recently studied (Chang et al. 2010).

In summary, our findings show the relevance of the identified conserved active modules with regard to the biological process of interest. By requiring the active modules to contain a certain fraction of conserved nodes, xHeinz identifies the main core proteins involved in the differentiation of Th17. Our analysis confirms that these proteins are very likely to have similar roles in both species.

3.2.4.3 Comparison to neXus

We compare the 48 h xHeinz modules (see figs. 3.4 to 3.6) with subnetworks computed by neXus version 3 (Deshpande et al. 2010). In contrast to our exact approach, neXus uses a heuristic technique to grow subnetworks from seed nodes simultaneously in two species in an iterative fashion. Neighborhoods of the two current modules are determined using a depth-first search. This search is restricted to only consider nodes that have a path to the seed node with a confidence larger than the user-specified parameter `dfscutoff`. The confidence of a path is defined as the product of the confidences of the edges comprising that path. The modules are extended to include the most active pair of orthologous nodes in the neighborhoods – where activity is defined as normalized log fold change and thus differs from the definition of activity used

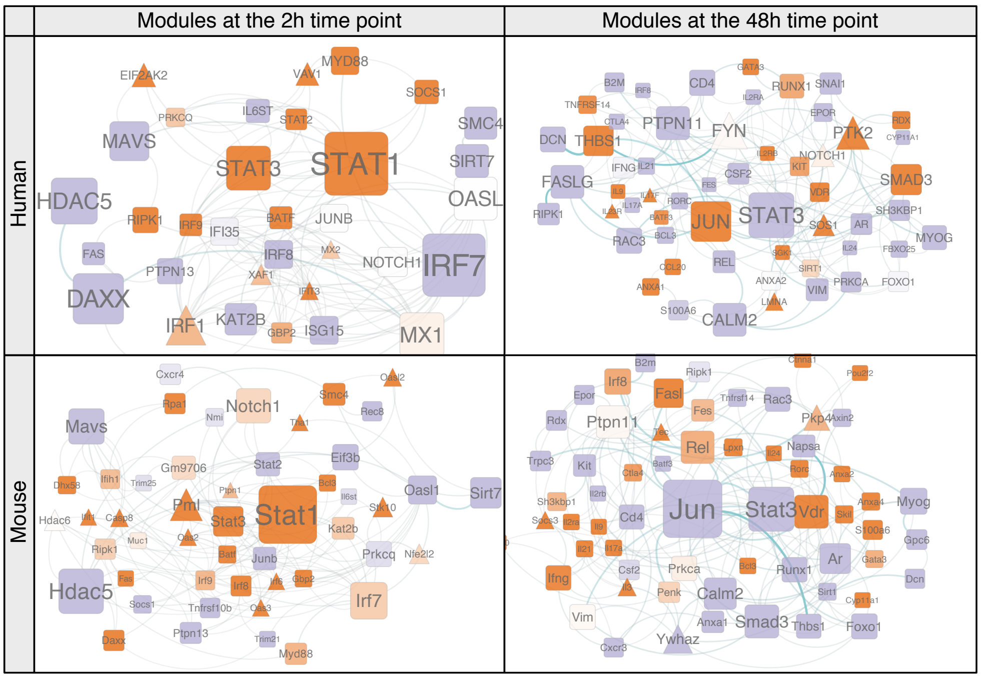

Figure 3.4: **Conserved active Th17 differentiation modules in human and mouse at** 2 h **and** 48 h**.** We obtained node activity scores capturing the significance of differential gene expression between the Th17 and Th0 conditions in human and mouse using the BUM model with FDR = 0.1. xHeinz uses these scores to search for conserved active modules in the STRING protein action network. The first row shows the human counterparts of the best scoring conserved modules for the 2 h (left) and 48 h (right) samples. The second row depicts the mouse counterparts. Rounded squares depict genes for which a homolog – as defined by Ensembl – is present in the counterpart, whereas triangles denote non-conserved genes. Node color gradually indicates activity scores. Orange: larger than 2; white: between -2 and 2; violet: smaller than -2. Node labels and sizes are proportional to betweenness centrality and edge width to edge-betweenness – both centralities are with respect to the subnetwork module. Only nodes having a degree larger than 2 (resp. 3) are displayed for the 2 h (resp. 48 h) module. The full networks in tabular format are available on the accompanying website and in appendix A.1.

in xHeinz. This whole procedure is repeated until either the cluster coefficient drops below the user-specified parameter `cc`, or the average activity scores of one of the two modules drops below parameter `scorecutoff`.

We ran neXus with the default parameters `cc` $= 0.1, 0.2$, `scorecutoff` $= 0.15$ and `dfscutoff` $= 0.3, 0.8$ for mouse and human respectively for all time points. Table 3.1 gives the resulting modules sizes for human and mouse.

solution	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	avg.	#sols
0.5h	7 (6)	4 (4)	7 (6)	3 (3)												5.25 (4.75)	4
1h	15 (10)	10 (9)	12 (12)	13 (13)	7 (7)	5 (5)	15 (13)	10 (11)	9 (10)	18 (16)	25 (24)	14 (14)	6 (7)	5 (5)	6 (6)	9.95 (9.58)	19
2h	15 (17)	6 (5)	12 (10)	12 (11)	10 (10)	13 (13)	17 (15)	12 (12)	8 (9)	5 (5)	11 (12)	19 (18)	3 (3)	9 (8)	23 (21)	10 (9.83)	30
4h	6 (9)	4 (4)	6 (5)	4 (4)	4 (4)	3 (3)	7 (8)	9 (10)	4 (4)	3 (3)						5 (5.40)	10
48h	5 (5)															5 (5)	1

Table 3.1: **Modules calculated with neXus for all time points.** Shown are the sizes in number of nodes of the first 15 representative solutions and the average sizes for the human subnetwork and for the mouse subnetwork in brackets. The last column lists the number of solutions for each time point. No solutions were obtained for time points 24 h and 72 h.

neXus finds 1 module for time point 48 h which is shown in fig. 3.7 for human (A) and mouse (B). In total 5 genes are contained in the module, which are identical for human and mouse, but the number of edges differs. Only one of the genes is significantly differentially expressed, CCL20, which has an absolute log fold change bigger than 1 and a BH FDR smaller than 0.1. Since neXus does not use p-values as an input, but log fold-changes which are normalized to activity values, the genes CCL20 and CXCR3 are considered as active nodes with a value above 0.15. These genes show changes in expression, but only two of these changes are statistically significant.

The low number of active nodes points to a drawback in the neXus algorithm: due to the locality of the greedy search strategy it may happen that the average activity of the subnetwork in construction keeps on degrading without reaching the next active node. The effects of this issue can be seen, for example, in fig. 3.7, where CCL20 is the seed node and the majority of other neighboring nodes are not differentially expressed. Furthermore, since the activity score of a single gene is just the log fold-change, and does not reflect both fold change and variability as a p-value, the neighboring nodes of the seed node in subnetwork fig. 3.7 might merely originate from the noise in the data and represent nothing biologically relevant for the interpretation of the data.

Another consequence of the neXus search strategy is that the module sizes are small (see table 3.1) and thus only give a limited view of the molecular mechanisms at play. Theoretically, the parameter `dfscutoff` can be decreased to increase the module size. Doing so, however, produces only slightly larger modules, but drastically increases the running time. Table 3.2 shows the neXus solutions for time point 48 h with different values for parameter `dfscutoff`.

Also note that changes in the clustering coefficient parameter `cc` only reduce

Table 3.2: **neXus solutions for time point** 48 h **with different values for parameter `dfscutoff`.** Shown are the number of solutions, the average and maximum number of nodes and running times for the human network.

dfscutoff	no. sols.	avg. size	max. size	CPU time [s]
0.1	2	5.00	5	176039.91
0.2	3	7.00	9	110282.61
0.3	5	6.60	9	77502.13
0.4	4	6.62	10	54972.92
0.5	6	5.58	9	35360.42
0.6	4	5.38	6	20538.4
0.7	3	5.33	6	11164.19
0.8	60	4.04	6	4639.59
0.9	0	0.00	0	1359.87

the module size with increasing `cc`. Table 3.3 shows the neXus solutions for time point 48 h with different values for parameter `cc`.

Table 3.3: **neXus solutions for time point** 48 h **with different values for parameter `cc`.** Shown are the number of solutions, the average and maximum number of nodes and running times for the human network.

cc	no. sols.	avg. size	max. size	CPU time [s]
0.1	1	5	5	23367.68
0.2	1	5	5	23243.83
0.3	1	5	5	23903.09
0.4	1	5	5	24285.30
0.5	1	3	3	24318.78
0.6	1	3	3	24059.59
0.7	1	3	3	24518.68
0.8	1	3	3	24321.92
0.9	1	3	3	24246.56

Conservation in neXus is enforced stringently by only allowing pairs of orthologous genes or genes that are only present in one of the networks to be included in the subnetworks (see fig. 3.7). This is too restrictive if the underlying mechanisms in the two species differ. For instance, for time point 48 hours and all but $\alpha = 1$ values, xHeinz finds the non-conserved gene IL23R (BH FDR 3.52e-8, score 14.50, logFC 1.38) in human, which is involved in Th17 autocrine signaling (Wei et al. 2007) but which is not differentially expressed in mouse. xHeinz also finds JUNB, which at the 2 hour time point is up-regulated in human data (BH FDR 1e-2, score 0.02, logFC 1.3) and not detected as differentially expressed in the mouse data (BH FDR 0.48, score -4.01, logFC 0.65). JUNB is a known partner of BATF with which it heterodimerizes preferentially during Th17 differentiation (Schraml et al. 2009), indicating its relevance. Both important genes would have been missed by a more restrictive conservation setting. Indeed, both neXus and xHeinz at $\alpha = 1$ fail to find these genes showing that a more flexible view on conservation is required to adequately deal with transferability.

3.3 Implementation and evaluation details

3.3.1 Implementation

xHeinz is implemented in modern C++14 using the best practice patterns recently introduced by Meyers (2014), and makes use of the boost libraries and the LEMON graph library (Dezső et al. 2011). CPLEX 12.6 is used as a black-box solver for our Mixed-Integer Linear Program formulation. The source code is publicly available in a git repository linked to from `http://software.cwi.nl/xheinz`.

Given two species, xHeinz takes as input:

1. a network for each of the species: G_1 and G_2,
2. a mapping between the nodes of the two networks,
3. scores associated to each of the nodes, *e.g.*, derived from the p-value of the moderated t-test,
4. the threshold value α, and
5. an optional time limit.

xHeinz returns two node sets corresponding to a solution found within the time limit together with an upper bound on the optimal objective value. If the objective value equals this upper bound, the computed solution is provably optimal.

3.3.2 Data processing pipeline

The full pipeline, implemented using Snakemake, from data downloading to running xheinz, goes as follows:

1. Retrieve human and mouse ENSEMBL orthologs, STRING species specific network

2. Retrieve human dataset from GEO (all time points, all conditions)

3. Annotate human probes with ENSEMBL

4. Select human probes based on variance filter

5. Perform linear modeling of the whole human dataset

6. Retrieve mouse dataset (all time points, all conditions)

7. Annotate mouse probes with ENSEMBL

8. Select mouse probes based on variance filter

9. Perform linear modeling of the whole mouse dataset

10. For each time point of interest:

    a) Call differentially expressed human genes by contrasting the Th17 with the Th0 condition $\Rightarrow$ p-value for each gene at this time point
    b) Call differentially expressed mouse genes by contrasting the Th17 with the Th0 condition $\Rightarrow$ p-value for each gene at this time point
    c) For an FDR threshold of 0.1, fit a BUM model for the human genes $\Rightarrow$ positive and negative scores for human genes
    d) For an FDR threshold of 0.1, fit a BUM model for the mouse genes $\Rightarrow$ positive and negative score for mouse genes
    e) Rank normalize the mouse score based on the human scores $\Rightarrow$ update the scores of the mouse genes
    f) Map human and mouse genes to proteins on the STRING network
    g) For each value of interest for the conservation threshold α:
        i. Run xHeinz with the following inputs:
            A. the human STRING network
            B. the mouse STRING network
            C. the BUM scored human proteins
            D. the BUM scored mouse proteins
            E. the human and mouse orthologs

3.3.3 Significance of results

We computed empirical p-values to assess the significance of the obtained scores at 2 h and 48 h, for each value of $\alpha \in \{0.1, 0.2, \ldots, 1.0\}$, with the following two procedures:

1. *Weights permutation:* We shuffled the node weights by generating random permutations of the activity scores of all genes.
2. *Topology permutation:* We repeated a million times the following operation: given two randomly selected edges A1–A2 and B1–B2, if the edges A1–B2 and B1–A2 are not present in the network we add them and remove the original edges, thus generating a random network with the same node weights and degree distribution.

For each resulting permuted network we applied a modified version of xHeinz a fixed number of times (500 times for $\alpha = 0.8$, 100 times otherwise). This modified version allows to check whether the optimal score on the new random network would exceed the best score we found on the original network.

To speed up these computations, we used the observation that solving a relaxed version of the conserved modules problem is sufficient, since we only need a sufficiently low upper bound (less than the score we obtained with unshuffled data) on the optimal values for those shuffled instances to make this decision.

We therefore consider the following ILP for the shuffled instances:

$$\max \sum_{v \in V_1 \cup V_2} w_v x_v \tag{3.23}$$

$$\text{s.t. } (3.2), (3.3), (3.4), (3.6) \tag{3.24}$$

that is, we drop the connectivity constraints. Note that the objective value is an upper bound of the optimal score of the original problem since this new ILP is a relaxation of the original one. In addition, we can stop the branch-and-cut algorithm as soon as the upper bound on (3.23) is lower than the best found solution of the original ILP on the unshuffled data. These two observations help to speed up the significance computations tremendously.

Table 3.4 shows the results of these runs, which we performed using a cumulative time limit of 500 h. It can be seen that only at extreme values of α for the 48 h time point the upper bound on (3.23) was not good enough to prove significance. This result was actually expected, since the original run was the only one with a very high gap (33.36%) at the end of the allocated timelimit.

Note that the p-values $\hat{p}$ that we demonstrate here are actually upper bounds to the underlying p-values p that we would obtain without the relaxation to the ILP.

For all other combinations, including the ones we chose to compute the Th17 modules and where we computed even more permutations, our procedure demonstrates that the signal in the real network is useful to obtain a statistically significant score.

Timepoint	α	FDR	k	k'	$\hat{p}$
2 h	0.1	0.1	100	0	0.0
2 h	0.2	0.1	100	0	0.0
2 h	0.3	0.1	100	0	0.0
2 h	0.4	0.1	100	0	0.0
2 h	0.5	0.1	100	0	0.0
2 h	0.6	0.1	100	0	0.0
2 h	0.7	0.1	100	0	0.0
2 h	**0.8**	**0.1**	**500**	**0**	**0.0**
2 h	0.9	0.1	100	0	0.0
2 h	1.0	0.1	100	0	0.0
48 h	0.1	0.1	100	7	**0.07**
48 h	0.2	0.1	100	4	**0.04**
48 h	0.3	0.1	100	0	0.0
48 h	0.4	0.1	100	0	0.0
48 h	0.5	0.1	100	0	0.0
48 h	0.6	0.1	100	0	0.0
48 h	0.7	0.1	100	0	0.0
48 h	**0.8**	**0.1**	**500**	**0**	**0.0**
48 h	0.9	0.1	100	0	0.0
48 h	1.0	0.1	100	100	**1.0**

Timepoint	α	FDR	k	k'	$\hat{p}$
2 h	0.1	0.1	100	0	0.0
2 h	0.2	0.1	100	0	0.0
2 h	0.3	0.1	100	0	0.0
2 h	0.4	0.1	100	0	0.0
2 h	0.5	0.1	100	0	0.0
2 h	0.6	0.1	100	0	0.0
2 h	0.7	0.1	100	0	0.0
2 h	**0.8**	**0.1**	**500**	**0**	**0.0**
2 h	0.9	0.1	100	0	0.0
2 h	1.0	0.1	100	0	0.0
48 h	0.1	0.1	100	7	0.0
48 h	0.2	0.1	100	4	0.0
48 h	0.3	0.1	100	0	0.0
48 h	0.4	0.1	100	0	0.0
48 h	0.5	0.1	100	0	0.0
48 h	0.6	0.1	100	0	0.0
48 h	0.7	0.1	100	0	0.0
48 h	**0.8**	**0.1**	**500**	**0**	**0.0**
48 h	0.9	0.1	100	0	0.0
48 h	1.0	0.1	100	20	**0.2**

Table 3.4: Results of significance experiments. For $\alpha \in \{0.1, 0.2, \dots, 1.0\}$ at time points 2 h and 48 h we computed upper bounds on k instances with permuted scores using the procedures described above (weights permutation on the left, topology permutation on the right). In k' of these cases, resulting upper bound was not lower than the score of the best found conserved active module on the original network, resulting in a p-value $p \leq \hat{p} = k'/k$. Values at the threshold $\alpha = 0.8$ we used to compute the Th17 modules and non-zero p-values are highlighted.

3.3.4 Robustness of modules for varying α

Given two modules $V_1' \subseteq V_1$ and $V_2' \subseteq V_2$, the Jaccard index is defined as $(V_1' \cap V_2')/(V_1' \cup V_2')$. Fig. 3.8 shows for each time point the Jaccard index for all pairs of conservation ratios. For consecutive values of α we can see that the module contents do not change much.

Fig. 3.9 shows the human gene module contents for time point 2 h for varying values of α, one gene per line. Genes in the left panel (FALSE) are negatively scored, that is, they are not differentially expressed at an FDR of 0.1. Genes in the right panel (TRUE) have a positive score. Squares represent conserved genes, whereas triangles represent non-conserved genes. An xHeinz-module solution is thus the set of genes marked with a square or triangle at a given value of α. The coloring is as follows:

- Genes that are selected at all values of α or are coloured red. These are the core genes.
- Genes that are selected at $\alpha = 0$ (no conservation) but not at $\alpha = 1$ are green.
- Genes that are selected by $\alpha = 1$ (strict conservation) but not at $\alpha = 0$ are turquoise.
- Genes that can only be picked by xHeinz(intermediate α values) are orchid.

This figure shows that a large fraction of genes only occur in an intermediate α regime (orchid color). Furthermore, the module content smoothly changes as α varies, which illustrates the robustness of the approach. xHeinzthus allows the investigator to make an informed choice on the conservation of genes in the modules, which is not possible with methods assuming no or strict conservation.

3.4 Discussion

We presented a module discovery method that simultaneously searches across two species for interesting gene sets. A key feature of the gene sets that we extract is the guaranteed lower bound on the number of conserved genes that they contain. This user defined lower bound, $\alpha \in [0, 1]$, provides a flexible conservation requirement between the two species, which allows for the discovery of conserved modules, including genes that would be missed with a stringent conservation requirement.

We have translated our model into an integer linear programming formulation and have devised and implemented an exact branch-and-cut algorithm that computes provably optimal or near-optimal conserved active modules in our model.

As a validation of our approach, our computational experiments for understanding the mechanisms underlying Th17 T cell differentiation in both mouse and human demonstrate that the flexibility in the definition of conservation is crucial for the computation of meaningful conserved active modules. We have found two conserved Th17 modules at time points 2 h ($\alpha = 0.8$) and 48 h ($\alpha = 0.8$) that thoroughly encompass the biphasic Th17 differentiation process. This result can not be revealed by requiring full conservation ($\alpha = 1$) or by independent modules without requiring conservation ($\alpha = 0$). Likewise, neXus, an alternative approach based on a stringent conservation model, is not able to capture the key regulatory program of the differentiation process.

A key characteristics of our model is its flexibility. This allows its extension to multiple species and time points, which we will address in future work. In this case, however, realistic instances will be harder to compute to optimality. Indeed, the number of interactions between multiple species or time points increase at least quadratically the number of both ILP variables and constraints. It would requires the development of powerful algorithm engineering techniques. Interestingly, the complexity of our branch-and-cut modelization of connectivity remains linear on the number of graphs involved.

We presented a general model and its MIP program to solve any instance of the cross-species module discovery problem. An extensive complexity analysis of the problem is presented in chapter 4, where we show that some instances can be solved in polynomial time with specific algorithms.

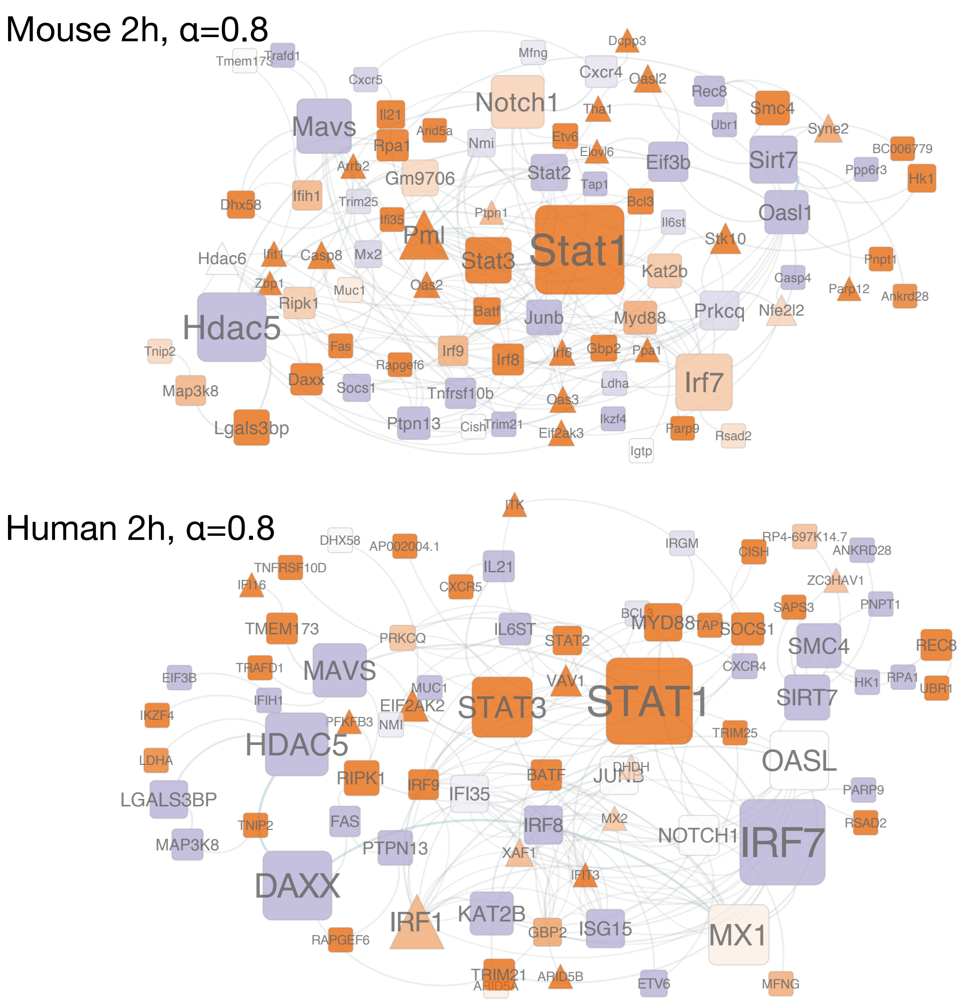

Figure 3.5: Full module at 2 h for $\alpha = 0.8$ conservation ratio

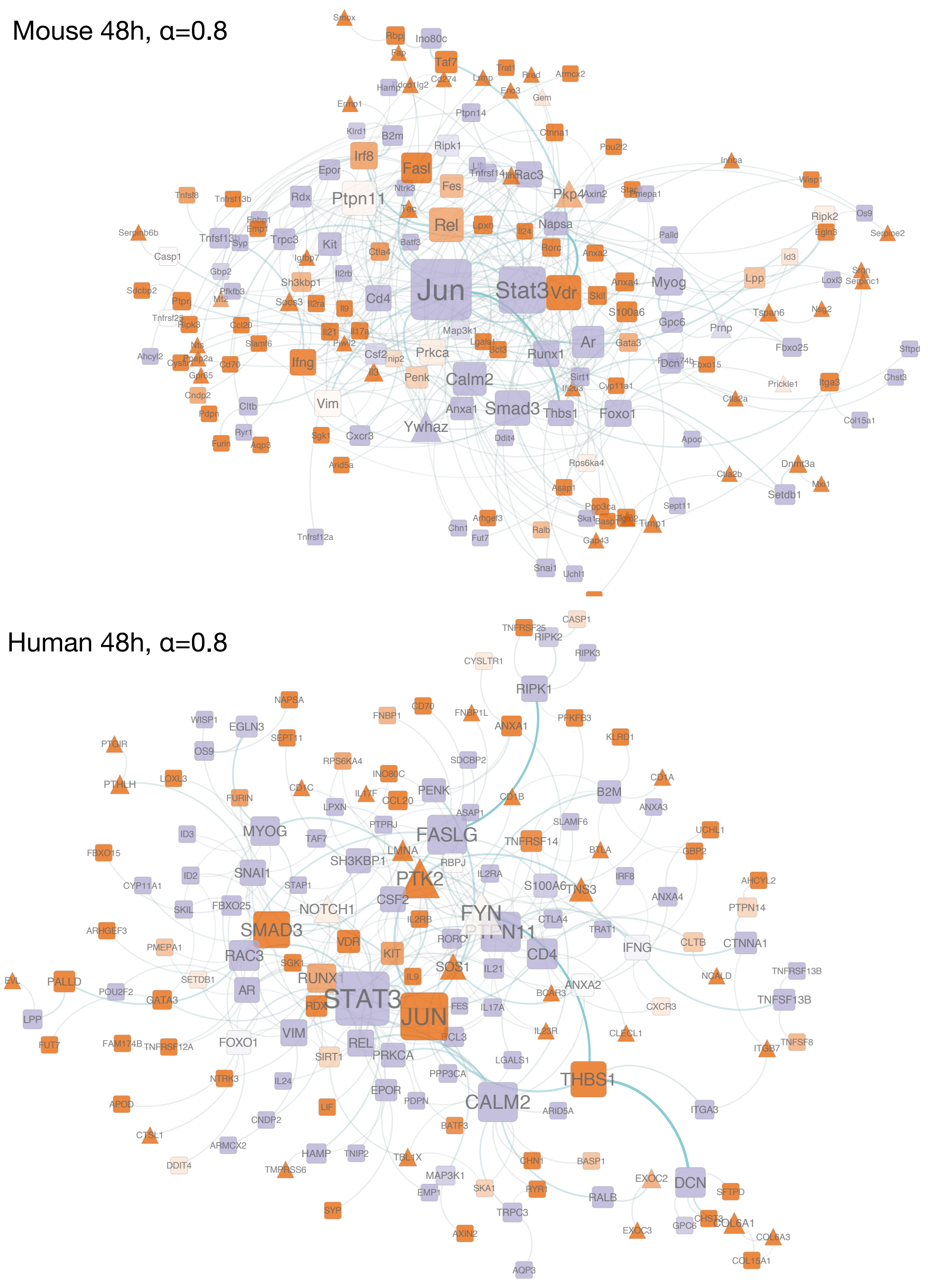

Figure 3.6: Full module at 48 h for $\alpha = 0.8$ conservation ratio

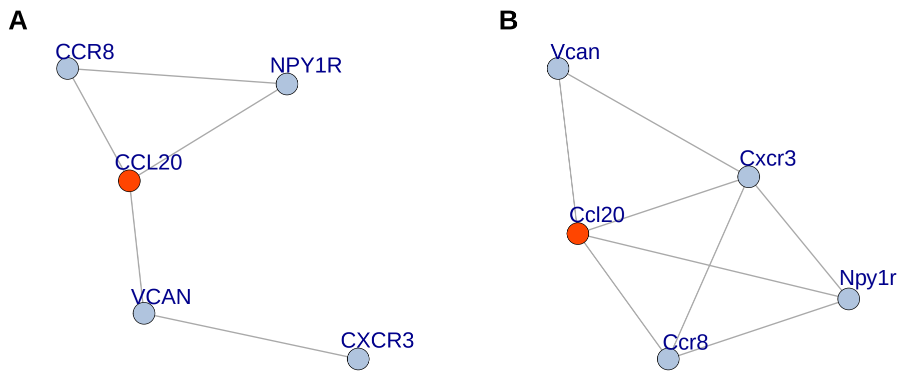

Figure 3.7: **neXus modules for the time point 48 hours for human (A) and mouse (B).** Orange coloring indicates genes with significant differential expression (BH FDR ≤ 0.1, $|\log \mathrm{FC}| \geq 1$). Here only one gene is significantly differentially expressed (CCL20).

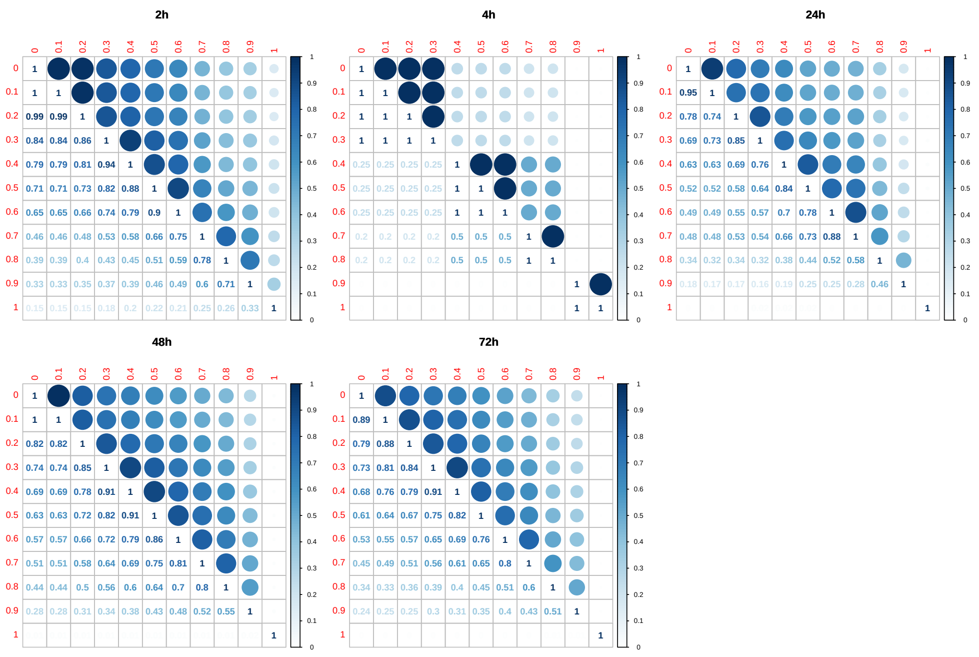

Figure 3.8: Jaccard index evolution over changes of α

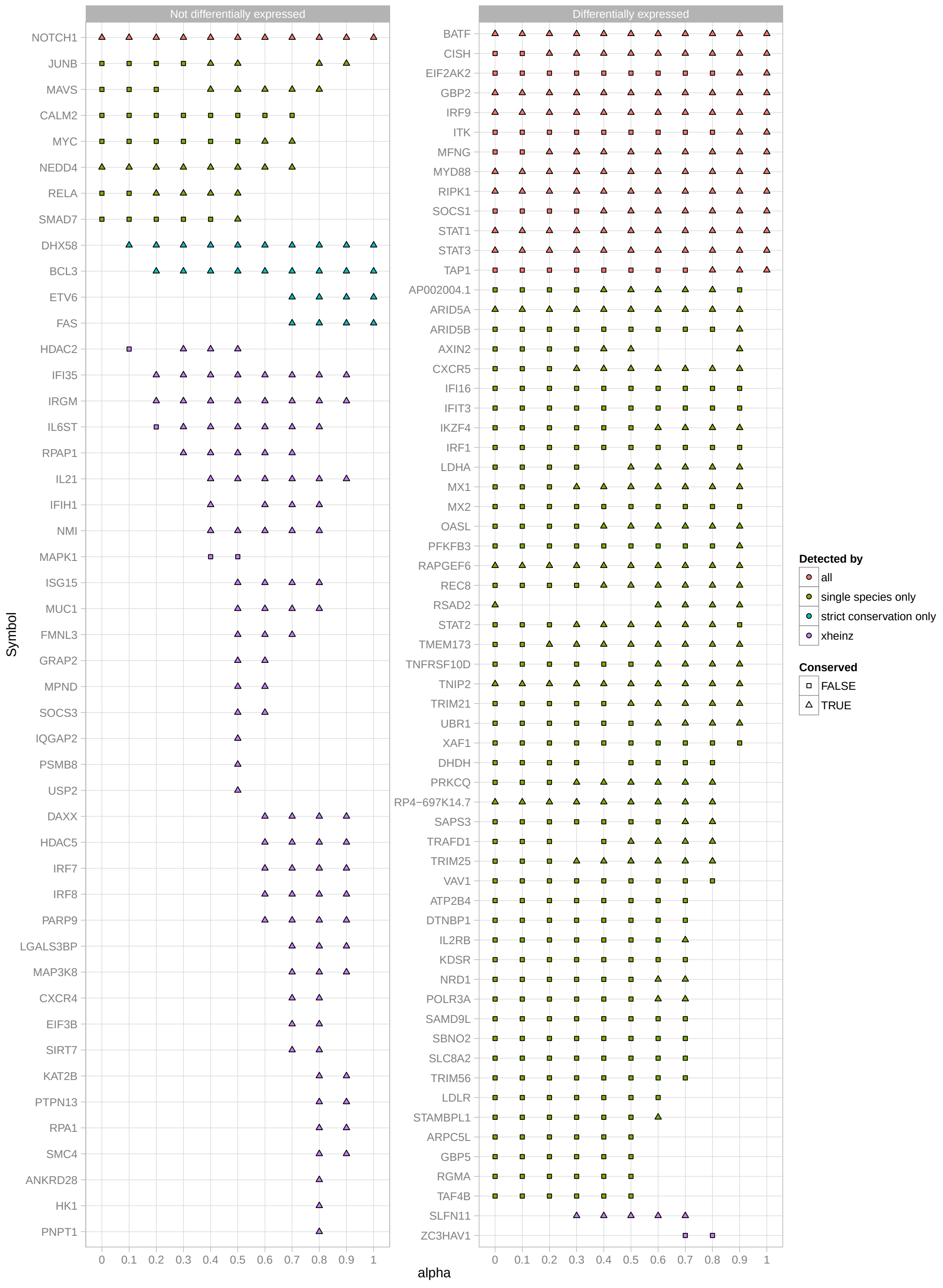

Figure 3.9: Human module contents of the 2 h time point with varying α

Chapter 4

Tight hardness bounds for the mwccs problem

In this chapter, we outline the frontier of complexity that characterizes the Maximum-Weight Cross-Connected Subgraph (mwccs) problem. We demonstrate that the bipartite relationship plays a major role in the separation between complexity classes, as do the types of input graphs. Furthermore, we provide constructive proofs, and algorithmic solutions that efficiently solve some instances of the problem in polynomial time.

The difficulty of the problem is not only dependent on the characteristics of the two main input graphs, but also of the relationship between the nodes of these two graphs. We hypothesised that the problem might be polynomial-time solvable in some instances with a simpler relationship function.

In section 4.1 we demonstrate that the mwccs problem is inapproximable (NP-hard to approximate) up to any arbitrary factor $\sigma < 1.0014$, it is thus an APX-hard problem. More precisely, we show that the APX-hardness holds for all inputs complex enough to represent the following cases:

1. one of the two graphs is a *binary caterpillar tree*[1], the other a *binary tree*, and the bipartite relationship is *injective*,

2. one of the two graphs is a *binary tree*, the other a general *graph*, and the bipartite relationship is *bijective*.

Section 4.1.1 contains the first proof, and section 4.1.2 contains the second.

Section 4.2 is separated into two main subsections dedicated to polynomially solvable cases of the mwccs. Both these cases are first formally defined, and their complexity is analyzed by reduction and constructive algorithmic

[1]In our original publication (Hume et al. 2015) we have erroneously used the term *comb graph*.

solutions are proposed. In Hume et al. 2015 we suggested that the problem might be solvable in polynomial time when both graphs are *trees* and the relationship is *bijective*; we give a proof for this conjecture in section 4.2.1. This draws a clear complexity frontier, dependent on the bipartite relationship. Finally, in section 4.2.2 we provide a general algorithm to solve MWCCS. Doing so we introduce a new problem, RB-MWCS, to which section 4.3 is dedicated. Given that one of the two graphs is polynomially enumerable, we provide an efficient reduction from an MWCCS instance to a polynomial number of RB-MWCS instances. In this case, MWCCS is as difficult as RB-MWCS: it is solvable in polynomial time when the second graph has a bounded treewidth. The details for the problem reduction are in section 4.2.2 and the RB-MWCS hardness analysis is in section 4.3.

The complexity is analyzed in a similar fashion in the two contexts. Adding constraints to one of the graphs allows to relax some of the constraints for the other graph, in order to stay within the same category of difficulty. In the context of the hardness proof, in order for the problem to stay difficult, the change from an injective to a bijective mapping function requires that one of the two trees become a general graph. In the context of polynomial-time algorithms, requiring that one of the graph that was previously a binary tree becomes more general requires that the other becomes polynomially enumerable to stay solvable in polynomial time.

4.1 APX-hardness of the mwccs problem

In this section we prove the inapproximability of two specific cases of the MWCCS problem. First, we prove that if the mapping between G_1 and G_2 is an injective function, if G_1 is a binary caterpillar tree, and if G_2 is a binary tree, MWCCS is APX-hard and can not be approximated within factor 1.0014. Then, we prove that if the mapping is a bijective function, the problem is as hard to approximate as when considering a binary tree and a graph. These results in themselves illustrate the role of the mapping function with respect to the difficulty of the problem.

Both proofs consist in L-reductions from the APX-hard MAX-3SAT(B) problem (see chapter *state of the art*).

4.1.1 Injective mapping, binary trees

Proposition 1. *The* MWCCS *problem for a binary caterpillar tree and a binary tree is APX-hard and not approximable within factor* 1.0014 *even when the mapping M is an injective function.*

We first describe how we build an instance of MWCCS corresponding to an instance of MAX-3SAT(B). Given any instance (C_q, V_n) of MAX-3SAT(B), we build a binary caterpillar tree $G_1 = (V_1, E_1)$ with weight function w_1, a binary tree $G_2 = (V_2, E_2)$ with weight function w_2, and a mapping M as follows.

The binary caterpillar graph G_1 is defined as follows. The vertex set is $V_1 = \{r, l_i, c_j, dl_i,\ dc_j \mid\ 1 \leq i \leq n, 1 \leq j \leq q\}$. The edge set is given by the following equation.

$$\begin{aligned} E_1 =& \{(c_j, dc_j), (l_i, dl_i) \mid 1 \leq i \leq n, 1 \leq j \leq q\} \cup \\ & \{(dc_q, r), (r, dl_1)\} \cup \\ & \{(dc_j, dc_{j+1}), (dl_i, dl_{i+1}) \mid 1 \leq i < n, 1 \leq j < q\}. \end{aligned}$$

The weight function w_1 is defined as follows: for all $1 \leq i \leq n$ and $1 \leq j \leq q$, $w_1(l_i) = B$, $w_1(c_j) = 1$ and $w_1(r) = w_1(dc_j) = w_1(dl_i) = 0$.

Roughly, in G_1 there is a node for each clause (denoted by c_j) and for each literal (denoted by l_i) that represent the leaves of the caterpillar. The spine of the caterpillar contains dummy nodes for each clause (denoted by dc_j) and for each literal (denoted by dl_i) separated by a central node (denoted by r).

The binary tree $G_2 = (V_2, E_2)$ with weight function w_2 is defined as follows. The vertex set is $V_2 = \{r, x_i, \overline{x_i}, c_j^k, dx_i, d\overline{x_i}, dc_j^i, dc_j^{\overline{i}} \mid 1 \leq i \leq n, 1 \leq j \leq q, 1 \leq k \leq 3\}$. The edge set E_2 is given by the following equation.

$$\begin{aligned} E_2 =& \{(r, dx_n)\} \cup \\ & \{(c_j^{k'}, dc_j^k) \mid x_k, \text{is the } k'\text{-th literal of clause } c_j\} \cup \\ & \{(c_j^{k'}, dc_j^{\overline{k}}) | \overline{x_k}, \text{is the } k'\text{-th literal of clause } c_j\} \cup \\ & \{(dx_i, d\overline{x_{i+1}}) | 1 \leq i < n\} \cup \\ & \{(dx_i, d\overline{x_i}), (dx_i, x_{n-i+1}), (d\overline{x_i}, \overline{x_{n-i+1}}), (x_i, dc_1^i), (\overline{x_i}, dc_1^{\overline{i}}) | 1 \leq i \leq n\} \cup \\ & \{(dc_j^i, dc_{j+1}^i), (dc_j^{\overline{i}}, dc_{j+1}^{\overline{i}}) | 1 \leq i \leq n, 1 \leq j < q\} \end{aligned}$$

The weight function w_2 is defined as follows: for all $1 \leq i \leq n$, $1 \leq j \leq q$ and $1 \leq k \leq 3$, $w_2(x_i) = w_2(\overline{x_i}) = -B$ and $w_2(r) = w_2(c_j^k) = w_2(dx_i) = w_2(d\overline{x_i}) = w_2(dc_j^i) = w_2(dc_j^{\overline{i}}) = 0$

Roughly, in G_2 there is a node for each literal of each clause (denoted by c_j^k) and for each value of each literal (denoted by x_i and $\overline{x_i}$). Dummy nodes for literals have been duplicated (one for each value of the literal - that is dx_i and $d\overline{x_i}$). Dummy nodes for clauses have also been duplicated (one for each value of all literals - dc_j^i and $dc_j^{\overline{i}}$). The structure is not as easy to informally describe as for G_1 but the reader may refer to an illustration provided in Figure 4.1.

Finally, the mapping M is an injective function from V_1 to V_2 defined as follows.

$$M(r) = r$$

$$
\begin{aligned}
M(l_i) &= \{x_i, \overline{x_i}\}, \text{for all } 1 \le i \le n \\
M(c_j) &= \{c_j^k | 1 \le k \le 3\}, \text{for all } 1 \le j \le q \\
M(dl_i) &= \{dx_i, d\overline{x_i}\}, \text{for all } 1 \le i \le n \\
M(dc_j) &= \{dc_j^i, dc_j^{\bar{i}}\}, \text{for all } 1 \le i \le n \text{ and } 1 \le j \le q
\end{aligned}
$$

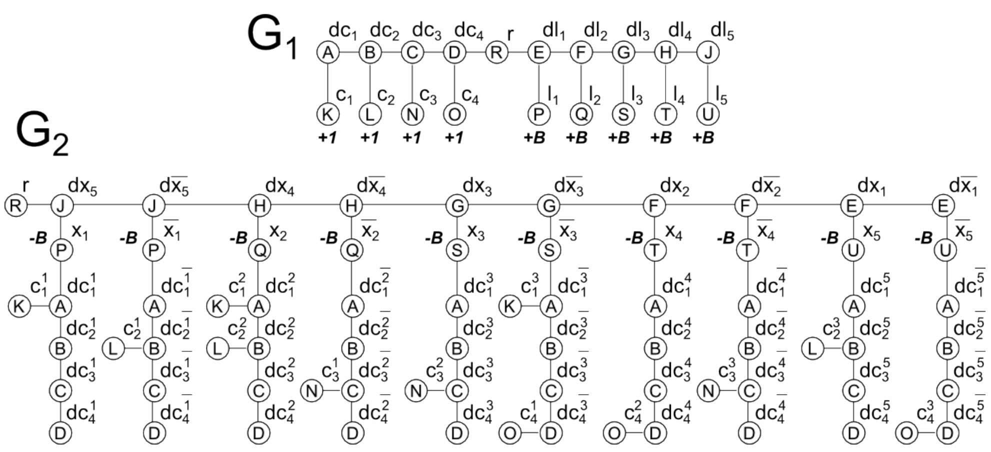
G1
dc1 dc2 dc3 dc4 r dl1 dl2 dl3 dl4 dl5
c1 c2 c3 c4 l1 l2 l3 l4 l5
+1 +1 +1 +1 +B +B +B +B +B
G2
r dx5 dx̄5 dx4 dx̄4 dx3 dx̄3 dx2 dx̄2 dx1 dx̄1
-B

Figure 4.1: Illustration of the construction of G_1, G_2, and M, given $C_q = \{(x_1 \vee x_2 \vee \neg x_3), (\neg x_1 \vee x_2 \vee x_5), (\neg x_2 \vee x_3 \vee \neg x_4), (\neg x_3 \vee x_4 \vee \neg x_5)\}$. For readability, the mapping M is not drawn but represented as labels located on the nodes: any pair of nodes (one in G_1 and one in G_2) of similar inner label are mapped in M.

Let us prove that this construction is indeed an L-reduction from MAX-3SAT(B). More precisely, we prove the following property.

Lemma 1. *There exists an assignment of V_n satisfying at least m clauses of C_q if and only if there exists a solution to* MWCCS *of weight at least m.*

Proof. $\boxed{\Rightarrow}$ Given an assignment $\mathcal{A}$ of V_n satisfying m clauses of C_q, we construct a solution to MWCCS of weight m as follows.

$$
\begin{aligned}
\text{Let } V_1^* =& V_1 \setminus \{c_j \mid c_j \text{ is not satisfied by the assignment}\} \text{ and} \\
V_2^* =& \{r\} \cup \\
& \{c_j^k \mid c_j \text{is satisfied by its } k\text{-th literal}\} \cup \\
& \{x_i, dc_j^i \mid x_i = 1, 1 \le j \le q\} \cup \\
& \{\overline{x_i}, dc_j^{\bar{i}} \mid x_i = 0, 1 \le j \le q\} \cup \\
& \{dx_i, d\overline{x_i} \mid 1 \le i \le n\}.
\end{aligned}
$$

By construction, $G_1[V_1^*]$ is connected since all the vertices of the spine of the caterpillar have been kept. Moreover, $G_1[V_1^*]$ contributes $B \times n + m$ to the overall weight of the solution, that is B for each of the l_i and $+1$ for each satisfied clause. By construction, all the sub-trees rooted at x_i (resp. $\overline{x_i}$) are kept in $G_2[V_2^*]$ if $x_i = 1$ (resp. $x_i = 0$) in $\mathcal{A}$. Moreover, all the dummy nodes for literals (dx_i and $d\overline{x_i}$) and the root r have been kept. Thus, $G_2[V_2^*]$ is also connected. Furthermore, $G_2[V_2^*]$ contributes to $-B \times n$ to the overall weight of the solution since exactly one of each variable node (x_i and $\overline{x_i}$) has been kept. One can easily check that any node of V_1^* has a mapping counterpart in V_2^*. The overall solution is valid and of total weight m.

$\boxed{\Leftarrow}$ Given any solution $\{V_1^*, V_2^*\}$ to MWCCS of weight m, we construct a solution to the MAX-3SAT(B) problem satisfying at least m clauses as follows.

First, note that we can assume that any such solution to MWCCS is *canonical*, meaning that V_2^* does not contain both vertices x_i and $\overline{x_i}$ for all $1 \leq i \leq n$. Indeed, by contradiction, suppose there exists a solution such that $\{x_i, \overline{x_i}\} \subseteq V_2^*$ for a given $1 \leq i \leq n$. Then, $\{x_i, \overline{x_i}\}$ in G_2 induce a negative weight of $-2B$. This negative contribution can at most be compensated by the weight of the corresponding literal node in G_1 ($w_1(l_i) = B$) and at most B clause nodes in G_1 ($B \geq \sum w_1(c_j)$ where $x_i \in c_j$ *or* $\overline{x_i} \in c_j$) since every literal occurs in at most B clauses in C_q. Therefore, such local configuration does not provide any positive contribution to the solution and can be transformed into a better solution by removing one of the sub-trees rooted in $\{x_i, \overline{x_i}\}$. We will consider hereafter that m is the weight of the resulting canonical solution. We further assume that $m > 1$ since otherwise we can build a trivial assignment $\mathcal{A} = \{c_1^1 = 1\}$ of V_n that is satisfying at least one clause of C_q.

Let $\mathcal{A}$ be an assignment of V_n such that for all $1 \leq i \leq n$ if $x_i \in V_2^*$ then $x_i = 1$ and $x_i = 0$ otherwise. Note that, since our solution is canonical, each literal has been assigned a single boolean value in $\mathcal{A}$. Let us now prove that this assignment satisfies at least m clauses of C_q.

First, note that since our solution is canonical and we require any node of V_1^* to have a mapping counterpart in V_2^*, this implies that if $l_i \in V_1^*$ then its contribution (that is $w_1(l_i) = B$) is cancelled by the negative contribution of either x_i or $\overline{x_i}$ in V_2^* (that is $w_2(x_i) = w_2(\overline{x_i}) = -B$). Therefore, the weight m of the solution can only be realized by m clause nodes of G_1, say $\mathcal{C}_1 \subseteq V_1^*$ – since $w_1(c_j) = 1$ for all $1 \leq j \leq q$.

As already stated, to be part of the solution any node in V_1^* has a mapping counterpart in V_2^*. Thus, for each node in $\mathcal{C}_1$, there should be a node of $\mathcal{C}_2 \subseteq \{c_j^k \mid 1 \leq j \leq q, 1 \leq k \leq 3\}$ in V_2^*. More precisely, by construction, any node c_j in V_1 has exactly three mapping counterparts in V_2 (that is $\{c_j^k \mid 1 \leq k \leq 3\}$) and for each $c_j \in \mathcal{C}_1$ at least one of these mapping counterparts has to belong to $\mathcal{C}_2$.

Finally, since both $G_1[V_1^*]$ and $G_2[V_2^*]$ have to be connected, each node in $\mathcal{C}_2$, say c_j^k, should be connected by a path to a node x_i or $\overline{x_i}$, say x_i, for some $1 \leq i \leq n$, in $G_2[V_2^*]$. By construction, this is the case if x_i is the k-th literal of the clause c_j for some $1 \leq k \leq 3$. Thus, $\mathcal{A}$ is an assignment that satisfies any clause c_j such that the clause node c_j belongs to V_1^*. As already stated $|\mathcal{C}_1| = m$. □

The above reduction linearly preserves the approximation since the weights of optimal solutions of the problems correspond and there exists an assignment of V_n satisfying at least m clauses of C_q if and only if there exists a solution to MWCCS of weight at least m. Hence, given an approximation to MWCCS, one can derive an algorithm for MAX-3SAT(B) with the same approximation ratio. Since MAX-3SAT(B), $B \geq 3$, is APX-hard (Papadimitriou and Yannakakis 1991) and MAX-3SAT(B) for $B = 6$ is not approximable within factor 1.0014 (Berman and Karpinski 1999), so is MWCCS, which proves Proposition 1.

Let us now prove a similar result for MWCCS problem when the mapping is a bijective function.

4.1.2 Bijective relationship function, tree, and graph

Proposition 2. *The* MWCCS *problem for a graph and a tree is APX-hard and not approximable within factor* 1.0014 *even when the mapping is a bijective function.*

Given any instance (C_q, V_n) of MAX-3SAT(B), we build a graph $G_1 = (V_1, E_1)$ with weight function w_1, a tree $G_2 = (V_2, E_2)$ with weight function w_2 and a mapping M as follows. The graph G_1 has the vertex set $V_1 = \{r, l_i, x_i, \overline{x_i}, c_j, c_j^k \mid 1 \leq i \leq n, 1 \leq j \leq q, 1 \leq k \leq 3\}$ and the edge set defined by the following equation.

$$\begin{aligned} E_1 =& \{(l_i, x_i), (l_i, \overline{x_i}), (r, x_i), (r, \overline{x_i}) \mid 1 \leq i \leq n\} \cup \\ & \{(c_j, c_j^k), (r, c_j^k) \mid 1 \leq k \leq 3, 1 \leq j \leq q\}. \end{aligned}$$

The weight function w_1 is defined as follows: for all $1 \leq k \leq 3, 1 \leq i \leq n$ and $1 \leq j \leq q$, $w_1(l_i) = B$, $w_1(c_j) = 1$ and $w_1(r) = w_1(c_j^k) = w_1(x_i) = w_1(\overline{x_i}) = 0$.

Roughly, in G_1 there is a node for each clause (denoted by c_j), for each of the three literals of each clause (denoted by c_j^k), for each literal (denoted by l_i) and for each valuation of each literal (denoted by x_i, $\overline{x_i}$). Clause nodes and literal nodes are separated by a central node r.

The tree G_2 is defined as follows. The vertex set is $V_2 = V_1$, the edge set is given by the following equation:

$$E_2 =\{(l_i, r), (c_j, r), (x_i, r), (\overline{x_i}, r) \mid 1 \leq i \leq n, 1 \leq j \leq q\} \cup$$

$$\{(c_j^k, x_i) \mid x_i \text{ is the } k\text{-th literal of clause } c_j\} \cup$$
$$\{(c_j^k, \overline{x_i}) \mid \overline{x_i} \text{ is the } k\text{-th literal of clause } c_j\}.$$

The weight function w_2 is defined as follows: for all $1 \leq k \leq 3$, $1 \leq i \leq n$ and $1 \leq j \leq q$, $w_2(x_i) = w_2(\overline{x_i}) = -B$, $w_2(r) = w_2(c_j^k) = w_2(l_i) = w_2(c_j) = 0$.

Roughly, in G_2 all the nodes except the ones in $\{c_j^k \mid 1 \leq j \leq q, 1 \leq k \leq 3\}$ form a star centered in node r. The nodes representing the literal of the clause (that is c_j^k) are connected to their corresponding variable nodes (that is x_i or $\overline{x_i}$).

Finally, the mapping M is a bijective function from V_1 to V_2 defined as the identity (that is each node in V_1 is mapped to the node of similar label in V_2).

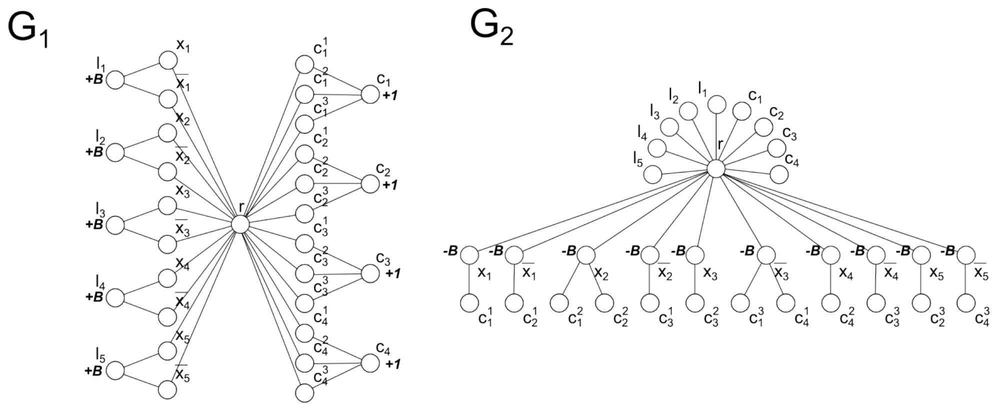

Figure 4.2: Illustration of the construction of G_1, G_2, and M, given $C_q = \{(x_1 \vee x_2 \vee \neg x_3), (\neg x_1 \vee x_2 \vee x_5), (\neg x_2 \vee x_3 \vee \neg x_4), (\neg x_3 \vee x_4 \vee \neg x_5)\}$. For readability, the mapping M is not drawn but deduced from the labels of the nodes; any pair of nodes (one in G_1 and one in G_2) of similar label are mapped in M.

Let us prove that this construction is indeed an L-reduction from MAX-3SAT(B). More precisely, we prove the following property.

Lemma 2. *There exists an assignment of V_n satisfying at least m clauses of C_q if and only if there exists a solution (not necessarily optimal) to* MWCCS *of weight at least m.*

Proof. $\boxed{\Rightarrow}$ Given an assignment $\mathcal{A}$ of V_n satisfying m clauses of C_q, we construct a solution to MWCCS of weight m as follows.

Let $V_1^* = V_2^* = \{c_j \mid c_j \text{ is satisfied by } \mathcal{A}\} \cup \{c_j^k \mid c_j^k \text{ is satisfying } c_j \text{ by } \mathcal{A}\} \cup \{x_i \mid x_i = 1\} \cup \{\overline{x_i} \mid x_i = 0\} \cup \{r, l_i \mid 1 \leq i \leq n\}$.

By construction, $G_1[V_1^*]$ and $G_2[V_2^*]$ are connected. Moreover, $G_1[V_1^*]$ contributes $B \times n + m$ to the overall weight of the solution, that is B for each of the l_i and $+1$ for each satisfied clause, while $G_2[V_2^*]$ contributes $-B \times n$ to the overall weight of the solution since exactly one of each variable node (*i.e.*, x_i and $\overline{x_i}$) has been kept. The overall solution is valid and of total weight m.

$\boxed{\Leftarrow}$ Given any solution $V^* \subseteq V_1$ to MWCCS of weight m, we construct a solution to the MAX-3SAT(B) problem satisfying at least m clauses as follows.

First, note that, as in the previous construction, we can assume that any such solution to MWCCS is *canonical* meaning that V^* does not contain both vertices x_i and $\overline{x_i}$ for any $1 \leq i \leq n$.

Let $\mathcal{A}$ be an assignment of V_n such that for all $1 \leq i \leq n$, if $x_i \in V^*$ then $x_i = 1$ and $x_i = 0$ otherwise. Note that, since our solution is canonical, each literal has been assigned a single boolean value in $\mathcal{A}$. Let us now prove that this assignment satisfies at least m clauses of C_q.

First, note that since our solution is canonical, as in the previous construction, the weight m of the solution can only be induced by m clause nodes of G_1, say $\mathcal{C}_1 \subseteq V^*$.

Since both $G_1[V^*]$ and $G_2[V^*]$ have to be connected, any solution with $m > 1$ will include node r in V^*. Thus, for each node $c_j \in \mathcal{C}_1$ there should be a node of $\{c_j^k \mid 1 \leq k \leq 3\}$ in $G_1[V^*]$ to connect c_j to r. In $G_2[V^*]$, in order for nodes r and c_j^k to be connected, the corresponding literal node (that is x_i or $\overline{x_i}$), say x_i – has to be kept in V^*. By construction, this is the case if x_i is the k-th literal of clause c_j. Thus, $\mathcal{A}$ is an assignment that satisfies any clause c_j such that the clause node c_j belongs to V^*. As already stated $|\mathcal{C}_1| = m$. $\square$

The above reduction linearly preserves the approximation and proves Proposition 2.

4.2 Polynomial-time cases for the mwccs problem

4.2.1 Two bounded-degree trees and a bijective mapping

We have proved in section 4.1.1 that for instances with two binary trees, and an injective mapping function, the problem is APX-hard. We will now consider the case of two bounded-degree trees, $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$, with a bijective relationship, or mapping function, $V_1 \, M \, V_2$. Without loss of generality, lets now suppose that their respective degrees are $d_{T_1} = d_{T_2} = d$, and that $|V_1| = |V_2| = n$.

Since $\alpha = 1$, a node can only be selected if its counterpart is also selected. However, note that a node in one of the trees can have its counterpart at a completely different depth in the other tree (see fig. 4.3). Consequently, adding a child node might require the addition of a parent by mutual requirement between the two trees and the mapping function (see fig. 4.3). This exhibit the main difficulty with this problem: the non-isomorphic mapping function makes it impossible to use the usual local dynamic programming scheme to solve the MWCS problem and its variants on trees.

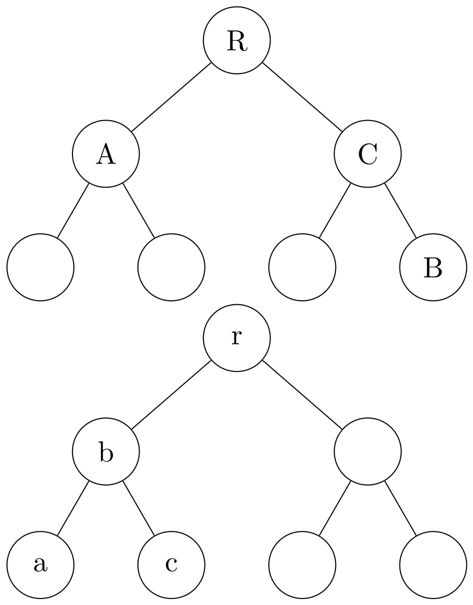

Figure 4.3: Nodes labeled with the same letter are counterpart of one another. When starting from an empty solution[2], the inclusion of node A requires only the inclusion of a. However when starting from the root nodes, the inclusion of node A requires the inclusion of node a, which recursively requires the inclusions of b, B, C, and c. A local dynamic programming approach is impossible since the context is important.

In this section, we introduce a *functional dynamic programming* approach for this setup that solve the MWCCS problem in polynomial time parameterized on the degree of the trees. A functional dynamic programming approach is a *top-down* dynamic programming technique, that is recursive and using subproblem overlaps, that uses a *memoization* scheme to store intermediate results instead of the usual tabular approach.

Our algorithm is composed of two main operations. ComputeDecisionTree is the main functionnal dynamic programming procedure and finds the optimal solution to the problem given two trees and a root for one of them. ComputeDecisionTree calls upon ConstructRequiredSet to maintain the connectivity and matching constraints of the candidate solution.

[2]As is the case with regular dynamic programming, which depend only on local subproblems.

ConstructRequiredSet: Informally, ConstructRequiredSet takes as input all the nodes that have been included up intil now, from both T_1 and T_2, and add into those sets all nodes that are required to be added if u is included to keep 1) connectivity and 2) perfect matching (see fig. 4.3).

Adding a node $u \in V_1$ from T_1 to a candidate solution[3] requires the unconditional inclusion of its counterpart $v \in V_2$ from T_2 to the candidate solution. In turn, the addition of v requires the addition of all nodes from T_2 that are necessary for the candidate solution to remain connected: the (possibly empty) shortest path $v_0 v_1 \ldots v_i$ between v and, if one exists, its closest node from T_2 already in the solution. Recursively, this requires the addition of all counterparts of the nodes $v_0, v_1, \ldots, v_i$ to the candidate solution, and the nodes required for the candidate solution to remain connected. This recursion continues until no more nodes are required to be included to conserve connectivity.

It is formally defined as follows. Given two sets $S_1 \subseteq V_1$ and $S_2 \subseteq V_2$ of nodes already selected, and a node $u \in V_1 \setminus S_1$ such that u is a neighbor of a node in S_1 ($v \in V_2$ its counterpart). We define a double recursion procedure over V_1 and V_2. Add u to S_1; then, find the shortest path from its counterpart v to any node of S_2. Add all nodes that pertain to this shortest path to S_2 (including v), and recursively find all shortest paths from their counterparts to the nodes of S_1. Continue until no more nodes need to be added.

Proposition 3. *The induced graphs $T_1[S_1]$ and $T_2[S_2]$ obtained as results of ConstructRequiredSet are connected.*

Proof. Indeed, since we recursively add nodes with their shortest path to already selected nodes, there exists a path connecting every pairs of nodes of S_1 and of S_2. $\square$

Proposition 4. *ConstructRequiredSet is a quadratic operation in the worst case.*

Proof. The shortest path toward a set of nodes in a tree is equivalent to the minimum sum subsequence problem and can be solved in linear time using a dynamic programming approach. We recursively add up to $O(n)$ nodes (all). Hence the resulting complexity. $\square$

ComputeDecisionTree: The main procedure, ComputeDecisionTree, is a recursive decision tree traversal of T_1 that uses the ConstructRequiredSet operation. It is formally defined as follows. Given T_1, T_2, and a node $u \in V_1$, this function returns the set S_1^u and S_2^u: the best selection of nodes, respectively

[3]At least one of the neighbors of u is in the candidate solution, or the candidate solution is empty. That is, its addition leaves the candidate solution connected.

from T_1 and T_2, when optimizing in the subtree rooted in u and which includes u. Note that S_1^u might contain nodes that are parents to u, as a requirement for the matching and connectivity constraints. Lets define u_i the i-th children of u, and $C^u = \bigcup\limits_i \{u_i\}$ the set of these children. For all nodes $u_i \in C^u$, if any:

1. recursively call ComputeDecisionTree with u_i, which will return the sets $S_1^{u_i}$ and $S_2^{u_i}$, then
2. call the ConstructRequiredSet procedure with $S_1^{u_i}$, $S_2^{u_i}$, and u, which will return $\overset{\circ}{S}{}_1^{u_i}$ and $\overset{\circ}{S}{}_2^{u_i}$.

Then compute a score $s(u)$, which correspond to the optimal combination of children such that their inclusion maximizes the local sum of weights. This score only consider the subtree rooted in u and including u in a typical dynamic programming approach.

Formally, the computation of this score is as follows. For notation sake, lets define $\mathcal{P}^u = \mathcal{P}(C^u)$: the *power set* containing all the combinations of children of u; and $\mathcal{P}_k^u$ the k-th element of this set. Lets also define $S^{\mathcal{P}^u_k} = \bigcup\limits_{u^i \in \mathcal{P}_k^u} \overset{\circ}{S}{}_1^{u_i} \cup \overset{\circ}{S}{}_2^{u_i}$: the union of the optimal nodes in the k-th combination of children[4], including u. To simplicity notation, we further define $S^{\mathcal{P}^u_0}$, with the 0-th combination being the usual empty set[5], as the set $\{u, v\}$: if no children are selected, only the current node and its counterpart are part of the solution, which is a connected and matched solution by definition. Finally, we have:

$$s(u) = \max_k \sum_{n \in S^{\mathcal{P}^u_k}} w(n)$$

Computing the optimal combination of children, required for the max function, is not an easy problem. It looks similar to the WEIGHTED MAXIMUM COVERAGE problem (Hochbaum 1996), relaxed both with possibly negative weights, and with unconstrained number of sets. It also looks quite similar to the SET-UNION KNAPSACK problem (Goldschmidt et al. 1994), but again with real-valued weights and without budget constraint. Both are sensibly related and difficult, NP-hard problems (R. Cohen and Katzir 2008; Hochbaum 1996). However, since we are working with bounded-degree trees, the number of combinations remains (parameterized-)constant. The overall algorithm is thus in FPT (Fixed-Parameter Tractable), see proposition 5 for proof.

Proposition 5. *ComputeDecisionTree is a parameterized complexity algorithm which requires at most $O(2^d n^2 + n^3)$ steps for d-ary trees.*

[4]Note that in general, this union is certainly not optimal. Unless there are multiple optimal solutions, only one combination of the children maximizes the score, and possibly the empty one.

[5]Here of children.

Proof. The recursion tree closely follows the topology of T_1, and contains at most $O(n)$ calls. At each step we use the ConstructRequiredSet procedure, which is a quadratic time operation. We also compute the score, which itself requires to compute a set union operation for each combination of children. There are at worst $O(2^d)$ combination of children (the cardinality of the power set), and set union is at worst an $O(n)$ operation with a bitvector representation of the sets. Hence the resulting number of steps. □

Proposition 6. *One of the recursive calls of ComputeDecisionTree contains the optimal solution to the global problem.*

Proof. Even though two subtrees could actually have the exact same solution, since parent nodes can be included for matching and connectivity constraint reasons, it should be clear from the standard tree traversal that the whole tree T_1 is considered and that all its subtrees (that respect the constraints) are evaluated. To see that the optimal solution found by this procedure, in T_1, is also the optimal in regard to T_2, note that any given subtree of T_1, that is valid under the constraints, have one and only one counterpart subtree in T_2. Indeed, since the matching is a bipartite function, each node have one and only one counterpart; any subtree of size m in T_1 have one and only one counterpart subtree of size m in T_2, the one that contains all counterpart nodes. Finally, since all possible subtrees of T_1 (that respect the constraints) are considered, it must be that all possible subtrees of T_2 are also considered by this procedure. □

The optimum is in the recursive call with the highest total score, the optimal objective, and the optimal solution is the corresponding set of nodes S^u.

4.2.2 Polynomial-time scheme for some inputs

Notice that in the previous sections the conservation constraint α for the solution, that defines which fraction of nodes has to be kept, has been strict; that is $\alpha = 1$. Here we consider where $\alpha \in [0, 1]$. In addition, we further relax the constraint on the mapping function, which can be any partial injective function. That is, any element of V_1 can have at most one image in V_2 (the elements of V_2 can have 0, 1, or more antecedents). Finally, we suppose that there is a polynomial number of connected induced subgraphs of G_1.

We consider as many candidate solutions as there are connected subgraphs of G_1, and for each one of them we try to find the best corresponding subgraph in G_2. The best corresponding subgraph in G_2 is the subgraph that maximizes the total weight of the candidate solution and such that at least an α-fraction of the nodes of G_1 and G_2 in the solution are M-related. For a given subgraph of G_1, the sum of the weights of its nodes is fixed. Hence maximizing the total weight of the candidate solution is equivalent to finding the optimal solution in G_2 where the α-fraction constraint holds.

To solve this problem, we introduce a new problem, the RATIO-BOUNDED MAXIMUM-WEIGHT CONNECTED SUBGRAPH (RB-MWCS) problem. Informally, it consists in a variant of the MWCS problem where an additional contribution function is associated to each node and where an additional ratio constraint is introduced, not unlike the *budget-constrained* variant of the MWCS problem. This new problem is formally defined in section 4.3.

The reduction to this new subproblem is as follows. The contribution function needs to *encode* the relationship function between the nodes of both G_1 and G_2. To do so, and since we fixed a partial solution to the problem in the form of the subgraph of G_1, we note for each node of G_2 its number of inverse images in G'_1 plus one if and only if at least one exists, zero otherwise. The reason we need to make the distinction between the two cases is that when there is no counterpart in G_1, selecting a node in the optimal solution in G_2 will not increase the number of mapped node overall. However, selecting a node in G_2 for which there exists at least one counterpart will increase the number of mapped node by the number of counterpart, plus one for the selected node itself.

Formally, given a connected subgraph $G'_1 = (V'_1, E'_1)$ of G_1, we define the corresponding G_2 *antecedent function* $a\colon V_2 \to \mathbb{N}$ to be $a(v) = |\{v, u \mid M(u, v), u \in V'_1\}|$. The *contribution function* $c\colon V_2 \to \mathbb{N}$ is defined as follows:

$$c(v) = \begin{cases} a(v) + 1 & \text{if } a(v) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Given $G_2 = (V_2, E_2)$, its weight-function w_2 and its contribution function c, the problem now corresponds to the discovery of the connected subgraph of maximum weight such that:

$$\sum_{v \in V_2^*} c(v) \geq \alpha \times (|V'_1| + |V_2^*|)$$

$$\sum_{v \in V_2^*} c(v) - \alpha \times |V'_1| \geq \alpha \times |V_2^*|$$

Where $\alpha \times |V'_1|$ is constant.

Finally, given the optimal score of all candidate solutions, for which there are one for each subgraphs G'_1, the optimal solution to the MWCCS problem is the one candidate solution among them which has the best score.

Clearly, constructing a contribution function for G_2 given a subgraph G'_1 is a linear time operation. Choosing the best candidate solution is as time consuming as enumerating all subgraphs of G_1, which is a polynomial time operation in this context. The complexity of this algorithm hence depends on the difficulty to solve the RB-MWCS subproblem.

Section 4.3 provides an analysis of this problem with a more general contribution function. It provides polynomial scheme for d-ary trees and an optimized algorithm for paths. Thus, there exists a polynomial-time algorithm to solve MWCCS when one of the graph if polynomially enumerable, the second is a bounded-degree tree, and the relationship is a partial injective function, for any $\alpha \in [0, 1]$.

4.3 The Ratio-Bounded Maximum-Weight Connected Subgraph problem

In this section, we introduce a slightly more general version than required in section 4.2.2[6]. The RATIO-BOUNDED MAXIMUM-WEIGHT CONNECTED SUBGRAPH (RB-MWCS) problem, is formally defined as follows.

rb-mwcs: Given a node-weighted graph $G = (V, E)$, its node-weighting function $w\colon V \to \mathbb{R}$, its contribution function $c\colon V \to \mathbb{R}$, a ratio $\alpha \in [0, 1]$ and a constant $C \in \mathbb{R}$, find a subset $V^* \subseteq V$ such that:

1. the induced graph $G[V^*]$ is connected, and
2. the ratio of the sum of contributions plus some constant over the number of nodes in the solution is greater than or equal to α, that is: $\sum\limits_{v \in V^*} c(v) + C \geq \alpha \times |V^*|$, and
3. $\sum\limits_{v \in V^*} w(v)$ is maximum.

Proposition 7. RB-MWCS *is at least as difficult as* MWCS.

Proof. Indeed, when $c(v) = 1$, $\forall v \in V$, the ratio $\sum\limits_{v \in V^*} c(v)/|V^*| = 1 \geq \alpha$, and the MWCS and RB-MWCS problems are equivalent. □

Let us show now that it is in PTIME for bounded-degree trees.

Proposition 8. RB-MWCS *is solvable in $O(n^{d+2})$ time for d-ary trees.*

Proof. Let us consider the RB-MWCS problem for a d-ary tree. We define a dynamic programming strategy with a $O(n^{d+2})$ time complexity. This leads to a polynomial algorithm for d-ary trees. The basic idea is to define a 3-dimensional table T of size $|V| \times \sum_{v \in V} c(v) \times |V|$ that stores the maximum weight of a subtree rooted in v of size s and of total contribution tc.

[6]The contribution function can represent inputs that we don't actually use.

Formally, $\forall v \in V$, $0 \leq tc \leq \sum_{v \in V} c(v)$, $0 \leq s \leq |V|$, let us note $v_{\langle i \rangle}$ the i-th child of v, $1 \leq i \leq d$, we have:

$$\begin{aligned}
T[v][0][0] &= 0 \\
T[v][tc][s] &= \max_{tc_1,\dots,tc_d,\ s_1,\dots,s_d} \left(w(v) + \sum_{1 \leq i \leq d} T[v_{\langle i \rangle}][tc_i][s_i] \right) \\
\\
s.t. \quad tc &= c(v) + \sum_{1 \leq i \leq d} tc_i \\
s &= 1 + \sum_{1 \leq i \leq d} s_i
\end{aligned}$$

The optimal subtree can be reconstructed from the table by finding the entry with the maximal weight and where the contribution ratio is not violated, and backtracking from that entry on the selected tc_i's and s_i's from the max function. Each entry of the table can be computed in $O(n^{d-1})$ (that is, an integer partition of $|V|$ into d parts) time, and since $\sum_{v \in V} c(v) \in O(n)$, there are $O(n^3)$ of them, which leads to the overall complexity. □

As paths and cycles are trees of degree 1, using the preceding result leads to an $O(n^3)$ algorithm for these cases. However, one can achieve a better complexity.

Proposition 9. RB-MWCS *is solvable in $O(n^2)$ time for paths and cycles.*

Proof. Let us first consider the RB-MWCS problem for paths. Leveraging the linearity of the graph structure, we define a dynamic programming strategy with an $O(n^2)$ time complexity.

The idea is to define two 2-dimensional tables T_w and T_{tc} with n^2 entries each and that store respectively, for each pair of indices, the maximum weight and the total contribution, of the corresponding graph. Let us consider a given orientation in the path with the node at the starting end as the reference node, of index 0. Every candidate solution (a subpath) in the path can then be defined as a pair of positions, the first element being the starting position as an index number, the second element being the size of the candidate solution. The main idea being that increasing the indices one by one enables us to update the weights and total contributions incrementally.

Formally, let us denote the k-th node of the graph in the predefined orientation by n_k, we have for all $0 \leq i \leq j \leq n$:

$$\begin{aligned}
T_w[i][i] &= 0 \\
T_w[i][j] &= w(n_{i+j-1}) + T_w[i][j-1] \\
\\
T_{tc}[i][i] &= 0 \\
T_{tc}[i][j] &= c(n_{i+j-1}) + T_{tc}[i][j-1]
\end{aligned}$$

The optimal subpath is defined by the indices of the entry with the maximal weight and where the contribution ratio is not violated (*i.e.*, for any (i, j) s.t. $T_{tc}[i][j] \geq \alpha \cdot j$). Each $O(n^2)$ entry of the tables can be computed in constant time, leading to the overall complexity. For cycles, the trick consists in taking any linearization of the cycle and merging two copies of the corresponding linearization as the input path. This ensures that we will consider any candidate solution (*i.e.*, simple subpath of the cycle). The time complexity is preserved. □

4.4 Related questions and further work

In this contribution we provide the first deep complexity analysis of the MWCCS problem. A fair number of interesting problems and questions still remains.

First of all, generalizing the problem to more than two graphs is of practical interest. We hypothesize that the most general expression of the problem is to have a, possibly empty, mapping function between each graph. However, as we have shown throughout this chapter, the mapping function is an important factor for the complexity of the problem. Since the MWCCS problem is APX-hard for two graphs, the hardness results should hold for $k > 2$ graphs. However it is unclear whether the polynomial-time solvable cases would remain so, and if that is the case what would be the polynomial exponent.

Another interesting avenue is to study the effects of a relaxation of the connectivity contraints in MWCCS. The problem would then become similar to the SET-UNION KNAPSACK, with a bipartite partition of the sets, and where the maximum weight becomes a moving target (α-ratio).

In section 4.2.2 we presented a reduction from some instances of MWCCS to instances of RB-MWCS. For some classes of input, their exists trivial reductions from RB-MWCS to MWCCS. However, the question whether MWCCS is more general than RB-MWCS and can thus encode any of RB-MWCS instances remains open.

Interestingly, analyses of the relationship between RB-MWCS and the variants of MWCS, such as the BUDGET CONSTRAINED problem, is an already ongoing direction of our research. Indeed, the ratio constraint $\sum_{v \in V^*} c(v) + C \geq \alpha \times |V^*|$ of the RB-MWCS problem is a generalization of the costs constraint $\sum_{v \in V^*} \text{cost}(v) \leq B$ of the B-MWCS problem. Since RB-MWCS $\supset$ BUDGET CONSTRAINED MWCS $\supset$ k-CARDINALITY MWCS $\supset$ MWCS, all algorithms that apply to RB-MWCS also apply to the other problems.

Going in this direction, instead of the *integer partition* enumeration scheme used in proposition 8, we can probably use a tabular dynamic programming approach quite similar to KNAPSACK where the weights are the contributions. It would effectively remove the degree bound for the trees and make the algorithm pseudo-polynomial.

Finally, we have started work on an improvement to the algorithm introduced by Bateni et al. (2011) that solves the MWCS problem in polynomial time for all graphs up to bounded tree-width. Introducing KNAPSACK for the contributions in Bateni et al. (2011)'s algorithm should provide a solution for RB-MWCS in pseudo-polynomial time for this same category of graphs.

Chapter 5

Conclusion

Appendix A

xHeinz appendices

A.1 Full Th17 module tables

We provide a tabular overview of the module contents (table A.1 and table A.2). In the tables, each row lists unmatched genes or homologuous gene pairs with their activity scores. On the left side are the mouse genes and on the right side the human genes. The gene activity scores are in parentheses. Names in bold represent genes for which no homologous counterpart exist in the underlying networks.

Note that very few (overall only 2 in mouse for 48 h) genes with negative activity score and without conserved counterpart are selected. Some of them even have a strong positive activity score. These genes would be missed in a strict conservation model.

Mouse	Human
Batf (8.29)	BATF (13.90)
Stat3 (4.22)	STAT3 (8.81)
Rapgef6 (5.67)	RAPGEF6 (6.68)
Stat1 (4.07)	STAT1 (7.95)
Rec8 (-2.85)	REC8 (11.56)
Rsad2 (0.44)	RSAD2 (7.99)
Cish (-0.08)	CISH (8.04)
Tnip2 (0.53)	TNIP2 (6.04)
Ifi35 (6.68)	IFI35 (-0.44)
Irf9 (1.07)	IRF9 (4.89)
Gbp2 (3.85)	GBP2 (1.21)
Parp9 (7.95)	PARP9 (-2.98)
Tnfrsf10b (-3.47)	TNFRSF10D (7.99)
Trim21 (-3.08)	TRIM21 (7.58)
Etv6 (7.90)	ETV6 (-3.54)
Stat2 (-1.88)	STAT2 (6.21)
Bcl3 (4.82)	BCL3 (-0.88)
Tmem173 (-0.15)	TMEM173 (3.96)
Dhx58 (3.96)	DHX58 (-0.20)
Fas (7.58)	FAS (-4.21)
Myd88 (1.11)	MYD88 (2.15)
Trim25 (-0.96)	TRIM25 (4.22)
Socs1 (-2.82)	SOCS1 (5.96)
Ubr1 (-3.27)	UBR1 (6.15)
Lgals3bp (6.52)	LGALS3BP (-3.64)
Trafd1 (-2.87)	TRAFD1 (5.67)
Ikzf4 (-3.34)	IKZF4 (6.14)
BC006779 (1.85)	RP4-697K14.7 (0.87)
Ripk1 (0.71)	RIPK1 (1.93)
Tap1 (-4.38)	TAP1 (6.91)
Arid5a (2.06)	ARID5A (0.23)
Saps3 (-3.51)	SAPS3 (5.62)
Daxx (5.36)	DAXX (-3.31)
Casp4 (-2.56)	AP002004.1 (4.14)
Cxcr5 (-1.36)	CXCR5 (2.63)
Notch1 (0.60)	NOTCH1 (-0.06)
Ldha (-1.10)	LDHA (1.61)
Mfng (-0.63)	MFNG (1.10)
Hk1 (4.89)	HK1 (-4.47)
Prkcq (-1.03)	PRKCQ (0.84)
Irf8 (2.70)	IRF8 (-3.31)
Igtp (0.02)	IRGM (-0.92)
Mx2 (-1.23)	MX1 (0.20)
Pnpt1 (3.12)	PNPT1 (-4.46)
Nmi (-0.87)	NMI (-0.55)
Il21 (2.68)	IL21 (-4.69)
Ifih1 (1.00)	IFIH1 (-3.19)
Oasl1 (-2.29)	OASL (0.01)
Rpa1 (1.82)	RPA1 (-4.36)
Ankrd28 (1.66)	ANKRD28 (-4.34)
Smc4 (2.15)	SMC4 (-5.12)
Muc1 (0.27)	MUC1 (-3.25)
Isg15 (0.53)	ISG15 (-3.57)
Kat2b (0.79)	KAT2B (-4.23)
Junb (-4.02)	JUNB (-0.08)
Map3k8 (1.02)	MAP3K8 (-5.33)
Cxcr4 (-0.74)	CXCR4 (-3.63)
Il6st (-1.06)	IL6ST (-3.37)
Irf7 (0.74)	IRF7 (-5.39)
Mavs (-1.97)	MAVS (-5.26)
Eif3b (-4.46)	EIF3B (-3.51)
Hdac5 (-3.20)	HDAC5 (-4.84)
Sirt7 (-4.06)	SIRT7 (-5.13)
Ptpn13 (-3.87)	PTPN13 (-5.40)
Oas2 (15.42)	
Elovl6 (13.90)	
Ppa1 (11.56)	
Pml (8.81)	
Zbp1 (8.55)	
Ifit1 (8.02)	
Parp12 (7.99)	
Oas3 (6.15)	
Arrb2 (5.67)	
Casp8 (3.69)	
Eif2ak3 (2.80)	
Irf6 (2.63)	
Stk10 (2.30)	
Syne2 (1.23)	
Ptpn1 (1.03)	
Nfe2l2 (0.59)	
Hdac6 (0.04)	
Dcpp3 (6.14)	
Tha1 (4.61)	
Oasl2 (2.19)	
	ITK (6.49)
	ARID5B (4.61)
	PFKFB3 (3.78)
	VAV1 (3.00)
	IFIT3 (2.70)
	EIF2AK2 (2.29)
	IRF1 (1.07)
	XAF1 (0.97)
	ZC3HAV1 (0.90)
	DHDH (0.55)
	IFI16 (6.52)
	MX2 (0.78)

Table A.1: Timepoint 2 h, $\alpha = 0.8$, FDR $= 0.1$

Mouse	Human
Napsa (-3.86)	NAPSA (32.07)
Il9 (5.03)	IL9 (22.36)
Il21 (32.07)	IL21 (-4.97)
Cd70 (7.24)	CD70 (19.85)
Cysltr1 (22.36)	CYSLTR1 (0.30)
Il17a (27.88)	IL17A (-5.46)
Fbxo15 (3.11)	FBXO15 (18.26)
Wisp1 (24.44)	WISP1 (-4.63)
Slamf6 (22.16)	SLAMF6 (-5.38)
Rbpj (14.79)	RBPJ (0.04)
Il24 (17.69)	IL24 (-3.66)
Emp1 (16.64)	EMP1 (-2.83)
Chn1 (-5.02)	CHN1 (18.50)
Egln3 (14.50)	EGLN3 (-1.79)
Ccl20 (9.84)	CCL20 (2.50)
Aqp3 (17.54)	AQP3 (-5.38)
Vdr (2.22)	VDR (9.74)
Apod (-4.84)	APOD (16.41)
Pdpn (16.86)	PDPN (-5.29)
Ahcyl2 (-3.39)	AHCYL2 (14.79)
Sgk1 (4.04)	SGK1 (5.55)
Lgals1 (12.94)	LGALS1 (-3.52)
Lpxn (14.49)	LPXN (-5.30)
Anxa2 (8.87)	ANXA2 (-0.12)
Loxl3 (-4.22)	LOXL3 (12.94)
Tnfrsf25 (-0.02)	TNFRSF25 (7.75)
Arhgef3 (4.10)	ARHGEF3 (2.72)
Basp1 (5.83)	BASP1 (0.89)
Gata3 (1.19)	GATA3 (5.24)
Fut7 (-1.83)	FUT7 (7.58)
Furin (4.28)	FURIN (1.46)
Ryr1 (-4.25)	RYR1 (8.54)
Axin2 (-4.95)	AXIN2 (7.83)
Tnfrsf13b (6.13)	TNFRSF13B (-3.57)
Tnfsf8 (1.41)	TNFSF8 (1.04)
Il2rb (-3.52)	IL2RB (5.91)
Fasl (7.75)	FASLG (-5.37)
Anxa3 (7.83)	ANXA3 (-5.48)
Jun (-2.95)	JUN (5.14)
Sdcbp2 (5.24)	SDCBP2 (-3.21)
Klrd1 (-1.64)	KLRD1 (3.51)
Uchl1 (-5.07)	UCHL1 (6.91)
Rps6ka4 (0.22)	RPS6KA4 (1.34)
Ifng (1.77)	IFNG (-0.32)
S100a6 (2.72)	S100A6 (-1.60)
Pfkfb3 (-5.16)	PFKFB3 (6.07)
Rdx (-3.21)	RDX (4.10)
Palld (-4.63)	PALLD (5.31)
Gbp2 (-1.37)	GBP2 (2.04)
Sftpd (-5.12)	SFTPD (5.71)
Asap1 (4.97)	ASAP1 (-4.42)
Col15a1 (-4.82)	COL15A1 (5.34)
Casp1 (0.03)	CASP1 (0.48)
Chst3 (-5.14)	CHST3 (5.61)
Arid5a (5.34)	ARID5A (-4.90)
Lif (-2.49)	LIF (2.92)
Thbs1 (-4.55)	THBS1 (4.87)
Fam174b (-3.95)	FAM174B (4.09)
Ptprj (5.47)	PTPRJ (-5.43)
Syp (-3.59)	SYP (3.45)
Trat1 (5.14)	TRAT1 (-5.35)
Tnfrsf12a (-4.23)	TNFRSF12A (3.90)
Id2 (5.02)	ID2 (-5.38)
Armcx2 (4.41)	ARMCX2 (-4.95)
Cyp11a1 (2.69)	CYP11A1 (-3.46)
Itga3 (2.81)	ITGA3 (-3.60)
Batf3 (-2.57)	BATF3 (1.64)
Pmepa1 (-2.08)	PMEPA1 (0.89)
Ctla4 (1.49)	CTLA4 (-2.69)
Anxa1 (-4.66)	ANXA1 (3.43)
Skil (2.30)	SKIL (-3.67)
Ripk2 (0.25)	RIPK2 (-1.64)
Ino80c (-3.62)	INO80C (2.22)
Stap1 (3.19)	STAP1 (-4.67)
Tnip2 (0.48)	TNIP2 (-1.99)
Smad3 (-3.58)	SMAD3 (1.95)
Sept11 (-5.00)	SEPT11 (3.21)
Tnfrsf14 (-3.50)	TNFRSF14 (1.70)
Ntrk3 (-5.01)	NTRK3 (3.19)
Ctnna1 (3.21)	CTNNA1 (-5.04)
Taf7 (3.41)	TAF7 (-5.28)
Ripk3 (1.93)	RIPK3 (-4.05)
Ddit4 (-2.48)	DDIT4 (0.31)
Anxa4 (3.28)	ANXA4 (-5.53)
Id3 (0.38)	ID3 (-2.65)
Foxo1 (-2.09)	FOXO1 (-0.28)
Map3k1 (-0.93)	MAP3K1 (-1.47)
Rorc (2.25)	RORC (-4.79)
Cndp2 (1.21)	CNDP2 (-3.76)
Il2ra (1.95)	IL2RA (-4.53)
Cxcr3 (-2.88)	CXCR3 (0.29)
Ppp3ca (1.84)	PPP3CA (-4.64)
Setdb1 (-3.16)	SETDB1 (0.31)
Pou2f2 (2.50)	POU2F2 (-5.48)
Lpp (0.89)	LPP (-4.05)
Sh3kbp1 (1.09)	SH3KBP1 (-4.29)
Bcl3 (2.19)	BCL3 (-5.51)
Irf8 (1.29)	IRF8 (-5.04)
Fes (1.03)	FES (-4.82)
Runx1 (-5.09)	RUNX1 (1.29)
Kit (-5.24)	KIT (1.41)
Ptpn14 (-4.47)	PTPN14 (0.58)
Fnbp1 (-5.10)	FNBP1 (1.09)
Rel (1.20)	REL (-5.37)
Ralb (0.98)	RALB (-5.21)
Sirt1 (-4.83)	SIRT1 (0.60)
Ska1 (-5.08)	SKA1 (0.66)
Cltb (-5.23)	CLTB (0.76)
Penk (0.66)	PENK (-5.23)
Csf2 (-1.22)	CSF2 (-3.46)
Ptpn11 (0.12)	PTPN11 (-4.91)
Vim (0.10)	VIM (-5.10)
Prkca (0.24)	PRKCA (-5.36)
Gpc6 (-4.26)	GPC6 (-1.59)
Ripk1 (-0.81)	RIPK1 (-5.60)
Calm2 (-2.76)	CALM2 (-4.28)
Snai1 (-5.08)	SNAI1 (-2.21)
Fbxo25 (-4.90)	FBXO25 (-2.49)
Stat3 (-3.22)	STAT3 (-4.60)
Epor (-5.17)	EPOR (-3.07)
Rac3 (-4.44)	RAC3 (-3.86)
B2m (-4.57)	B2M (-4.45)
Os9 (-5.20)	OS9 (-4.11)
Tnfsf13b (-4.10)	TNFSF13B (-5.24)
Trpc3 (-4.28)	TRPC3 (-5.21)
Dcn (-4.39)	DCN (-5.27)
Ar (-4.20)	AR (-5.67)
Myog (-5.25)	MYOG (-5.01)
Cd4 (-4.94)	CD4 (-5.53)
Hamp (-5.21)	HAMP (-5.53)
Gap43 (22.44)	
Nts (18.50)	
Inhba (13.50)	
Tec (10.10)	
Serpine2 (9.94)	
Lrmp (9.92)	
Eno3 (9.74)	
Serpinc1 (9.69)	
Fap (9.64)	
Ifih1 (8.54)	
Tspan6 (8.41)	
Smox (6.91)	
Ppap2a (6.35)	
Piwil2 (6.11)	
Cd274 (5.61)	
Ermp1 (5.31)	
Tgm2 (5.21)	
Srgn (4.72)	
Serpinb6b (4.53)	
Mxi1 (4.39)	
Dnmt3a (4.32)	
Nsg2 (3.71)	
Igfbp7 (3.45)	
Gpr65 (3.30)	
Pdcd1lg2 (3.01)	
Rrad (2.97)	
Socs3 (1.90)	
Mt2 (1.31)	
Pkp4 (1.12)	
Gem (0.37)	
Prickle1 (0.30)	
Prnp (-1.10)	
Ywhaz (-3.30)	
Timp1 (18.48)	
Ifi203 (18.26)	
Ctla2b (10.18)	
Ctla2a (6.54)	
Il3 (1.86)	
	PTHLH (24.44)
	TMPRSS6 (16.86)
	COL6A3 (16.64)
	IL23R (14.50)
	PTK2 (11.74)
	EXOC3 (10.18)
	BCAR3 (9.84)
	IL17F (8.67)
	PTGIR (6.95)
	ITGB7 (6.54)
	COL6A1 (5.47)
	TNS3 (5.20)
	LMNA (4.70)
	NCALD (4.41)
	EVL (4.28)
	BTLA (3.41)
	FNBP1L (1.93)
	SOS1 (1.84)
	EXOC2 (1.19)
	NOTCH1 (0.23)
	FYN (0.08)
	CD1A (27.88)
	CLECL1 (22.16)
	CD1C (18.48)
	CD1B (14.49)
	CTSL1 (13.50)
	TBL1X (4.04)

Table A.2: Timepoint 48 h, $\alpha = 0.8$, FDR = 0.1

Bibliography

Aaronson, Scott, Greg Kuperberg, and Christopher Granade (2016). *Complexity Zoo.* URL: `https://complexityzoo.uwaterloo.ca/Complexity_Zoo` (visited on 05/20/2016) (cit. on p. 10).

Althaus, Ernst and Markus Blumenstock. "Algorithms for the Maximum Weight Connected Subgraph and Prize-collecting Steiner Tree Problems". In: (cit. on p. 29).

Althaus, Ernst, Markus Blumenstock, Alexej Disterhoft, Andreas Hildebrandt, and Markus Krupp (2014). "Algorithms for the Maximum Weight Connected k-Induced Subgraph Problem". In: *Combinatorial Optimization and Applications.* Springer, pp. 268–282 (cit. on p. 29).

Altman, Russ B and Soumya Raychaudhuri (2001). "Whole-genome expression analysis: challenges beyond clustering". In: *Current opinion in structural biology* 11.3, pp. 340–347 (cit. on pp. 19, 22).

Álvarez-Miranda, Eduardo, Ivana Ljubić, and Petra Mutzel (2013a). "The maximum weight connected subgraph problem". In: *Facets of Combinatorial Optimization.* Springer, pp. 245–270 (cit. on pp. 25, 28, 29, 35).

Álvarez-Miranda, Eduardo, Ivana Ljubić, and Petra Mutzel (2013b). "The rooted maximum node-weight connected subgraph problem". In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems.* Springer, pp. 300–315 (cit. on p. 28).

Annunziato, Francesco, Lorenzo Cosmi, Francesco Liotta, Enrico Maggi, and Sergio Romagnani (2009). "Human Th17 cells: are they different from murine Th17 cells?" In: *European journal of immunology* 39.3, pp. 637–640 (cit. on p. 37).

Annunziato, Francesco and Sergio Romagnani (2009). "Do studies in humans better depict Th17 cells?" In: *Blood* 114.11, pp. 2213–2219 (cit. on p. 37).

Backes, Christina, Alexander Rurainski, Gunnar W Klau, Oliver Müller, Daniel Stöckel, Andreas Gerasch, Jan Küntzer, Daniela Maisel, Nicole Ludwig, Matthias Hein, et al. (2012). "An integer linear programming approach for finding deregulated subgraphs in regulatory networks". In: *Nucleic acids research* 40.6, e43–e43 (cit. on pp. 22, 27, 28).

Bandyopadhyay, Sourav, Roded Sharan, and Trey Ideker (2006). “Systematic identification of functional orthologs based on protein network comparison”. In: *Genome research* 16.3, pp. 428–435 (cit. on p. 20).

Bateni, M, Chandra Chekuri, Alina Ene, Mohammad Taghi Hajiaghayi, Nitish Korula, and Dániel Marx (2011). “Prize-collecting Steiner problems on planar graphs”. In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, pp. 1028–1049 (cit. on p. 75).

Beisser, Daniela, Gunnar W Klau, Thomas Dandekar, Tobias Müller, and Marcus T Dittrich (2010). “BioNet: an R-Package for the functional analysis of biological networks”. In: *Bioinformatics* 26.8, pp. 1129–1130 (cit. on p. 39).

Beriou, Gaëlle, Elizabeth M Bradshaw, Ester Lozano, Cristina M Costantino, William D Hastings, Tihamer Orban, Wassim Elyaman, Samia J Khoury, Vijay K Kuchroo, Clare Baecher-Allan, et al. (2010). “TGF-β induces IL-9 production from human Th17 cells”. In: *The Journal of Immunology* 185.1, pp. 46–54 (cit. on p. 45).

Berman, Piotr and Marek Karpinski (1999). *On some tighter inapproximability results*. Springer (cit. on p. 64).

Bienstock, Daniel, Michel X Goemans, David Simchi-Levi, and David Williamson (1993). “A note on the prize collecting traveling salesman problem”. In: *Mathematical programming* 59.1-3, pp. 413–420 (cit. on p. 24).

Boykov, Yuri and Vladimir Kolmogorov (2004). “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26.9, pp. 1124–1137 (cit. on p. 36).

Camerini, PM, G Galbiati, and F Maffioli (1979). “On the complexity of Steiner-like problems”. In: *Proceedings 17th Ann. Allerton Conf. on Communication, Control, and Computing*. Department of Electrical Engineering and the Coordinated Science Laboratory, University of Illinois Urbana, Ill, pp. 969–977 (cit. on p. 24).

Campain, Anna and Yee H Yang (2010). “Comparison study of microarray meta-analysis methods”. In: *BMC bioinformatics* 11.1, p. 408 (cit. on p. 23).

Carvajal, Rodolfo, Miguel Constantino, Marcos Goycoolea, Juan Pablo Vielma, and Andrés Weintraub (2013). “Imposing connectivity constraints in forest planning models”. In: *Operations Research* 61.4, pp. 824–836 (cit. on p. 27).

Chang, Seon Hee, Yeonseok Chung, and Chen Dong (2010). “Vitamin D suppresses Th17 cytokine production by inducing C/EBP homologous protein (CHOP) expression”. In: *Journal of Biological Chemistry* 285.50, pp. 38751–38755 (cit. on p. 45).

Chen, Chao-Yeh and Kristen Grauman (2012). “Efficient activity detection with max-subgraph search”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 1274–1281 (cit. on p. 27).

Chimani, Markus, Maria Kandyba, Ivana Ljubić, and Petra Mutzel (2009). “Obtaining optimal k-cardinality trees fast”. In: *Journal of Experimental Algorithmics (JEA)* 14, p. 5 (cit. on pp. 28, 29).

Ciofani, Maria, Aviv Madar, Carolina Galan, MacLean Sellars, Kieran Mace, Florencia Pauli, Ashish Agarwal, Wendy Huang, Christopher N Parkurst, Michael Muratet, et al. (2012). “A validated regulatory network for Th17 cell specification”. In: *Cell* 151.2, pp. 289–303 (cit. on p. 42).

Cohen, Nathann (2010). *Several Graph problems and their Linear Program formulations*. URL: `https://hal.inria.fr/inria-00504914` (cit. on p. 29).

Cohen, Reuven and Liran Katzir (2008). “The generalized maximum coverage problem”. In: *Information Processing Letters* 108.1, pp. 15–22 (cit. on p. 69).

Conrad, Jon, Carla P Gomes, Willem-Jan van Hoeve, Ashish Sabharwal, and Jordan Suter (2007). “Connections in networks: Hardness of feasibility versus optimality”. In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, pp. 16–28 (cit. on pp. 24, 27).

Crescenzi, Pierluigi and Viggo Kann (2016). *A compendium of NP optimization problems*. URL: `http://www.nada.kth.se/~viggo/problemlist/compendium.html` (visited on 05/17/2016) (cit. on p. 17).

Crescenzi, Pierluigi and Viggo Kann (1995). *A compendium of NP optimization problems* (cit. on p. 17).

Crick, Francis et al. (1970). “Central dogma of molecular biology”. In: *Nature* 227.5258, pp. 561–563 (cit. on p. 3).

Crome, Sarah Q, Adele Y Wang, Christine Y Kang, and Megan K Levings (2009). “The role of retinoic acid-related orphan receptor variant 2 and IL-17 in the development and function of human CD4+ T cells”. In: *European journal of immunology* 39.6, pp. 1480–1493 (cit. on p. 45).

Csermely, Peter, Tamás Korcsmáros, Huba JM Kiss, Gábor London, and Ruth Nussinov (2013). “Structure and dynamics of molecular networks: a novel paradigm of drug discovery: a comprehensive review”. In: *Pharmacology & therapeutics* 138.3, pp. 333–408 (cit. on p. 31).

Dede, Duygu and Hasan Oğul (2014). “TriClust: A Tool for Cross-Species Analysis of Gene Regulation”. In: *Molecular Informatics* 33.5, pp. 382–387 (cit. on p. 23).

Deshpande, Raamesh, Shikha Sharma, Catherine M Verfaillie, Wei-Shou Hu, and Chad L Myers (2010). “A scalable approach for discovering conserved active subnetworks across species”. In: *PLoS computational biology* 6.12, e1001028 (cit. on pp. 23, 45).

Dezső, Balázs, Alpár Jüttner, and Péter Kovács (2011). “LEMON – an open source C++ graph template library”. In: *Electronic Notes in Theoretical Computer Science* 264.5, pp. 23–45 (cit. on p. 49).

Dilkina, Bistra and Carla P Gomes (2010). “Solving connected subgraph problems in wildlife conservation”. In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, pp. 102–116 (cit. on pp. 27, 35).

Dittrich, Marcus, Gunnar W Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller (2008). “Identifying functional modules in protein–protein interaction networks: an integrated exact approach”. In: *Bioinformatics* 24.13, pp. i223–i231 (cit. on pp. 22, 27, 28, 32, 39, 41).

Dunning, Mark J, Mike L Smith, Matthew E Ritchie, and Simon Tavaré (2007). “beadarray: R classes and methods for Illumina bead-based data”. In: *Bioinformatics* 23.16, pp. 2183–2184 (cit. on p. 38).

Estevez-Garcia, Purificacion, Fernando Rivera, Sonia Molina-Pinelo, Marta Benavent, Javier Gómez, Maria Luisa Limón, Maria Dolores Pastor, Julia Martinez-Perez, Luis Paz-Ares, Amancio Carnero, et al. (2015). “Gene expression profile predictive of response to chemotherapy in metastatic colorectal cancer.” In: *Oncotarget* 6.8, pp. 6151–6159 (cit. on p. 20).

Feigenbaum, Joan, Christos Papadimitriou, and Scott Shenker (2001). “Sharing the cost of multicast transmissions”. In: *Journal of Computer and System Sciences* 63.1, pp. 21–41 (cit. on pp. 25, 26).

Fischetti, Matteo, Horst W Hamacher, Kurt Jørnsten, and Francesco Maffioli (1994). “Weighted k-cardinality trees: Complexity and polyhedral structure”. In: *Networks* 24.1, pp. 11–21 (cit. on pp. 28, 29).

Flicek, Paul, Ikhlak Ahmed, M Ridwan Amode, Daniel Barrell, Kathryn Beal, Simon Brent, Denise Carvalho-Silva, Peter Clapham, Guy Coates, Susan Fairley, et al. (2013). “Ensembl 2013”. In: *Nucleic acids research* 41.D1, pp. D48–D55 (cit. on p. 40).

Franceschini, Andrea, Damian Szklarczyk, Sune Frankild, Michael Kuhn, Milan Simonovic, Alexander Roth, Jianyi Lin, Pablo Minguez, Peer Bork, Christian von Mering, et al. (2013). “STRING v9. 1: protein-protein interaction networks, with increased coverage and integration”. In: *Nucleic acids research* 41.D1, pp. D808–D815 (cit. on p. 40).

Garey, Michael R and David S Johnson (1979). “Computers and intractability: a guide to the theory of NP-completeness”. In: *San Francisco, LA: Freeman* (cit. on p. 25).

Goemans, Michel X and David P Williamson (1995). “A general approximation technique for constrained forest problems”. In: *SIAM Journal on Computing* 24.2, pp. 296–317 (cit. on p. 27).

Goemans, Michel X and David P Williamson (1997). “The primal-dual method for approximation algorithms and its application to network design problems”. In: *Approximation algorithms for NP-hard problems*, pp. 144–191 (cit. on p. 28).

Goldschmidt, Olivier, David Nehme, and Gang Yu (1994). “Note: On the set-union knapsack problem”. In: *Naval Research Logistics (NRL)* 41.6, pp. 833–842 (cit. on p. 69).

Golub, Todd R, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. (1999). “Molecular classification of cancer: class discovery and class prediction by gene expression monitoring”. In: *Science* 286.5439, pp. 531–537 (cit. on p. 19).

Gomes, Carla P, Willem-Jan Van Hoeve, and Ashish Sabharwal (2008). “Connections in networks: A hybrid approach”. In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, pp. 303–307 (cit. on p. 27).

Hamburg, Jan Piet van, Marjolein JW De Bruijn, Claudia Ribeiro de Almeida, Marloes van Zwam, Marjan van Meurs, Edwin de Haas, Louis Boon, Janneke N Samsom, and Rudi W Hendriks (2008). “Enforced expression of GATA3 allows differentiation of IL-17-producing cells, but constrains Th17-mediated pathology”. In: *European journal of immunology* 38.9, pp. 2573–2586 (cit. on p. 45).

Harris, Timothy J, Joseph F Grosso, Hung-Rong Yen, Hong Xin, Marcin Kortylewski, Emilia Albesiano, Edward L Hipkiss, Derese Getnet, Monica V Goldberg, Charles H Maris, et al. (2007). “Cutting edge: An in vivo requirement for STAT3 signaling in TH17 development and TH17-dependent autoimmunity”. In: *The Journal of Immunology* 179.7, pp. 4313–4317 (cit. on p. 44).

Hauptmann, Mathias and Marek Karpinski (2014). *A compendium on steiner tree problems* (cit. on p. 24).

Hochbaum, Dorit S (1996). *Approximation algorithms for NP-hard problems.* PWS Publishing Co. (cit. on p. 69).

Hochbaum, Dorit S and Anu Pathria (1994). “Node-optimal connected k-subgraphs”. In: *manuscript, UC Berkeley* (cit. on p. 26).

Hromkovič, Juraj (2013). *Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics.* Springer Science & Business Media (cit. on p. 18).

Hu, Pingzhao, Celia MT Greenwood, and Joseph Beyene (2006). “Statistical methods for meta-analysis of microarray data: a comparative study”. In: *Information Systems Frontiers* 8.1, pp. 9–20 (cit. on p. 23).

Hume, Thomas, Hayssam Soueidan, Macha Nikolski, and Guillaume Blin (2015). “Approximation Hardness of the Cross-Species Conserved Active Modules Detection Problem”. In: *SOFSEM 2015: Theory and Practice of Computer Science*. Springer, pp. 242–253 (cit. on pp. 59, 60).

Ideker, Trey, Owen Ozier, Benno Schwikowski, and Andrew F Siegel (2002). “Discovering regulatory and signalling circuits in molecular interaction networks”. In: *Bioinformatics* 18.suppl 1, S233–S240 (cit. on pp. 19, 22, 25).

Jada, Joseph (1992). *An introduction to parallel algorithms*. Addison Wesley (cit. on p. 7).

Johnson, David S (1985). “The NP-completeness column: an ongoing guide”. In: *Journal of Algorithms* 6.1, pp. 145–159 (cit. on p. 25).

Johnson, David S, Maria Minkoff, and Steven Phillips (2000). “The prize collecting steiner tree problem: theory and practice”. In: *SODA*. Vol. 1. 0.6. Citeseer, p. 4 (cit. on pp. 26, 28).

Johnson, W Evan, Cheng Li, and Ariel Rabinovic (2007). “Adjusting batch effects in microarray expression data using empirical Bayes methods”. In: *Biostatistics* 8.1, pp. 118–127 (cit. on p. 6).

Karatsuba, Anatolii and Yu Ofman (1963). “Multiplication of multidigit numbers on automata”. In: *Soviet physics doklady*. Vol. 7, p. 595 (cit. on p. 9).

Karmarkar, Narendra (1984). “A new polynomial-time algorithm for linear programming”. In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM, pp. 302–311 (cit. on pp. 14, 15).

Karp, Richard M (1972). *Reducibility among combinatorial problems*. Springer (cit. on pp. 24–26).

El-Kebir, Mohammed, Jaap Heringa, and Gunnar W Klau (2011). “Lagrangian relaxation applied to sparse global network alignment”. In: *Pattern Recognition in Bioinformatics*. Springer, pp. 225–236 (cit. on p. 20).

El-Kebir, Mohammed and Gunnar W Klau (2014). “Solving the Maximum-Weight Connected Subgraph Problem to Optimality”. In: *arXiv preprint arXiv:1409.5308* (cit. on p. 29).

El-Kebir, Mohammed, Hayssam Soueidan, Thomas Hume, Daniela Beisser, Marcus Dittrich, Tobias Müller, Guillaume Blin, Jaap Heringa, Macha Nikolski, Lodewyk FA Wessels, et al. (2015). “xHeinz: An algorithm for mining cross-species network modules under a flexible conservation model”. In: *Bioinformatics*, btv316 (cit. on p. 32).

Kelley, Ryan and Trey Ideker (2005). “Systematic interpretation of genetic interactions using protein networks”. In: *Nature biotechnology* 23.5, pp. 561–566 (cit. on p. 21).

Khachiyan, Leonid G (1980). “Polynomial algorithms in linear programming”. In: *USSR Computational Mathematics and Mathematical Physics* 20.1, pp. 53–72 (cit. on p. 14).

Klee, Victor and George J Minty (1970). *How good is the simplex algorithm.* Tech. rep. DTIC Document (cit. on p. 14).

Kristiansson, Erik, Tobias Österlund, Lina Gunnarsson, Gabriella Arne, DG Joakim Larsson, and Olle Nerman (2013). “A novel method for cross-species gene expression analysis”. In: *BMC bioinformatics* 14.1, p. 70 (cit. on p. 23).

Kuzniar, Arnold, Roeland CHJ van Ham, Sándor Pongor, and Jack AM Leunissen (2008). “The quest for orthologs: finding the corresponding gene across genomes”. In: *Trends in Genetics* 24.11, pp. 539–551 (cit. on p. 4).

Land, Ailsa H and Alison G Doig (1960). “An automatic method of solving discrete programming problems”. In: *Econometrica: Journal of the Econometric Society*, pp. 497–520 (cit. on p. 17).

Lee, Heungsoon Felix and Daniel R Dooly (1998). “Decomposition algorithms for the maximum-weight connected graph problem”. In: *Naval Research Logistics* 45.8, pp. 817–837 (cit. on p. 26).

Lenstra Jr, Hendrik W and Carl Pomerance (2002). “Primality testing with Gaussian periods”. In: *Lecture Notes in Computer Science*, pp. 1–1 (cit. on p. 9).

Ljubić, Ivana, René Weiskircher, Ulrich Pferschy, Gunnar W Klau, Petra Mutzel, and Matteo Fischetti (2005). “Solving the prize-collecting Steiner tree problem to optimality”. In: *ALENEX/ANALCO*, pp. 68–76 (cit. on p. 28).

Ljubić, Ivana, René Weiskircher, Ulrich Pferschy, Gunnar W Klau, Petra Mutzel, and Matteo Fischetti (2006). “An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem”. In: *Mathematical Programming* 105.2-3, pp. 427–449 (cit. on p. 28).

Lu, Yong, Roni Rosenfeld, Gerard J Nau, and Ziv Bar-Joseph (2009). “Cross species expression analysis of innate immune response”. In: *Research in Computational Molecular Biology.* Springer, pp. 90–107 (cit. on p. 23).

Lucena, Abilio and Mauricio GC Resende (2004). “Strong lower bounds for the prize collecting Steiner problem in graphs”. In: *Discrete Applied Mathematics* 141.1, pp. 277–294 (cit. on p. 28).

Magnanti, Thomas L and Laurence A Wolsey (1995). “Optimal trees”. In: *Handbooks in operations research and management science* 7, pp. 503–615 (cit. on p. 35).

Maier, David (1978). “The complexity of some problems on subsequences and supersequences”. In: *Journal of the ACM (JACM)* 25.2, pp. 322–336 (cit. on p. 16).

Makarova, Kira S, Alexander V Sorokin, Pavel S Novichkov, Yuri I Wolf, and Eugene V Koonin (2007). “Clusters of orthologous genes for 41 archaeal genomes and implications for evolutionary genomics of archaea”. In: *Biol Direct* 2, p. 33 (cit. on p. 4).

McCune, Amy R and John C Schimenti (2012). “Using genetic networks and homology to understand the evolution of phenotypic traits”. In: *Current genomics* 13.1, p. 74 (cit. on p. 20).

McGeachy, Mandy J and Daniel J Cua (2008). “Th17 cell differentiation: the long and winding road”. In: *Immunity* 28.4, pp. 445–453 (cit. on p. 37).

Meyers, Scott (2014). *Effective Modern C++: 42 Specific Ways to Improve Your Use of C++ 11 and C++ 14*. O’Reilly Media, Inc. (cit. on p. 49).

Mihail, Christos Gkantsidist Milena and Ellen Zegura (2003). “The markov chain simulation method for generating connected power law random graphs”. In: *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments*. Vol. 111. SIAM, p. 16 (cit. on p. 44).

Miller, Clair E, Albert W Tucker, and Richard A Zemlin (1960). “Integer programming formulation of traveling salesman problems”. In: *Journal of the ACM (JACM)* 7.4, pp. 326–329 (cit. on p. 28).

Mitra, Koyel, Anne-Ruxandra Carvunis, Sanath Kumar Ramesh, and Trey Ideker (2013). “Integrative approaches for finding modular structure in biological networks”. In: *Nature Reviews Genetics* 14.10, pp. 719–732 (cit. on pp. 22, 27).

Nesterov, Yurii, Arkadii Nemirovskii, and Yinyu Ye (1994). *Interior-point polynomial algorithms in convex programming*. Vol. 13. SIAM (cit. on p. 14).

Noort, Vera van, Berend Snel, and Martijn A Huynen (2003). “Predicting gene function by conserved co-expression”. In: *Trends in Genetics* 19.5, pp. 238–242 (cit. on p. 22).

O’Brien, Kevin P, Maido Remm, and Erik LL Sonnhammer (2005). “Inparanoid: a comprehensive database of eukaryotic orthologs”. In: *Nucleic acids research* 33.suppl 1, pp. D476–D480 (cit. on pp. 20, 23).

O’Garra, Anne, Brigitta Stockinger, and Marc Veldhoen (2008). “Differentiation of human TH-17 cells does require TGF-β!” In: *Nature immunology* 9.6, pp. 588–590 (cit. on p. 37).

Okyere, John, Ekow Oppon, Daniel Dzidzienyo, Lav Sharma, and Graham Ball (2014). “Cross-Species Gene Expression Analysis of Species Specific

Differences in the Preclinical Assessment of Pharmaceutical Compounds". In: *PLoS One* 9.5 (cit. on p. 31).

Orchard, Sandra, Samuel Kerrien, Sara Abbani, Bruno Aranda, Jignesh Bhate, Shelby Bidwell, Alan Bridge, Leonardo Briganti, Fiona SL Brinkman, Gianni Cesareni, et al. (2012). "Protein interaction data curation: the International Molecular Exchange (IMEx) consortium". In: *Nature methods* 9.4, pp. 345–350 (cit. on p. 20).

Papadimitriou, Christos H and Kenneth Steiglitz (1982). *Combinatorial optimization: algorithms and complexity.* Courier Corporation (cit. on pp. 14, 17, 18).

Papadimitriou, Christos H and Mihalis Yannakakis (1991). "Optimization, approximation, and complexity classes". In: *Journal of Computer and System Sciences* 43.3, pp. 425–440 (cit. on p. 64).

Park, Heon, Zhaoxia Li, Xuexian O Yang, Seon Hee Chang, Roza Nurieva, Yi-Hong Wang, Ying Wang, Leroy Hood, Zhou Zhu, Qiang Tian, et al. (2005). "A distinct lineage of CD4 T cells regulates tissue inflammation by producing interleukin 17". In: *Nature immunology* 6.11, pp. 1133–1141 (cit. on p. 37).

Perou, Charles M, Therese Sørlie, Michael B Eisen, Matt van de Rijn, Stefanie S Jeffrey, Christian A Rees, Jonathan R Pollack, Douglas T Ross, Hilde Johnsen, Lars A Akslen, et al. (2000). "Molecular portraits of human breast tumours". In: *Nature* 406.6797, pp. 747–752 (cit. on p. 20).

Polzin, Tobias (2003). "Algorithms for the Steiner problem in networks". PhD thesis. Universitätsbibliothek (cit. on p. 27).

Pounds, Stan and Stephan W Morris (2003). "Estimating the occurrence of false positives and false negatives in microarray studies by approximating and partitioning the empirical distribution of p-values". In: *Bioinformatics* 19.10, pp. 1236–1242 (cit. on p. 39).

Quintão, Frederico P, Alexandre Salles da Cunha, and Geraldo R Mateus (2008). "Integer programming formulations for the k-cardinality tree problem". In: *Electronic Notes in Discrete Mathematics* 30, pp. 225–230 (cit. on p. 28).

Quintão, Frederico P, Alexandre Salles da Cunha, Geraldo R Mateus, and Abilio Lucena (2010). "The k-cardinality tree problem: reformulations and lagrangian relaxation". In: *Discrete Applied Mathematics* 158.12, pp. 1305–1314 (cit. on p. 28).

Räihä, Kari-Jouko and Esko Ukkonen (1981). "The shortest common supersequence problem over binary alphabet is NP-complete". In: *Theoretical Computer Science* 16.2, pp. 187–198 (cit. on p. 16).

Reddy, TBK, Alex D Thomas, Dimitri Stamatis, Jon Bertsch, Michelle Isbandi, Jakob Jansson, Jyothi Mallajosyula, Ioanna Pagani, Elizabeth A Lobos, and Nikos C Kyrpides (2014). "The Genomes OnLine Database (GOLD) v. 5: a metadata management system based on a four level (meta) genome project classification". In: *Nucleic acids research*, gku950 (cit. on p. 5).

Regalado, A (2014). "EmTech: Illumina Says 228,000 Human Genomes Will Be Sequenced This Year". In: *Technology Review* 24 (cit. on p. 5).

Reiss, David J, Nitin S Baliga, and Richard Bonneau (2006). "Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks". In: *BMC bioinformatics* 7.1, p. 280 (cit. on p. 22).

Reiss, David J, Christopher L Plaisier, Wei-Ju Wu, and Nitin S Baliga (2015). "cMonkey2: Automated, systematic, integrated detection of co-regulated gene modules for any organism". In: *Nucleic acids research*, gkv300 (cit. on p. 22).

Rekatsinas, Theodoros, Xin Luna Dong, Lise Getoor, and Divesh Srivastava (2015). "Finding Quality in Quantity: The Challenge of Discovering Valuable Sources for Integration." In: *CIDR* (cit. on p. 13).

Rhodes, Daniel R, Scott A Tomlins, Sooryanarayana Varambally, Vasudeva Mahavisno, Terrence Barrette, Shanker Kalyana-Sundaram, Debashis Ghosh, Akhilesh Pandey, and Arul M Chinnaiyan (2005). "Probabilistic model of the human protein-protein interaction network". In: *Nature biotechnology* 23.8, pp. 951–959 (cit. on p. 21).

Richard, Arianne C, Paul A Lyons, James E Peters, Daniele Biasci, Shaun M Flint, James C Lee, Eoin F McKinney, Richard M Siegel, and Kenneth GC Smith (2014). "Comparison of gene expression microarray data with count-based RNA measurements informs microarray interpretation". In: *BMC genomics* 15.1, p. 649 (cit. on p. 6).

Richard, Mélisande, Jamila Louahed, Jean-Baptiste Demoulin, and Jean-Christophe Renauld (1999). "Interleukin-9 regulates NF-κB activity through BCL3 gene induction". In: *Blood* 93.12, pp. 4318–4327 (cit. on p. 45).

Ross, Douglas T, Uwe Scherf, Michael B Eisen, Charles M Perou, Christian Rees, Paul Spellman, Vishwanath Iyer, Stefanie S Jeffrey, Matt Van de Rijn, Mark Waltham, et al. (2000). "Systematic variation in gene expression patterns in human cancer cell lines". In: *Nature genetics* 24.3, pp. 227–235 (cit. on p. 20).

Ruan, Qingguo, Shi-Jun Zheng, Scott Palmer, Ruaidhri J Carmody, and Youhai H Chen (2010). "Roles of Bcl-3 in the pathogenesis of murine type 1 diabetes". In: *Diabetes* 59.10, pp. 2549–2557 (cit. on p. 45).

Schatz, Michael C (2015). “Biological data sciences in genome research”. In: *Genome research* 25.10, pp. 1417–1422 (cit. on p. 5).

Schraml, Barbara U, Kai Hildner, Wataru Ise, Wan-Ling Lee, Whitney A-E Smith, Ben Solomon, Gurmukh Sahota, Julia Sim, Ryuta Mukasa, Saso Cemerski, et al. (2009). “The AP-1 transcription factor Batf controls TH17 differentiation”. In: *Nature* 460.7253, pp. 405–409 (cit. on p. 49).

Sealfon, Stuart C and Tearina T Chu (2011). “RNA and DNA microarrays”. In: *Biological Microarrays*. Springer, pp. 3–34 (cit. on p. 6).

Sharan, Roded and Trey Ideker (2006). “Modeling cellular machinery through biological network comparison”. In: *Nature biotechnology* 24.4, pp. 427–433 (cit. on p. 21).

Sipser, Michael (2012). *Introduction to the Theory of Computation*. Cengage Learning (cit. on p. 8).

Smyth, Gordon K (2005). “Limma: linear models for microarray data”. In: *Bioinformatics and computational biology solutions using R and Bioconductor*. Springer, pp. 397–420 (cit. on p. 38).

Smyth, Gordon K, Joëlle Michaud, and Hamish S Scott (2005). “Use of within-array replicate spots for assessing differential expression in microarray experiments”. In: *Bioinformatics* 21.9, pp. 2067–2075 (cit. on p. 6).

Sorlie, Therese, Ch M Perou, R Tibshirani, T Aas, S Geisler, H Johnsen, T Hastie, MB Eisen, M Van de Rijn, SS Jeffrey, et al. (2001). “Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications”. In: *Proc Natl Acad Sci USA* 98.19, pp. 10869–74 (cit. on p. 20).

Steinman, Lawrence (2007). “A brief history of TH17, the first major revision in the TH1/TH2 hypothesis of T cell–mediated tissue damage”. In: *Nature medicine* 13.1, pp. 139–145 (cit. on p. 37).

Stormo, Gary D (2000). “Gene-finding approaches for eukaryotes”. In: *Genome research* 10.4, pp. 394–397 (cit. on p. 4).

Szklarczyk, Damian, Andrea Franceschini, Stefan Wyder, Kristoffer Forslund, Davide Heller, Jaime Huerta-Cepas, Milan Simonovic, Alexander Roth, Alberto Santos, Kalliopi P Tsafou, et al. (2014). “STRING v10: protein–protein interaction networks, integrated over the tree of life”. In: *Nucleic acids research*, gku1003 (cit. on p. 20).

Tatusov, Roman L, Eugene V Koonin, and David J Lipman (1997). “A genomic perspective on protein families”. In: *Science* 278.5338, pp. 631–637 (cit. on pp. 20, 23).

Thiel, Christoph, Sebastian Schneckener, Markus Krauss, Ahmed Ghallab, Ute Hofmann, Tobias Kanacher, Sebastian Zellmer, Rolf Gebhardt, Jan G Hengstler, and Lars Kuepfer (2015). “A Systematic Evaluation of the

Use of Physiologically Based Pharmacokinetic Modeling for Cross-Species Extrapolation". In: *Journal of pharmaceutical sciences* 104.1, pp. 191–206 (cit. on p. 31).

Trapnell, Cole, David G Hendrickson, Martin Sauvageau, Loyal Goff, John L Rinn, and Lior Pachter (2013). "Differential analysis of gene regulation at transcript resolution with RNA-seq". In: *Nature biotechnology* 31.1, pp. 46–53 (cit. on p. 31).

Tseng, George C, Debashis Ghosh, and Eleanor Feingold (2012). "Comprehensive literature review and statistical considerations for microarray meta-analysis". In: *Nucleic acids research*, gkr1265 (cit. on p. 23).

Tuomela, Soile, Verna Salo, Subhash K Tripathi, Zhi Chen, Kirsti Laurila, Bhawna Gupta, Tarmo Äijö, Lotta Oikari, Brigitta Stockinger, Harri Lähdesmäki, et al. (2012). "Identification of early gene expression changes during human Th17 cell differentiation". In: *Blood* 119.23, e151–e160 (cit. on pp. 37, 38, 42, 44).

Van De Vijver, Marc J, Yudong D He, Laura J van't Veer, Hongyue Dai, Augustinus AM Hart, Dorien W Voskuil, George J Schreiber, Johannes L Peterse, Chris Roberts, Matthew J Marton, et al. (2002). "A gene-expression signature as a predictor of survival in breast cancer". In: *New England Journal of Medicine* 347.25, pp. 1999–2009 (cit. on p. 20).

Van't Veer, Laura J, Hongyue Dai, Marc J Van De Vijver, Yudong D He, Augustinus AM Hart, Mao Mao, Hans L Peterse, Karin van der Kooy, Matthew J Marton, Anke T Witteveen, et al. (2002). "Gene expression profiling predicts clinical outcome of breast cancer". In: *nature* 415.6871, pp. 530–536 (cit. on p. 20).

Vergis, A (1983). "manuscript" (cit. on p. 25).

Waltman, Peter, Thadeous Kacmarczyk, Ashley R Bate, Daniel B Kearns, David J Reiss, Patrick Eichenberger, and Richard Bonneau (2010). "Multi-species integrative biclustering". In: *Genome Biol* 11.9, R96 (cit. on p. 23).

Wei, Lai, Arian Laurence, Kevin M Elias, and John J O'Shea (2007). "IL-21 is produced by Th17 cells and drives IL-17 production in a STAT3-dependent manner". In: *Journal of Biological Chemistry* 282.48, pp. 34605–34610 (cit. on p. 48).

Wilke, Cailin Moira, Keith Bishop, David Fox, and Weiping Zou (2011). "Deciphering the role of Th17 cells in human disease". In: *Trends in immunology* 32.12, pp. 603–611 (cit. on p. 37).

Williamson, David P and David B Shmoys (2011). *The design of approximation algorithms*. Cambridge university press (cit. on p. 7).

Xu, Lin, Hong Chen, Xiaohua Hu, Rongmei Zhang, Ze Zhang, and ZW Luo (2006). "Average gene length is highly conserved in prokaryotes and eukaryotes and diverges only between the two kingdoms". In: *Molecular biology and evolution* 23.6, pp. 1107–1108 (cit. on p. 3).

Yamamoto, Takanori, Hideo Bannai, Masao Nagasaki, and Satoru Miyano (2009). "Better decomposition heuristics for the maximum-weight connected graph problem using betweenness centrality". In: *Discovery Science*. Springer, pp. 465–472 (cit. on pp. 22, 27).

Yang, Xuexian O, Bhanu P Pappu, Roza Nurieva, Askar Akimzhanov, Hong Soon Kang, Yeonseok Chung, Li Ma, Bhavin Shah, Athanasia D Panopoulos, Kimberly S Schluns, et al. (2008). "T helper 17 lineage differentiation is programmed by orphan nuclear receptors RORα and RORγ". In: *Immunity* 28.1, pp. 29–39 (cit. on p. 45).

Yosef, Nir, Alex K Shalek, Jellert T Gaublomme, Hulin Jin, Youjin Lee, Amit Awasthi, Chuan Wu, Katarzyna Karwacz, Sheng Xiao, Marsela Jorgolli, et al. (2013). "Dynamic regulatory network controlling TH17 cell differentiation". In: *Nature* 496.7446, pp. 461–468 (cit. on pp. 38, 42, 44, 45).

Zhang, Lan V, Oliver D King, Sharyl L Wong, Debra S Goldberg, Amy HY Tong, Guillaume Lesage, Brenda Andrews, Howard Bussey, Charles Boone, and Frederick P Roth (2005). "Motifs, themes and thematic maps of an integrated Saccharomyces cerevisiae interaction network". In: *Journal of biology* 4.2, p. 6 (cit. on p. 21).

Zheng, Wei-ping and Richard A Flavell (1997). "The transcription factor GATA-3 is necessary and sufficient for Th2 cytokine gene expression in CD4 T cells". In: *Cell* 89.4, pp. 587–596 (cit. on p. 45).

Zinman, Guy E, Shoshana Naiman, Dawn M O'Dee, Nishant Kumar, Gerard J Nau, Haim Y Cohen, and Ziv Bar-Joseph (2015). "ModuleBlast: identifying activated sub-networks within and across species". In: *Nucleic acids research* 43.3, e20–e20 (cit. on p. 23).