University of Ljubljana
Faculty of Computer and Information Science

Mihael Rajh

Integration of gene expression data with causal networks

master's thesis

the 2nd cycle master's study programme
computer and information science
Track: Computer and Information Science

Supervisor: izr. prof. dr. Tomaž Curk
Co-supervisor: dr. Carissa Robyn Bleker

Ljubljana, 2023

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Mihael Rajh

Integracija podatkov o genskem izražanju s kavzalnimi omrežji

magistrsko delo

magistrski študijski program druge stopnje
računalništvo in informatika
Smer: Računalništvo in Informatika

Mentor: izr. prof. dr. Tomaž Curk
Somentor: dr. Carissa Robyn Bleker

Ljubljana, 2023

Acknowledgments

I would first like to thank my co-supervisor, Dr. Carissa Robyn Bleker, for her consistent guidance and encouragement throughout the process of developing this thesis. Her invaluable advice was a cornerstone of this work. I would like to express my gratitude to my supervisor, Assoc. Prof. Dr. Tomaž Curk, for his constructive feedback and supervision. In addition, my thanks extend to Assist. Prof. Dr. Anže Županič for his time and valuable insights. Lastly, I would like to acknowledge the support of my family and friends during my studies.

Mihael Rajh, 2023

Contents

List of used acronmys

acronym	meaning
CBN	causal biological network
BEL	Biological Expression Language
UBE	upstream biological entity
NPA	Network Perturbation Amplitude
TopoNPA	Topological Network Perturbation Amplitude
RCR	Reverse Causal Reasoning
CMPA	Candidate Mechanism Perturbation Amplitude
IPA	Ingenuity Pathway Analysis
ILP	integer linear programming
DSV	delimiter-separated value
MDS	multidimensional scaling
GEO	Gene Expression Omnibus
SNR	signal-to-noise ratio
FAIR	findability, accessibility, interoperability, and reusability

Abstract

Title: Integration of gene expression data with causal networks

With the increased availability of large gene expression datasets comes an increased need for informed methods of data analysis. One class of recent methods involves the use of causal biological networks, which depict causal relationships between molecular events inside the cell. These networks offer the advantage of representing prior biological knowledge in a form that is suited for both computation and human interpretation. However, many of the current methods are held back by implementational challenges, which make them difficult to apply to novel networks. In this thesis, we develop and extend an implementation of the TopoNPA algorithm in the form of a Python package. We present PerturbationX, which features support for custom network syntax, Cytoscape integration, as well as improvements in both edge pruning and permutations. Alongside the implementation, we also provide an estimate of the algorithm's scalability and analyse its sensitivity to noise, missing data, and edge modifications. With the introduction of this robust, open-source tool, we hope to facilitate advancement in the development of causal network algorithms. We aim for the tool to promote insight into experimental data from multiple biological domains.

Keywords

bioinformatics, causal reasoning, gene expression

Povzetek

Naslov: Integracija podatkov o genskem izražanju s kavzalnimi omrežji

S povečanjem razpoložljivosti velikih naborov podatkov o izražanju genov se povečuje potreba po informiranih metodah analize podatkov. Eden izmed novejših razredov metod temelji na uporabi kavzalnih bioloških omrežij, ki prikazujejo vzročne odnose med molekularnimi dogodki v celici. Prednost teh omrežij je predstavitev obstoječega biološkega znanja v obliki, ki je primerna tako za računanje kot človeško interpretacijo. Po drugi strani pa številne sorodne metode zavirajo implementacijske težave, ki otežujejo njihovo uporabo z novimi omrežji. V sklopu magistrskega dela smo razvili in razširili implementacijo algoritma TopoNPA v obliki Python programskega paketa. Predstavljamo PerturbationX, ki podpira poljubno sintakso omrežij, integracijo z ogrodjem Cytoscape, ter izboljšave pri odstranjevanju povezav in permutacijah. Poleg implementacije podamo tudi oceno skalabilnosti algoritma in analiziramo njegovo občutljivost na šum, manjkajoče podatke ter spremembe povezav. Z uvedbo robustnega, odprtokodnega orodja želimo spodbuditi napredek v razvoju algoritmov za kavzalna omrežja. Upamo, da orodje prispeva k spoznanjem o eksperimentalnih podatkih iz različnih področij biologije.

Ključne besede

bioinformatika, vzročno sklepanje, izražanje genov

Razširjeni povzetek

Biološka omrežja so ključnega pomena v biologiji za predstavitev raznih interakcij. Posebej pomembna so v sistemski biologiji, ki si prizadeva natančno opredeliti in modelirati biološke sisteme. Junker in Schreiber [1] podrobno opisujeta biološka omrežja in njihove glavne vrste. Številne med njimi lahko sklepajo o interakcijah in spremembah na podlagi meritev. V tem delu se osredotočamo na kavzalna oz. vzročna omrežja, ki vsebujejo le tiste interakcije, za katere je bila vzročnost zadostno potrjena.

Ena izmed izstopajočih metod za uporabo kavzalnih omrežij s podatki o genski ekspresiji je TopoNPA [2]. Ta magistrska naloga predstavlja PerturbationX [3], odprtokodni Python paket za kvantifikacijo sistemske perturbacije z uporabo kavzalnih bioloških omrežij. Paket je bil razvit v sodelovanju z Nacionalnim inštitutom za biologijo in implementira razširjeno različico TopoNPA metode. Poleg tega omogoča tudi uvoz, izvoz in urejanje poljubnih omrežij, ter vizualizacijo rezultatov preko integracije Cytoscape [4].

I Kavzalna biološka omrežja

Kavzalna biološka omrežja (CBN) združujejo znane biološke interakcije sistema in omogočajo informirano analizo izražanja genov. Za razliko od drugih metod lahko upoštevajo strukturo regulacijskih mehanizmov ter znanje o bioloških mehanizmih, ki je omejeno s kontekstom raziskav. Ponujajo natančen prikaz vzročnih odnosov mehanizma, kar omogoča tako človeško interpretacijo kot računalniško sklepanje.

Namesto sklepanja naprej (angl. *forward assumption*), ki gene povezuje z aktivnostjo njihovih produktov, omogočajo tudi sklepanje nazaj (angl. *backward assumption*). Pri tem pristopu gene združujemo glede na biološke entitete, ki regulirajo njihovo izražanje. To je pomembno, ker je korelacija med ekspresijo genov in njihovimi produkti nezanesljiva [5].

Leta 2005 so Pollard idr. [6] predstavili računalniški pristop z uporabo CBN za modeliranje biologije človeških mišic. Pristop najprej določi regulatorje, ki so najbolj verjetno povzročili spremembe v izražanju genov, nato pa jih razvrsti glede na razlagalno moč in statistične značilnosti. Chindelevitch idr. [7] so leta 2012 predlagali sorodno metodologijo, ki uporablja obstoječe baze znanja za določanje najbolj verjetnih regulatorjev izbranih genov. Istega leta so Martin idr. [8] uvedli štiri metode za ocenjevanje CBN, ki jih imenujejo amplituda omrežne perturbacije (NPA). Z združevanjem ocen manjših mrež so pokazali, da se lahko te metode uporablja tudi pri napovedovanju fenotipov [9]. Catlett idr. [10] so leta 2013 objavili implementacijo sorodne metode vzvratnega vzročnega sklepanja (RCR).

Martin idr. [2] so leta 2014 predstavili metodo TopoNPA, ki za razliko od ostalih pristopov ne zahteva vzročne konsistence biološkega omrežja. Poleg tega uporablja celoten nabor podatkov genskega izražanja in ne zahteva predhodne identifikacije reguliranih genov. Leta 2019 so Martin idr. [11] objavili implementacijo TopoNPA, ki pa je omejena na omrežja, ki so del implementacije. Talikka idr. [12] so leta 2015 predstavili podatkovno bazo s kavzalnimi biološkimi omrežji, ki so shranjena v sintaksi jezika za biološko izražanje (BEL). Python paket za uporabo omrežij v BEL sintaksi so leta 2017 predstavili Hoyt idr. [13]. Karki idr. [14] so leta 2019 predlagali CMPA kot alternativo k obstoječim metodam NPA, ki pa ne omogoča uporabe omrežij z vzročnimi zankami.

Kramer idr. [15] so leta 2013 predstavili programski paket *Ingenuity Pathway Analysis* (IPA), ki uporablja poenostavljen način določanja regulacijskih mehanizmov. Kot alternativo komercialnim rešitvam sta Bradley in Barrett [16] leta 2017 objavila CausalR, odprtokodno implementacijo vzročne

analize bioloških omrežij. Podobno strukturo omrežja so leta 2014 uporabili tudi Jaeger idr. [17]. Predlagali so šest metod za razvrščanje vozlišč v bioloških omrežjih, med katerimi se je najbolje obnesla metoda SigNet. Paull idr. [18] so leta 2013 predstavili TieDIE, ki lahko poišče vzročno konsistentne poti v omrežju brez predpostavk o njegovi strukturi. Leta 2019 so Liu idr. [19] objavili CARNIVAL, ki identificira signalizacijske poti s pomočjo celoštevilskega linearnega programiranja (ILP). CARNIVAL so Dugourd idr. [20] leta 2021 razširili v metodo COSMOS, ki lahko integrira nabore podatkov različnih omskih področij.

Širši pregled metod biologije, ki temeljijo na signalizacijskih omrežjih, so leta 2022 objavili Garrido-Rodriguez idr. [21]. Housseini-Germani idr. [22] so leta 2023 primerjali metode SigNet, CausalR in CARNIVAL pri določanju signalizacijskih poti in disreguliranih proteinov. Poudarili so, da je učinkovitost metod odvisna od značilnosti omrežja. Toure idr. [23] so leta 2021 analizirali vzročnost v bioloških podatkovnih zbirkah. Opozarjajo na potrebo po dodatni dokumentaciji in orodjih za interoperabilnost podatkov.

II Topološka amplituda omrežne perturbacije

Algoritem topološke amplitude omrežne perturbacije (TopoNPA) kvantificira odziv kavzalne mreže na biološko perturbacijo. Sledi dvostopenjskemu procesu, kjer najprej določi optimalne koeficiente za vsa vozlišča v omrežju, iz teh pa izračuna oceno amplitude omrežne perturbacije (NPA). Rezultat NPA je mogoče razčleniti na prispevke vozlišč, kar pomaga pri identifikaciji najbolj vpletenih omrežnih komponent. Algoritem izračuna tudi intervala zaupanja in permutacijske statistike za oceno statistične značilnosti rezultata. Diagram algoritma je podan na Sliki 3.1.

TopoNPA zahteva omrežje z dvema vrstama vozlišč. En del omrežja sestavljajo vozlišča, ki predstavljajo regulatorje, drug del pa vozlišča, ki predstavljajo zapise genov. Regulacija je predstavljena z usmerjenimi, uteženimi povezavami med vozlišči. Drug zahtevan vhod so podatki o spremembah

v izražanju genov skupaj z oceno standardne napake. V splošnem lahko vhodno omrežje razumemo kot osrednji podgraf skupaj z robnimi vozlišči v neposredni povezavi z njim. Za slednje je zahtevan vnos vhodnih vrednosti, ki jih imenujemo koeficienti.

Glavni rezultat TopoNPA je ocena NPA. Koeficienti vozlišč v osrednjem podgrafu so izračunani na podlagi optimizacije enačbe 3.1. Edinstvena rešitev problema je zagotovljena pod pogojem, da je osrednji podgraf šibko povezan in ima vsaj eno povezavo na robno vozlišče z vhodnim koeficientom. Ocena NPA se nato izračuna s pomočjo enačbe 3.3. Postopek je podoben optimizaciji v prvem koraku, s ključno razliko nasprotnega predznaka med koeficienti vozlišč. Povezava med korakoma je prikazana na Sliki 3.2. Z enačbo 3.5 lahko izračunano vrednost razčlenimo na prispevke posameznih vozlišč.

Naslednji korak algoritma TopoNPA je izračun variance in intervalov zaupanja za koeficiente jedrnih vozlišč in ocene NPA. Pri tem se uporabijo ocene standardne napake vhodnih koeficientov. Varianca NPA se izračuna z enačbo 3.7. Na podlagi variance se določi 95-odstotni interval zaupanja. Nazadnje se izvedejo še permutacijski testi, ki ocenijo statistično pomembnost strukture omrežja. Permutacijski test "O" premeša oznake robnih vozlišč in oceni pomembnost njihovega vrstnega reda. Permutacijski test "K" naključno spremeni strukturo osrednjega podgrafa in oceni pomembnost povezav v njem.

III Razvoj in implementacija funkcionalnosti

PerturbationX je programski paket v Pythonu, ki temelji na NetworkX [24], pandas [25, 26] in Cytoscape [4]. Razvoj je potekal iterativno, s poudarkom na dobri organizaciji programske kode. Struktura paketa je prikazana na Sliki 4.1. Funkcionalne zahteve so bile zbrane v obliki uporabniških zgodb, delno preko rednih srečanj z Dr. Carisso Robyn Bleker, Dr. Anžetom Županičem in Dr. Tomažem Curkom, ter delno v obliki skupinskega pogovora s širšo skupino uslužbencev Nacionalnega inštituta za biologijo. Vsem

vpletenim se iskreno zahvaljujemo za njihove ideje in prispevke k razvoju.

Izboljšave ter nadgradnje algoritma so strnjene v Tabeli 4.1. Večina funkcionalnosti paketa je izpostavljena preko razreda `CausalNetwork`. Razred se lahko inicializira z obstoječim NetworkX usmerjenim grafom, dodane pa so tudi pomožne metode za uvoz iz datotek DSV, datotek Cytoscape JSON, ter instanc pandas `DataFrame`. Uporabljeno omrežje mora imeti za vse povezave navedeno relacijo in tip. Relacija določa vrsto regulacije in je obvezen podatek. Zavzame lahko katerokoli vrednost. Med potekom algoritma se preslika v številčno utež povezave, kar lahko uporabnik nastavlja preko razreda `RelationTranslator`. Primer preslikave je prikazan na Sliki 4.2. Tip povezave je neobvezen in ima privzeto vrednost `infer`. Če ni ustrezno naveden, se določi glede na lastnosti sosednjih vozlišč.

Zagon algoritma TopoNPA zahteva vhodne koeficiente robnih vozlišč v obliki slovarja Python, ki ime naborov slika v ustrezajoče instance pandas `DataFrame`. Izračunani rezultati se shranijo v instanco razreda `NPAResult`, ki vsebuje štiri tipe rezultatov: globalne informacije, informacije vozlišč, distribucije in metapodatke. Metapodatki so kopirani iz razreda `CausalNetwork`, kjer jih lahko navede uporabnik. Avtomatsko pa so dodani tudi podatki o datumu, času in programskem okolju. To zagotavlja skladnost rezultatov s principi FAIR (ugotovljivost, dostopnost, interoperabilnost in ponovna uporabnost). Razred `NPAResult` omogoča tudi prikaz rezultatov permutacij in inicializacijo vizualizacije Cytoscape.

Permutacijski testom je bil dodan parameter stopnje permutacije, ki uporabnikom omogoča določiti obseg permutacij vozlišč in povezav. Permutacija "K"je bila spremenjena, da zagotavlja šibko povezanost osrednjega podgrafa. Uvedeni sta bili dve različici permutacije "K". Prva dodeli povezave naključno, druga pa glede na obstoječe zaporedje stopenj vozlišč. Obe različici se lahko izvedeta delno, z vzorčenjem koeficientov osrednjih vozlišč ali pa v polnem obsegu, z neposrednim vzorčenjem ocene NPA. Razredu `CausalNetwork` sta bili dodani metodi `wire_edges` in `rewire_edges`, ki omogočata spreminjanje omrežja pred zagonom algoritma. Postopek se

lahko dodatno prilagodi z navedbo posebnih uteži zaupanja (angl. *confidence*).

Učinkovitost algoritma TopoNPA je bila izboljšana z uvedbo načina, ki uporablja razpršene matrike SciPy. Dodana je bila metoda za ocenjevanje permutacij, ki jih ustvarita metodi `wire_edges` in `rewire_edges`. Prilagojeno je bilo odstranjevanje povezav v primeru manjkajočih vrednosti. Poleg obstoječega načina, ki uteži teh povezav izniči, je bil dodan način odstranitve s ponovnim uravnoteženjem. V prvem primeru se uteži ostalih povezav ne spremenijo, v drugem primeru pa se nastavijo na vrednosti, ki bi jih imele, če izničenih povezav v mreži ne bi bilo. Ponazoritev je podana na Sliki 4.3. Odstranjevanje povezav je bilo podprto tudi v primeru, ko izmerjene spremembe v izražanju genov nasprotujejo predvideni vzročnosti povezav. Uporabniki lahko nastavijo prag amplitude, ki razlikuje pomenljivo nasprotujoče vrednosti od tistih, ki so zanemarljivo majhne.

Integracija s Cytoscape je bila implementirana s py4cytoscape [27], ki uporablja Cytoscape REST API [28]. Prikaz se inicializira prek razreda `NPAResult`, ki ustvari instanco `NPAResultDisplay`. Ta omogoča obarvanje vozlišč glede na poljuben atribut. Pomembna funkcionalnost TopoNPA je tudi prepoznavanje vozlišč, ki največ prispevajo k oceni NPA. PerturbationX omogoča poudarjanje in ekstrahiranje teh vozlišč. Poleg tega sta na voljo še dve metodi za določitev podgrafa z vodilnimi vozlišči. Prva metoda poišče najkrajše poti med njimi glede na podano odstopanje. Druga metoda poišče vsa sosednja vozlišča z vmesnimi povezavami glede na podano maksimalno razdaljo. Metodi se lahko uporabljata tudi skupaj, primeri pa so podani na Sliki 4.4 in Sliki 4.5.

IV Generiranje in analiza omrežij

Proces ustvarjanja novih kavzalnih bioloških omrežij po navadi vključuje ročni pregled literature ter ovrednotenjem s transkriptomskimi podatki. Ker implementacija TopoNPA v R ponuja omejen nabor, smo raziskali umetno

generiranje kavzalnih omrežij z uporabo modela Barabási-Albert. Za namen vrednotenja smo omrežjem generirali še sintetične nabore podatkov.

Omrežja v implementaciji R smo najprej analizirali z uporabo 14 primarnih in sekundarnih statističnih značilnosti. Primarne značilnosti lahko določimo neposredno in vključujejo števila vozlišč ter povezav. Sekundarne značilnosti vključujejo mere razdalje in nakopičenosti vozlišč. Zaradi visoke korelacije med značilnostmi smo uporabili hierarhično združevanje za zmanjševanje dimenzionalnosti podatkov. Gruče značilk so prikazane na Sliki 5.1. Na podlagi spremenjenih značilk smo omrežja razdelili v štiri skupine, prikazane na Sliki 5.3. Tri skupine vsebujejo dve omrežji za vsak vključen proces, pri čemer eno opisuje človeka (*Homo sapiens*), drugo pa miš (*Mus musculus*). Zadnja skupina vsebuje eno samo omrežje, ki opisuje kardiotoksičnost v navadni cebrici (*Danio rerio*).

Sintetična omrežja smo ustvarili za določeno število osrednjih vozlišč. Ostale primarne značilnosti so bile vzorčene z dvostopenjskim procesom. Najprej smo uporabili linearno regresijo med značilnostmi in številom osrednjih vozlišč, nato pa napovedanim vrednostim prišteli vzorec iz multivariatne normalne porazdelitve ostankov. Osrednji podgraf je bil ustvarjen s standardno uporabo modela Barabási-Albert glede na vrednosti parametrov. Pri generiranju robnih povezav pa je bilo ugotovljeno, da izhodne stopnje vozlišč ne sledijo potenčnemu zakonu. Zaradi tega so bile izhodne stopnje najprej vzorčene iz združene distribucije vseh vhodnih omrežij. Ustvarjena omrežja smo primerjali z vhodnimi s pomočjo algoritma MDS, kar je prikazano na Sliki 5.5. Podrobne primerjave so podane v Dodatku A.

Umetni nabori podatkov so bili osnovani na podatkih iz baze *Gene Expression Omnibus* (GEO). Uporabljeni so bili vzorci pod kodami GSE205434, GSE218120, GSE158763 [29], GSE216395 [30] in GSE225001 [31]. Za generiranje naborov je bil uporabljen genetski algoritem. V vsaki iteraciji je bila populacija vzorcev razvrščena glede na oceno NPA, naslednje generacije pa so bile ustvarjene s procesi križanja in mutacije. Ustvarjeni so bili nabori podatkov treh tipov. Tretjina jih je bila ustvarjenih naključno, ostali pa tako,

da so čim bolj oziroma čim manj nasprotovali vzročnosti omrežnih povezav. Primeri procesa generiranja so prikazani na Sliki 5.6.

V Zmogljivost in vrednotenje algoritma

Pri vrednotenju algoritma so bile uporabljene tri vrste vhodnih podatkov: nabor COPD1 z omrežjem *Mus musculus* Apoptoza, sintetični nabori podatkov z omrežji iz implementacije TopoNPA v jeziku R, ter sintetični nabori podatkov s podmnožico petnajstih omrežij, ustvarjenih z modelom Barabási-Albert. Pri analizi občutljivosti na manjkajoče vrednosti, ki je prikazana na Sliki 6.1, smo preizkusili vse strategije odstranjevanja povezav. Najbolje se je obnesla strategija izničevanja uteži s točnimi stopnjami vozlišč. V primeru šuma se je algoritem izkazal za manj občutljivega, kar je razvidno iz Slike 6.2. Nazadnje smo izvedli še analizo občutljivosti na spremembe osrednjih povezav omrežja, ki je prikazana na Sliki 6.3. Pri 10% spremembi povezav sta natančnost in priklic vodilnih vozlišč padli za približno 20%.

Izvedli smo vrednotenje strategij za odstranjevanje povezav v primeru vzročno nasprotujočih vrednosti med podanimi podatki, ki je prikazano na Sliki 6.4. Uporabili smo le sintetične nabore podatkov z vnaprej določeno stopnjo nasprotovanja. Rezultati so pokazali znatni padec ocen NPA za vzročno nasprotujoče podatke ne glede na izbrano strategijo. V primeru vzročno skladnih podatkov je odstranjevanje povzročilo delno povečanje ocen NPA, nabori podatkov z naključnim nasprotovanjem pa so zasedli širok vmesni spekter. To nakazuje, da nekatere izmed vhodnih vrednosti pomembno vplivajo na izračunane rezultate. V splošnem priporočamo, da uporabniki omejijo delež odstranjenih podatkov z nastavitvijo ustreznega praga.

Pri ocenjevanju permutacij smo se osredotočili le na oceno NPA. Rezultati permutacije “O”, ki zamenja vrstni red vhodnih podatkov, je podan na Sliki 6.5. Permutacija vedno zniža oceno NPA, kar poudarja pomembnost dodeljevanja podatkov pravilnim vozliščem. Rezultati permutacij “K”, ki so prikazani na Sliki 6.6. Omrežja, ustvarjena z uporabo modela Barabási-

Albert, so pokazala drugačen odziv v primerjavi z biološko pomenljivimi omrežji. To nakazuje, da imajo ta omrežja posebne lastnosti, ki jih naš proces generiranja ni zajel. V splošnem sklepamo, da sta obe vrsti permutacij veljavni metodi ocenjevanja statistične pomembnosti rezultatov.

Računsko in spominsko zahtevnost implementacije smo ocenili na prenosnem računalniku z AMD Ryzen 7 5800H procesorjem in 16 GB RAM spomina. V primerjavi z obstoječo implementacijo v jeziku R se je izkazala za več kot $2,5$-krat hitrejšo. Poleg tega smo izmerili čas trajanja in porabo spomina v primeru večjih, računalniško generiranih omrežij. Rezultati so prikazani na Sliki 6.7. Implementacija z razpršenimi matrikami je za omrežja z 10.000 jedrnimi in več kot 200.000 robnimi vozlišči algoritem izvedla v približno uri in pol, pri tem pa potrebovala manj kot 8 GB spomina.

VI Sklepi

Magistrsko delo predstavlja kavzalne biološke mreže (CBN) kot metodo za vodenje analize izražanja genov s predhodnim biološkim znanjem. Razvili smo paket Python PerturbationX, ki implementira algoritem TopoNPA in uporabnikom omogoča uporabo poljubnih omrežij. Podpira vizualizacijo preko integracije s Cytoscape, ekstrakcijo podgrafa na podlagi vodilnih vozlišč in izvoz rezultatov, obogatenimi z metapodatki. Dodane so bile razširitve algoritma TopoNPA, vključno z različnimi strategijami odstranjevanja odvečnih povezav in prilagodljivimi testi permutacije.

Za ovrednotenje algoritma so bila ustvarjena sintetična omrežja skupaj z ustrezajočimi nabori podatkov. Analiza občutljivosti je pokazala relativno robustnost algoritma v primeru šuma, medtem ko so manjkajoče vrednosti in spremembe povezav povzročile večje spremembe v rezultatih. Pri uporabi odstranjevanja vzročno nasprotujočih povezav pa svetujemo ustrezno omejevanja obsega odstranjevanja. Računska in prostorska zahtevnost sta bili optimizirani z uporabo razpršenih matrik. Upamo, da bo orodje spodbudilo razvoj in uporabo kavzalnih mrež pri analizi izražanja genov.

Chapter 1

Introduction

Networks have a long-standing and integral presence within the field of biology. Biological study includes the study of interactions and emergent properties at various levels of biological organisation, ranging from the molecular to the community level. Networks provide a logical representation of these systems and can be easily analysed further. This is most prominent within the domain of systems biology, which aims to understand biological systems through accurate modelling and inference of their components and interactions.

Junker, B. H. and Schreiber, F. [1] present an overview of biological networks. They specifically detail signal transduction and gene regulation networks, protein-protein interaction networks, metabolic networks, phylogenetic networks, ecological networks, and correlation networks. Many of these can be used in conjunction with measured data to infer the interactions within the networks or identify changes in network activity under different conditions. For example, gene expression data has been used to infer gene regulatory networks (e.g. Emmert-Streib et al. [32]), and to refine metabolic networks in the process of drug target prediction (e.g. Foler et al. [33]).

In this Master's thesis, we focus specifically on causal networks, which are a subset of biological networks with strictly causal relationships. While many of the aforementioned network groups contain causal connections, their focus is typically on building a complete model of the subject of study, regardless of the types of interaction present. Many relationships are therefore correlative rather than causal or occur due to transitive effects and unobserved variables. Causal networks can therefore be understood as a refined type of network, in which the validity and causality of each interaction has been adequately verified.

Causal networks provide valuable insight into the mechanistic nature of biological phenomena while simultaneously allowing for easy computational inference. Several methods for computation on causal graphs using gene expression data have been developed in the last two decades. One prominent method is the TopoNPA pipeline, which was published in 2014 by Martin et al. [2], and released in 2019 in the form of an R package [11]. However, this package has notable limitations, including only allowing computation for a pre-specified set of networks. Implementations of related methods suffer from similar problems, with many being either closed-source [15] or written in deprecated programming languages [18].

The purpose of this thesis is to develop and implement a fully functional pipeline for quantifying response-specific system perturbations from gene expression data and a corresponding causal biological network. The purpose and extent of functionality were determined in collaboration with the National Institute of Biology, and implemented in an open-source Python package named PerturbationX [3]. It supports flexible network import, export, and manipulation, an extended version of the TopoNPA pipeline, and visualisation of computed results with Cytoscape [4]. The package was developed using NetworkX [24] and pandas [25, 26] libraries.

In Chapter 2, we provide an overview of existing computational approaches that make use of causal networks. In Chapter 3, we describe the TopoNPA algorithm in greater detail, breaking down each of its computational steps. We then cover the process of the package feature specification and implementation in Chapter 4. In Chapter 5, we move on to the process of synthetic network and dataset generation. This is followed by an evaluation of algorithm robustness and performance in Chapter 6. Finally, we summarise our findings and conclusions in Chapter 7.

Chapter 2

Causal biological networks

The proliferation of high-throughput measurement technologies has presented itself as both an opportunity and a challenge. The comprehensive datasets that are produced can generate many useful observations about the biological systems in question. However, precise inference of biological mechanisms and causal relationships underlying the observed data remains difficult. One of the potential avenues for addressing this involves the use of causal biological networks (CBNs), which incorporate published information on known biological interactions. In this chapter, we present CBNs and several notable approaches that have been developed for their application.

CBNs offer a way to integrate prior knowledge in informed gene expression analyses. The use of prior biological knowledge is well-established through gene set and pathway analysis methods, categorised by Khatri et al. into over-representation analysis, gene set enrichment analysis and pathway topology [34]. While these methods provide valuable insights, they are not without drawbacks. Notably, gene set methods do not consider the regulatory structure governing gene expression changes [7], while pathway databases may fail to record context-specific knowledge of biological mechanisms [14].

This is where CBNs offer an advantage, aiming to capture data within well-defined contextual boundaries. They are typically composed of signed, directed edges, representing regulatory interactions between entities at vari-

ous levels of biological organisation. Such models depict the mechanisms in question with greater precision than other comparable resources. Additionally, the connections within the network are strictly causal. This allows for human interpretability as well as computational inference of the network [12]. Consequently, both the construction and analysis of CBNs promote a deeper understanding of the mechanisms driving gene expression patterns.

An important distinction between CBNs and other analysis methods based on prior knowledge is the direction of causal inference. Many methods rely on the "forward assumption" that gene expression changes are directly correlated with changes in the activity of their corresponding products. This assumption can introduce problems, as the correlation is typically modest and variable [5]. Instead, CBNs allow for the use of the "backward assumption". This approach groups genes that are regulated by the same biological entity and uses measured gene expression of these groups to infer information about the upstream regulators.

The rest of this chapter is split into two sections. We first discuss computational approaches related to the TopoNPA algorithm, which was implemented as part of the PerturbationX package [3]. We then move on to other approaches that incorporate causal networks, including a recent comparison study and a review of causality in biological network databases.

2.1 TopoNPA and related methods

An early computational approach using CBNs was presented in 2005 by Pollard et al. [6]. They describe causal models as a middle ground between more abstract association models (e.g. derived through co-occurrence or correlation), and more detailed quantitative models (such as those used for dynamic simulations). Their model, describing human skeletal muscle biology, was constructed from scientific literature using an ontological framework. Their computational method consists of two phases, called reverse and forward causal analysis.

Reverse causal analysis uses the model to identify the regulators most likely responsible for the observed changes in gene expression. During subsequent forward causal analysis, regulators are ranked according to the number of observed changes that they can explain. Using two statistical measures, the best hypothetical causes are then evaluated based on the likelihood that the changes could have occurred by chance alone. The measure of richness considers all changes mapped to the model, while concordance focuses on the changes whose directionality matches the model's prediction.

In 2012, Chindelevitch et al. [7] described a related methodology based on existing causal knowledgebases provided by Selventa Inc. and Ingenuity Inc. They first identified the shortest paths from each regulator node to gene transcript nodes within a specified distance threshold. The paths consist of signed edges that represent either up- or down-regulation. By multiplying the signs of all the edges on the path, they classified the direction of regulation for nodes in the transcript layer. Additionally, they duplicated every regulator to represent a decrease or increase in its quantity and adjusted the regulatory connections accordingly.

From this model, the directionality of gene expression changes can be predicted for every regulator node. Predictions are then compared with measured data, and the regulator node is assigned a score. This score is based on the number of correctly and incorrectly predicted changes in the data. As such a computation favours hubs, further selection is performed through statistical testing. They remark that the statistical measures of richness and concordance do not take into account the imbalance between up- and down-regulated transcripts. Consequently, they may over- or underestimate the significance of a hypothesis.

In the same year, Martin et al. [8] introduced four new CBN scoring methods, collectively named Network Perturbation Amplitude (NPA). They specified the structure of their CBNs in greater detail, defining them as a collection of interconnected hypothesis networks, also called HYPs. A HYP contains one upstream biological entity or process, along with several

measurable entities that it regulates. Larger HYPs are constructed through aggregation, combining networks while maintaining causal consistency for a reference regulator. The networks they constructed were assembled from literature and are encoded in the Biological Expression Language (BEL).

The four scoring methods introduced are intended for different data modalities. Although they focus on different aspects of biological activity, they were shown to perform similarly in the example study on TNFα-induced signalling. When measurements represent mRNA expression changes, HYPs can also be understood as gene expression signatures for the upstream entity or process. Additionally, Hoeng et al. [9] elaborated that scores for different HYPs can be combined into a more abstract Biological Impact Factor (BIF). With an appropriate selection of regulators and adequate calibration, this score can be used in phenotype prediction.

A Java implementation of the Reverse Causal Reasoning (RCR) method, named Whistle, was published by Catlett et al. [10] in 2013. This method is highly concordant with the one presented by Chindelevitch et al. [7]. By first inferring regulatory signatures for upstream entities, smaller HYP networks are constructed and assigned a score. However, the scoring methods introduced by Martin et al. [8] for BEL-encoded HYPs were not supported. Statistical measures were also limited to the richness and concordance measures first introduced by Pollard et al. in 2005 [6].

In 2014, two years after the initial publication of the NPA scoring methods, Martin et al. [2] proposed a separate topological scoring algorithm named TopoNPA. This algorithm relaxes the important constraint of causal consistency in CBNs. As a result, the required structure is simplified into a two-layer configuration. The upper layer contains regulators at different levels of biological organisation, called upstream biological entities (UBEs). The lower layer contains nodes representing gene expression changes or other measurable quantities.

TopoNPA is a threshold-free approach and does not require prior identification of up- and down-regulated genes. The network-scale perturbation amplitude is formulated as a function of all causal edges and may be used in phenotype prediction without further aggregation. Individual regulators (UBEs) are ranked according to their contribution to the perturbation. Additional measures of statistical significance are then computed to accompany the results. We discuss the algorithm in greater detail in Chapter 3.

The TopoNPA algorithm has been used in several systems toxicology studies, including studies on mouse [35, 36], rat [37], zebrafish [38], and human [39, 40] biology. Other NPA scoring methods have also been used to score causal networks with transcriptomic data [41]. An R implementation of TopoNPA was made available by Martin et al. [11] in 2019. This implementation, called simply NPA, allows the user to analyse any given dataset of gene expression change. However, it is limited to the networks provided in a companion package and does not allow their construction or modification. As such, it is not useful for the development or use of novel CBNs.

In 2015, Talikka et al. [12] presented the causal biological network database (CBND). It contains a collection of publicly available and manually curated CBNs. They are encoded in the BEL syntax supported by NPA, RCR and TopoNPA algorithms, and can be exported in both SIF (simple interaction file) and JSON formats. However, only the layer containing UBEs is available for each of the networks. A Python package for parsing and handling BEL-encoded networks was presented by Hoyt et al. in 2017 [13].

In 2019, Karki et al. [14] proposed another way of estimating perturbation using the Candidate Mechanism Perturbation Amplitude (CMPA) algorithm. CMPA does not consider perturbation to be a function of one central node as NPA [8], or a function of edges as TopoNPA [2]. Instead, perturbation is computed as a sum of hub scores. Hubs are defined as any node with both incoming and outgoing edges. Unfortunately, the algorithm is unable to process networks with causal loops. This makes it less flexible than the TopoNPA approach, despite not demanding strict causal consistency.

2.2 Other causal network methods

Four causal algorithms implemented in the commercial Ingenuity Pathway Analysis (IPA) software were presented by Krämer et al. [15] in 2013. Unlike other causal network approaches, these algorithms do not require a context-specific network. Instead, they use all available causal connections to infer the upstream regulators most likely to be involved. The scoring method is an approximation of the one defined by Chindelevitch et al. [7], and is easier to compute.

Once the regulators have been determined, they are used to construct dataset-specific networks containing relevant causal connections. These networks are further evaluated in a manner similar to the RCR method [10]. As the biological relevance of input connections is less strict than in related approaches, the generated networks will depend on the quality of input data. Finally, a causal method for identifying downstream mechanisms most likely affected by differentially expressed genes was also implemented.

As an alternative to commercial software, Bradley G. and Barrett S. J. presented CausalR [16] in 2017. CausalR is a causal reasoning implementation in R based on the network structure defined by Chindelevitch et al. [7]. However, instead of considering shortest paths, a user-specified maximum path length is used to generate predictions for gene transcript changes. If paths with conflicting causality are identified within the range, they are classified under ambiguous regulation.

Although the case of ambiguous regulation is also described in the work of Chindelevitch et al., it is significantly less likely to occur when considering only the shortest paths. To alleviate this issue, CausalR supports an alternative mode of computation called ScanR, where the user specifies multiple maximum path lengths. In this case, only the regulators that score highly across multiple parametrisations are retained. Additionally, CausalR also supports the generation of dataset-specific subnetworks in Cytoscape [4] compatible formats.

The same network structure was used in the work of Jaeger et al. [17] in 2014. They proposed and evaluated several methods for ranking nodes in directed networks. Their networks were constructed from Metabase [42] and STRING [43] interaction databases, duplicating every node to represent its up- and down-regulation. While their regulatory signatures were also computed using shortest paths, their scoring methods considered the magnitude of fold change measurements alongside directionality.

Out of the six proposed methods, SigNet performed the best at predicting causal targets from differential gene expression measurements. SigNet is a consensus approach, yielding the lowest rank among those produced by three of the other methods, namely Lambda, Power Weights and Exponential Weights. The Lambda score reflects only the proportion of differentially expressed genes that can be reached from a node, while the remaining two compute a weighted sum of normalised absolute fold changes in the dataset. Importantly, these methods were shown to perform better than the two that also incorporated the sign of the shortest paths.

Various approaches based on different formulations of causal networks and input variables have also been developed. One such method is TieDIE, an algorithm and software package presented by Paull et al. [18] in 2013. TieDIE does not make any assumptions about the graph structure underlying the causal network. Instead, it performs tied heat diffusion from nodes in specified source and target sets. By identifying the nodes lying in between, causally consistent pathways can be recovered. Heat diffusion strategies are also less sensitive to hubs, as nodes with many connections will both gain and lose more heat.

More recently, Liu et al. [19] published a causal reasoning implementation called CARNIVAL in 2019. CARNIVAL is based on an integer linear programming (ILP) approach proposed by Melas et al. [44]. CARNIVAL predicts the most likely signalling pathways for a given network. It supports a standard mode of computation, where targets of perturbation are given in advance, and inverse computation, where they are inferred alongside the

pathways. Using an ILP formulation allows for efficient computation, but also restricts it to networks without any positive feedback loops.

Originally, CARNIVAL was used on a protein-level signalling network constructed from OmniPath [45]. The inputs consisted of inferred transcription factor activities, although Liu et al. remark that the algorithm may be used on networks that map gene expression measurements to nodes directly. Additionally, the results can be further refined by providing the algorithm with predicted pathway activities. In 2021, Dugourd et al. [20] expanded on the approach by demonstrating how it may be used to integrate multi-omics datasets through the COSMOS method. COSMOS can generate mechanistic hypotheses for a prior knowledge network consisting of signalling, metabolic and gene regulatory networks.

Other recent systems biology approaches based on signalling networks can be found in the review by Garrido-Rodriguez et al. [21], published in 2022. To be included, approaches were required to have an open-source implementation, and output either a cellular signalling network or a ranking of networks. Notably, the methods were not restricted to causal networks and many incorporated gene regulatory interactions.

SigNet, CausalR and CARNIVAL were benchmarked in recovering dysregulated proteins and signalling pathways by Housseini-Gerami et al. [22] in 2023. SigNet always recovered the most direct targets, although performance varies by network. CARNIVAL and SigNet performed better on the smaller OmniPath network than the larger MetaBase networks, while the opposite was true for CausalR. The ScanR mode of CausalR was also shown to heavily favour hubs in target recovery.

A similar pattern of network dependence was observed in pathway recovery, with CausalR further depending on the selected mode of computation. SigNet once again performed the best on the OmniPath network. For the MetaBase networks, both SigNet and the ScanR mode of CausalR were competitive. All methods outperformed pathway enrichment using differentially expressed genes. Overall, Housseini-Gerami et al. conclude that algorithm

performance will depend on the properties of the input network, and more independent studies are required to draw firm conclusions.

The status of causality in biological databases was reviewed by Touré et al. [23] in 2021. They emphasised that a wide range of causal data is available, but that the concept of causality and its biological context varies between resources. The differences in accompanying metadata make mapping information between resources a challenging process. Additional documentation or tools are required to alleviate the difficulties in interoperability.

Chapter 3

Topological network perturbation amplitude

In this chapter, we describe the TopoNPA algorithm [2] and detail its individual steps. The goal of TopoNPA is to quantify a network-scale response to biological perturbations. This is performed using a two-step process, which first determines optimal coefficients for all nodes in the network. It then uses these coefficients to compute the Network Perturbation Amplitude (NPA) score. This score may be decomposed into contributions of individual nodes, enabling users to identify the most prominent parts of the network. Additionally, TopoNPA derives confidence intervals for all computed coefficients as well as the NPA score. It then performs two permutation tests to estimate the significance of the results. A diagram of the algorithm can be seen in Figure 3.1.

The rest of this chapter is composed of four sections, each covering a different part of the algorithm. In the first section, we define the input in both a biological and mathematical context. Next, we specify the central equations used for deriving the NPA score. We then move on to the computation of variance and confidence intervals. Finally, we conclude with a description of the accompanying permutation tests.

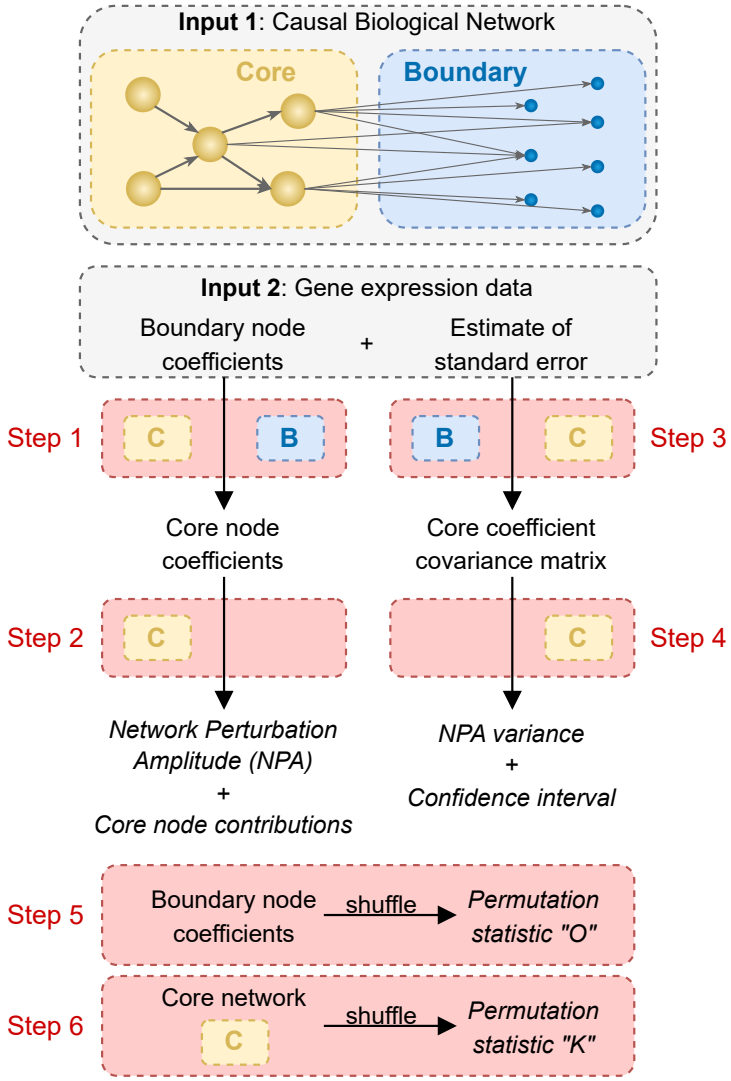

Figure 3.1: A diagram of the TopoNPA algorithm. Individual steps are denoted in red, while final results are italicised. The usage of CBN core and boundary is denoted with yellow and blue boxes, respectively. Permutation statistics are computed from NPA scores derived from the shuffled inputs.

3.1 Algorithm input

As TopoNPA is based on the "backward assumption" of causal inference, it expects an input network that is in line with this paradigm. This means that the network should contain two types of nodes: upstream and downstream nodes. In the intended biological setting downstream nodes represent gene transcripts for which we have obtained relevant gene expression change measurements. Upstream nodes are understood as regulators of both downstream nodes and other upstream nodes. They can represent proteins and other biological entities, as well as various molecular mechanisms. Regulation is represented with signed outgoing edges, where the sign indicates up- or down-regulation.

Typically, there are many more regulatory connections specified for downstream nodes than between upstream nodes. Outgoing regulatory connections are not expected for downstream nodes, as they do not help the TopoNPA optimisation process. Gene expression changes are the second required input to the algorithm and must be accompanied by an estimate of their standard error, such as the t-statistic. If a gene expression measurement is not provided for a downstream node, the node and its associated edges are ignored. On the other hand, if a measurement is provided for an upstream node, the measurement is ignored.

In a more general setting, the algorithm expects a core subgraph along with its boundary. The core subgraph corresponds to upstream nodes and the connections between them. The subgraph's boundary corresponds to downstream nodes and their incoming edges. The second input can therefore be understood as a set of values specified for vertices in the outer boundary of the core subgraph. An example network structure can be seen in Figure 3.1. The core subgraph is denoted in yellow and contains vertices that are upstream in terms of causality, representing biological regulators. The outer boundary is denoted in blue and contains vertices that are downstream, representing gene transcripts.

3.2 Computing NPA

The main result of TopoNPA is the NPA score, which is computed in two steps. They are denoted in Figure 3.1 as Step 1 and Step 2. The first step infers values for core vertices, i.e. the upstream regulators. The inferred values are understood as analogues to the given input values, typically fold change measurements. As several types of values can be assigned to vertices, we refer to these as coefficients for greater specificity. Once coefficients have been determined for all vertices, the second step computes a network-scale response as the NPA score. Both steps are formulated primarily in terms of network edges, in order for the results to adequately reflect the network's structure.

We first specify the necessary mathematical notation. In this notation, if a scalar function is used on a vector, it is applied element-wise. Let $G = (V, E, w)$ be the weighted directed graph underlying the input network. Let V_c be the core vertices corresponding to upstream nodes, and V_b the boundary vertices corresponding to downstream nodes, so that $V_c \cup V_b = V$ and $V_c \cap V_b = \emptyset$. Let $\boldsymbol{v}$, $\boldsymbol{v_c}$ and $\boldsymbol{v_b}$ be vectors representing a fixed ordering of V, V_c and V_b, respectively. Let $E_c \subset V_c \times V_c$ be the edges in the core subgraph, and $E_b \subset V_c \times V_b$ the subgraph's edge boundary, such that $E_c \cup E_b = E$ and $E_c \cap E_b = \emptyset$. Let $w : E \to \mathbb{R}$ be the weight function on edges, and $\sigma : E \to \{-1, +1\}$ the sign function on edges, such that $\forall e \in E : w(e) = \sigma(e)|w(e)|$. Finally, let $f : V \to \mathbb{R}$ be the function that maps vertices to their coefficients.

3.2.1 Core vertex coefficients

The step S1 performs coefficient inference through constrained optimisation. The process can be understood as an estimation of missing measurements for the function f. The loss function for this optimisation is defined as

$$L(f, G) = \sum_{(x,y)\in E} \left[f(x) - \sigma(x, y)f(y)\right]^2 |w(x, y)|. \tag{3.1}$$

Intuitively, this process aligns endpoint coefficients for positive edges, while opposing endpoint coefficients for negative edges, in the sense of the additive inverse. Martin et al. [2] remark that this optimisation is related to a Dirichlet boundary condition problem in spectral graph theory [46].

The absolute values of edge weights control edge significance during optimisation. Edges with larger absolute weights contribute more to the loss function. They are therefore better accommodated by the optimisation. The weights of all core edges are set to 1. The weight of a boundary edge depends on its inner vertex. For a vertex on the inner boundary with k boundary edges, the absolute weight of all its boundary edges is set to $\frac{1}{k}$. This ensures that the boundary does not overtake the core subgraph in significance, as there may be many more regulatory connections for gene transcripts.

The solution to the optimisation $\min_f L(f, G)$ can be derived analytically and computed using matrices. Let $\boldsymbol{A}$ be the weighted adjacency matrix of G, defined as $A_{x,y} = w(x, y)$, where the weight of non-existent edges is 0. We can define the signed Laplacian of the undirected graph as $\boldsymbol{L} = \text{diag}(\deg(\boldsymbol{v})) - \left(\boldsymbol{A} + \boldsymbol{A}^T\right)$. Notably, the degree of a vertex is computed using absolute edge weights, such that $\forall x \in V : \deg(x) = \sum_{y \in V, y \neq x} |w(x, y) + w(y, x)|$. Additionally, let $\boldsymbol{L_c}$ denote the sub-matrix of $\boldsymbol{L}$ with rows and columns corresponding to core vertices, and $\boldsymbol{L_b}$ denote the sub-matrix with rows corresponding to core vertices and columns corresponding to boundary vertices.

We can compute the coefficients of core vertices according to the equation

$$f(\boldsymbol{v_c}) = -{\boldsymbol{L_c}}^{-1} \boldsymbol{L_b}\, f(\boldsymbol{v_b}), \tag{3.2}$$

where $f(\boldsymbol{v_c})$ and $f(\boldsymbol{v_b})$ are coefficient vectors for core and boundary vertices, respectively. For the solution to uniquely exist, the matrix $\boldsymbol{L_c}$ must be invertible. While Laplacian matrices are typically only weakly diagonally dominant, the degrees of core vertices here also account for boundary edges. Due to how weights are assigned, core vertices with boundary edges have their degrees raised by 1. Then, as long as the core subgraph is connected, the sub-matrix $\boldsymbol{L_c}$ will be irreducibly diagonally dominant and invertible.

3.2.2 Perturbation amplitude

In the second step S2, we compute the network-scale perturbation amplitude using the equation

$$TopoNPA(f, G) = \frac{1}{|E_c|} \sum_{(x,y)\in E_c} (f(x) + \sigma(x, y)f(y))^2 |w(x, y)|. \tag{3.3}$$

It is formulated similarly to the loss function used for coefficient inference, with three key differences. Firstly, the result is weighted by the total number of core edges. Secondly, it considers only edges in the core subgraph, meaning that boundary vertices are no longer relevant. Finally, the edge sign is inverted. If we assume that the optimisation in the first step produced a perfect fit for a given edge, then this equation will return the combined amplitude of its vertex coefficients. If an edge's weight is positive, then the coefficients will have equal signs, and vice versa. This relationship is depicted as a heatmap in Figure 3.2. Edge significance is once again controlled by absolute edge weights, which are set to the same values as before.

The result of this step can also be computed using appropriate matrices. Let $\boldsymbol{A_c}$ be the weighted adjacency matrix of G, restricted to only the rows and columns that correspond to core vertices. Let $\deg_c : V_c \to \mathbb{R}$ be the function that maps vertices to their degrees in the core subgraph, such that $\forall x \in V_c : \deg_c(x) = \sum_{y\in V_c, y\neq x} |w(x, y) + w(y, x)|$. We can then define a second Laplacian-like matrix $\boldsymbol{Q} = \text{diag}(\deg_c(\boldsymbol{v_c})) + \boldsymbol{A_c} + \boldsymbol{A_c}^T$. Apart from having opposite edge signs, this matrix differs from $\boldsymbol{L_c}$ in that the degrees of vertices with boundary edges are not raised by 1. The resulting amplitude can then be computed with the equation

$$TopoNPA(f, G) = \frac{1}{|E_c|} f(\boldsymbol{v_c})^T \boldsymbol{Q} f(\boldsymbol{v_c}). \tag{3.4}$$

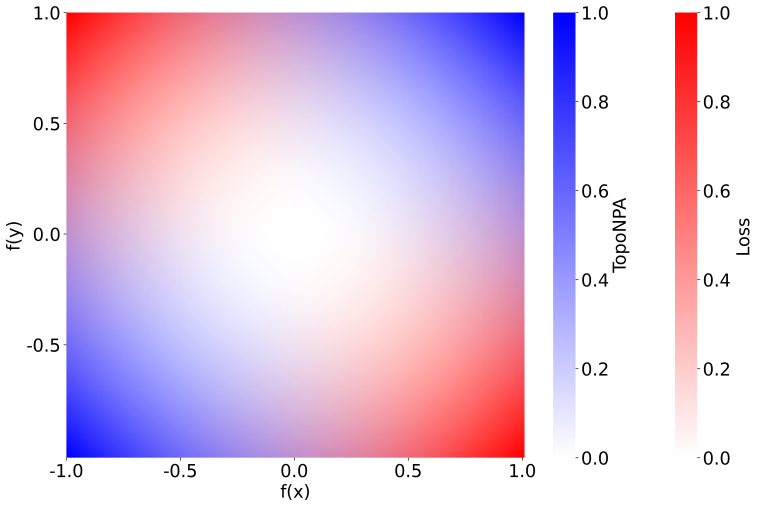

Figure 3.2: Visualisation of loss and TopoNPA score for a single positively weighted edge with endpoint coefficients $f(x)$ and $f(y)$. The two scores are encoded using different colour channels, with red representing loss and blue representing the TopoNPA score. For a negatively signed edge, the plot would be flipped horizontally.

By rearranging the terms in equation (3.3), we can express the perturbation amplitude as a sum over vertices, which is given in equation

$$TopoNPA(f, G) = \frac{1}{|E_c|} \sum_{x \in V_c} f(x) \left(\deg_c(x) f(x) + \sum_{y \in V_c, y \neq x} w(x, y) f(y) \right). \tag{3.5}$$

This is useful because it allows us to break down the amplitude into vertex contributions. Consequently, we can identify vertices with the most perturbed edges. The nodes corresponding to these vertices are called leading nodes and are the first statistic that the TopoNPA algorithm produces. By limiting the network to the most perturbed nodes, it is easier to identify the biological mechanisms that drive the network's response. By default, the set of leading nodes is defined as the most perturbed nodes that together make up 80% of the perturbation. However, while the total amplitude is necessarily non-negative, this does not hold for individual contributions.

3.3 Variance and confidence intervals

The second statistic that TopoNPA computes are variances and confidence intervals, both for inferred core vertex coefficients, as well as the final network perturbation amplitude. This is depicted in Figure 3.1 with Steps 3 and 4. It requires additional input, an estimate of standard error for the boundary vertex coefficients, e.g. the t-statistic. It is assumed that the input variables, i.e. fold changes for transcript nodes, are mutually independent normal random variables. Under this assumption, it is easy to compute the covariance matrix for core vertex coefficients using the equation

$$\boldsymbol{\Sigma} = \boldsymbol{L_c}^{-1}\boldsymbol{L_b}^{T}\text{diag}(\text{Var}(f(\boldsymbol{v_b})))\boldsymbol{L_b}(\boldsymbol{L_c}^{-1})^{T}. \tag{3.6}$$

This is because the core vertex coefficients are linear combinations of the input variables. From there, the variance of the final TopoNPA score is calculated with the equation

$$\text{Var}(TopoNPA) = 2\text{tr}(\boldsymbol{Q\Sigma Q\Sigma}) + 4f(\boldsymbol{v_c})^{T}\boldsymbol{Q\Sigma Q}f(\boldsymbol{v_c}). \tag{3.7}$$

By default, a 95% confidence interval based on the variance is returned.

3.4 Permutation tests

Finally, permutation tests evaluate whether the network's structure is relevant to the computed perturbation score. This is depicted in Figure 3.1 with Steps 5 and 6. TopoNPA defines two permutation p-statistics, called the "O" and "K" statistic. The "O" permutation test shuffles the labels of boundary vertices. Equivalently, it reassigns fold change measurements to transcript nodes at random. The "K" permutation test instead permutes the network structure, reassigning edges in the core subgraph during inference of core vertex coefficients. Once the coefficients have been sampled, the NPA score is computed using the network's original structure. The distributions of NPA scores are then used to determine the p-statistics of the TopoNPA result.

By default, both permutation tests operate on the entire network structure. The "O" permutation shuffles all given boundary vertex coefficients and the "K" permutation shuffles all the edges between core vertices. In this way, the "O" statistic estimates the significance of the mapping between input data and boundary vertices, testing the null hypothesis that the ordering of boundary coefficients does not affect the NPA score. The "K" statistic instead tests the significance of edges in the core subgraph, with the null hypothesis being that they have no importance in quantifying the perturbation. A TopoNPA result is considered significant if both the "O" and "K" statistics are under 0.05.

Chapter 4

Python development and feature implementation

In this chapter, we describe the process of development, and the implementation details of our Python tool, PerturbationX [3], available for installation from both GitHub and PyPI. PerturbationX is a Python package based on NetworkX [24] and pandas [25, 26]. While it is primarily focused on the TopoNPA pipeline [2], Cytoscape [4] visualisation, and related use cases, it is designed to be easily extensible with additional perturbation-based algorithms. The structure of the package is depicted in Figure 4.1.

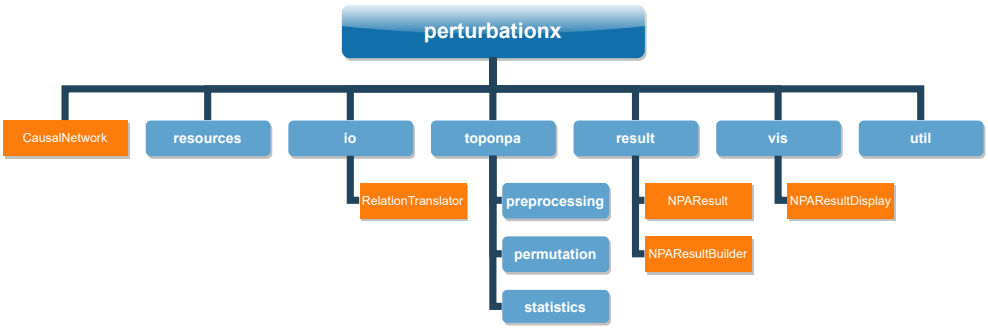

Figure 4.1: Structure of the PerturbationX package. Subpackages are depicted in blue, and classes are depicted in orange. Supporting modules are omitted for clarity.

Development of the package began with an implementation of the TopoNPA algorithm that is faithful to the original R pipeline [11]. Afterwards, additional features were scoped and implemented as extensions to the functional interface, aiming to preserve existing functionality. Throughout development, emphasis was placed on good separation of concerns, refactoring and reorganising code as necessary. Feature requirements were gathered iteratively in the form of user stories, emphasising their purpose and scope. Gathering requirements took place partly in the form of bi-weekly meetings with Dr. Carissa Robyn Bleker, Dr. Anže Županič, and Dr. Tomaž Curk, and partly in the form of a group interview with a larger group of employees at the National Institute of Biology. We extend our heartfelt gratitude to everyone involved for their valuable insight and ideas throughout this process.

The package is composed of multiple subpackages, each containing code related to a specific set of features. The `io` subpackage contains code for the import and export of causal networks, and the `RelationTranslator` class. The `resources` subpackage contains global default parameter values and the default visualisation style. The majority of the code is part of the `toponpa` subpackage, which implements features related to the extended TopoNPA pipeline. As the operations are complex, some are organised within additional subpackages. In the `result` subpackage is the `NPAResult` class, intended for containing computed results, as well as the associated `NPAResultBuilder` class. The `vis` subpackage contains the `NPAResultDisplay class`, which creates and adjusts a network display in Cytoscape, and also the code necessary for this visualisation. Lastly, the `util` subpackage contains helper functions that are non-specific to any other group of features. At the top level is an additional `CausalNetwork` class, which is intended to serve as the main interface for accessing implemented functionality.

In the following sections, we describe in greater detail the implementation of specific features. First, we describe features related to the import and export of causal networks and computational results. Next, we cover the implementation of different network permutations. We then move on to

the TopoNPA algorithm, including all the adjustments and extensions to its pipeline. Next, we describe the implementation of network visualisation in Cytoscape. Finally, we provide a summary of identified limitations and implemented changes.

4.1 Import and export

This section describes the import of networks and datasets, the interpretation of edge relations and types, as well as the data structures used to represent computed results and the associated metadata.

4.1.1 Network initialisation and computation

The `CausalNetwork` class is initialised with an existing NetworkX `DiGraph` (directed graph) instance. If a different type of graph is passed as an argument, a conversion to `DiGraph` will be attempted. Initialising an empty `CausalNetwork` by passing a `None` value is also allowed. Additional class methods are present to help create a `CausalNetwork` from DSV (delimiter-separated value) files, Cytoscape JSON files, and pandas `DataFrame` instances. As some users may wish to store different parts of the network in different files, edges may be added to an existing `CausalNetwork` instance in a similar manner.

The `DiGraph` instance is first validated by assuring that every edge has a specified relation and type. The relation of an edge describes the type of interaction between the two entities and may take any value. This property is mandatory, and edges without a specified relation are removed. The type of an edge is optional and may be `core`, `boundary`, or `infer`. Missing and unexpected values are replaced with `infer`. Afterwards, datasets can be passed in the form of a Python dictionary that maps dataset names to pandas `DataFrame` instances. Each `DataFrame` must have columns specifying node identifiers and coefficients and must match at least some of the boundary nodes in the network.

4.1.2 Translation of edge relations and types

While the BEL syntax was developed for specifying causal edge relations, we wanted to enable users to import networks in any syntax. For this reason, the `RelationTranslator` class was implemented to help with parsing. The `CausalNetwork` instance retains the original relations for inspection and visualisation purposes. A `RelationTranslator` is then used to convert relations into numeric weights whenever necessary. By default, `increases`, `directlyIncreases`, `->`, `=>`, and `1` are mapped to a weight of 1, while `decreases`, `directlyDecreases`, `-|`, `=|`, and `-1` are mapped to a weight of -1.

Additionally, a conversion to a floating-point number will be attempted for unknown relations. Importantly, this enables the use of floating-point weights and allows the user to adjust the importance of individual edges. The user may override and expand the default map of relations to whatever syntax they are using, and may also disable the conversion to floating-point numbers. In cases where a relation cannot be correctly mapped, a weight of 0 is returned. An example of translating relations into weights is displayed in Figure 4.2.

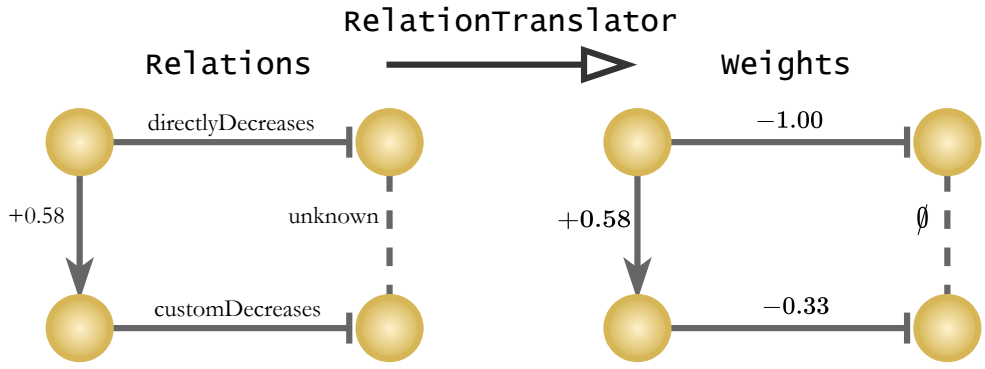

Figure 4.2: Example use of `RelationTranslator` to translate relations to weights. Causal BEL relations are mapped to weights of 1 or -1, numeric values are parsed directly, and a weight of 0 is assigned to unknown relations. Custom mappings between relations and weights can also be specified.

The type of the edge may also be automatically inferred by our implementation. This action will be performed if the type of an edge is not specified or if the user explicitly sets the edge type to `infer`. Inference is performed based on the expected network structure. Due to the nature of the algorithm, boundary nodes are not expected to have any outgoing connections. Consequently, any edge that points to a node with outgoing connections is considered to be a part of the network core. Otherwise, it is considered to be a part of the network boundary. While users will likely wish to specify edge types themselves during computation, they may want to infer edge types during network construction, in order to easily identify dangling nodes and edges.

4.1.3 Computed results

When storing and returning computational results, the `NPAResult` class was designed to offer a high degree of flexibility. The class is initialised with four types of results: global information, node information, distributions, and metadata. A graph instance representing the network state at the time of computation is also required.

Global and node information are stored in pandas `DataFrame` instances. Global information is indexed by dataset name, and stores computed results in columns. Node information is indexed by both dataset name and node attribute, with columns representing individual nodes. When accessing this information, the user may specify either a dataset name or node attribute, and a transposed cross-section will be returned.

Distributions are stored in a Python dictionary as pairs of a value list and a reference value, the latter of which may be `None`. They are indexed by permutation and dataset name. Metadata is the last type of attribute and is also stored in a simple Python dictionary.

As the construction of the required data structures can be quite complex, an associated builder class was implemented. The `NPAResultBuilder` requires only a graph and list of datasets during initialisation, and stores all results in Python dictionaries. The list of core nodes, for which attributes are stored, is inferred from the graph. Different kinds of results may then be added either individually or in bulk through a more fine-grained functional interface. Once finished, the build method will produce a valid `NPAResult` instance based on currently stored information.

Apart from direct retrieval of stored data, the `NPAResult` class offers some additional functionality. Individual distributions can be plotted as histograms. A list of leading nodes can be retrieved based on any given node attribute and cut-off ratio, allowing users to deviate from the original definition of leading nodes. Lists of nodes and edges in a node-induced subgraph can be obtained for the stored graph and a given set of nodes. A copy of the stored graph can be retrieved, with all node information embedded as node attributes. Results can be converted into a combined Python dictionary, and optionally stored as a JSON file. Finally, a Cytoscape display of the network and results can also be initialised.

4.1.4 Metadata

Importantly, the metadata associated with results is intended to store rich context information. This information can ease the process of publishing generated results in accordance with FAIR (findability, accessibility, interoperability, and reusability) principles [47]. Upon generation, the `NPAResult` class will automatically add information about the current date and time, package version, Python version and implementation, as well as system name, release, and version. Additionally, all attributes of the graph instance are copied to the metadata map. If the attribute key starts with `network` or `dataset`, it is copied without change, otherwise, a `network_` prefix is added.

In this way, the user may add any amount of optional information about the graph or datasets before running the computation. After initialising a `CausalNetwork` instance, the metadata may also be directly specified through the metadata attribute. The computational pipeline also adds information about the number of nodes and edges in the network, both before and after matching them with the provided datasets, as well as the parameter values used during computation.

While most metadata is optional, the attributes of `title` and `collection` are mandatory, as they are necessary for visualisation through the Cytoscape API [28]. If they are not specified, they default to `Untitled network` and `Untitled collection`, respectively. When initialising the `CausalNetwork` from a file, the file and folder names are used to automatically infer the network title and collection.

4.2 Permutations

This section covers the implementation of network permutations. We first cover the adjustments to permutation tests, before moving on to network modifications through edge rewiring.

4.2.1 Permutation tests

The permutation tests that are used to verify the significance of TopoNPA results were adjusted in their implementation. The first implemented change was the addition of a permutation rate parameter. Originally, both permutations permuted all of the relevant network data, with the "O" permutation permuting all boundary nodes, and the "K" permutation permuting all core network edges. By reducing the number of nodes and edges that are being permuted, the user can generate a stronger proof of significance.

Additional changes were made to the "K" permutation. In the R implementation of TopoNPA, the network core is permuted completely at random. Consequently, permuted networks are not guaranteed to be weakly connected. As the algorithm requires the core Laplacian matrix to be weakly diagonally dominant, each weakly connected component must have at least one node with boundary edges. When this condition isn't met, invalid results can be produced. Therefore, we adjusted our implementation to connect the weakly connected components in the permuted network cores.

Additionally, we split our implementation of the "K" permutation into two variants. The "K1" permutation is similar to the one from the R implementation of TopoNPA, producing connected network cores with randomly distributed edges. The "K2" permutation instead constructs network cores according to the configuration model [48], respecting the degree sequence. We believe that preserving the degree distribution creates network cores that are more biologically likely, producing a stronger statement of significance.

Another aspect of the core "K" permutation that was reconsidered is that only one of the core Laplacian matrices is changed during computation. Conceptually, this means that we are sampling core coefficients from the permuted networks, and then computing the NPA score using the initial network structure. Instead, we allow the user to compute full core permutations, permuting both core Laplacian matrices. In this case, we instead sample the final NPA score directly from the permuted networks.

4.2.2 Edge rewiring

Another use case related to permutations was determined while gathering of feature requirements. When adjusting the network structure, the user may want to randomly permute a portion of the network, and evaluate the modifications. Instead of modifying individual edges, the user may want to establish a connection between two parts of the network without specifying the exact location of the edges. To support this feature, two methods were added to the `CausalNetwork` class, called `wire_edges` and `rewire_edges`.

The method `wire_edges` accepts the number of edges, a list of node names, and a list of possible relations, and at each iteration randomly generates the requested number of edges between the specified nodes. The method `rewire_edges` instead accepts only a list of nodes and a permutation rate. At each iteration, it selects a subset of edges between the specified nodes to rewire, either completely at random, or by respecting the existing degree distribution through the configuration model.

The permutations here are not generated as adjacency or Laplacian matrices, but rather as a list of edge modifications that can be passed to a `CausalNetwork` instance to change its existing structure. If datasets are provided, TopoNPA scores are computed for each list of modifications, otherwise, the modifications alone are returned. As the user would like to know the actual NPA scores produced by the modified networks, we only support full core permutations with this method.

The process of rewiring can be further customised by specifying a confidence weight for each network edge separately. This property is completely optional and defaults to a weight of 1. It is also distinct from the weights generated by `RelationTranslator`, as those determine the significance of the edge during the computation of the NPA score, and can be understood to represent the strength of the edge. Confidence weights instead represent the user's confidence in the edge and are ignored outside of the `rewire_edges` method. They may also be imported in the same way as the other edge properties.

Once confidence weights are specified, the user may pass `confidence` instead of a numeric value as the permutation rate. The method will then perform a random check at every edge to determine whether or not to rewire it. This is useful when the specified subset of nodes also contains edges that we do not wish to rewire. In either case, the user is notified if a generated network permutation is identical to the original due to high confidence settings. However, such networks are still accepted as valid and returned, in order for the program to avoid infinite looping.

4.3 TopoNPA algorithm

To improve the scalability of the TopoNPA algorithm and related computations, we implemented a sparse mode of computation using SciPy [49] sparse arrays. This allows TopoNPA to scale to much larger networks than would otherwise be possible due to the large memory complexity of Laplacian matrices.

Alongside the standard pipeline, which supports the computation of NPA scores, confidence intervals, and permutations, a second method of running the algorithm was also enabled. This method computes only NPA scores for a given set of network modifications, which are generated by the `wire_edges` and `rewire_edges` methods. Additionally, the user may run only the first part of the pipeline, inferring graph attributes such as edge types and node indices and verifying their network structure before computation.

The rest of this section focuses on alterations regarding edge pruning in the phase of dataset validation. We first introduce the concept of opposing edge pruning and then present the different variations of edge pruning that are supported by PerturbationX.

4.3.1 Opposing edge pruning

An aspect of the TopoNPA algorithm that was examined in greater detail is the way it handles directionality in measured coefficients. For example, the network boundary might contain an edge that states entity A activates measurable entity B. If measurements suggest an upregulation of entity B, the algorithm will infer a similar upregulation of entity A. However, the opposite is also true. In the case of the downregulation of entity B, the downregulation of entity A will also be inferred.

This can be beneficial when both cases are sensible, e.g. when entity A represents the abundance of a biological compound. However, this condition may not always hold. If entity A represents the inhibition of a biological process, then a decrease in inhibition may not be an attributable cause for the

downregulation of entity B. Instead, a second node representing activation of the same process should be constructed, inverting all relevant edges.

To prevent the TopoNPA algorithm from falsely interpreting measured data, a second mode of computation was implemented. This mode performs a stricter pruning of boundary nodes and edges, removing more than just those for which data was not given. Additionally, because TopoNPA is a threshold-free approach, many of the boundary coefficients are expected to be insignificant or otherwise low in amplitude. The removal of edges for all opposing values, regardless of their amplitudes, could lead to a significant loss of information. PerturbationX therefore allows the user to specify a minimum amplitude for this process. Only boundary coefficients with amplitude higher than this threshold are considered during opposing value pruning.

4.3.2 Types of edge pruning

Initially, two variants of opposing edge pruning were proposed. In the first variant, only edges with conflicting causality would be removed. In the second variant, both nodes and edges would be removed if the percentage of conflicting edges reached a specified threshold. However, as boundary nodes frequently have edges with highly conflicting causality, only the first variant was implemented. While the different types of edge pruning were developed in the context of opposing boundary coefficients, they may also be used when dealing with missing boundary coefficients.

When removing an edge, the algorithm has two options. In the first case, the edge is removed and boundary weights are rebalanced, resulting in a state equivalent to the edge being removed from the network. In the second case, the edge weight is removed from the matrix, but no rebalancing is performed. This results in the edge's weight being set to zero, while maintaining the rest of the weights as if an additional edge were present. In this case, the boundary outdegree for core nodes is no longer equal to 1, but to the sum of remaining edge weights. We refer to the first case as edge removal, and the second case as edge nullification.

The two types of pruning can be applied to both missing values and opposing values separately. The original implementation of TopoNPA in R nullified the edges of nodes with missing values and kept the edges with opposing causality. Additionally, due to the pre-computation of network matrices, they did not account for the reduction of boundary outdegrees during nullification.

Consequently, all core nodes with boundary edges had their degrees increased by 1, regardless of the number of edges that were nullified. This results in two subtypes of nullification pruning, one using exact boundary outdegrees, and one using binary outdegrees. Examples of the different edge pruning variations can be seen in Figure 4.3.

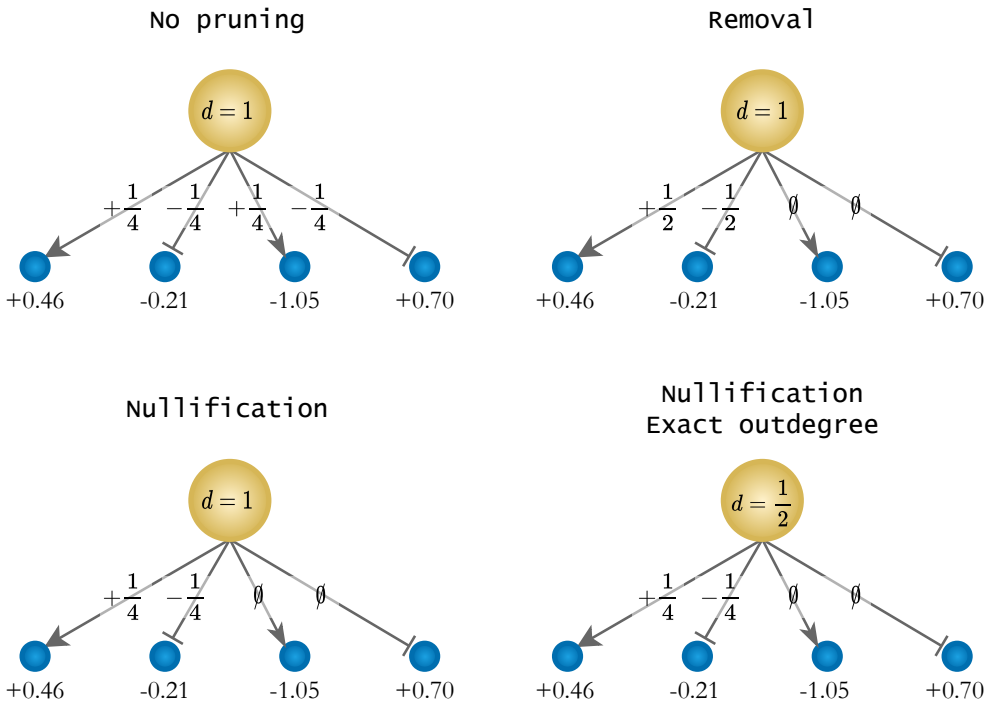

Figure 4.3: Variations of opposing edge pruning. Removal rebalances the weights of remaining edges, while nullification does not. As the core node outdegree is reduced during nullification, we can choose whether or not to use exact values during computation.

Another important consideration for boundary edges is that their influence on the NPA score should be limited. This is because there are typically one or two orders of magnitude more boundary edges in a network than there are core edges. When removing edges and rebalancing their weights, the importance of some boundary edges can therefore increase to an unfavourable amount. Consequently, an additional stage of clipping was added, which removes all boundary nodes of a core node with a boundary outdegree of 5 or less. The exact threshold can be adjusted by the user.

Because all types of pruning can be sensible from a biological standpoint, we decided to support all possible algorithm parametrisations. Removal of edges is primarily recommended when they are considered to be wrong, and should not have been included for a given dataset to begin with. Edge nullification is instead recommended when the edges are not wrong, but may be missing some biological context for the given situation. The remaining parameters of minimum opposing value amplitude, minimum boundary outdegree, and boundary outdegree type should be set according to the network and datasets being used.

4.4 Visualisation

Cytoscape visualisation was implemented through the py4cytoscape [27] library, which uses the Cytoscape REST API [28] to automate network operations. The display is initialised through a method of the `NPAResult` class and managed through the `NPAResultDisplay` class. This allows the same set of results to be visualised multiple times, enabling the user to create parallel visualisations that focus on different aspects of the network. Because network identifiers are not preserved across Cytoscape sessions, visualisation methods can also be accessed without the `NPAResultDisplay` class, instead using a network SUID and manual input of nodes and edges.

The visualisation can be manipulated in several ways. By default, only the network core is loaded and displayed, as the number of boundary nodes is typically too large to visualise comprehensibly. This can be changed both during the initial generation of display and afterwards through the `NPAResultDisplay` class. Additionally, nodes can be coloured using any node attribute specified in the node information `DataFrame`. While a default colour scheme is provided, the user may override it as they please.

4.4.1 Leading node display

An important feature of the TopoNPA pipeline is the easy identification of leading nodes, which contribute the most to the NPA score. Consequently, the visualisation of leading nodes was also implemented. As the method uses the `NPAResult` class internally, it is possible to define leading nodes using different attributes and cut-offs than in the original pipeline. The default parameter values will return the same set of leading nodes, defined as the top nodes contributing 80% of the NPA score together.

The `NPAResultDisplay` supports three ways of visualising leading nodes in Cytoscape. Firstly, they can be highlighted by adjusting the width of node borders and edges. Secondly, they can be extracted in place by hiding the remaining network nodes. Finally, they can extracted by generating a new Cytoscape network containing only leading nodes.

Because the leading nodes typically represent the most perturbed, and therefore the most interesting parts of the network, extracting a subnetwork based on these nodes was also prioritised. Although it was originally suggested to use a separate pathfinding tool for this purpose, such as TieDIE, this was found to be more complex than necessary. It is more likely that a user will be interested in extracting a subnetwork based on easily understandable criteria, such as shortest paths and closest neighbours.

Both of these options were implemented and can be used together. Shortest paths were implemented with a length tolerance, which can be given as an absolute number of edges, or a ratio of the shortest path length. Then, the shortest paths up to the specified tolerance are returned for each pair of leading nodes. Additionally, the method may be restricted to only consider directed paths, which are paths with consistent edge directions. Examples of path-induced subnetworks are displayed in Figure 4.4.

Closest neighbours are computed with a similar level of flexibility. The user can specify the maximum distance from a leading node, which determines whether a node is considered to be a neighbour. Concurrently, the neighbourhood type can be set to either union or intersection. If a neighbourhood type of union is selected, then a node must be selected as a neighbour of at least one leading node to be included. Otherwise, only nodes that are selected as neighbours of all leading nodes are returned. Examples of neighbour-induced subnetworks are displayed in Figure 4.5.

4.5 Summary

Table 4.1 contains a list of some identified shortcomings in both the TopoNPA pipeline and the R implementation. Alongside every limitation is a brief description of the solution implemented in PerturbationX.

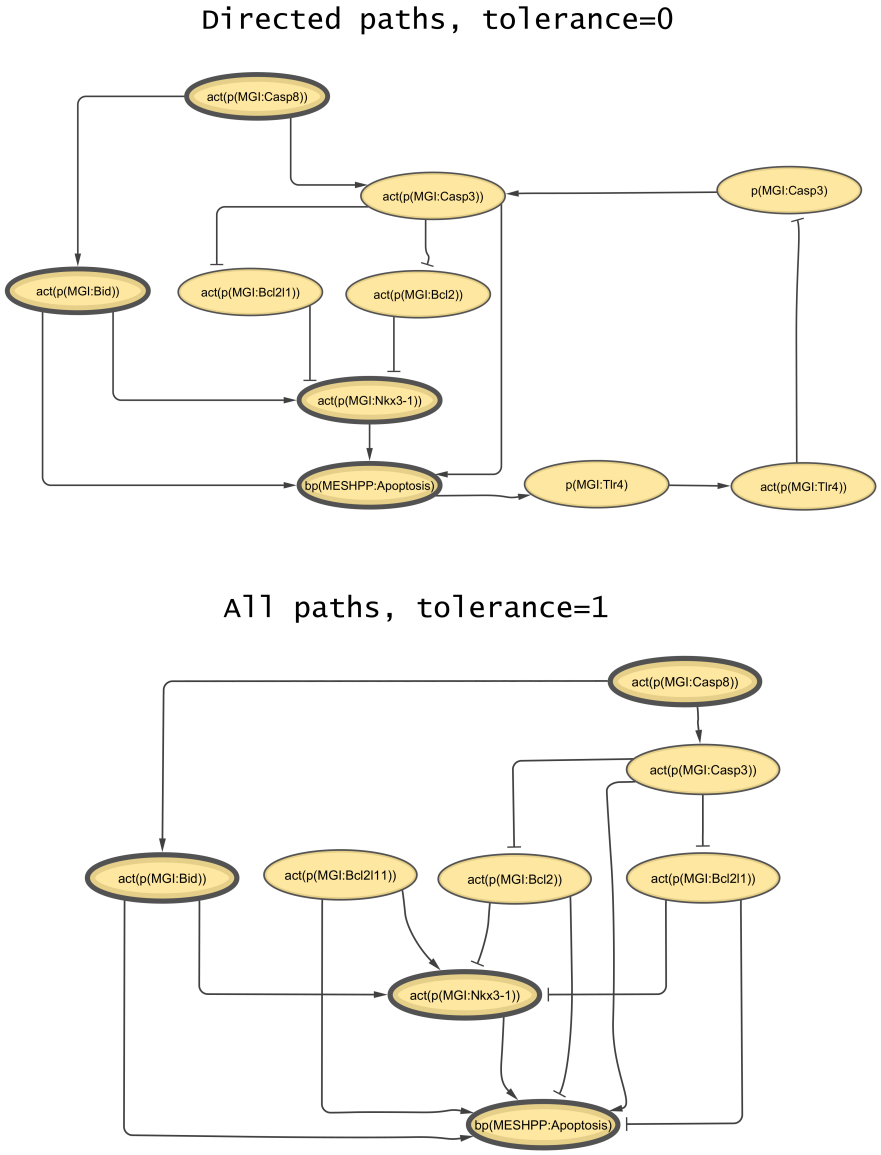

Figure 4.4: Examples of path-induced subnetwork visualisations. Leading nodes are displayed in bold. The top image shows all directed shortest paths between every pair of leading nodes and in both directions. The bottom image shows all shortest paths with a tolerance of 1. As most of the leading nodes are direct neighbours, this selects common neighbours of at least two leading nodes.

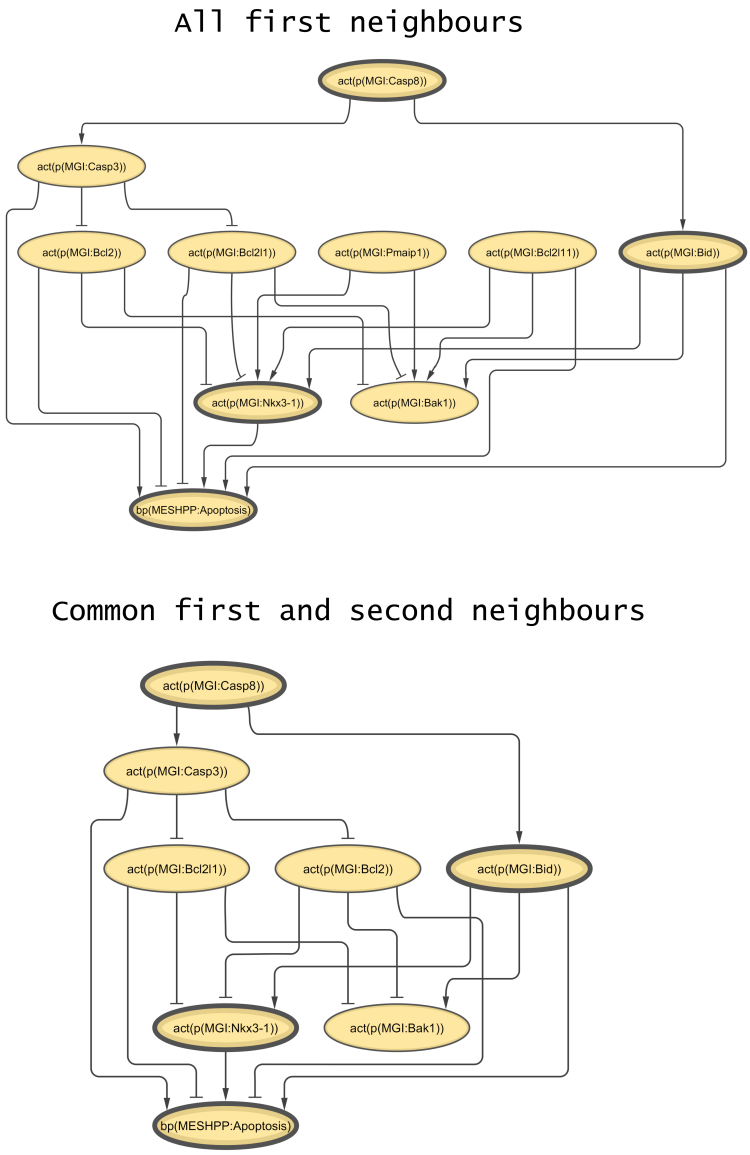

Figure 4.5: Examples of neighbour-induced subnetwork visualisations. Leading nodes are displayed in bold. The top image shows all first neighbours of any leading node. The bottom image shows common first and second neighbours of all leading nodes. In other words, this selects nodes that are at most two steps away from every leading node.

Table 4.1

Identified limitation	**Implemented solution**
Limited selection of networks	Support for network import from various sources
Limited syntax and weights	Support for custom syntax and weight mapping
Inability to modify networks	Modification of individual edges and random rewiring
Increased memory complexity due to dense matrices	Sparse mode of computation
Inability to detect causally opposing edges	Optional pruning of causally opposing edges
Fixed method of edge pruning	Several edge pruning variants
Fixed scope of permutation	Optional permutation rate
Random network permutations	A permutation variant respecting the degree sequence
Core permutations limited to sampling core coefficients	Support for partial and full core permutations
Lack of contextual metadata	Automatic inclusion of metadata in computed results to ensure FAIR-ness
Limited visualisation capabilities	Cytoscape integration
Lack of subnetwork visualisation	Visualisation of path- and/or neighbour-induced subnetworks

Chapter 5

Network generation and analysis

While the network structure required by the TopoNPA algorithm is flexible, the networks are typically constructed manually. This process can be arduous, involving a detailed review of both literature and existing biological networks within the relevant context. They are then verified using transcriptomic data, validating the predictive power of the NPA score, as well as the relevance of identified leading nodes. While the R implementation of the TopoNPA algorithm provides 17 such networks through the companion NPAModels package, this is a rather small number of samples for a general algorithm evaluation.

Due to this limitation, we investigated the possibility of artificially increasing the number of available networks, including the generation of larger and computationally more expensive networks. In this chapter, we first look at the properties of networks in the NPA R package. We then describe the process of generating synthetic networks using the Barabási-Albert model [50], and assess the similarity of generated networks to the NPA R networks. Finally, we describe a process of generating synthetic datasets that induce a perturbation in any type of network. The generated networks and datasets were used to supplement our evaluation of algorithm performance.

5.1 Network analysis

In order to analyse the properties of the NPA R networks, 14 statistical features were measured in each network. These can be split into primary features, which can be controlled directly, and secondary features. Primary features include core node and edge count, boundary node and edge count, inner boundary node count, and negative edge ratios in both the network core and boundary. Boundary nodes refer specifically to outer boundary nodes, while inner boundary nodes are core nodes with at least one boundary edge.

Secondary features include radius, diameter, shortest path length, transitivity, average clustering, and degree assortativity coefficient of the core network, as well as the degree assortativity coefficient of the network boundary. As the network boundary is a directed bipartite graph, measures of network distance, transitivity and average clustering are not informative. The correlation between network features can be seen in Figure 5.1.

Because of the high correlation between features, we decided to use agglomerative clustering. First, min-max scaling was used to limit the range of all features to $[0, 1]$. Rescaled features were then agglomerated with ward linkage, partitioning them into six clusters. The resulting dendrogram can be seen in Figure 5.1. The features in each cluster were combined using average pooling, and PCA was then used to determine the effective dimensionality of this feature space. PCA results are depicted in Figure 5.2.

Agglomerative clustering with average linkage was then run on the networks, using both agglomerated features and the first four principal components obtained with PCA. It was found that further dimensionality reduction using PCA did not have any significant effect on clustering. Consequently, agglomeration was chosen as the sole dimensionality reduction tool to maintain interpretability. Based on the dendrogram depicted in Figure 5.3, the networks were then split into four clusters.

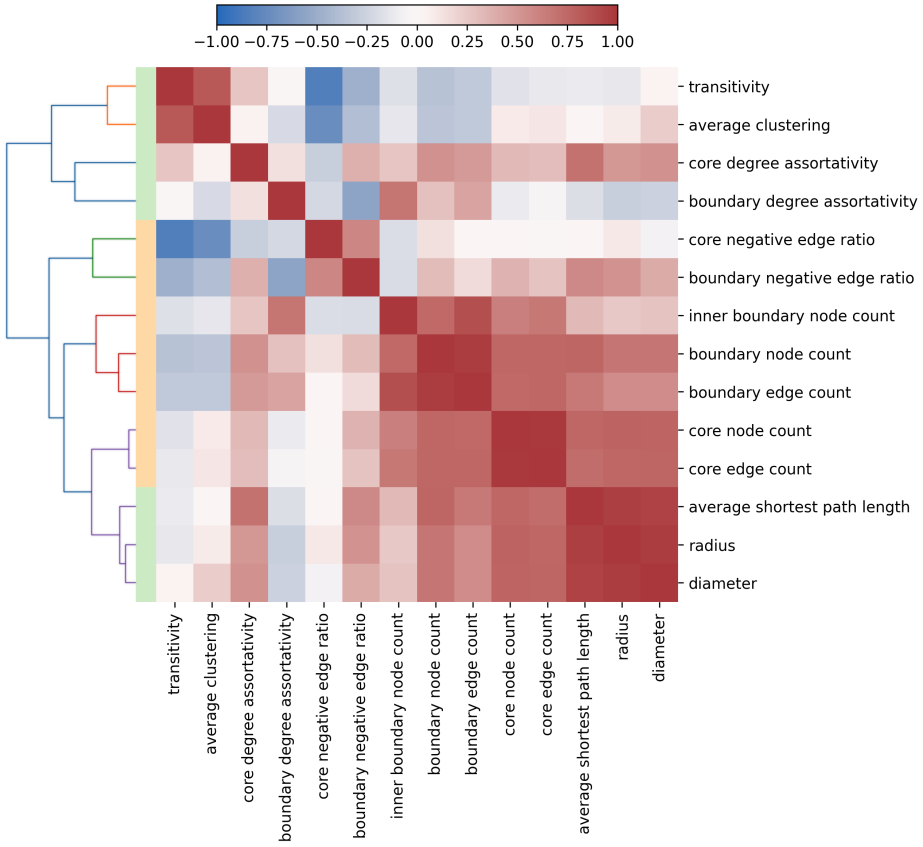

Figure 5.1: A correlation heatmap combined with a dendrogram depicting the agglomeration of scaled network features using ward linkage. Primary features are labelled orange, while secondary features are labelled green.

There is a high similarity between the networks that describe the same process in mice (*Mus musculus*) and humans (*Homo sapiens*). The zebrafish (*Danio rerio*) cardiotoxicity network is the sole exception and forms its own cluster, group D. The largest cluster, designated as group A, is formed by networks describing neutrophil signalling, epithelial innate immune activation, apoptosis and oxidative stress response. The remaining two clusters describe two processes each. Group B contains networks describing extracellular matrix degradation and the cell cycle, while group C contains networks describing JAK/STAT signalling and xenobiotic metabolism response.

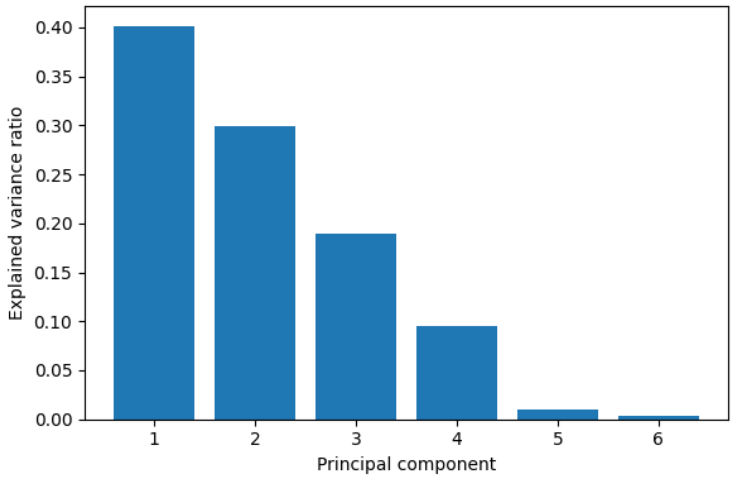

Figure 5.2: Histogram depicting the variance explained by each principal component of agglomerated network features. The feature space consists of four dimensions with significant variance.

5.2 Network generation

All primary network features had to be specified in order to generate synthetic networks. Since they are correlated, we decided to fix the number of core nodes and sample the rest using a two-step approach. First, a linear regression was computed between each parameter and the core node count. A multivariate normal distribution was then fit on the residuals. The correlation between the residuals can be seen in Figure 5.4. Parameters were generated by combining the prediction of the linear regression for a given core node count with a sample from the residual normal distribution.

Network generation was split into two phases. The first phase generated the network core for a given number of core nodes, edges, and negative edges. As degree distributions were observed to follow a power law, the Barabási-Albert model was selected. Edges were iteratively added to an initial star graph based on the degree sequence. Each edge was randomly assigned a weight of 1 or -1 according to the remaining number of negative edges.

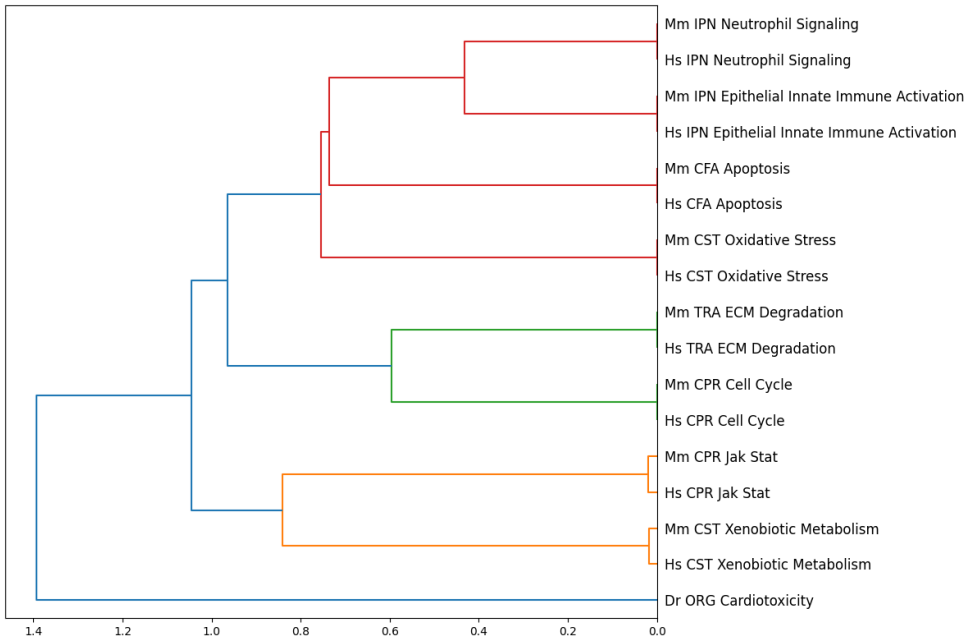

Figure 5.3: Dendrogram depicting the agglomeration of NPA R networks using average linkage and agglomerated features.

The generation of the boundary part of the network was slightly more complex. While the indegree distributions follow a power law, the outdegree distributions do not. We believe that this is partly due to the minimum number of boundary edges that are required per core node so that their weights remain reasonably low. Consequently, we decided to first create a combined outdegree distribution of all NPA R networks.

The network boundary was generated for a given number of inner and outer boundary nodes, edges and negative edges. First, a subset of core nodes was selected as the inner boundary nodes. For each node, an outdegree was sampled with replacement from the outdegree distribution and scaled to match the specified number of edges. Then, each outer boundary node was assigned one random edge to ensure connectivity. The remaining edges were assigned targets based on the degree distribution of outer boundary nodes. Edge signs were randomised in the same way as in core network generation.

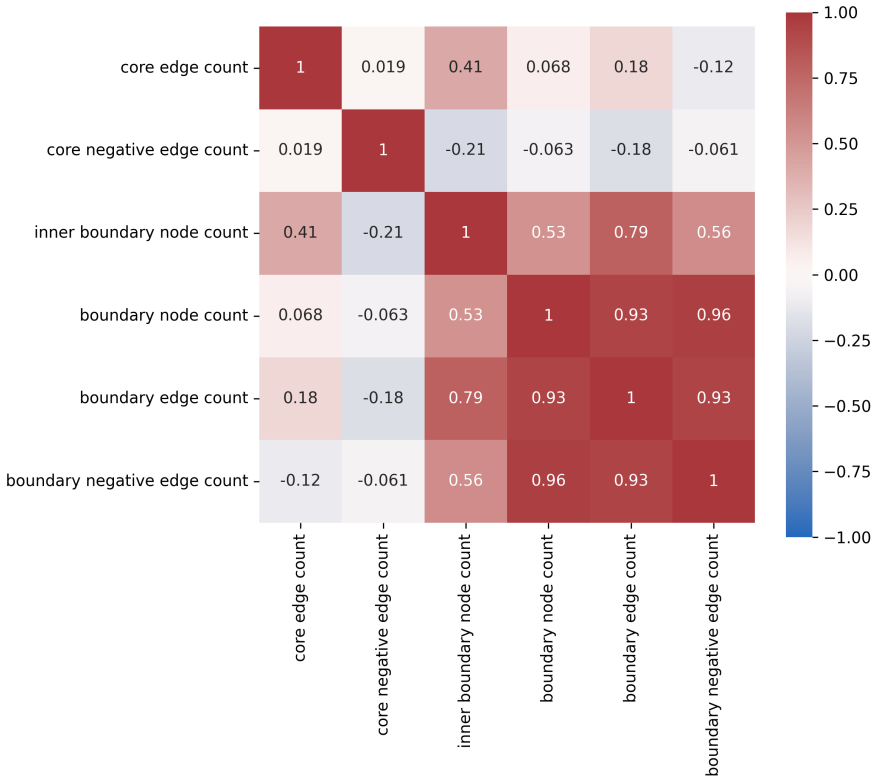

Figure 5.4: A correlation heatmap of the residual distributions of network generation parameters, after regressing out the core node count.

Generated networks were compared to the different groups of NPA R networks using non-metric multidimensional scaling (MDS), which can be seen in Figure 5.5. As the number of core nodes was fixed for each generated network, this parameter was regressed out of the remaining 13 network features using linear regression. The MDS algorithm was then run on the residual distributions. Additional comparisons of network properties and degree distributions can be found in Appendix A.

To account for the differences between the four NPA network clusters, we considered generating cluster-specific networks. Because the clusters are small, the process could be based on distances to cluster centroids. This distance could be used to assign network weights while sampling parameters.

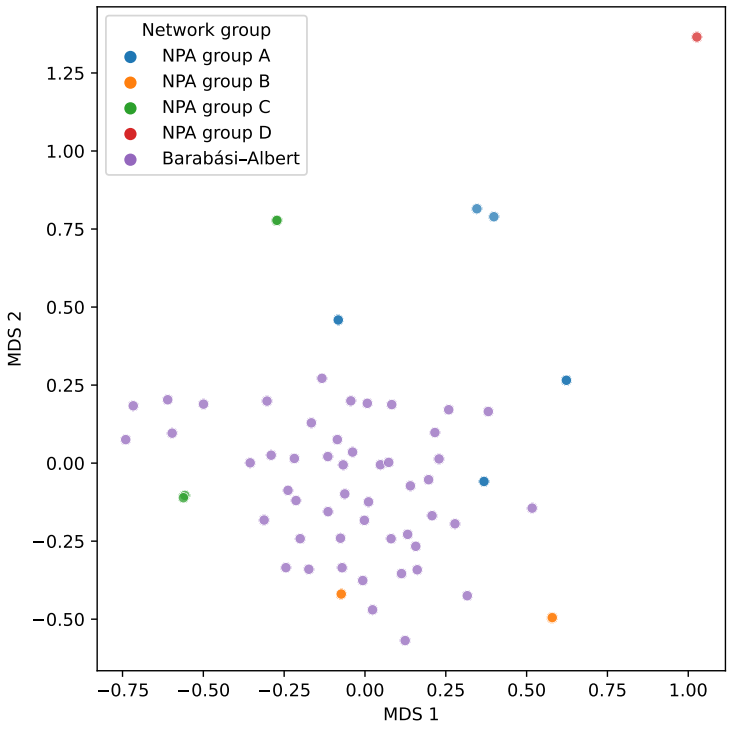

Figure 5.5: Distances between networks visualised using multidimensional scaling (MDS).

Additionally, boundary outdegree distributions could be pooled within clusters. However, we decided against it, as the general generation procedure produced networks sufficiently similar to the NPA networks for evaluation.

5.3 Dataset generation

While the low number of network samples was challenging, the lack of relevant input data was a bigger obstacle. For NPA networks, one would need to obtain fold change measurements with relatively complete coverage of boundary nodes, which induce a perturbation in the network. For Barabási-Albert networks, such datasets cannot be obtained. Instead, synthetic datasets inducing a perturbation were generated using a genetic algorithm.

The gene expression change measurements used to generate samples were not taken from COPD1 data, in order to reduce the dependence of evaluation on these datasets. Instead, data was sourced from the Gene Expression Omnibus (GEO) database under accession codes GSE205434, GSE218120, GSE158763 [29], GSE216395 [30], and GSE225001 [31]. Gene expression fold change measurements were then pooled together and sampled with replacement.

Data sets were generated using a genetic algorithm. Each network was first preprocessed to obtain the relevant Laplacian matrices. Then an initial population of datasets were created at random from the combined pool of gene expression measurements. The algorithm was run for a fixed number of iterations, and at every iteration, the population was scored using the TopoNPA algorithm. Crossover parents were selected preferentially based on their NPA score, and the next population was generated using uniform crossover and mutation. The best-scoring network was stored throughout and returned as the final result.

The genetic algorithm parameters were fixed at 2000 iterations, a population size of 500, and a mutation rate of 0.002. Advanced hyper-parameter optimisation was not performed, as this parametrisation was already able to generate datasets that induce a significantly higher perturbation than what is observed for the COPD1 datasets. However, it was observed that increasing the population size typically had a greater effect than increasing the number of iterations.

Another aspect of the generated datasets was the degree to which they conform to the directionality of boundary edges. To evaluate the effect of opposing value edge pruning, datasets with varying degrees of edge consistency were required. For this purpose, the genetic algorithm was expanded to account for the number of positively and negatively signed edges at every boundary node. Depending on the setting, only positive or negative values were sampled for specified boundary nodes during the initial dataset generation and the mutation process.

We can define the edge consistency of a given dataset and network pair as the percentage of edges that do not oppose the sign of boundary coefficients. The highest possible edge consistency for NPA R networks ranges between 79.41% and 93.36%, while for the selected Barabási-Albert networks it ranges between 76.26% and 82.47%. The lowest possible edge consistency can be computed as the remaining percentage. Examples of dataset generation for the *Mus musculus* apoptosis network are shown in Figure 5.6. Different edge consistency types attain similar perturbations, which are significantly higher than the maximum NPA score of 0.1026 observed using COPD1 data.

Different runs of dataset generation for the same network tend to produce datasets with similar perturbation scores, regardless of the selected edge consistency. On the other hand, datasets generated for different networks displayed a large variance in induced perturbation. We circumvented this issue by evaluating the algorithm on a per-network basis, normalizing the results accordingly. Generally, the threshold for what is considered a significant perturbation will depend on the input network.

Because different datasets for the same network produced similar scores, we also investigated if there was any collinearity between the coefficients of individual boundary nodes. In Figure 5.7, we can see correlation plots for input values of three COPD1 datasets with the highest induced perturbation, as well as three datasets with random edge consistency generated for the *Mus musculus* apoptosis network. While there is a notable correlation between the log fold change measurements in the COPD1 datasets, such a correlation is not observed between the generated datasets. Therefore, it is possible to induce perturbation in a network in several different data configurations, although only a few may be biologically plausible.

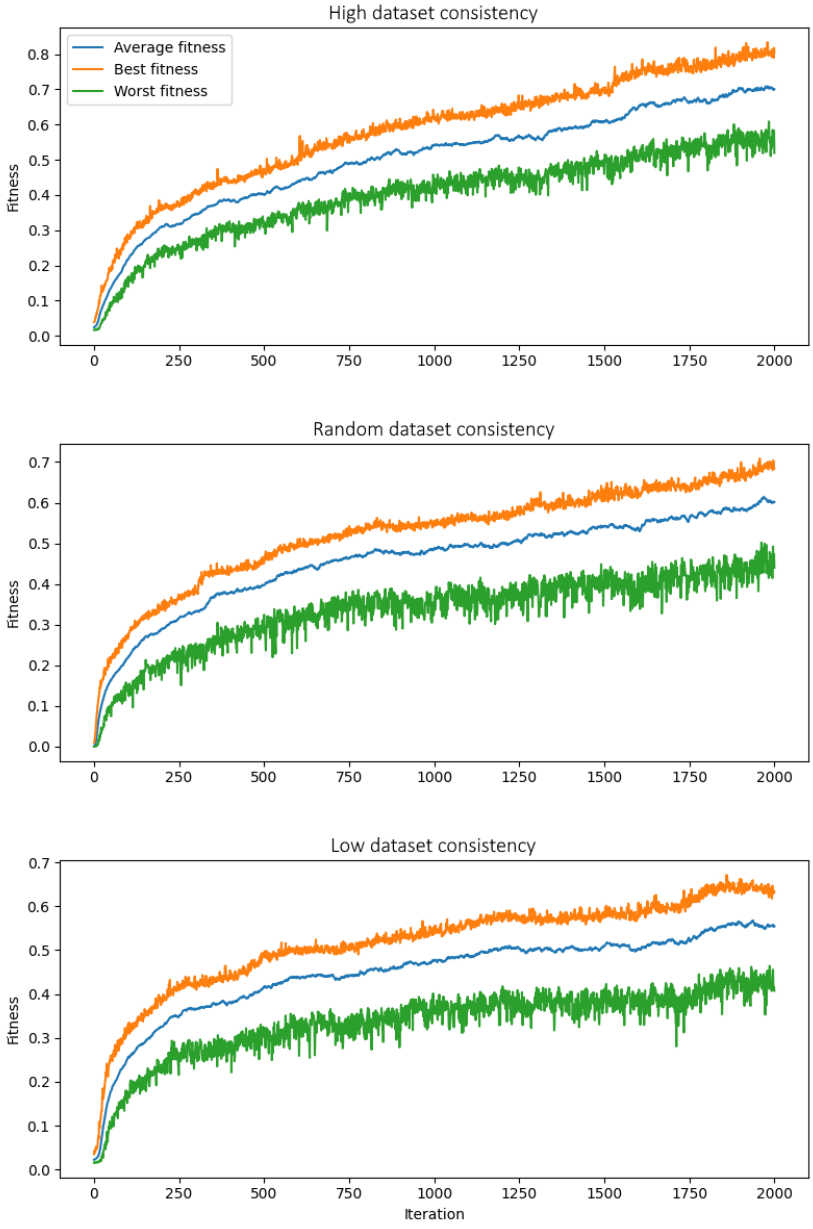

Figure 5.6: Examples of dataset generation for the *Mus musculus* apoptosis network using a population size of 500 and a mutation rate of 0.002. Dataset fitness was evaluated using the TopoNPA algorithm without any edge pruning.

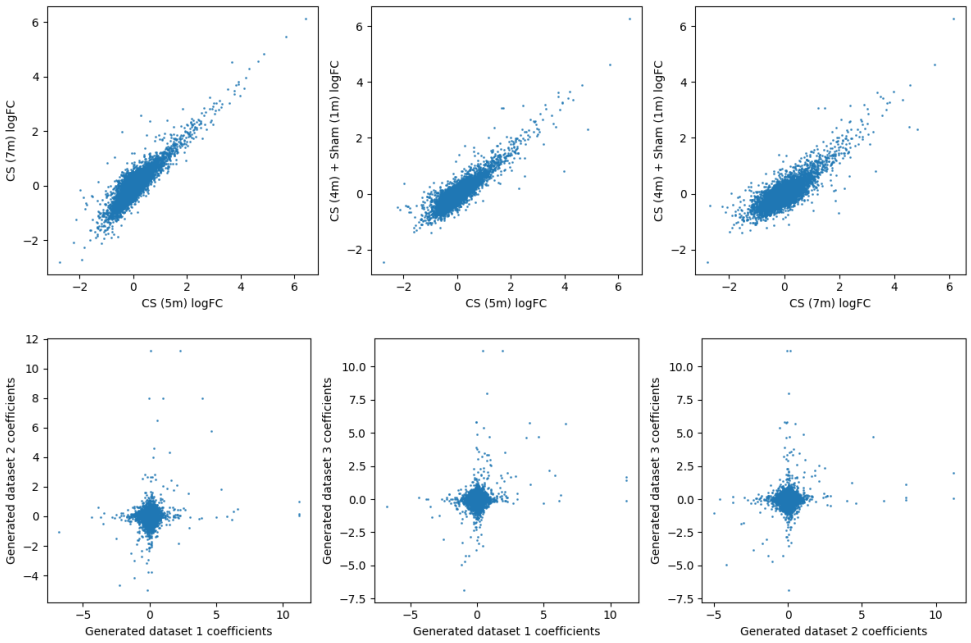

Figure 5.7: Correlation plot of boundary coefficients for different *Mus musculus* apoptosis network datasets. The top row displays correlations between three COPD1 datasets with the highest induced perturbation, while the bottom row displays correlations between three generated datasets with random edge consistency.

Chapter 6

Algorithm performance and evaluation

In this chapter, we analyse the performance of the implemented algorithm and present our evaluation results. The evaluation was performed using three different kinds of input data. First, the COPD1 data, which is available as part of the NPA R package, was used along with the *Mus musculus* Apoptosis network. Next, synthetic datasets were used with sixteen of the networks available in the NPAModels R package, which is a companion package to the R implementation of the TopoNPA algorithm. Finally, synthetic datasets were used with a subset of fifteen synthetic networks generated with the Barabási-Albert model [50], ranging in size from 100 to 500 core nodes.

Among all the networks that are available in the NPAModels R package, only the *Homo sapiens* Xenobiotic Metabolism network was excluded. This is due to the fact that our implementation requires core and boundary nodes to have distinct names, as the same node cannot appear in both parts of the network. The *Homo sapiens* network appears to use fully capitalised names for both genes and proteins, thus breaking the rule of distinct naming conventions. As this problem did not occur in any of the remaining NPA networks, we decided against attempting to adjust the node names.

The COPD1 data contains six distinct sets of measurements. Each set of measurements covers the same subset of boundary nodes, representing 91.203% of all boundary nodes in the *Mus musculus* Apoptosis network. For evaluations performed using NPA networks, 15 synthetic sets of data were used per network. For networks generated using the Barabási-Albert model, 9 synthetic sets of data were used per network. In the latter two cases, a third of the datasets was generated randomly, while a third was generated with high edge consistency and a third was generated with low edge consistency. Unless otherwise specified, results were aggregated across all datasets for every given network.

In the following sections, we cover different aspects of the algorithm's performance. First, we describe the results of our sensitivity analysis, evaluating the different types of missing value pruning. We also assess the general performance with noisy input data and changes to the core network's structure. Next, we evaluate the different types of strict edge pruning when given data with opposing causality. We then move on to estimating the effect of all implemented permutation variants on the NPA score. Finally, we discuss the scalability of the algorithm in terms of time and memory complexity.

6.1 Sensitivity analysis

In this section, we analyse the algorithm's sensitivity to missing data and noise. We investigate the effect of missing and noisy data, as well as changes to core network edges, on two of the TopoNPA results. The first result is the NPA score, where we measured relative absolute error. The second are the leading nodes, where we measured their recall and precision. Leading nodes were defined in the same way as in the original TopoNPA pipeline. Edge pruning for causally opposing data was not used, and a global minimum boundary outdegree of 6 was used. This matched the minimum number of boundary edges observed across the NPA networks.

6.1.1 Missing data

When pruning edges of missing data, the strategies of removal, nullification with exact boundary outdegrees, and nullification with binary boundary outdegrees were used. We refer to the last strategy as only nullification for brevity. It is the same strategy that was used in the R NPA package. Results were plotted for each type of input data separately, to highlight the differences between the three strategies.

For analysis using COPD1 data, reference results were set at 8.797% missing data and no added noise. This is due to the dataset itself not having measurements for all boundary nodes of the *Mus musculus* Apoptosis network. For analysis using all NPA networks and synthetic networks, reference results were set at no missing data and no added noise. The ratio of missing values varied between 2.5% and 25% for evaluation using synthetic data, and between 10% and 30% for evaluation using the COPD1 data. At every set ratio of missing values, 10 reduced datasets were randomly generated per base dataset. The results can be seen in Figure 6.1.

Among the three strategies of missing value pruning, nullification with exact boundary outdegrees performed the best at maintaining the NPA score when increasing the number of missing values. The strategy of removal with rebalancing was second, while the strategy of nullification used in the NPA R package performed the worst. When looking at the precision and recall of leading nodes, the strategy of removal performed the worst throughout. Nullification with exact boundary outdegrees exhibited better leading node recall, while nullification with binary boundary outdegrees exhibited better leading node precision. This suggests that the use of exact outdegrees tends to increase the number of leading nodes among results.

Importantly, the decrease in accuracy appears to occur on a much larger scale when using real measurements from the COPD1 data. Even the best edge pruning strategy produced a relative absolute error of upwards of 50% when increasing the number of missing values by 10%. This is in contrast to the relative absolute error of 10% observed on the same scale with synthetic

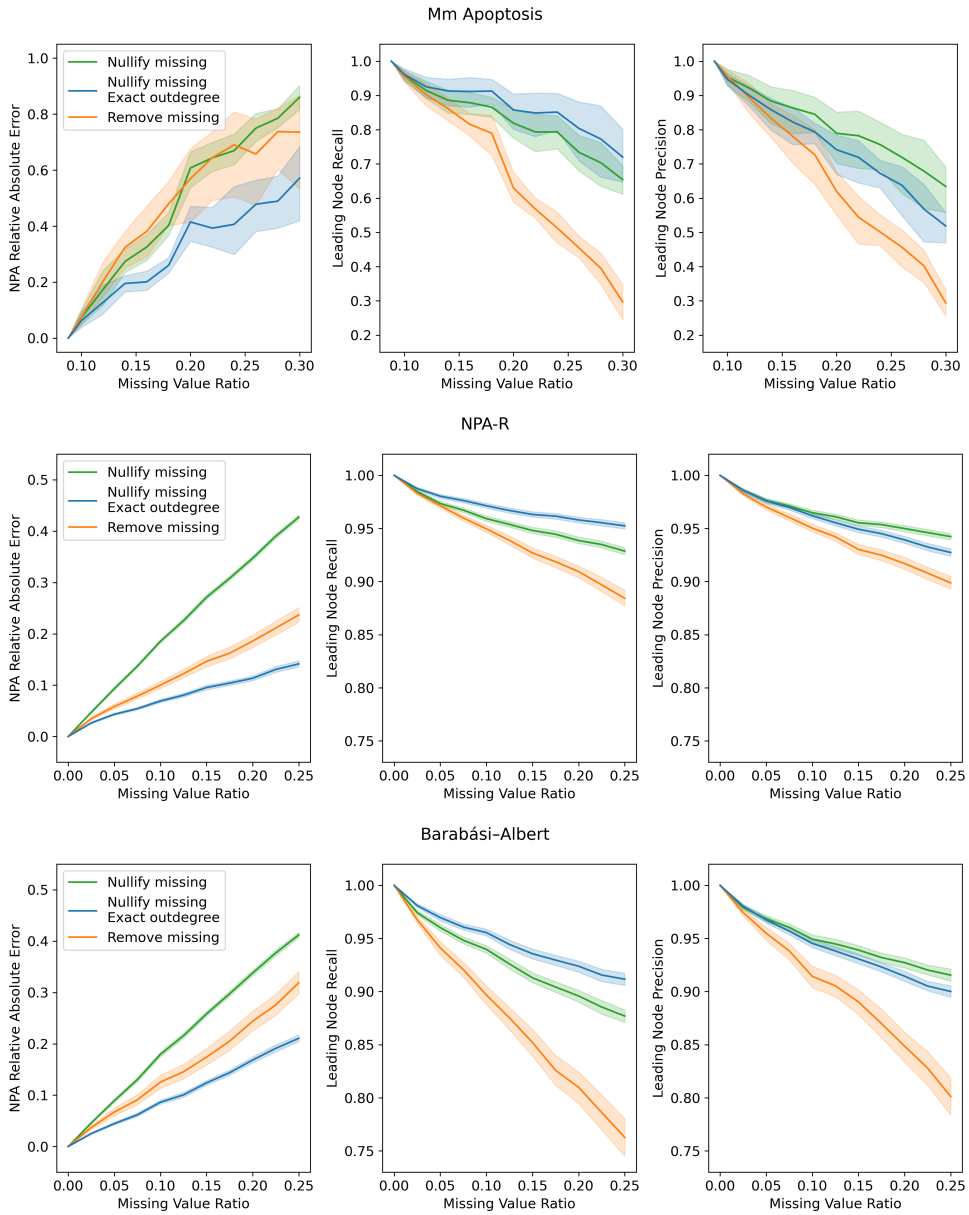

Figure 6.1: Sensitivity analysis of the TopoNPA variants to missing values. The first row of results was generated using one NPAModels network and real gene expression measurements. The second and third rows of results were generated using synthetic datasets that induced perturbation in NPAModels networks and Barabási-Albert networks, respectively.

datasets, regardless of the networks used. Similarly, leading node precision and recall dropped by upwards of 30% using real measurements, compared to a drop of less than 10% when using generated data.

Overall, we conclude that both strategies of nullification perform similarly when looking at leading node precision and recall. Due to the improved performance with exact boundary outdegrees when looking at the error in NPA scores, we decided to use this parametrisation as the default setting in our algorithm's implementation. However, care should still be taken when analysing datasets with a large amount of missing data, as it is likely that the decrease in accuracy will be closer to what was observed with the COPD1 data than the synthetic datasets.

6.1.2 Noisy data

When performing sensitivity analysis for noisy data, noise was added according to a pre-specified signal-to-noise ratio (SNR). The SNR was varied between 25 and 0, and at every set SNR, 10 noisy datasets were generated per base dataset. As the amount of missing data does not vary, the results for all three different kinds of input data were plotted together. A baseline strategy of nullification with exact boundary outdegrees for pruning edges of missing values was used throughout. The results can be seen in Figure 6.2.

Once again, we can see that the effect of noise on observed error was significantly larger when using real measurements rather than synthetic data. Interestingly, the combination of synthetic data with Barabási-Albert networks was less sensitive to noise than the combination of synthetic data and NPA networks. Overall, we can conclude that the algorithm is less sensitive to noise than to missing values. Even at an SNR of 0, where added noise has a similar strength to the underlying signal, the error in NPA score does not exceed 30%, and both leading node recall and precision remain above 85% on average.

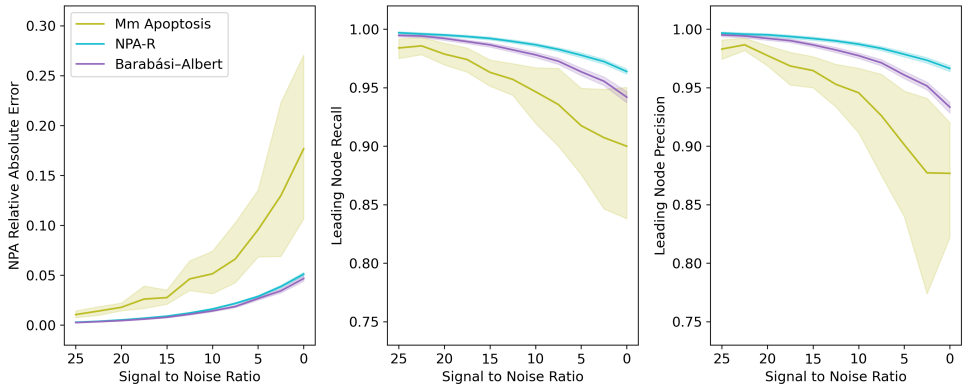

Figure 6.2: Sensitivity analysis of the TopoNPA algorithm to noisy data. As the choice of missing value pruning does not affect response to noise, results are plotted together for all the different kinds of input data.

6.1.3 Shuffled edges

Changes to the core network were performed by randomly shuffling edges. Permutation rate was varied between 1% and 10%, and 10 permutations were performed per given rate. Each permutation randomly selected a percentage of core edges equal to the permutation rate and reassigned them to random nodes. Results were computed for each permuted network structure similarly to the full "K1" permutation test. The pruning strategy of nullification with exact boundary outdegrees was used, although the amount of missing data was not varied. Results can be seen in Figure 6.3.

The effect of edge shuffling induced the greatest error in the NPA score and leading node recall when using real measurements. On the other hand, leading node precision was worsened the most when using synthetic networks and datasets. Overall, the algorithm is more sensitive to changes in the network core than input data, which is expected. However, even when reassigning 10% of the core network edges, up to 80% of the leading nodes can still be reliably recovered. Therefore, the algorithm does exhibit some robustness to mistakes in the core network structure.

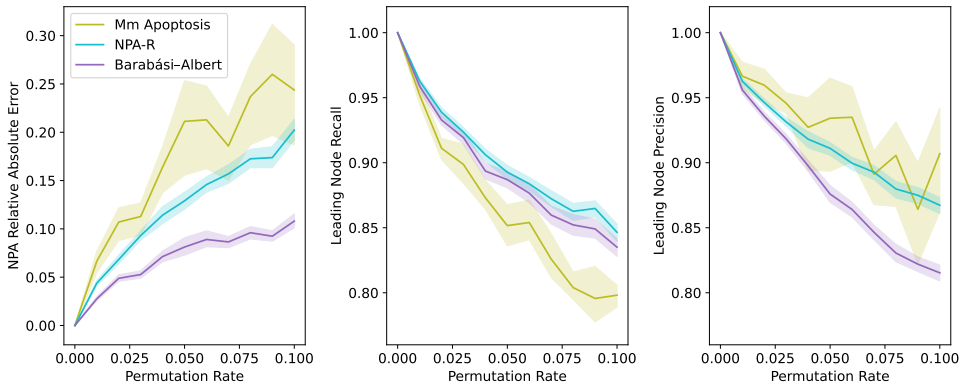

Figure 6.3: Sensitivity analysis of the TopoNPA algorithm to edge modifications. Results for different kinds of input data are plotted together.

6.2 Opposing edge pruning

Evaluation of edge pruning for causally opposing values was performed in relation to the causal consistency of datasets. Because the causal consistency of the COPD1 dataset cannot be controlled, it was excluded from this evaluation. Generated datasets for NPA networks and Barabási-Albert networks were categorised by causal consistency for separate analyses. A minimum boundary outdegree of 6 was used throughout while missing value pruning was irrelevant due to the lack of missing values in generated data.

We were interested in the effect that different edge removal strategies have on both the NPA score and leading nodes. For each network, we set the baseline NPA score as the average of NPA scores obtained without opposing edge pruning across all datasets. Computed NPA scores were then displayed relative to this baseline, aggregating them across networks. The baseline set of leading nodes was defined per network and dataset as those determined without the use of opposing edge pruning, in the same way as in the original TopoNPA pipeline. Recall and precision were then computed for every pruning strategy and aggregated across datasets and networks.

As in the analysis of sensitivity to missing values, three strategies of edge pruning for opposing data were investigated. These are edge removal with weight rebalancing, edge nullification with binary boundary outdegrees, and edge nullification with exact boundary outdegrees. The minimum amplitude for a measurement to be considered causally opposing was set to 0 so that all edges that did not match the measurement were removed. Results were displayed using boxplots and can be seen in Figure 6.4.

When looking at NPA scores, we can see that for causally inconsistent datasets, all pruning strategies produce a stark drop in the computed NPA score. Out of the pruning strategies investigated, this drop is least severe when using edge removal, although it is still significant. For causally consistent datasets, we can see that there is an overall increase in measured NPA scores. In this case, the strategy of nullification with binary boundary outdegrees produced results most similar to the original distribution.

The recall of leading nodes is high across different types of edge pruning for datasets with high causal consistency. For low causal consistency, the strategy of removal appears to somewhat outperform the others, as in the case of NPA networks, it produced the highest mean recall, while in the case of Barabási-Albert networks, it has the highest minimum and lower quartile. Precision was more varied, with removal producing worse results than the alternatives. In the case of Barabási-Albert networks and causally consistent datasets, nullification with binary boundary outdegrees produced the best results, followed by nullification with exact outdegrees.

If we focus instead on datasets with random edge causality, which is most likely to occur naturally, generated datasets respond on a spectrum between what we observe in causally consistent and causally inconsistent datasets. This suggests that certain measurements have a more significant effect on computed results, and the alterations produced by pruning depend on whether they are affected by the process.

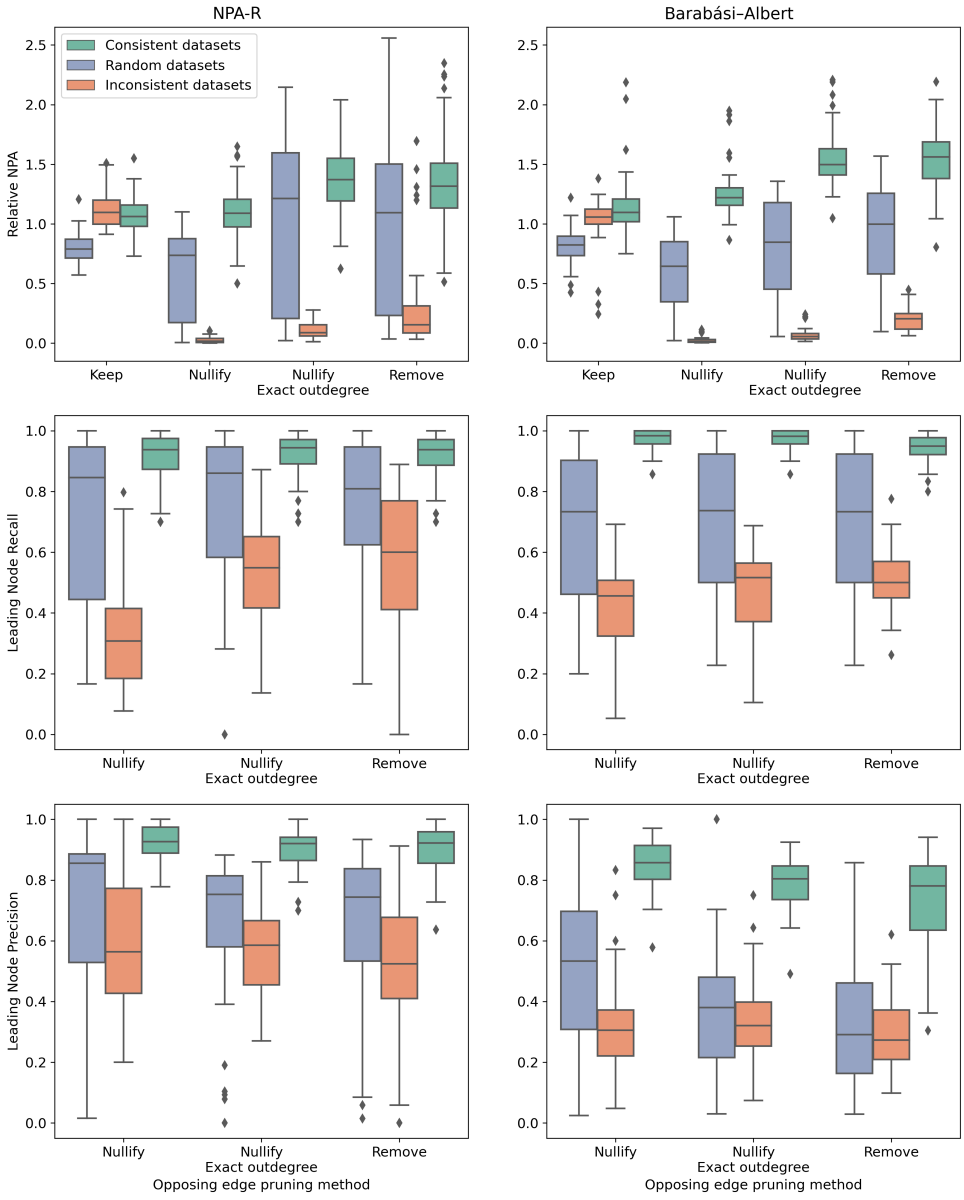

Figure 6.4: Evaluation of opposing edge pruning variants using synthetic datasets. The first column was generated using NPAModels networks, while the second was generated using Barabási-Albert networks.

Overall, we conclude that no pruning strategy is suitable for pruning a large number of causally inconsistent edges. While for our evaluation, we constructed datasets with high and low amounts of causal consistency, we encourage users to limit the number of removed measurements in practice. This can be varied by adjusting the minimum amplitude required for a measurement to be considered significant enough to oppose edge causality. When processing a limited number of causally inconsistent edges, we recommend the strategy of nullification with binary outdegrees, as it performed the best in our estimation.

6.3 Permutation tests

Due to the introduction of several variants of the core permutation test "K", all permutations were evaluated in terms of their effect on the NPA score. A minimum boundary outdegree of 6 was used throughout, along with nullification with exact outdegrees for missing values (which are present at a fixed rate in the COPD1 data). Opposing value edge pruning was not used. We first examine the boundary permutation test "O", which was evaluated similarly to noise sensitivity.

At each specified ratio of shuffled values (i.e. the permutation rate), ten permuted datasets were generated per base dataset. Then the average NPA score was computed relative to the baseline score, and results were aggregated across datasets and networks. As we are interested in this process primarily as a permutation test rather than a robustness test, leading nodes were not further investigated. Obtained results can be seen in Figure 6.5.

For core permutations, we investigate all four possible variants separately. We refer to the random permutation of core edges as "K1" while permuting core edges using the configuration model [48] is referred to as "K2". Each permutation of the adjacency matrix can be used to infer either only the L_c Laplacian matrix or both the L_c and Q Laplacian matrices. Therefore, both the "K1" and "K2" permutations can be performed partially or in full.

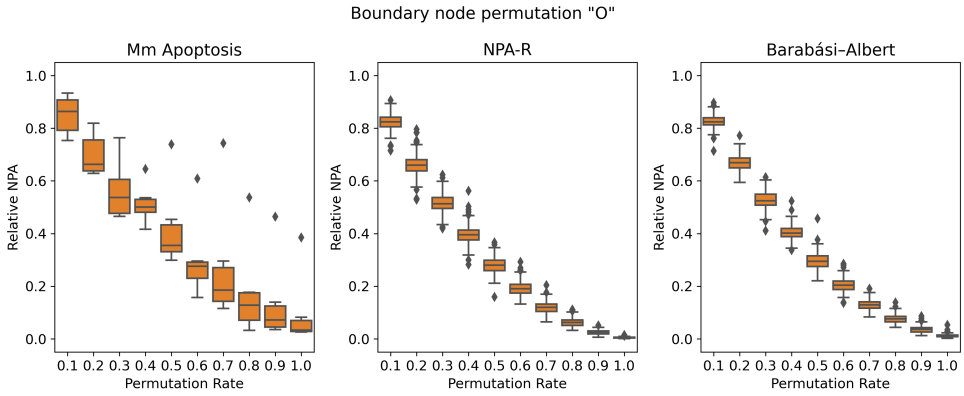

Figure 6.5: Evaluation of boundary permutation "O".

A partial permutation test draws samples of core coefficients and evaluates them using the original network, while a full permutation test draws NPA scores directly. The NPA R package implements the partial "K1" permutation as the canonical "K" permutation, albeit without ensuring connectivity of the permuted networks. Results for the "K1" and "K2" permutations can be seen in Figure 6.6.

The permutation test "O" appears to produce a similar effect regardless of the type of input data. The process of shuffling invariably reduces the computed NPA score. This indicates that the pairing of nodes and data has a significant effect on the estimated perturbation. Amplitude alone cannot be used to quantify a network response.

In the plot for COPD1 data, one set of outliers can be seen to experience a lesser drop in NPA score. These results correspond to the "CS (2m) + Sham (5m)" dataset, which did not induce a significant perturbation in the *Mus musculus* Apoptosis network to begin with. It thus differs from the remaining COPD1 datasets, as well as the generated datasets.

On the other hand, the type of input data is a significant factor in core permutations. Networks generated using the Barabási-Albert model respond differently than those in the NPAModels R package. For the latter, a decrease in NPA score is consistently observed across all permutations.

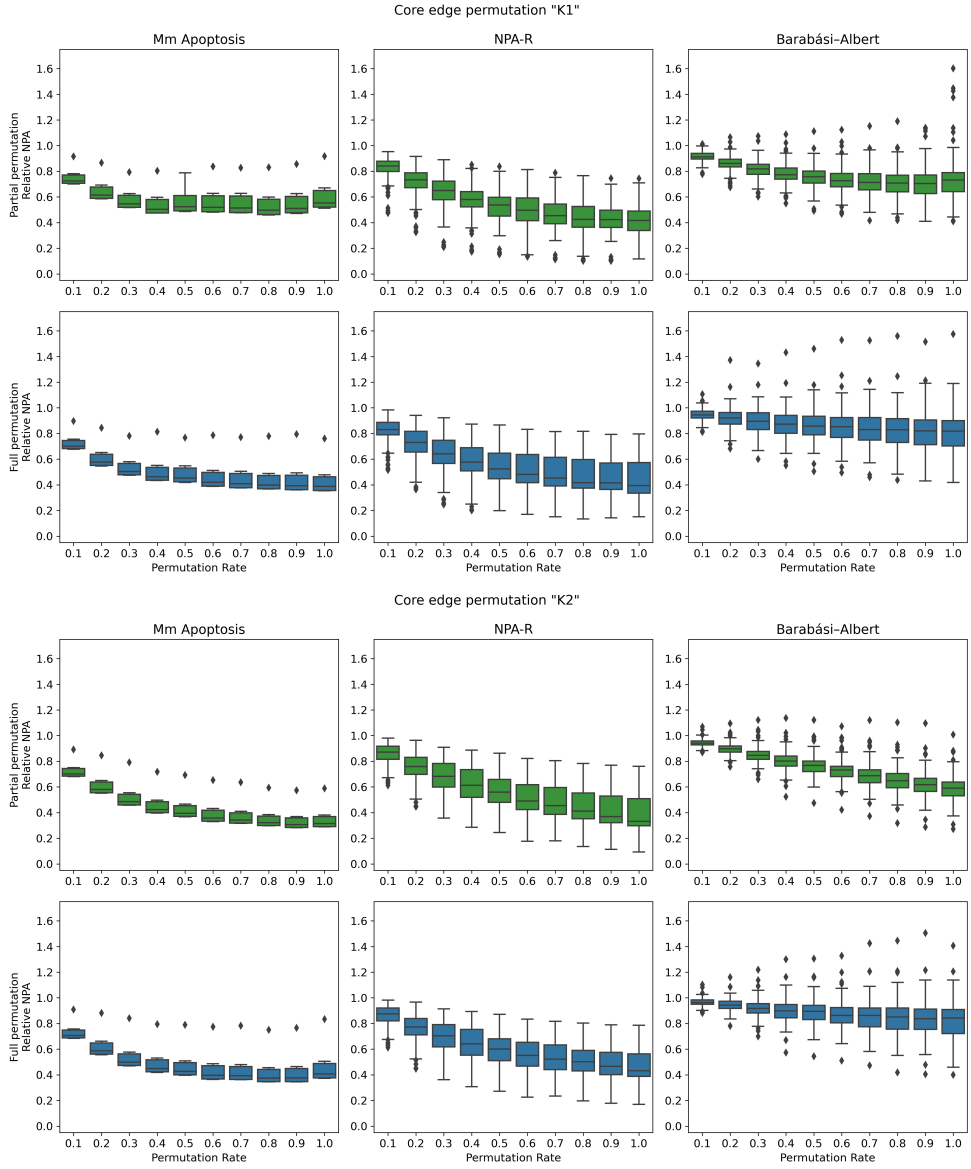

Figure 6.6: Evaluation of core permutations. Permutation "K1" shuffles core edges at random, while permutation "K2" shuffles them according to the original degree sequence.

Permutations of synthetic networks can, however, produce a higher NPA score with significant likelihood. This suggests that there is something specific to the structure of biologically meaningful networks that is not captured by the Barabási-Albert model. Additionally, we can once again observe an outlier in the COPD1 data, representing the dataset with insignificant perturbation.

While the "K2" permutation was implemented to produce network permutations that are closer to the original network structure, and thus more biologically likely, this is not supported by our evaluation results. The average NPA score produced by partial "K2" permutations is noticeably lower than that of partial "K1" permutations. For full permutations, they appear to be comparable. Interestingly, the difference between a partial and full permutation is much more pronounced in the "K2" variant.

Overall, we conclude that both boundary and core permutations are valid methods for evaluating the significance of TopoNPA results. In the case of boundary permutations, we can observe that reassigning measurements to nodes does have a significant effect on the final perturbation. This effect is less pronounced for datasets that do not induce a significant perturbation.

For core permutations, the network structure also affects the NPA score distributions. While "K2" permutations produce edge configurations that respect the original degree distribution, "K1" permutations produce NPA scores that are closer to the original and thus provide a stronger proof of significance. If users still wish to generate permutation tests using the configuration model, we recommend the use of full permutations over partial permutations.

6.4 Scalability

An important consideration for any software that deals with networks is how it scales with network size. For this purpose, we evaluated both the execution time and memory usage of our implementation. Evaluation of scalability was performed on a Lenovo laptop with an AMD Ryzen 7 5800H processor and 16 GB of RAM. Memory usage in Python was measured using the Memory Profiler package [51].

We first compared our implementation of the TopoNPA pipeline to the implementation in the NPA R package. Both pipelines were run on the *Mus musculus* apoptosis network and COPD1 data with the same parameters, computing 500 "O" and partial "K1" permutations each. Loading the network and input data into memory was included in the measurement. Each implementation was run five times, and the lowest achieved runtime and memory usage was kept.

The TopoNPA pipeline in the NPA R package finished in $30.07s$ and required $5604.2MB$ of memory. The large memory usage is due to the package loading all available networks before selecting the requested one. The non-sparse mode of computation in PerturbationX finished in $10.48s$ and required $264.29MB$ of memory. The sparse mode of computation in PerturbationX finished in $16.45s$ and required $195.56MB$ of memory. We conclude that PerturbationX improves on both runtime and memory usage when compared to the NPA R package.

A broader evaluation of scalability was performed on generated networks with the number of core nodes ranging from 500 to 10,000. These networks were generated with deterministic parameters, which were sampled directly using linear regression and were not randomised with samples from the residual distribution. The networks of lower size were evaluated using both sparse and non-sparse modes of computation, while the rest were only evaluated with the sparse mode. The networks were evaluated with randomly sampled gene expression measurements, not optimised to induce perturbation.

The algorithm was run without edge pruning, computing 500 "O" and full "K2" permutations. Loading the network and input data was included in the measurement. Each evaluation was run once. The results can be seen in Figure 6.7. Based on the results we conclude that our implementation easily scales to networks that contain several hundreds of thousands of nodes and edges. While the non-sparse implementation was only 35.99% slower on a network with 500 core nodes, it was 154.76% slower on a network with 1500 core nodes. 16 GB of RAM was no longer sufficient to run non-sparse computation on a network with 2000 core nodes.

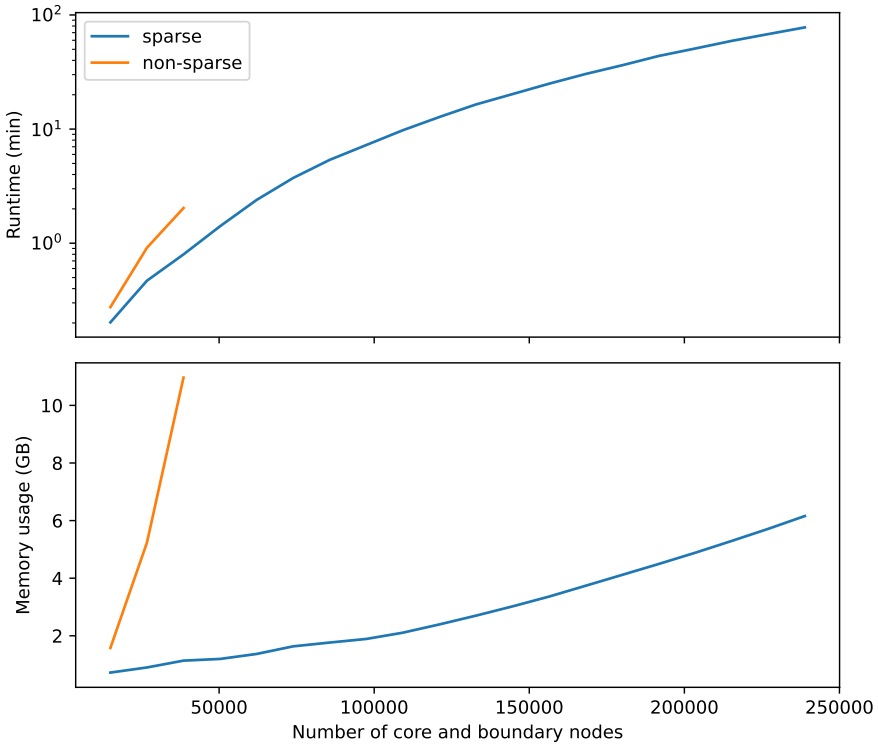

Figure 6.7: Evaluation of runtime and memory usage of the sparse and non-sparse modes of computation. Non-sparse computation was not run on networks with 2000 or more core nodes.

Chapter 7

Conclusions

In this work, we presented causal biological networks (CBNs) as a method of guiding gene expression analysis using prior biological knowledge. We described the development and Python implementation of an extension to the TopoNPA algorithm [2]. The implementation, named PerturbationX [3], was developed in collaboration with the National Institute of Biology.

PerturbationX allows users to import causal networks encoded in any syntax, enabling them to run the algorithm on a wide range of input data. It supports visualisation through Cytoscape [4] integration and extraction of subnetworks based on leading nodes. Results can be exported and are automatically coupled with rich contextual metadata, making it easier to publish findings in accordance with FAIR principles.

Several extensions of the TopoNPA algorithm were implemented as part of the package. Different types of edge pruning were investigated in cases when network edges cannot or should not be included. Pruning strategies were enabled for edges with causality that contradict the directionality of observed changes in datasets. This effectively prevents the algorithm from attributing regulatory changes to biologically improbable regulators.

The implementation of permutation tests was also adjusted. They allow the user to vary the percentage of the network that is permuted, as well as adjust the type of permutation samples generated during the process. Specifically, when permuting edges in the core network, the user can choose to sample either core node coefficients or the final NPA scores from permuted networks.

To evaluate the different modes of computation, we generated synthetic networks and datasets that induce a perturbation for a given network. Networks were generated using linear regression and the Barabási–Albert model. They were shown to be similar to existing CBNs that they were based on. Datasets were generated using a genetic algorithm and were shown to induce a perturbation several times higher than what was observed using experimentally validated gene expression measurements.

Synthetic data enabled us to expand the breadth of evaluation without constructing additional CBNs by hand, as well as evaluate algorithm performance on very large networks. The evaluation was performed under three settings. Firstly, real gene expression measurements were used in combination with the *Mus musculus* Apoptosis network. Secondly, synthetic datasets were used with a selection of CBNs exported from the R NPA package. Finally, synthetic datasets and networks were evaluated together.

When evaluating robustness to missing and noisy data, it was shown that real gene expression datasets were much more sensitive to errors in input, while network type was not a significant factor. On the other hand, different network types produced very different results during the evaluation of core network permutation variants. In general, the algorithm appears to be more robust against data with noise than data with missing values. Removing 10% of input values produced a relative absolute error of upwards of 50% when using real gene expression measurements. The same error did not exceed 30% when combining measurement data with noise of equal amplitude (signal-to-noise ratio of 0).

Sensitivity to missing values was further emphasised during the evaluation of opposing edge pruning, where pruning of a large number of edges produced a significant drop in relative NPA scores. As such, we recommend users to limit the number of edges pruned during the process. This can be done by setting an appropriate threshold for the amplitude of input values that may be considered causally opposing rather than insignificant.

Boundary permutations were shown to perform similarly regardless of the type of network and data being used. As shuffling input values severely decreases the final NPA score, we conclude that this test is a good measure of result significance. On the other hand, core permutations did not necessarily produce a decrease in computed perturbation when used with networks constructed using the Barabási–Albert model. This is partly because the algorithm optimises coefficients for the current network structure. It is unlikely to produce a significantly lower perturbation score when the network boundary remains fixed. Therefore, the test is a better evaluation of network structure than the specific results of a given dataset.

The algorithm was optimised to greatly reduce both memory and computational complexity. While the implementation enables the use of the algorithm for very large networks, further optimisation can be performed through the parallelisation of permutation tests. Another aspect of permutations that can be improved is the network structure generated by core permutations. While networks generated using the configuration model are more biologically plausible than random networks, they still do not exhibit the clustering and community structures that are typically observed in biological networks.

One way to achieve this would be to incorporate the approach presented by Sah et al. [52] to create more plausible network cores. The same process could also be used during the process of synthetic network generation. This generation procedure could then be additionally improved by generating cluster-specific networks. This would induce a more significant variation in the properties of generated networks, further diversifying testing data.

Finally, the package can be easily extended with both existing and novel perturbation scoring algorithms, such as the four NPA metrics presented in 2012 by Martin et al. [8]. In this case, some additional constraints would have to be placed on the import of networks to ensure that their structure conforms to what is expected by the selected algorithm. However, the remaining functionality could be implemented as a simple extension to the existing package structure and user interface. We hope that the PerturbationX package eases both the development and evaluation of novel causal biological networks and empowers researchers to uncover meaningful insights in complex biological systems.

Appendix A

Comparison of NPAModels and generated networks

In this appendix, we compare networks from the NPAModels R package with networks generated using the Barabási-Albert model. Networks exported from R are grouped into four clusters. Group A contains networks describing neutrophil signalling, epithelial innate immune activation, apoptosis and oxidative stress response. Group B contains networks describing extracellular matrix degradation and the cell cycle. Group C contains networks describing JAK/STAT signalling and xenobiotic metabolism response. These three groups contain two networks per biological process, one for mice (*Mus musculus*), and one for humans (*Homo sapiens*). Group D contains a single network describing cardiotoxicity in zebrafish (*Danio rerio*).

Network features were split into four clusters for easier pairwise comparison. Primary core features include core node count, core edge count, and core negative edge count. Primary boundary features include inner boundary node count, (outer) boundary node count, boundary edge count, and boundary negative edge count. Secondary distance features include radius, diameter, and average shortest path length. Secondary local structure features include transitivity, average clustering, core degree assortativity coefficient, and boundary degree assortativity coefficient.

A total of 50 synthetic networks were generated with core node counts ranging from 50 to 500. These are displayed in all figures showing pairwise relationships between features. For figures displaying node degree distributions, only 5 synthetic networks are displayed to improve legibility. These networks have core node counts of 100, 200, 300, 400, and 500.

Primary features were controlled directly through a combination of linear regression and sampling from the residual distribution. The relationships between them are displayed in Figures A.1, A.2, and A.3. A linear regression was fit for relationships between these features, however, only networks in NPA groups were included in the calculation.

The relationships between secondary features are displayed in Figures A.4, A.5, and A.6. The relationships between primary and secondary features are displayed in Figures A.7, A.8, A.9, and A.10. Core degree distributions are displayed in Figure A.11, and boundary degree distributions are displayed in Figure A.12.

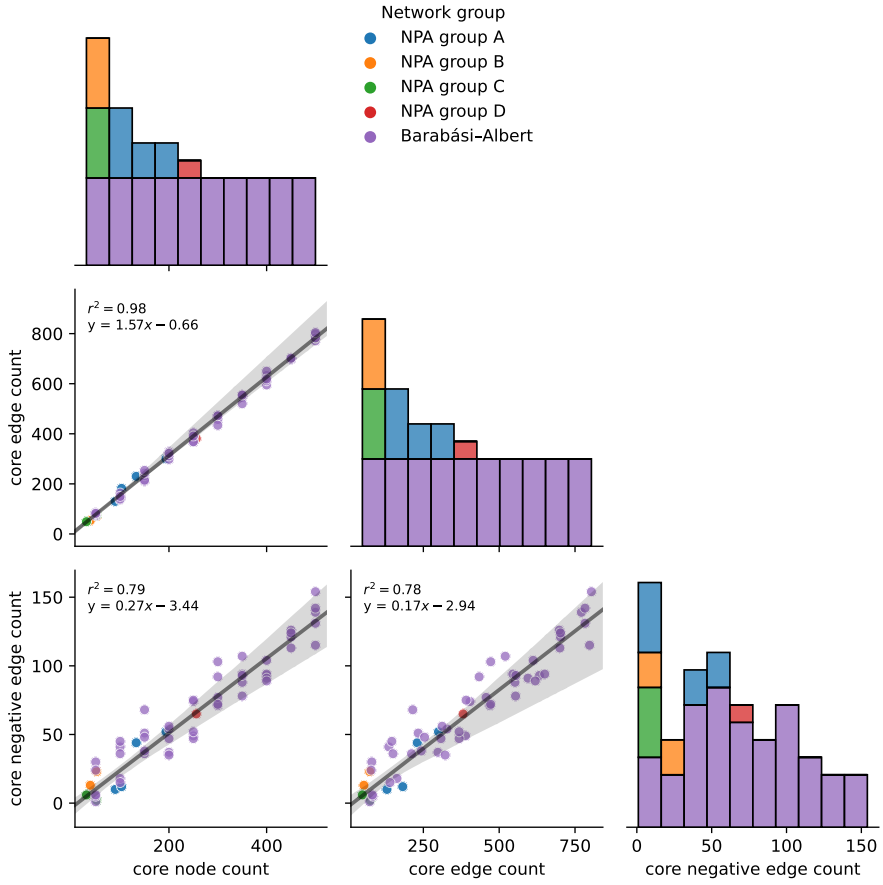

Figure A.1: Pairwise relationships between primary core features. Linear regression was fit for networks in the four NPA groups.

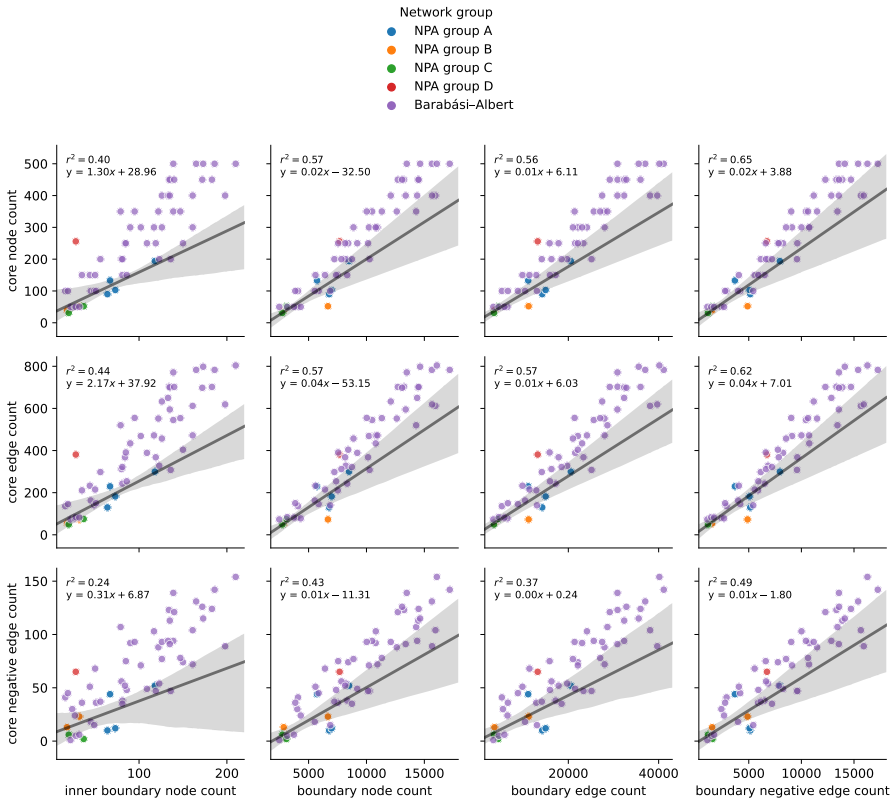

Figure A.2: Pairwise relationships between primary core and primary boundary features. Linear regression was fit for networks in the four NPA groups.

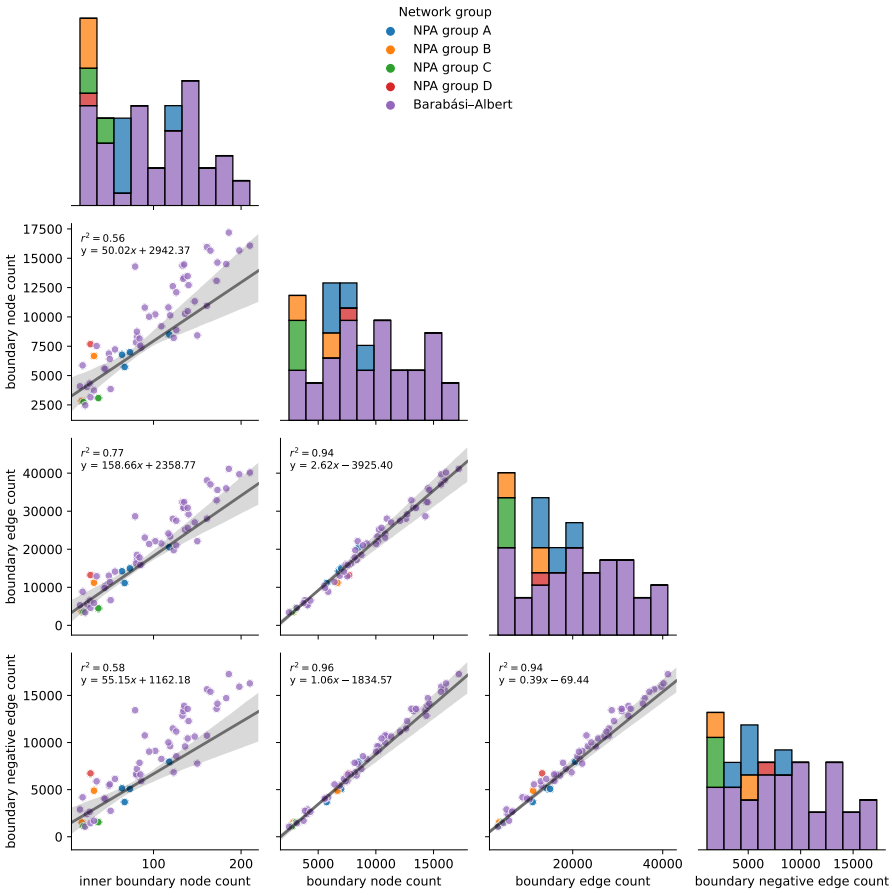

Figure A.3: Pairwise relationships between primary boundary features. Linear regression was fit for networks in the four NPA groups.

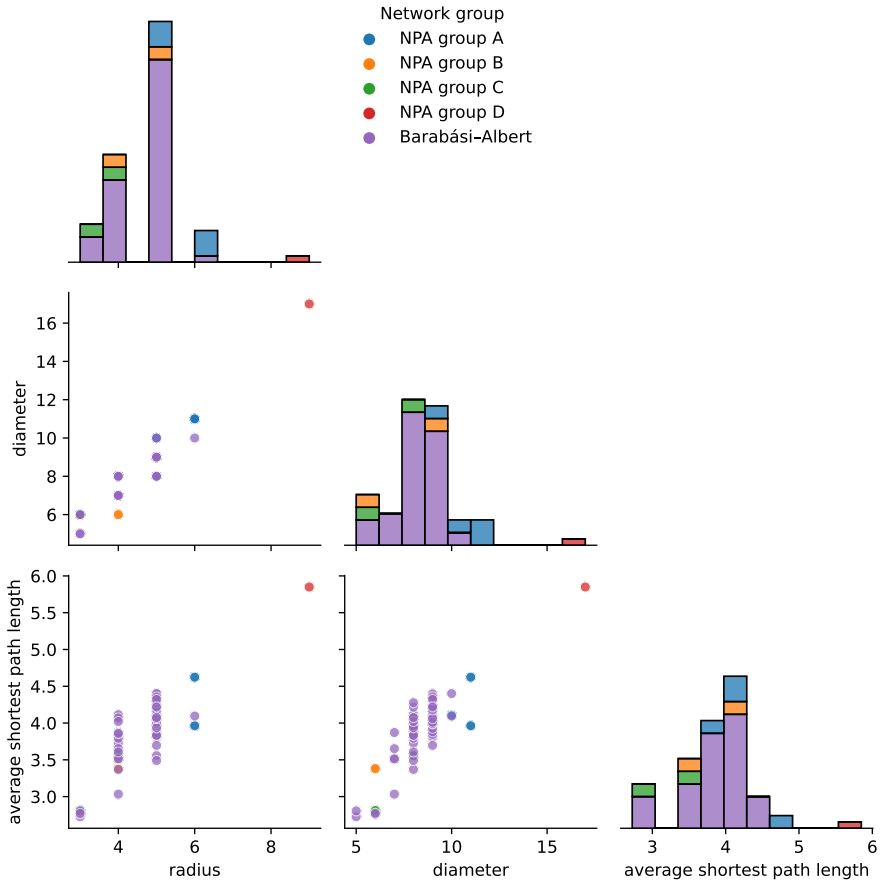

Figure A.4: Pairwise relationships between secondary distance features.

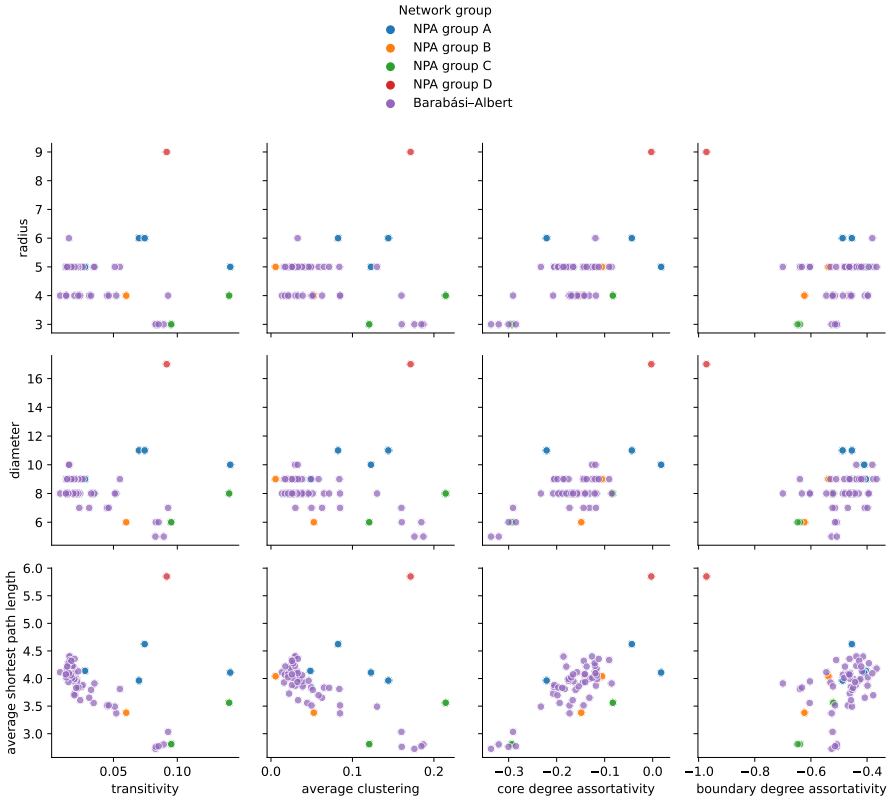

Figure A.5: Pairwise relationships between secondary distance and secondary local structure features.

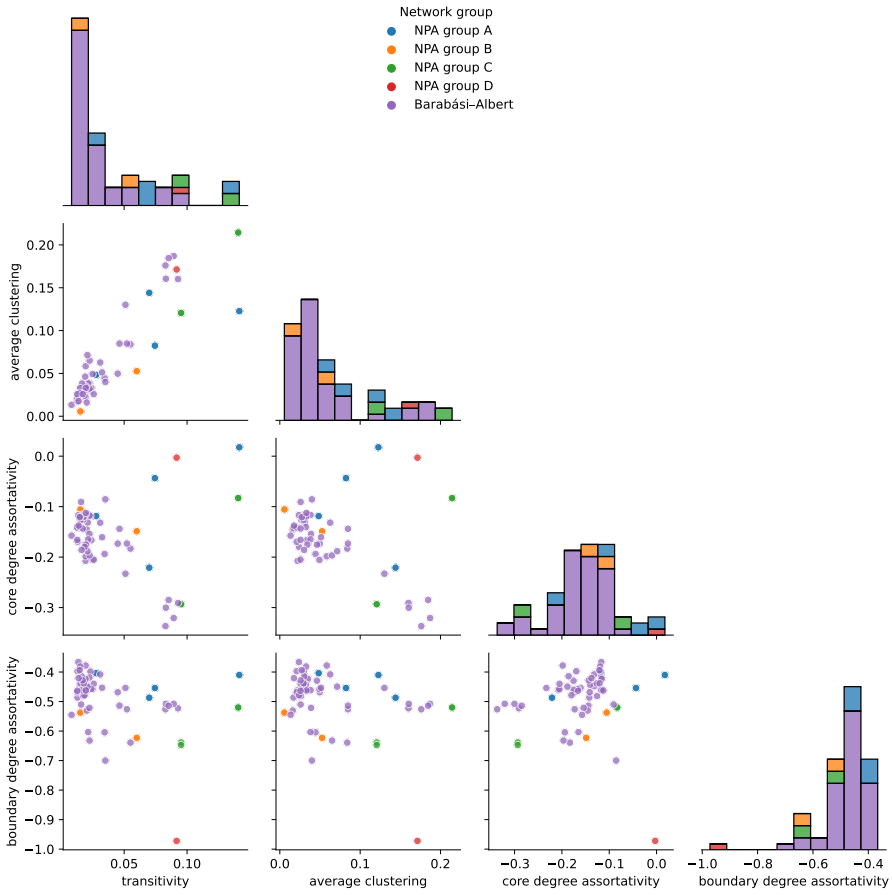

Figure A.6: Pairwise relationships between secondary local structure features.

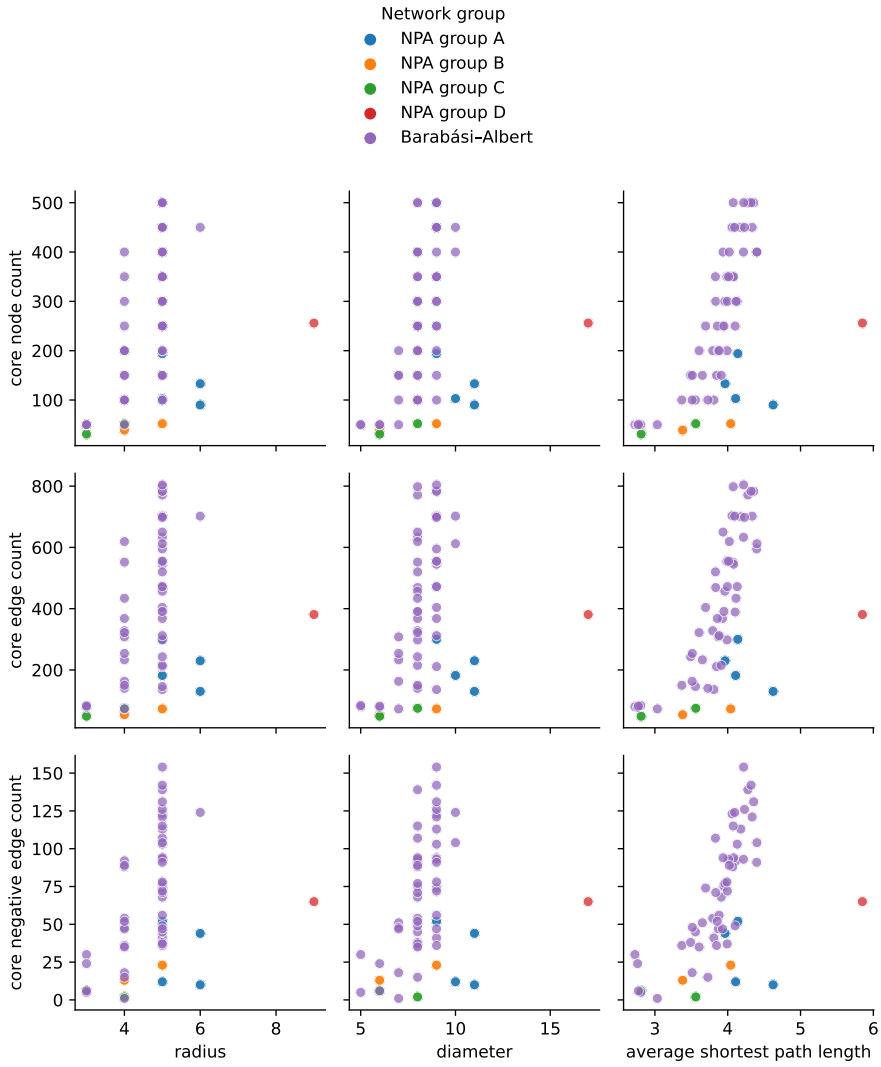

Figure A.7: Pairwise relationships between primary core and secondary distance features.

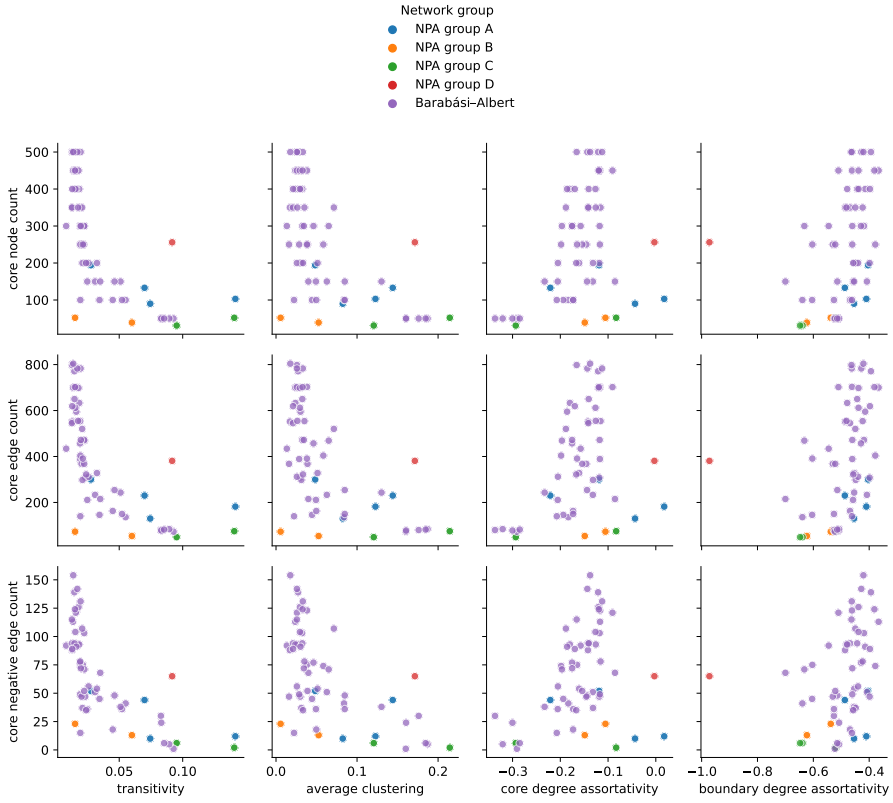

Figure A.8: Pairwise relationships between primary core and secondary local structure features.

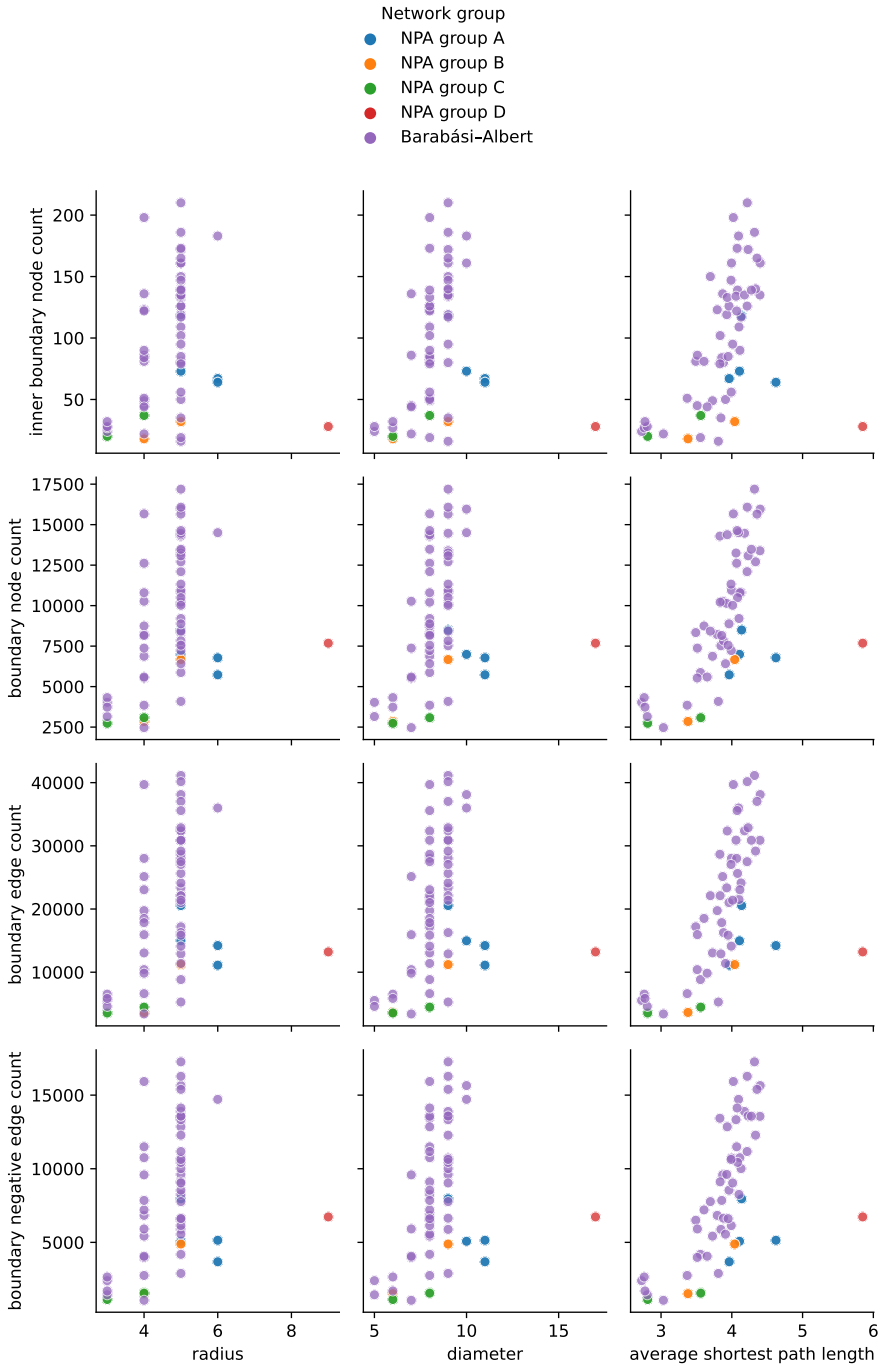

Figure A.9: Pairwise relationships between primary boundary and secondary distance features.

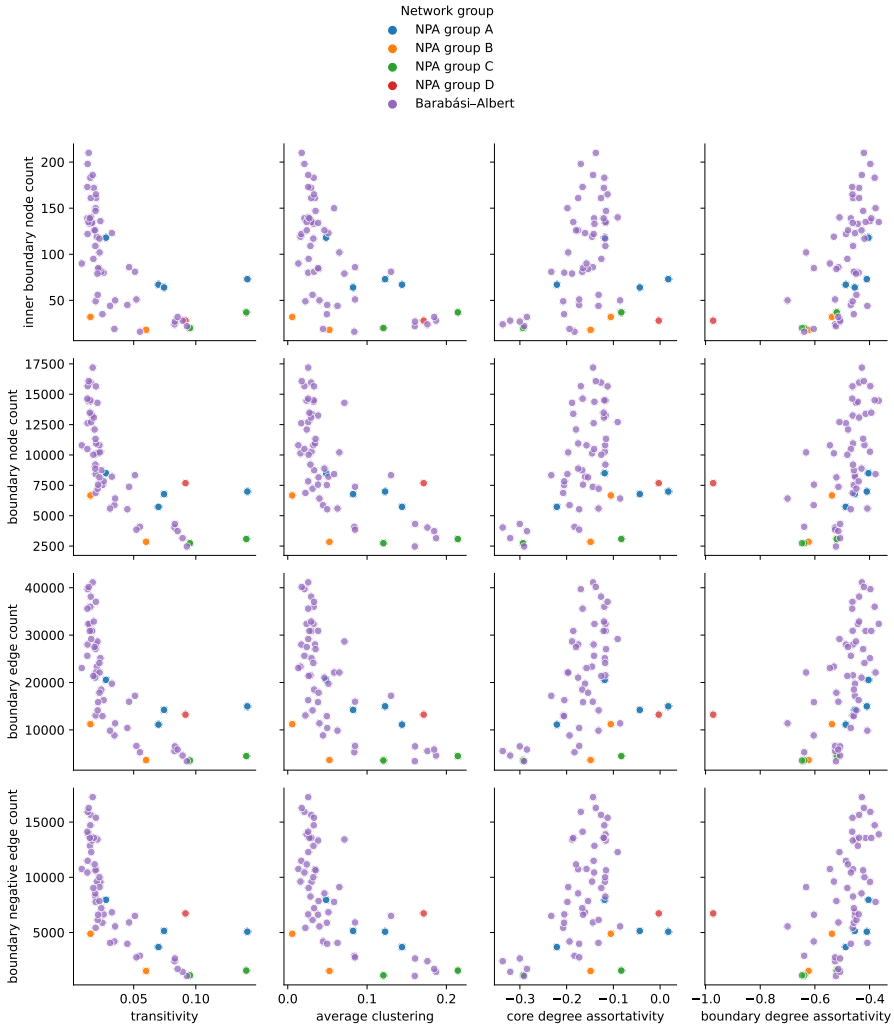

Figure A.10: Pairwise relationships between primary boundary and secondary local structure features.

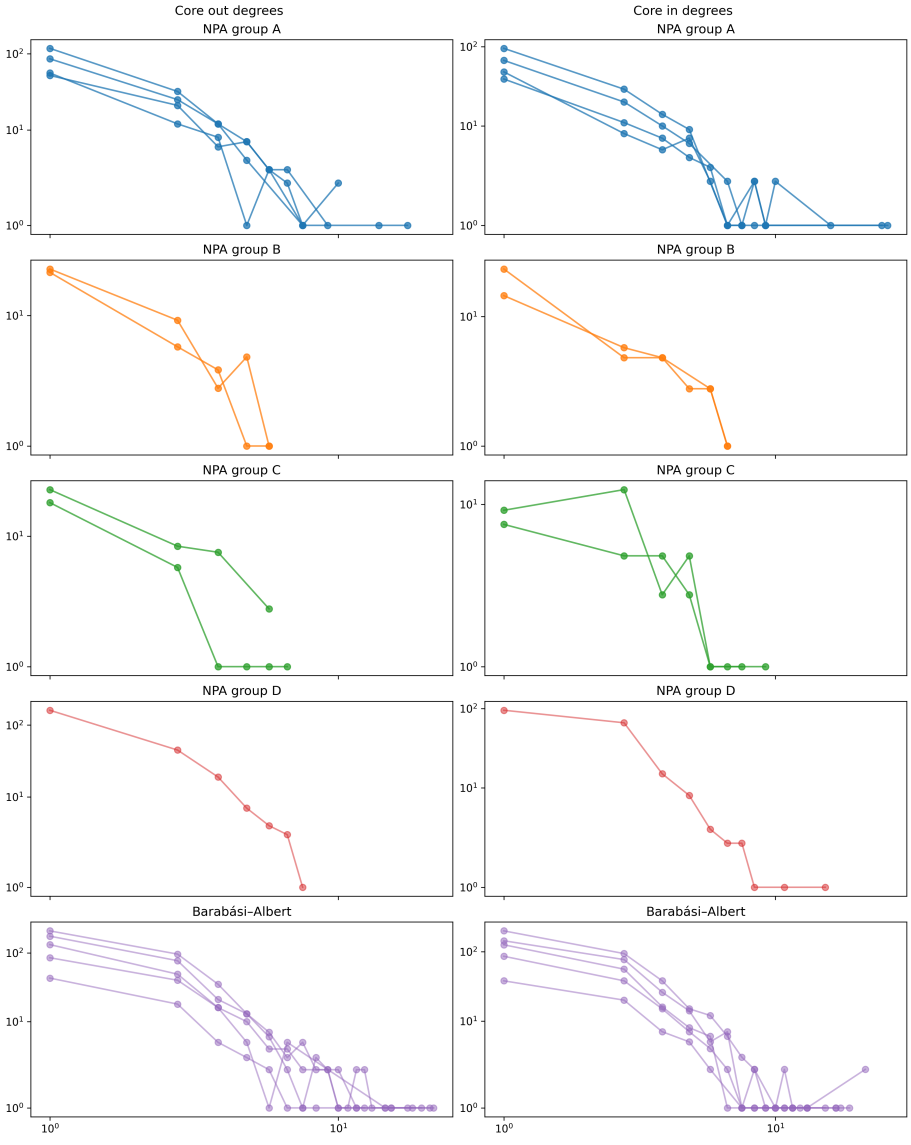

Figure A.11: Outdegree and indegree distributions of the core networks.

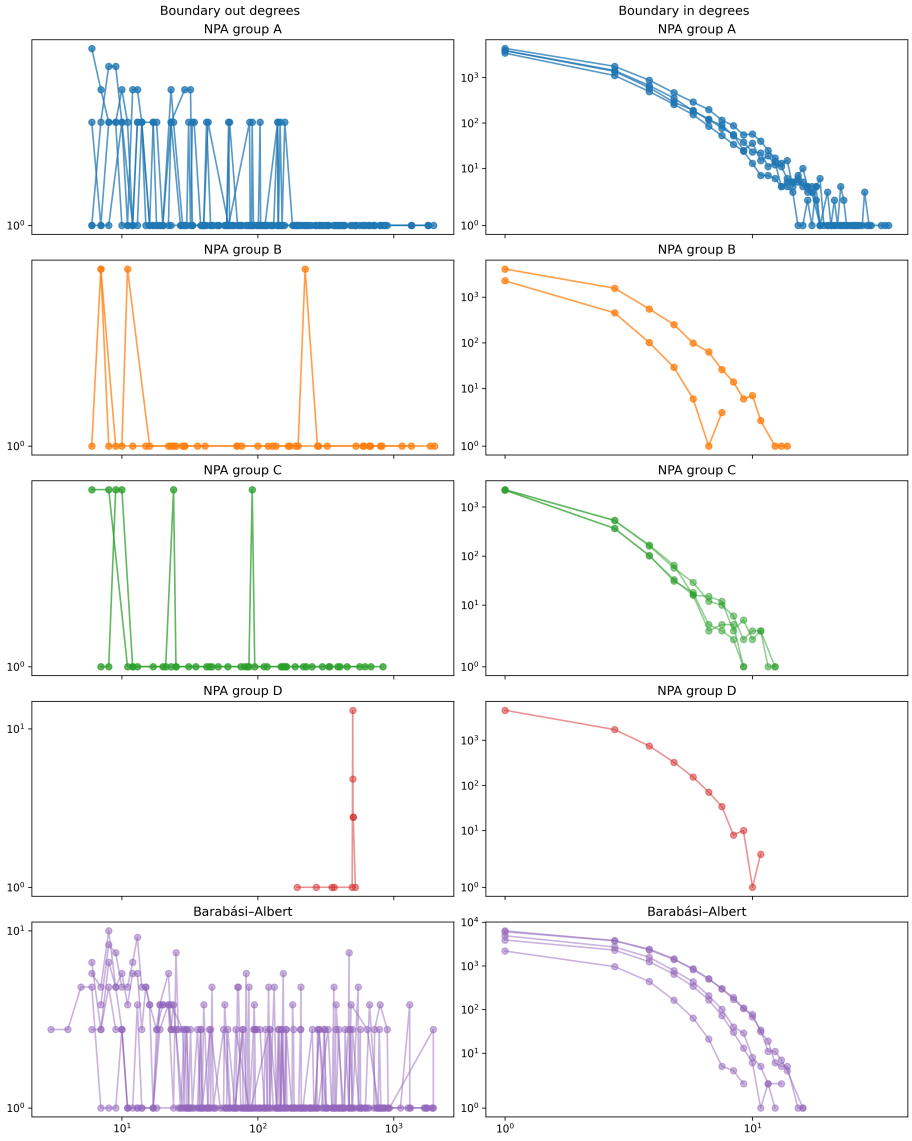

Figure A.12: Outdegree and indegree distributions of the boundary networks.

Bibliography

[1] B. H. Junker, F. Schreiber, Analysis of Biological Networks, John Wiley & Sons, Ltd, 2008. doi:10.1002/9780470253489.

[2] F. Martin, A. Sewer, M. Talikka, Y. Xiang, J. Hoeng, M. C. Peitsch, Quantification of biological network perturbations for mechanistic insight and diagnostics using two-layer causal models, BMC Bioinformatics 15 (1) (2014) 238. doi:10.1186/1471-2105-15-238.

[3] M. Rajh, Perturbationx, https://github.com/mikethenut/perturbationx, accessed: 2023-10-02 (2023).

[4] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, T. Ideker, Cytoscape: A software environment for integrated models of biomolecular interaction networks, Genome Research 13 (11) (2003) 2498–2504. doi:10.1101/gr.1239303.

[5] Y. Guo, P. Xiao, S. Lei, F. Deng, G. G. Xiao, Y. Liu, X. Chen, L. Li, S. Wu, Y. Chen, H. Jiang, L. Tan, J. Xie, X. Zhu, S. Liang, H. Deng, How is mRNA expression predictive for protein expression? a correlation study on human circulating monocytes, Acta Biochimica et Biophysica Sinica 40 (5) (2008) 426–436. doi:10.1111/j.1745-7270.2008.00418.x.

[6] J. Pollard, A. J. Butte, S. Hoberman, M. Joshi, J. Levy, J. Pappo, A computational model to define the molecular causes of type 2 diabetes

mellitus, Diabetes Technology & Therapeutics 7 (2) (2005) 323–336, pMID: 15857235. doi:10.1089/dia.2005.7.323.

[7] L. Chindelevitch, D. Ziemek, A. Enayetallah, R. Randhawa, B. Sidders, C. Brockel, E. S. Huang, Causal reasoning on biological networks: interpreting transcriptional changes, Bioinformatics 28 (8) (2012) 1114–1121. doi:10.1093/bioinformatics/bts090.

[8] F. Martin, T. M. Thomson, A. Sewer, D. A. Drubin, C. Mathis, D. Weisensee, D. Pratt, J. Hoeng, M. C. Peitsch, Assessment of network perturbation amplitudes by applying high-throughput data to causal biological networks, BMC Systems Biology 6 (1) (2012) 54. doi:10.1186/1752-0509-6-54.

[9] J. Hoeng, R. Deehan, D. Pratt, F. Martin, A. Sewer, T. M. Thomson, T. M. Thomson, D. A. Drubin, C. A. Waters, D. de Graaf, M. C. Peitsch, A network-based approach to quantifying the impact of biologically active substances, Drug discovery today 17 (9-10) (2012) 413—418. doi:10.1016/j.drudis.2011.11.008.

[10] N. L. Catlett, A. J. Bargnesi, S. Ungerer, T. Seagaran, W. Ladd, K. O. Elliston, D. Pratt, Reverse causal reasoning: applying qualitative causal knowledge to the interpretation of high-throughput data, BMC Bioinformatics 14 (1) (2013) 340. doi:10.1186/1471-2105-14-340.

[11] F. Martin, S. Gubian, M. Talikka, J. Hoeng, M. C. Peitsch, NPA: an R package for computing network perturbation amplitudes using gene expression data and two-layer networks, BMC Bioinformatics 20 (1) (2019) 451. doi:10.1186/s12859-019-3016-x.

[12] M. Talikka, S. Boue, W. K. Schlage, Causal biological network database: A comprehensive platform of causal biological network models focused on the pulmonary and vascular systems, in: J. Hoeng, M. C. Peitsch (Eds.), Computational Systems Toxicology, Springer New York, New York, NY, 2015, pp. 65–93. doi:10.1007/978-1-4939-2778-4_3.

[13] C. T. Hoyt, A. Konotopez, C. Ebeling, PyBEL: a computational framework for biological expression language, Bioinformatics 34 (4) (2017) 703–704. doi:10.1093/bioinformatics/btx660.

[14] R. Karki, A. T. Kodamullil, C. T. Hoyt, M. Hofmann-Apitius, Quantifying mechanisms in neurodegenerative diseases (NDDs) using candidate mechanism perturbation amplitude (CMPA) algorithm, BMC Bioinformatics 20 (1) (2019) 494. doi:10.1186/s12859-019-3101-1.

[15] A. Krämer, J. Green, J. Pollard, Jack, S. Tugendreich, Causal analysis approaches in Ingenuity Pathway Analysis, Bioinformatics 30 (4) (2013) 523–530. doi:10.1093/bioinformatics/btt703.

[16] G. Bradley, S. J. Barrett, CausalR: extracting mechanistic sense from genome scale data, Bioinformatics 33 (22) (2017) 3670–3672. doi:10.1093/bioinformatics/btx425.

[17] S. Jaeger, J. Min, F. Nigsch, M. Camargo, J. Hutz, A. Cornett, S. Cleaver, A. Buckler, J. L. Jenkins, Causal network models for predicting compound targets and driving pathways in cancer, Journal of Biomolecular Screening 19 (5) (2014) 791–802, pMID: 24518063. doi:10.1177/1087057114522690.

[18] E. O. Paull, D. E. Carlin, M. Niepel, P. K. Sorger, D. Haussler, J. M. Stuart, Discovering causal pathways linking genomic events to transcriptional states using tied diffusion through interacting events (TieDIE), Bioinformatics 29 (21) (2013) 2757–2764. doi:10.1093/bioinformatics/btt471.

[19] A. Liu, P. Trairatphisan, E. Gjerga, A. Didangelos, J. Barratt, J. Saez-Rodriguez, From expression footprints to causal pathways: contextualizing large signaling networks with CARNIVAL, npj Systems Biology and Applications 5 (1) (2019) 40. doi:10.1038/s41540-019-0118-z.

[20] A. Dugourd, C. Kuppe, M. Sciacovelli, E. Gjerga, A. Gabor, K. B. Emdal, V. Vieira, D. B. Bekker-Jensen, J. Kranz, E. Bindels, A. S. Costa, A. Sousa, P. Beltrao, M. Rocha, J. V. Olsen, C. Frezza, R. Kramann, J. Saez-Rodriguez, Causal integration of multi-omics data with prior knowledge to generate mechanistic hypotheses, Molecular Systems Biology 17 (1) (2021) e9730. doi:https://doi.org/10.15252/msb.20209730.

[21] M. Garrido-Rodriguez, K. Zirngibl, O. Ivanova, S. Lobentanzer, J. Saez-Rodriguez, Integrating knowledge and omics to decipher mechanisms via large-scale models of signaling networks, Molecular Systems Biology 18 (7) (2022) e11036. doi:https://doi.org/10.15252/msb.202211036.

[22] L. Hosseini-Gerami, I. A. Higgins, D. A. Collier, E. Laing, D. Evans, H. Broughton, A. Bender, Benchmarking causal reasoning algorithms for gene expression-based compound mechanism of action analysis, BMC Bioinformatics 24 (1) (2023) 154. doi:10.1186/s12859-023-05277-1.

[23] V. Touré, Å. Flobak, A. Niarakis, S. Vercruysse, M. Kuiper, The status of causality in biological databases: data resources and data retrieval possibilities to support logical modeling, Briefings in Bioinformatics 22 (4) (2021) bbaa390. doi:10.1093/bib/bbaa390.

[24] A. A. Hagberg, D. A. Schult, P. J. Swart, Exploring network structure, dynamics, and function using NetworkX, in: G. Varoquaux, T. Vaught, J. Millman (Eds.), Proceedings of the 7th Python in Science Conference, Pasadena, CA USA, 2008, pp. 11 – 15.

[25] Wes McKinney, Data Structures for Statistical Computing in Python, in: Stéfan van der Walt, Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 56 – 61. doi:10.25080/Majora-92bf1922-00a.

[26] The pandas development team, pandas-dev/pandas: Pandas (Aug. 2023). doi:10.5281/zenodo.8301632.

[27] Cytoscape Consortium, py4cytoscape, https://github.com/cytoscape/py4cytoscape, accessed: 2023-08-09 (2023).

[28] D. Otasek, J. H. Morris, J. Bouças, A. R. Pico, B. Demchak, Cytoscape Automation: empowering workflow-based network analysis, Genome Biology 20 (1) (2019) 185. doi:10.1186/s13059-019-1758-4.

[29] J. Oh, A. E. Riek, K. T. Bauerle, A. Dusso, K. P. McNerney, R. A. Barve, I. Darwech, J. E. Sprague, C. Moynihan, R. M. Zhang, G. Kutz, T. Wang, X. Xing, D. Li, M. Mrad, N. M. Wigge, E. Castelblanco, A. Collin, M. Bambouskova, R. D. Head, M. S. Sands, C. Bernal-Mizrachi, Embryonic vitamin D deficiency programs hematopoietic stem cells to induce type 2 diabetes, Nat Commun 14 (1) (2023) 3278.

[30] K. J. Baines, M. S. Klausner, V. S. Patterson, S. J. Renaud, Interleukin-15 deficient rats have reduced osteopontin at the maternal-fetal interface, Front Cell Dev Biol 11 (2023) 1079164.

[31] J. Neuckermans, S. Lequeue, P. Claes, A. Heymans, J. H. Hughes, H. Colemonts-Vroninks, L. Marcélis, G. Casimir, P. Goyens, G. A. Martens, J. A. Gallagher, T. Vanhaecke, G. Bou-Gharios, J. De Kock, Hereditary tyrosinemia type 1 mice under continuous nitisinone treatment display remnants of an uncorrected liver disease phenotype, Genes (Basel) 14 (3) (Mar. 2023).

[32] F. Emmert-Streib, M. Dehmer, B. Haibe-Kains, Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks, Frontiers in Cell and Developmental Biology 2 (2014). doi:10.3389/fcell.2014.00038.

[33] O. Folger, L. Jerby, C. Frezza, E. Gottlieb, E. Ruppin, T. Shlomi, Predicting selective drug targets in cancer through metabolic networks,

Molecular Systems Biology 7 (1) (2011) 501. doi:10.1038/msb.2011.35.

[34] P. Khatri, M. Sirota, A. J. Butte, Ten years of pathway analysis: Current approaches and outstanding challenges, PLOS Computational Biology 8 (2) (2012) 1–10. doi:10.1371/journal.pcbi.1002375.

[35] B. Phillips, E. Veljkovic, S. Boué, W. K. Schlage, G. Vuillaume, F. Martin, B. Titz, P. Leroy, A. Buettner, A. Elamin, A. Oviedo, M. Cabanski, H. De León, E. Guedj, T. Schneider, M. Talikka, N. V. Ivanov, P. Vanscheeuwijck, M. C. Peitsch, J. Hoeng, An 8-Month Systems Toxicology Inhalation/Cessation Study in Apoe−/− Mice to Investigate Cardiovascular and Respiratory Exposure Effects of a Candidate Modified Risk Tobacco Product, THS 2.2, Compared With Conventional Cigarettes, Toxicological Sciences 149 (2) (2015) 411–432. doi:10.1093/toxsci/kfv243.

[36] B. Phillips, E. Veljkovic, M. J. Peck, A. Buettner, A. Elamin, E. Guedj, G. Vuillaume, N. V. Ivanov, F. Martin, S. Boué, W. K. Schlage, T. Schneider, B. Titz, M. Talikka, P. Vanscheeuwijck, J. Hoeng, M. C. Peitsch, A 7-month cigarette smoke inhalation study in c57bl/6 mice demonstrates reduced lung inflammation and emphysema following smoking cessation or aerosol exposure from a prototypic modified risk tobacco product, Food and Chemical Toxicology 80 (2015) 328–345. doi:10.1016/j.fct.2015.03.009.

[37] A. Oviedo, S. Lebrun, U. Kogel, J. Ho, W. T. Tan, B. Titz, P. Leroy, G. Vuillaume, M. Bera, F. Martin, G. Rodrigo, M. Esposito, R. Dempsey, N. V. Ivanov, J. Hoeng, M. C. Peitsch, P. Vanscheeuwijck, Evaluation of the tobacco heating system 2.2. part 6: 90-day OECD 413 rat inhalation study with systems toxicology endpoints demonstrates reduced exposure effects of a mentholated version compared with mentholated and non-mentholated cigarette smoke, Regulatory Toxicology

and Pharmacology 81 (2016) S93–S122, evaluation of the Tobacco Heating System 2.2. doi:10.1016/j.yrtph.2016.11.004.

[38] R. Li, A. Zupanic, M. Talikka, V. Belcastro, S. Madan, J. Dörpinghaus, C. v. Berg, J. Szostak, F. Martin, M. C. Peitsch, J. Hoeng, Systems toxicology approach for testing chemical cardiotoxicity in larval zebrafish, Chemical Research in Toxicology 33 (10) (2020) 2550–2564. doi:10.1021/acs.chemrestox.0c00095.

[39] U. Kogel, I. Gonzalez Suarez, Y. Xiang, E. Dossin, P. Guy, C. Mathis, D. Marescotti, D. Goedertier, F. Martin, M. Peitsch, J. Hoeng, Biological impact of cigarette smoke compared to an aerosol produced from a prototypic modified risk tobacco product on normal human bronchial epithelial cells, Toxicology in Vitro 29 (8) (2015) 2102–2115. doi:10.1016/j.tiv.2015.08.004.

[40] F. Zanetti, A. Sewer, C. Mathis, A. R. Iskandar, R. Kostadinova, W. K. Schlage, P. Leroy, S. Majeed, E. Guedj, K. Trivedi, F. Martin, A. Elamin, C. Merg, N. V. Ivanov, S. Frentzel, M. C. Peitsch, J. Hoeng, Systems toxicology assessment of the biological impact of a candidate modified risk tobacco product on human organotypic oral epithelial cultures, Chemical Research in Toxicology 29 (8) (2016) 1252–1269. doi:10.1021/acs.chemrestox.6b00174.

[41] H. Yepiskoposyan, M. C. Peitsch, M. Talikka, Causal biological network model for inflammasome signaling applied for interpreting transcriptomic changes in various inflammatory states, International Journal of Inflammation 2022 (2022) 4071472. doi:10.1155/2022/4071472.

[42] CLARIVATE PLC, MetaBase & MetaCore, a Cortellis solution, https://web.archive.org/web/20230322023249/https://clarivate.com/products/biopharma/research-development/early-research-intelligence-solutions/, accessed: 2023-08-09 (2023).

[43] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork, L. J. Jensen, C. v. Mering, STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets, Nucleic Acids Res 47 (D1) (2019) D607–D613.

[44] I. N. Melas, T. Sakellaropoulos, F. Iorio, L. G. Alexopoulos, W.-Y. Loh, D. A. Lauffenburger, J. Saez-Rodriguez, J. P. F. Bai, Identification of drug-specific pathways based on gene expression data: application to drug induced lung injury, Integrative Biology 7 (8) (2015) 904–920. doi:10.1039/c4ib00294f.

[45] D. Türei, T. Korcsmáros, J. Saez-Rodriguez, OmniPath: guidelines and gateway for literature-curated signaling pathway resources, Nature Methods 13 (12) (2016) 966–967. doi:10.1038/nmeth.4077.

[46] F. Chung, Spectral Graph Theory, Conference Board of Mathematical Sciences, American Mathematical Society, 1997.

[47] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, The FAIR guiding principles for scientific data management and stewardship, Scientific Data 3 (1) (2016) 160018. doi:10.1038/sdata.2016.18.

[48] M. Newman, The configuration model, in: Networks, Oxford University Press, 2018, pp. 369–433. doi:10.1093/oso/9780198805090.003.0012.

[49] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental algorithms for scientific computing in python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.

[50] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Rev. Mod. Phys. 74 (2002) 47–97. doi:10.1103/RevModPhys.74.47.

[51] pythonprofilers, Memory profiler, https://github.com/pythonprofilers/memory_profiler, accessed: 2023-08-09 (2023).

[52] P. Sah, L. O. Singh, A. Clauset, S. Bansal, Exploring community structure in biological networks with random graphs, BMC Bioinformatics 15 (1) (2014) 220. doi:10.1186/1471-2105-15-220.