Understanding protein complexes by graph-theoretical analysis of their topology

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich 12
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

von
Jan Niclas Wolf
aus Gelnhausen

Frankfurt 2023

vom Fachbereich 12 der
Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Dekan: Prof. Dr. Martin Möller

Gutachter: Prof. Dr. Ina Koch, Prof. Dr. Marcel Schulz

Datum der Disputation: 30.10.2023

Abstract

Proteins are biological macromolecules playing essential roles in all living organisms. Proteins often bind with each other forming complexes to fulfill their function. Such protein complexes assemble along an ordered pathway. An assembled protein complex can often be divided into structural and functional modules. Knowing the order of assembly and the modules of a protein complex is important to understand biological processes and treat diseases related to misassembly.

Typical structures of the Protein Data Bank (PDB) contain two to three subunits and a few thousand atoms. Recent developments have led to large protein complexes being resolved. The increasing number and size of the protein complexes demand for computational assistance for the visualization and analysis. One such large protein complex is respiratory complex I accounting for 45 subunits in *Homo sapiens*. Complex I is a well understood protein complex that served as case study to validate our methods.

Our aim was to analyze time-resolved Molecular Dynamics (MD) simulation data, identify modules of a protein complex and generate hypotheses for the assembly pathway of a protein complex. For that purpose, we abstracted the topology of protein complexes to Complex Graphs of the Protein Topology Graph Library (PTGL). The subunits are represented as vertices, and spatial contacts as edges. The edges are weighted with the number of contacts based on a distance threshold. This allowed us to apply graph-theoretic methods to visualize and analyze protein complexes.

We extended the implementations of two methods to achieve a computation of Complex Graphs in feasible runtimes. The first method skipped checks for contacts using the information which residues are sequential neighbors. We extended the method to protein complexes and structures containing ligands. The second method introduced spheres encompassing all atoms of a subunit and skipped the check for contacts if the corresponding spheres do not overlap. Both methods combined allowed skipping up to 93 % of the checks for contacts for sample complexes of 40 subunits compared to up to 10 % of the previous implementation. We showed that the runtime of the combined method scaled linearly with the number of atoms compared to a non-linear scaling of the previous implementation

We implemented a third method fixing the assignment of an orientation to secondary structure elements. We placed a three-dimensional vector in each secondary structure element and computed the angle between secondary structure elements to assign an orientation. This method sped up the runtime especially for large structures, such as the capsid of human immunodeficiency virus, for which the runtime decreased from 43 to less than 9 hours.

The feasible runtimes allowed us to investigate two data sets of MD trajectories of respiratory complex I of *Thermus thermophilus* that we received. The data sets differ only by whether ubiquinone is bound to the complex. We implemented a pipeline, PTGLdynamics, to compute the contacts and Complex Graphs for all time steps of the trajectories. We investigated different methods to track changes of contacts during the simulation and created a heat map put onto the three-dimensional structure visualizing the changes. We also created line plots to visualize the changes of contacts over the course of the simulation. Both visualizations helped spotting outstandingly flexible or rigid regions of the structure or time points of the simulation in which major dynamics occur.

We introduced normalizations of the edge weights of Complex Graphs for identi-

fying modules and predicting the assembly pathway. The idea is to normalize the number of contacts for the number of residues of a subunit. We defined five different normalizations.

To identify structural and functional modules, we applied the Leiden graph clustering algorithm to the Complex Graphs of respiratory complex I and the respiratory supercomplex. We examined the results for the different normalizations of the weights of the Complex Graphs. The absolute edge weight produced the best result identifying three of four modules that have been defined in the literature for respiratory complex I.

We applied agglomerative hierarchical clustering to the edges of a Complex Graph to create hypotheses of the assembly pathway. The rationale was that subunits with an extensive interface in the final structure assemble early. We tested our method against two existing methods on a data set of 21 proteins with reported assembly pathways. Our prediction outperformed the other methods and ran in feasible runtimes of a few minutes at most.

We also tested our method on respiratory complex I, the respiratory supercomplex and the respiratory megacomplex. We compared the results for the different normalizations with an assembly pathway of respiratory complex I described in the literature. We transformed the assembly pathways to dendrograms and compared the predictions to the reference using the Robinson-Foulds distance and clustering information distance. We analyzed the landscape of the clustering information distance by generating random dendrograms and showed that our result is far better than expected at random. We showed in a detailed analysis that the assembly prediction using one normalization was able to capture key features of the assembly pathway that has been proposed in the literature.

In conclusion, we presented different applications of graph theory to automatically analyze the topology of protein complexes. Our programs run in feasible runtimes even for large complexes. We showed that graph-theoretic modeling of the protein structure can be used to analyze MD simulation data, identify modules of protein complexes and predict assembly pathways.

Zusammenfassung

Proteine sind biologische Makromoleküle, die eine essenzielle Rolle in allen lebenden Organismen spielen. Proteine erfüllen dabei verschiedenste Aufgaben, die von Stabilität für Zellen bis hin zur Erbgutvervielfältigung reichen. Häufig lagern sich Proteine dafür zusammen und bilden Proteinkomplexe. Daraus ergeben sich verschiedene Abstraktionsstufen für Proteinkomplexe: die Stufe der Atome, der Aminosäuren, der Sekundärstrukturelemente und der Proteinketten, beziehungsweise Untereinheiten im Kontext von Komplexen.

Ähnlich wie die Faltung von Proteinketten, assemblieren Proteinkomplexe entlang eines geordneten Pfades. Ein assemblierter Proteinkomplex kann oft in strukturelle und funktionelle Module unterteilt werden. Die Reihenfolge der Assemblierung und die Module eines Proteinkomplexes zu kennen, ist wichtig, um biologische Prozesse zu verstehen und Krankheiten zu heilen, die im Zusammenhang mit Fehlassemblierung stehen.

Typische Strukturen der *Protein Data Bank* (PDB) enthalten zwei bis drei Untereinheiten und einige tausend Atome. Kürzliche Entwicklungen in der Proteinstrukturvorhersage, besonders *single-particle cryo-electron microscopy with image reconstruction*, haben dazu geführt, dass große Proteinkomplexe aufgeklärt wurden. Die Größe der Proteinkomplexe erfordert Unterstützung durch Rechner für die Visualisierung und Analyse. Eines dieser großen Proteinkomplexe ist der respiratorische Komplex I, der in *Homo sapiens* 45 Untereinheiten ausmacht. Komplex I ist ein gut untersuchter Proteinkomplex, der uns als Fallbeispiel diente, um unsere Methoden zu validieren.

Unser Ziel war es, zeitaufgelöste *Molecular Dynamics*- (MD-)Simulationsdaten zu analysieren, Module eines Proteinkomplexes zu identifizieren und Hypothesen für den Assemblierungspfad eines Proteinkomplexes zu generieren. Dafür abstrahierten wir auf unterschiedlichen Abstraktionsstufen die Topologie von Proteinkomplexen auf Graphen der *Protein Topology Graph Library* (PTGL). In *Complex Graphs* werden die Untereinheiten durch Knoten repräsentiert und räumliche Kontakte als Kanten. Räumliche Kontakte basieren auf einem Distanzschwellwert zwischen Atomen, sodass zum Beispiel zwei Atome von Aminosäuren unterhalb einer Distanz von 4 Å als Kontakt definiert werden. Die Kanten des *Complex Graphs* sind mit der Anzahl an Kontakten gewichtet. Die Übertragung der Topologie von Proteinkomplexen auf Graphebene erlaubte uns, automatisiert graphentheoretische Methoden anzuwenden, um Proteinkomplexe zu visualisieren und zu analysieren.

Wir erweiterten die Implementierungen von zwei Methoden, um eine Berechnung von *Complex Graphs* in hinnehmbarer Laufzeit zu erreichen. Die erste Methode übersprang das Überprüfen von Kontakten unter Nutzung der Information, welche Aminosäuren sequentielle Nachbarn sind. Wir erweiterten die Methode, sodass sie auch auf Proteinkomplexe und Strukturen, die Liganden enthalten, angewandt werden kann. Die zweite Methode führte Sphären ein, die den Distanzvergleich aller Atome zweier Aminosäuren übersprang, wenn die Sphären der Aminosäuren nicht überlappten. Wir erweiterten die Methode, indem wir ebenfalls Sphären für Proteinketten einführten. Die beiden Methoden kombiniert erlaubten bis zu 93 % der Überprüfungen auf Kontakte für beispielhafte Komplexe von 40 Untereinheiten zu überspringen, verglichen mit bis zu 10 % der vorherigen Implementierung. Wir zeigten, dass die Laufzeit der kombinierten Methoden linear mit der Anzahl an Atomen skalierte, verglichen zu einer nicht-linearen Skalierung der vorherigen Implementierung.

Wir implementierten eine dritte Methode, die die Zuordnung von einer Orientierung zwischen zwei Sekundärstrukturelementen korrigierte. Wir platzierten einen dreidimensionalen Vektor entlang des Verlaufs jedes Sekundärstrukturelements und berechneten den Winkel zwischen Sekundärstrukturelementen, um eine Orientierung zuzuordnen. Die Methode reduzierte die Laufzeit besonders für große Strukturen wie die Hülle vom menschlichen Immundefizienz-Virus, für das die Laufzeit von 43 zu weniger als 9 Stunden reduziert wurde. Diese hinnehmbaren Laufzeiten erlaubten uns die Untersuchung umfangreicher Proteinkomplexe, wie Komplex I der Atmungskette.

Wir erhielten zwei Datensätze MD-Trajektorien des Komplex I von *Thermus thermophilus*. Die Datensätze unterscheiden sich nur darin, ob Ubiquinon an den Komplex gebunden ist. Wir implementierten eine Pipeline, *PTGLdynamics*, um die MD-Trajektorien einzulesen und Kontakte sowie *Complex Graphs* für alle Zeitschritte der Trajektorien zu berechnen. Wir untersuchten unterschiedliche Methoden, um die Änderungen von Kontakten über die Simulation hinweg zu verfolgen.

Zunächst untersuchten wir die Änderungen der Kantengewichte zwischen *Complex Graphs* aufeinander folgender Zeitschritte. Wir verglichen die absolute Anzahl an Änderungen und die Anzahl an Änderungen normalisiert mit der Kettenlänge. Die Änderung an Kontakten an den Kantengewichten zu bemessen erwies sich als ungenau, da sich hinzukommende und verlorene Kontakte gegenseitig in der Anzahl ausgleichen. Außerdem enthielten die Kantengewichte nur die Änderungen von Kontakten zwischen Proteinketten, aber nicht Änderungen von Kontakten innerhalb einer Proteinkette.

Um dieses Problem zu beheben, erweiterten wir die Pipeline um den Vergleich der einzelnen Kontakte zwischen Aminosäuren. Dadurch erhielten wir die exakte Anzahl an Kontaktänderungen und konnten zwischen Kontakten innerhalb und Kontakten zwischen Proteinketten unterscheiden. Da wir zuvor die Kantengewichte des *Complex Graphs* nutzten, konnten wir nur die Ebene der Proteinketten betrachten. Durch das Verfolgen der Kontakte zwischen Aminosäuren konnten wir nun auch auf der Ebene der Aminosäuren Beobachtungen anstellen.

Wir erzeugten für jeden Ansatz eine Heatmap, die wir auf die dreidimensionale Struktur legten, um Bereiche geringer oder großer Änderung der Kontakte zu visualisieren. Die Heatmaps der unterschiedlichen Ansätze erlaubten je nach Fragestellung besonders starre oder flexible Proteinketten oder einzelne Aminosäuren zu identifizieren. Bei der Heatmap auf der Ebene von Proteinketten, unter Einbezug von Kontakten zwischen und innerhalb von Proteinketten, konnten wir sehen, dass der Matrixarm größtenteils starr und der Membranarm größtenteils flexibel war. Wir konnten außerdem die Heatmaps der beiden Datensätze vergleichen und somit die Unterschiede zwischen den Simulationen, in Bezug auf geänderte Dynamiken, visualisieren. In der Heatmap von diesem Vergleich konnten wir sehen, dass die meisten Aminosäuren eine ähnliche Anzahl an Kontaktänderungen aufwiesen, sich einige Aminosäuren jedoch unterschiedlich in den Simulationen verhielten. Die Unterschiede waren in der Nähe von Ubiquinon besonders groß. Wir erzeugten außerdem Liniendiagramme, um die Änderungen von Kontakten über den Verlauf der Simulation zu visualisieren.

Für die nachfolgenden Betrachtungen führten wir Normalisierungen der Kantengewichte von *Complex Graphs* ein. Ziel war es, die Kantengewichte mit der Länge der beteiligten Proteinketten zu normalisieren, weswegen in jeder Normalisierung die Anzahl absoluter Kontakte durch einen Term geteilt wird, der die Längen der Proteinketten enthält. Je nach Term definierten wir fünf verschiedene Normalisierungen.

Beispielsweise werden bei der additiven Normalisierung die Längen der Proteinketten addiert und bei der multiplikativen multipliziert.

Wir wandten den Leiden-Graphenclusteralgorithmus auf *Complex Graphs* des respiratorischen Komplexes I und des respiratorischen Superkomplexes an, um eine strukturelle oder funktionelle Unterteilung vorherzusagen. Die Vorteile des Algorithmus sind, dass er keine Vorgabe einer gesuchten Anzahl an Clustern benötigt und Kantengewichte berücksichtigt. Wir testeten die Performanz der verschiedenen Normalisierungen und der absoluten Kantengewichte. Für Komplex I verglichen wir die Partitionen des Leiden-Graphenclusteralgorithmus mit der Einteilung in der Literatur in vier Module. Für den Superkomplex verglichen wir die Partitionen mit der Einteilung in die Komplexe und mit der Unterteilung von Komplex I in Module.

Das Clustering mit dem absoluten Kantengewicht erzeugte die beste Partition, da drei von vier Clustern exakt den Modulen von Komplex I aus der Literatur entsprachen. Das vierte Modul erstreckte sich über zwei Cluster. Bei dem Superkomplex erzeugte das Clustering mit den absoluten und den additiven Kantengewichten dieselbe Partition. In dieser entsprachen die Cluster jeweils den zwei Kopien des Komplexes III und dem Komplex IV. Ein Cluster entsprach dem Matrixarm von Komplex I und zwei Cluster entsprachen mit einer Ausnahme den zwei übrigen Modulen.

Wir wandten agglomeratives hierarchisches Clustern der Kanten eines *Complex Graph* an, um Hypothesen über den Assemblierungspfad zu erzeugen. Die Idee dahinter war, dass Untereinheiten mit einem extensiven Interface in der finalen Struktur früh assemblieren. Als Maßstab dafür, wie extensiv ein Interface ist, benutzten wir die Kantengewichte des *Complex Graphs*. Wir testeten das absolute Kantengewicht und die Normalisierungen. Wir benutzten keine klassische *linkage function*, sondern verschmolzen zwei Knoten des *Complex Graphs*, nachdem deren jeweilige Cluster in einem Schritt des agglomerativen Clusterings verbunden wurden. Daraus folgte, dass wir die Kantengewichte verschmolzener Kanten addierten und gegebenenfalls die Normalisierungen neu berechneten.

Wir verglichen unsere Methode mit zwei existierenden Methoden auf einem Datensatz von 21 Proteinen mit bekannten Assemblierungspfaden. Basierend auf der Literatur benutzten wir drei Bewertungsschemata, um die Güte einer Vorhersage zu bestimmen. Unsere Vorhersagen, basierend auf dem additiven Kantengewicht, übertrafen die anderen Methoden für alle drei Bewertungsschemata. Unsere Methode sagte 14 von 21 Komplexen und insgesamt 45 von 58 Zwischenschritten korrekt voraus. Darüber hinaus lieferte unsere Methode die Vorhersage höchstens in wenigen Minuten pro Komplex, verglichen mit tagelangen Berechnungen einer existierenden Methode.

Wir wandten unsere Methode auf den respiratorischen Komplex I, den respiratorischen Super- und Megakomplex an. Wir benutzten eine experimentelle Bestimmung des Assemblierungspfades von Komplex I aus der Literatur als Referenz. Um einen quantitativen Vergleich zu ermöglichen, der über die drei Bewertungsschemata von vorher hinausgeht, transformierten wir den Referenzpfad und unsere Vorhersagen zu Dendrogrammen. Auf diese Weise konnten wir die Robinson-Foulds-Distanz und die Clusterinformationsdistanz benutzen, um den Unterschied zwischen Referenz und Vorhersage zu quantifizieren. Die Robinson-Foulds-Distanz erwies sich als ungeeignet, da sie wenig sensitiv war. Der vorhergesagte Pfad für Komplex I mit der niedrigsten Clusterinformationsdistanz zur Referenz war der Pfad basierend auf den additiven Kantengewichten.

Wir analysierten die Landschaft der Clusterinformationsdistanz, um den erhaltenen Distanzwert besser einschätzen zu können. Wir erzeugten dafür drei Millionen zufällige Dendrogramme mit den Blattbeschriftungen von Komplex I. Keines der zufälligen Dendrogramme besaß eine Clusterinformationsdistanz zur Referenz, die ansatzweise so niedrig war wie die zwischen dem vorhergesagten Pfad und der Referenz. Der Mittelwert der Clusterinformationsdistanzen wich um den zwanzigfachen Betrag der Standardabweichung von der Clusterinformationsdistanz des vorhergesagten Pfades ab. Wir schlossen daraus, dass sehr viel mehr zufällige Dendrogramme nötig wären, um einen vergleichbar ähnlichen Pfad zu erhalten, und dass die Vorhersage besser als zufällig erwartbar ist.

Für einen qualitativen Vergleich diskutierten wir die Ähnlichkeiten und Unterschiede der vorhergesagten Pfade mit der Referenz. Eine betrachtete Eigenschaft war, ob die Topologie eines Dendrogramms modular oder einer Treppe entsprechend ist. Das Dendrogramm basierend auf den absoluten Kantengewichte war ein Extrembeispiel für eine Topologie wie eine Treppe, denn mit wenigen Ausnahmen kam in jedem Schritt eine neue Untereinheit zu dem bestehenden Komplex hinzu. Das Dendrogramm basierend auf der multiplikativen Normalisierung stellte das Gegenbeispiel dar, denn es bildeten sich Subkomplexe, die sich nach und nach modular miteinander verbanden. Diese unterschiedlichen Eigenschaften waren klar mit unserer benutzten *linkage function* und dem absoluten Kantengewicht sowie der multiplikativen Normalisierung zu erklären.

Das Dendrogramm basierend auf der additiven Normalisierung wies größtenteils eine modulare Topologie auf, allerdings nicht so strikt wie für die multiplikative Normalisierung. Wir konnten alle Subkomplexe der Vorhersage Subkomplexen der Referenz zuordnen. Die Zuordnungen waren unterschiedlich passend, denn manchmal fehlten Untereinheiten, die sich an anderer Stelle befanden, oder es waren zusätzliche Untereinheiten im Subkomplex enthalten. Dennoch wies die Vorhersage des Assemblierungspfades viele Ähnlichkeiten in der Bildung der Subkomplexe mit der Referenz auf. Wir konnten im qualitativen Vergleich die quantitative Einschätzung bestätigen, dass die Vorhersage unter Benutzung der additiven Kantengewichte am besten funktionierte.

Wir erweiterten den Betrachtungsfall von Komplex I auf den Super- und Megakomplex. Unsere Methode konnte für die additiven Kantengewichte teilweise zwischen den einzelnen Komplexen unterscheiden, indem diese einzeln assemblierten. Die beiden Kopien von Komplex III konnten jedoch nicht unterschieden werden. Die Vorhersage für Komplex I als Teil des Super- oder Megakomplexes ist weniger ähnlich zur Referenz als für Komplex I isoliert betrachtet. Wir schlossen daraus, dass unsere Methode auch für die Vorhersage von Assemblierungspfaden von Super- und Megakomplexen geeignet ist, aber die Vorhersage schwieriger ist und damit die Ergebnisse schlechter sind.

Die Arbeit war der graphenteoretischen Untersuchung von der Topologie von Proteinkomplexes gewidmet. Im Laufe der Untersuchung testeten wir verschiedene Herangehensweisen, zum Beispiel für die Behandlung von Liganden oder die Normalisierung von Kontakten zwischen Proteinketten. Der Umgang mit Liganden ist nicht trivial, da sie keine eigenen Proteinketten darstellen und in der Regel kürzer sind, aber gleichzeitig sind Liganden wichtige Bestandteile und Kontaktpartner innerhalb von Komplexen. Wir testeten Kontakte, die transitiv über Liganden definiert sind, doch fanden für unsere Fallbeispiele keine Verbesserungen für die Vorhersagen. Der

sinnvolle Einbezug von Liganden sollte weiter untersucht werden und könnte je nach Methode unterschiedlich ausfallen.

Die vorgeschlagenen Normalisierungen von Kontakten lieferten für die Modul- und Assemblierungsvorhersage zum Teil stark unterschiedliche Ergebnisse. Für die Modulvorhersage erzielte die Benutzung der absoluten Anzahl an Kontakten bessere Ergebnisse als irgendeine Normalisierung, danach folgte die additive Normalisierung. Bei der Assemblierungsvorhersage erzielte die Vorhersage unter Benutzung multiplikativer Kantengewichte gute Vorhersagen, da der Pfad wie in der Referenz eine modulare Topologie aufwies. Die Vorhersage unter Benutzung der additiven Normalisierung erzielte jedoch ein insgesamt besseres Ergebnis. Es sollten weitere Normalisierungen untersucht werden, um die verschiedenen Kettenlängen zu berücksichtigen. Es ist denkbar, dass für verschiedene Anwendungen unterschiedliche Normalisierungen bessere Ergebnisse liefern.

Die Vorhersage des Assemblierungspfades könnte weiter verbessert werden, indem die Auswirkungen verschiedener Parameter untersucht wird. Anfänge legten wir mit der Betrachtung des Distanzschwellwerts, dessen Erhöhung auf 7 Å bei schwierigen Fällen ein besseres Ergebnis erzielte. Bei anderen schwierigen Fällen schlagen wir eine unterschiedliche Gewichtung von Kontakten vor, basierend darauf, ob der Kontakt zwischen Sekundärstrukturelementen besteht. Um die Lesbarkeit der vorhergesagten Assemblierungspfade zu erhöhen und damit ihre praktische Anwendung zu ermöglichen, schlagen wir eine automatisierte Visualisierung im Stile von Schaubildern vor, die die gebildeten Subkomplexe in den Fokus stellen.

Zusammenfassend konnten wir zeigen, welches Potenzial in der Nutzung von Graphentheorie für die automatisierte Untersuchung von Proteinkomplexen steckt. Umfangreiche MD-Simulationsdaten können automatisiert auf herausragende Zeitpunkte und Regionen in der Struktur untersucht und die Ergebnisse visualisiert werden. Damit können Erkenntnisse aus den Daten gewonnen werden, die sonst verborgen blieben oder nur aufwendig erlangt würden. Die Vorhersage von strukturellen und funktionellen Modulen von Proteinkomplexen funktionierte gut für die betrachteten Beispiele und könnte benutzt werden, um Erkenntnisse über nicht untersuchte Proteinkomplexe zu erlangen. Die Vorhersage vom Assemblierungspfad von Komplex I zeigte umfangreiche Übereinstimmungen mit der Referenz. Angewandt auf weniger gut untersuchte Komplexe könnte die Vorhersage Experimente und deren Richtung motivieren, um die experimentelle Bestimmung von Assemblierungspfaden zu unterstützen. Aufgrund der auf Laufzeit optimierten Implementierungen können die Vorhersage von Modulen und des Assemblierungspfades in Sekunden bis Minuten für typische Proteinkomplexe ausgeführt werden. Auf diese Weise könnte eine Datenbank aufgebaut werden, welche die Vorhersagen für alle Komplexe der PDB enthält. So eine Datenbank könnte Forschende unterstützen, Proteinkomplexe besser zu verstehen, weiter zu erforschen und Heilmittel für Krankheiten zu finden.

Contents

List of Figures

List of Tables

Abbreviations

2D two-dimensional

3D three-dimensional

AAG Amino-Acid Graph

ATP adenosine triphosphate

BSA buried surface area

CG Complex Graph

CID clustering information distance

CPU central processing unit

cryo-EM single-particle cryo-electron microscopy with image reconstruction

CTL contact transitive by ligands

DD double distance

DNA deoxyribonucleic acid

DSSP Define Secondary Structure of Proteins

EM electron microscopy

ETE Environment for Tree Exploration

Fe-S iron-sulfur

FG Folding Graph

FMN flavin mononucleotide

LIS Large Interface Score

MCI molecule contact info

MCInf mutual clustering information

MD molecular dynamics

mmCIF marcomolecular crystallographic information file

noQ data set without ubiquinone

NADH reduced nicotinamide adenine dinucleotide

nRF normalized Robinson-Foulds

Nqo NADH-quinone oxidoreductase

OPM Orientations of Proteins in Membranes

PG Protein Graph

PDB Protein Data Bank

PPI protein-protein interaction

POPC 1-palmitoyl-2-oleoyl phosphatidylcholine

PTGL Protein Topology Graph Library

Qox data set with ubiquinone

rCI respiratory complex I

RCSB Research Collaboratory for Structural Bioinformatics

RMSD root mean square deviation

RNA ribonucleic acid

RF Robinson-Foulds

SDR short-chain dehydrogenase/reductase

SASA solvent accessible surface area

SCOP Structural Classification of Proteins

SSE secondary structure element

TIP3 transferable intermolecular potential with 3 points

VPLG Visualization of Protein Ligand Graphs

Entries of the Protein Data Bank

1ikn IκBα/NF-κB complex [1]

1jtv 17β-hydroxysteroid dehydrogenase [2]

1s5b cholera holotoxin with an A subunit [3]

1w88 pyruvate dehydrogenase E1 bound to binding domain of E2 [4]

2e9x human GINS core complex [5]

2h8o geranyltransferase from *Agrobacterium tumefaciens* (unpublished)

2hsdA 3α,20β-hydroxysteroid dehydrogenase, chain A [6]

3au1A antigen-presenting glycoprotein CD1d1, chain A [7]

3denA dihydrodipicolinate synthase, chain A [8]

3j3q capsid of human immunodeficiency virus [9]

3j3y capsid of human immunodeficiency virus (186 hexamers + 12 pentamers) [9]

3slo serine protease EspP N1023D mutant [10]

4gwp mediator head module bound to the carboxy-terminal domain of RNA polymerase II [11]

4hea respiratory complex I from *Thermus thermophilus* [12]

4oj5 tailspike protein 1 from *Escherichia coli* [13]

4ro0 MthK gating ring in a ligand-free form (unpublished)

5a22 L polymerase of vesicular stomatitis virus [14]

5im4 designed two-component self-assembling icosahedral cage [15]

5mx2 photosystem II depleted of the Mn4CaO5 cluster [16]

5nug motor domains from human cytoplasmic dynein-1 in the phi-particle conformation [17]

5ojs transcription-associated protein 1 of *Saccharomyces cerevisiae* [18]

5p4kA endothiapepsin, chain A [19]

5xtd human respiratory complex I [20]

5xth human respiratory supercomplex $I_1III_2IV_1$ [20]

5xti human respiratory megacomplex $I_2III_2IV_2$ [20]

6y11 respiratory complex I from *Thermus thermophilus* [21]

7tim triosephosphate isomerase-phosphoglycolohydroxamate complex [22]

1 Introduction

1.1 Motivation

Proteins are essential for all living organisms. Amid carbohydrates, lipids and nucleic acids, proteins are one of four major classes of biological macromolecules. Proteins fulfill a plethora of different tasks. For example, membrane proteins act as dynamic transporters through membranes, structural proteins give stability and enzymes catalyze chemical reactions. Proteins are able to work as microbiological machines composed of rigid as well as flexible parts.

Our understanding of the world of proteins is founded on many decades of research. Researchers have discovered the existence of [23] amino acids, amino acids as building blocks [24] and the process of folding of proteins [25]. More discoveries have been the flexibility of proteins, intrinsically disordered proteins [26] or the dynamic role of membrane proteins following the fluid mosaic model [27]. To understand the function of a protein, its sequence of amino acids and its structure is of utmost importance.

Proteins most often fulfill their function by binding to other proteins forming complexes. For a protein complex, the composition of single proteins and in which order they assemble is important. Misassembly can cause problems for cells [28]. The assembly of proteins has been investigated for a long time, for example, for simple oligomeric enzymes [29] and oxygen-related proteins [30]. Protein complexes are often organized in functional and structural modules.

The structure of protein complexes can be resolved experimentally or predicted computationally. The Protein Data Bank (PDB) contains more than 190,000 experimental and 1,000,000 computational structures (October 2022). Structures of more than 99,999 atoms or 62 subunits (*large structures*) cannot be saved in the legacy file format. Because of this, many bioinformatic tools and databases cannot handle *large structures*. Aside from the file format, *large structures* pose a special challenge for computational tools, because of long runtimes and the need for large memory. But computational tools are necessary to visualize, classify and analyze proteins, because the number and size of resolved proteins increases steadily.

In this work, we focus on protein complexes. We investigate the structure and interactions of the single subunits of a complex. We apply graph theory to abstract protein structures to a format that is machine-readable and for which plenty of methods exist for the analysis. We particularly focus on protein complexes of more than 15 subunits up to *large structures*.

The main use case of our approaches is respiratory complex I (rCI) composed of 45 subunits in the case of *Homo sapiens*. rCI is essential for cells, because it is involved in the electron transport chain and proton pumping to provide the cell with adenosine triphosphate (ATP) [31]. Many diseases are related to misassembly or malfunction of rCI [32, 32]. The respiratory chain has been investigated for a long time [33] and is still an important point of research.

We investigated three topics applying graph theory to protein complexes. We analyzed the movement of residues and changes of interactions during the time course of a simulation. Next, we identified modules within protein complexes that allow inferring structural and functional units. Lastly, we predicted the assembly pathway of proteins to generate hypotheses that can guide experiments.

1.2 Structure of proteins

Structures of proteins are the input for our work, and our work aims to shed light on protein structures. Because of this, we shortly introduce general aspects of protein structures. The aspects presented here are basic knowledge for understanding this work. Directly applied material and methods will be described later (see Section 2). The following subsection can be read, for example, in Branden and Tooze [34].

Amino acids are the building blocks of proteins. An amino acid contains a central carbon (C_α). An amino (NH_2) group, a carboxyl group (COOH), a side chain and a hydrogen are bound to the C_α. There are 20 primary amino acids in nature that differ in their side chain.

The bond between the amino group of one and the carboxyl group of another amino acid under elimination of water is called peptide bond. Amino acids joined by peptide bonds are called a protein or polypeptide chain. The amino group at one end of a protein chain is called N-terminus and the carboxyl group at the other end C-terminus.

Four levels of abstraction of protein structure are differentiated. The primary structure describes the sequence of amino acids. The secondary structure describes local, regular arrangements of amino acids that form patterns. Commonly defined are two secondary structure elements (SSEs): helices and strands. The tertiary structure describes the atom coordinates of a protein chain and ligands in three-dimensional (3D) space. The quaternary structure describes the formation of complexes consisting of multiple protein chains.

1.3 State of the art

In the last thirty years, many bioinformatic tools have been developed. Notable ones include CATH [35] and Structural Classification of Proteins (SCOP) which can be devided in the databases SCOP2 [36, 37] and SCOPe [38, 39]. CATH and SCOP define domains of protein chains and classify these domains. The classification groups structurally similar domains into the same class. Both databases aim to include and infer evolutionary relationships between domains.

TOPS [40] provides diagrams of the topology of SSEs and their analysis. ProSMoS [41] abstracts the topology of SSEs to matrices containing the information of contacts between SSEs. Pro-origami [42] draws editable structure cartoons comprising SSEs. The tools mentioned above work on the level of SSEs and provide searches for proteins similar in structure and function.

InterEvol [43] abstracts multimeric proteins to graphs and focuses on the interface of protein-protein interactions (PPIs). CORUM [44] is a manually curated database of mammalian complexes and relies on IntAct [45] for information about PPIs. The team of IntAct also developed Complex Portal [46] which is a manually curated database of protein and nucleic acid complexes. PCDq [47] is a database of human complexes that have been predicted from PPI network data. STRING [48] is a database of PPIs that are either physical or functional. PCFamily [49] visualizes and analyzes complexes graph-theoretically. The above mentioned tools visualize, abstract and collect information about proteins on the level of protein chains.

The approach of 3D Complex [50] is similar to the approach we present in this work. 3D Complex abstracts the topology of protein complexes to graphs where each vertex corresponds to a protein and an edge to an interface. An interface is defined by the number of residue contacts. A residue contact is assigned if the

residues are spatially close to each other. Edges are weighted with the average number of amino acids contributing to an interface. The graphs are used for the comparison of the topology and for the hierarchical classification of the complexes. 3D Complex explicitly omits complexes consisting of more than 62 subunits due to the new marcomolecular crystallographic information file (mmCIF) format they are saved in and due to the high computational costs. 3D Complex uses information of SCOP 1.69 and, consequently, can only cover proteins that are present in this release of SCOP.

Powerful tools exist to visualize and analyze molecular dynamics (MD) simulation data, such as VMD [51]. DynDom [52] determines domains and movement of domains for two conformations of the same structure which could be applied to MD simulation data. Bougueroua et al. [53] have presented a method that graph-theoretically analyzes MD simulation data to show transitions of states. Bougueroua et al. have applied their method to gases and liquids. We do not know of any application analyzing MD simulation data of protein complexes graph-theoretically.

Different clustering techniques have been applied to structural data. A typical approach is graph clustering of PPI networks to identify protein complexes. This is done by many tools, such as PEWCC [54], ClusterViz [55], ClusterONE [56] or MCODE [57]. SCODE [58] uses supervised learning with features specified in a Bayesian network structure to identify protein complexes in PPI networks. We do not know of any approach that applies graph clustering to graphs of topology on the level of chains to identify functional modules.

The computational prediction of assembly pathways is largely unexplored. To our knowledge, only two computational methods have been published. Similar to our approach, Levy et al. [59] and Marsh et al. [60] have used the contact definition of 3D Complex [50] to infer an order of assembly from large to small interfaces of the final structure. The method of Levy et al. and Marsh et al. is not publicly available and has not been applied to larger protein complexes, such as rCI. Path-LZerD [61] applies multi-docking to single protein chains to predict the structure of a complex and predict an assembly pathway along the way. Path-LZerD has not been applied to structures of more than seven subunits, such as rCI.

1.4 Aim and structure of this thesis

This work aims to investigate protein complexes by applying graph-theoretic methods. The work is structured into the sections: Introduction, Material and methods, Results and discussion, and Conclusion and outlook. Appendix and Bibliography are provided after that.

In section 1, Introduction, we lay out the motivation of the work embedding it in a general scientific context. We briefly introduce the structure of proteins. We present the state of the scientific discourse embedding the work in the specific scientific context.

In Section 2, Material and methods, we present the material and methods used. We give an overview of graph-theoretic methods, such as graph clustering, dendrograms and agglomerative clustering. We introduce biological methods, for example, for the determination of protein structures and assembly pathways, and the respiratory chain. We introduce two important databases, the PDB and DSSP, as well as software tools. Next, we explain the Protein Topology Graph Library (PTGL) in detail covering the definitions of contacts, different graphs, the user interface and

implementation. Lastly, we present the data sets and use cases we worked with, which are mainly complexes of the respiratory chain.

Section 3, Results and discussion, is the main part of this thesis. We present new implementations and discuss their runtime improvements in section 3.1. In section 3.2, we present the analysis of data from MD simulations. In Section 3.3, we introduce different types of normalizations of edge weights. We present the prediction of modules by graph clustering of graphs of topology in section 3.4. Lastly, we discuss the results of predicting the assembly pathway by agglomerative hierarchical clustering of topology in section 3.5.

In section 4, we conclude the work and give a perspective for further research. In the appendix, we present additional figures and tables. In the bibliography, we provide all citations of this work.

2 Material and methods

2.1 Graph Theory

2.1.1 Introduction

This sub-subsection is based on Newman [62].

A graph consists of vertices and edges, which join vertices. A graph with at most one edge between the same two vertices, and without edges connecting a vertex to itself is called simple graph. In an undirected graph, edges possess no direction differentiating between the connected vertices. In a weighted graph, each edge is assigned a weight. Consequently, a simple, undirected, weighted graph $G = (V, E, c)$ is a collection of a set of vertices, V, a set of edges, E, and a function c. The elements of E are sets of two elements of V each. The function c assigns a number to each element of V.

A path is a sequence of vertices connected by edges. A loop is a path, in which the start vertex and the end vertex are identical. A graph is connected if each vertex can be reached by every other vertex via a path.

A tree is a connected, undirected graph without loops. A rooted tree contains a vertex that is designated as root vertex. An hierarchy can be imposed from the root to the other vertices. Because of the hierarchy, vertices can be designated as parents and children in the means that of two connected vertices the one with a lower distance to the root is the parent and the other is the child. In a binary rooted tree, there is no vertex with more than two children. Vertices without children are called leaves. Vertices that are neither the root nor leaves are called inner vertices. A rooted tree is terminally labelled if all leaves are assigned a label. In an unordered tree, there is no order imposed among inner vertices with the same distance to the root.

2.1.2 Graph clustering

This sub-subsection about graph clustering is based on Fortunato [63].

Overview

Graphs contain entities of a system as vertices and relationships among them as edges. A general aspect of graph theory is finding clusters of vertices that are similar to each other. In the case of an unweighted graph, these clusters can only be identified by their topology. For weighted graphs, the weight can be interpreted as measure of similarity or distance.

No final, accepted definition of a cluster exists. In general, vertices within a cluster should be more densely connected than vertices between clusters. For weighted graphs, this extends to higher edge weights within clusters than between clusters. In this work, we use the term cluster which may also be called community or module in the literature.

In a partition of a graph, each vertex is assigned to exactly one cluster. A cover, on the contrary, allows vertices to be part of multiple clusters.

Modularity

Modularity [64] is a quality function for the assessment of partitions of a graph. It is derived from the idea that a random graph does not contain clusters. The partition

is compared to a null model of a randomly rewired graph such that chosen graph properties stay the same but no clusters can be expected. In a typical null model, the degree sequence of the rewired graph is the same as for the original graph. If the modularity is positive, the partition is considered good.

The modularity Q can be defined as the sum of the modularities of each cluster of the partition as

$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right] \tag{1}$$

where the sum iterates over the clusters, n_c is the number of clusters, l_c the number of edges within a cluster, d_c the sum of degrees of the vertices of a cluster and m the number of edges in the graph.

Similarily, for weighted graphs, the weighted modularity Q_w can be defined as

$$Q_w = \sum_{c=1}^{n_c} \left[\frac{W_c}{W} - \left(\frac{S_c}{2W} \right)^2 \right] \tag{2}$$

where W is the sum of all weights, W_c the sum of all weights within a cluster and S_c the sum of weights of the vertices of a cluster.

Louvain and Leiden algorithm

The Louvain algorithm [65] can be used for weighted graphs. The algorithm greedily searches a partition optimizing the weighted modularity. At the start, all vertices represent their own clusters. For each vertex, the gain in modularity is computed by placing it in the cluster of each neighbor. The vertex is placed in the cluster of the neighbor of highest increase of modularity if the increase is positive.

Each cluster is merged into a single vertex (super-vertex). Super-vertices are connected by an edge if any of the vertices of the corresponding clusters were connected. The edge weights are the sums of the initial edges. These two steps are repeated producing a partition in hierarchical steps until no gain in modularity can be achieved anymore.

In the first step, modularity is greedily optimized locally. By creation of super-vertices, modularity is greedily optimized globally. Compared to other algorithms, the Louvain algorithm has been reported to achieve partitions with higher modularity.

Traag et al. [66] have noted that the Louvain algorithm may yield internally poorly connected clusters or even clusters that are internally disconnected. The Leiden algorithm is an extension to the Louvain algorithm fixing this problem among other advantages such as better performance and better partitions. Among other adjustments, the biggest difference is the introduction of a new phase between local movement of the vertices and creating super-vertices. In this refinement phase, each cluster is split in single vertices and clustered again. This may create multiple clusters where previously only one cluster was created and ensures well-connected clusters. [66]

2.1.3 Dendrograms and Newick format

In this work, we refer to a rooted, binary, unordered, terminally-labeled tree as dendrogram (see Section 2.1.1). In case of trees where one vertex has more than two children, we refer to it as a non-binary dendrogram. We represent dendrograms in

the Newick format[1]. We use one of the most basic versions which is self-explanatory: Leaves are represented by an identifier and grouped together in brackets. For example, "((A,B),C);" stands for a dendrogram with an inner vertex with children A and B (see Figure 1). When used in the text, we omit the final semicolon.

Figure 1: Visualization of an example dendrogram. Leaves are labeled. The only possible split is indicated with an orange line.

A split is a bipartition of the set of leaves according to the structure of the tree which can be thought of as cutting the dendrogram at an edge. For a dendrogram, each inner vertex corresponds to a split. The leaves under the inner vertex belong to one set of the split and all the other leaves belong to the other set. For the dendrogram above (see Figure 1), the only split is: (A,B)|C.

Staircase and modular topology

We informally define a topology of a dendrogram as staircase if only one inner vertex exists whose two children are leaves. This means that all other inner vertices have exactly one inner vertex and one leaf as children. A staircase-like topology is when a sub-dendrogram of the dendrogram but not the whole dendrogram exhibits a staircase topology. The opposite of the staircase topology is a modular topology, where multiple inner vertices exist whose children are leaves, and these inner vertices do not exhibit a staircase topology if they were treated as leaves. We give no exact definition and instead use these labels to describe features of a dendrogram's topology which can be seen visually with an appropriate layout.

Number of different dendrograms

For the number BT_n of possible dendrograms with n distinct leaves, we refer to Lapointe and Legendre [67]. Please note that Lapointe and Legendre use the term dendrogram in different way than us. In their sense, what we call dendrogram in this work is an unweighted binary tree hence the abbreviation "BT". Lapointe and Legendre refer to Felsenstein [68] and Phipps [69] giving the formula $BT_n = (2n-3)!/2^{n-2}(n-2)!$ for the number of possible dendrograms of n distinct leaves.

Comparison of dendrograms

Different approaches can be used to compare dendrograms quantitatively. The goal is assigning a score, similarity or distance to a pair of dendrograms.

The Robinson-Foulds (RF) distance [70] is a classic metric for the comparison of dendrograms. The RF distance is the symmetric difference between the splits of both

[1] https://evolution.genetics.washington.edu/phylip/newicktree.html

trees. The normalized Robinson-Foulds (nRF) distance is obtained by normalizing the RF distance to a value between zero and one, where zero means that the dendrograms are equal.

Smith [71] has introduced three generalized RF metrics. Smith calls two splits, one of each dendrogram, a pairing. While the RF distance only assesses if a pairing is equal or not, the generalized RF metrics assign a similarity score to a split. A set of pairings, where each split of each tree occurs exactly once, is called matching. The score of a matching is the summed scores of its pairings. A matching is called optimal if there is no other matching with higher score.

The mutual clustering information (MCInf) interprets a split as a partition (see Section 2.1.2), where the sets of leaves of the split are clusters. The MCInf assigns a score to a pairing based on information theory:

$$\begin{aligned} I_{CI}(S_1; S_2) = P_{CI}(A_1, A_2) log \frac{P_{CI}(A_1, A_2)}{P_{CI}(A_1) P_{CI}(A_2)} \\ + P_{CI}(A_1, B_2) log \frac{P_{CI}(A_1, B_2)}{P_{CI}(A_1) P_{CI}(B_2)} \\ + P_{CI}(B_1, A_2) log \frac{P_{CI}(B_1, A_2)}{P_{CI}(B_1) P_{CI}(A_2)} \\ + P_{CI}(B_1, B_2) log \frac{P_{CI}(B_1, B_2)}{P_{CI}(B_1) P_{CI}(B_2)} \end{aligned} \tag{3}$$

where $S_1 = A_1|B_1$ is a split of the one dendrogram, $S_2 = A_2|B_2$ is a split of the other dendrogram and $P_{CI} = \frac{|A_1 \cap A_2|}{|X|}$. [71]

The idea is, to find how much knowing the partition inferred from one split helps correctly identifying the partition of the other split. The MCInf is defined as the score of the optimal matching using the scores of I_{CI}. The clustering information distance (CID) is obtained by subtracting the MCInf from half the entropy of all the splits in both trees, which is defined for one split as $-P_{CI}(A) log P_{CI}(A) - P_{CI}(B) log P_{CI}(B)$. The CID is a number between zero and one, where zero means that the dendrograms are equal. [71]

Smith has compared the generalized RF metrics regarding different properties of distances for comparing dendrograms. The CID has performed best among the three generalized RF metrics and other state-of-the-art methods. [71]

Generating random dendrograms

Furnas [72] has presented a method to uniformly draw a terminally labeled dendrogram from all existing dendrograms with n leaves. The method starts with an unrooted dendrogram of two vertices connected by an edge. An edge is randomly drawn from the dendrogram with uniform distribution. The edge is replaced by a new vertex v_i that is connected to the vertices to which the edge was adjacent. Next, a new leaf l_i is added to v_i. This is repeated until the dendrogram contains n leaves. The unrooted dendrogram is rooted at an randomly drawn edge with uniform distribution by replacing the edge with the root vertex and connecting it to both vertices that were adjacent to the edge.

2.1.4 Agglomerative hierarchical clustering

The following sub-subsection is based on Murtagh and Contreras [73]. We refer to agglomerative hierarchical clustering simply as agglomerative clustering.

Agglomerative clustering is a bottom-up clustering technique. At the beginning, all data points start in their own clusters (*singleton clusters*). In each step, two clusters are merged until only one cluster remains.

Before clustering, a similarity, dissimilarity or distance needs to be inferred from the attributes of the data to decide which data points should be grouped together. The choice of a similarity, dissimilarity or distance influences the outcome. Typical examples are the Manhattan distance, Euclidean distance and Chebyshev distance [74].

To decide which *singleton clusters* are merged, the similarity, dissimilarity or distance can be directly applied. To compare clusters with multiple data points, the user needs to choose a linkage method. The linkage method decides how similarity, dissimilarity or distance are computed between non-*singleton clusters*. Typical examples are single, complete, average and centroid linkage [62].

Typically, agglomerative clustering is applied greedily. In a greedy approach, only the information that is available up to the step is used to make the decision how to proceed. For agglomerative clustering, the decision is which clusters are merged at a given step. In a greedy approach, a score is optimized, such as choosing the highest edge weight. As a consequence, a greedy method may yield a solution that is not optimal. For example, a better result may be achieved when deciding non-greedily at a step enabling an overall better choice at a later step.

The output of agglomerative clustering can be interpreted as a dendrogram (see Section 2.1.3). The leaves of the dendrogram correspond to the *singleton clusters*, inner vertices to merged clusters and the root to the cluster containing all data points. Agglomerative clustering can be used to hierarchically structure the data in groups of similarity.

The dendrogram represents the hierarchy of the data points. To retrieve clusters, the dendrograms needs to be cut horizontally at a given stage. The position of the cut determines the number of clusters, which can be any number from 1 to n, where n is the number of data points. There is no straightforward method that can be applied to find the best cut, as it depends on what the user is interested in. Thus, finding a cut is a challenge when using agglomerative clustering to identify clusters. [62]

Another characteristic, to keep in mind, is that agglomerative clustering excels in grouping together similar data points, but needs to put distant data points to a cluster at some point. This means, that early merges of data points along the clustering represent trustworthy similarities, while later merges may add data points, which are only loosely connected to the group. [62]

2.2 Resolving the structure of proteins

Proteins are too small to be observed with the naked eye. Therefore, experimental techniques are required to elucidate the structure of proteins. Over the past, many different techniques have been applied. Here, we shortly present the most important ones with a special focus on large protein complexes. The following subsection is based on Miyaguchi [75].

X-ray crystallography is a technique that has been widely used to determine the

structure of molecules. Multiple copies of a molecule are crystallized in solution and frozen afterwards. The crystallization ensures that the molecules have the same orientation.

The crystal is exposed to an X-ray beam. The electrons of the molecules scatter the beam producing a diffraction pattern. The crystal is rotated to observe the diffraction pattern from all angles. As a last step, an electron-density map is built computationally based on the diffraction patterns. Based on the electron-density map, the structure is modeled.

The need for a crystal is one of the biggest limitations of X-ray crystallography. Some molecules may adopt unnatural conformations during crystallization or do not crystallize at all. An advantage of X-ray crystallography is that the technique is able to produce high-resolution structures and resolve structures of any sizes.

Electron microscopy (EM) is being used in different techniques, such as transmission EM with negative staining, rotary shadowing or freeze-etching. For high-resolution structure determination of large complexes mostly single-particle cryo-electron microscopy with image reconstruction (cryo-EM) is relevant. Multiple copies of the molecule in solution are put into a frozen-hydrated state. In contrast to X-ray crystallography, the molecules are in arbitrary orientation. The molecules in this context called particles are photographed using EM. All the two-dimensional (2D) images of the particles in different orientations are merged into a 3D structure.

A disadvantage of cryo-EM is the 'initial model problem' when no model for the structure is present which can be used to determine the orientation of the particles. However, cryo-EM can be used where crystallization is not possible. Cryo-EM is able to achieve near-atomic resolution and excels especially for large complexes.

2.3 Molecular dynamics simulation

The following subsection is based on a review by González [76].

MD simulation is a computational method to analyze the dynamics of an atomic system. The input is an atomic system where a position and velocity is assigned to each atom. The user needs to set system parameters, such as temperature, pressure and box size, and choose a force field when not applying *ab initio* MD simulation. The output are the trajectories of the atoms over the simulated time.

In the context of structural biology, usually a resolved protein structure is taken as input for MD simulation. Because the protein structure is the result of an experiment for structure determination, the system may be in a different configuration than desired for the simulation. For example, the structure might be crystallized for X-ray crystallography (see Section 2.2). The system is energetically minimized first and equilibrated before performing the simulation.

During simulation, the next time step is computed by solving Newton's equation of motion. The equation is solved numerically, because exact solutions cannot be achieved. Typically, the used force fields incorporate bond lengths, angle bending, dihedral torsions, improper torsions, electrostatic interactions and Van der Waals interactions.

MD simulations can be thought of as treating atoms as connected by springs. The springs allow to be stretched, which increases the energy. Atoms can be attracted to each other, but are repelled if they come too close. Based on these considerations of energy, the position and the velocity, the trajectory of an atom is computed.

Classical force fields do not allow breaking bonds between atoms. In MD simulations, neither electrons nor transfers of charges are modelled. Therefore, MD

simulations cannot simulate chemical reactions. Parameters are mostly based on empirical observations and differential equations are solved numerically. Therefore, MD simulations are approximations and require experimental validation.

The computation of MD simulations is far from trivial and demands software solutions. Software to perform MD simulation is, for example, CHARMM [77], NAMD [78] and Gromacs [79]. A software to analyze MD simulations is, for example, VMD [51].

Despite these limitations, MD simulations enable observing dynamics, for example, for protein complexes. With the development of runtime-saving methods and modern hardware, simulations of hundreds of thousand atoms over a timescale of microseconds are possible.

2.4 Assembly pathway

Most proteins function together with other proteins forming complexes. Complexes built by multiple copies of the same subunit are called homomers and complexes of different subunits are called heteromers. For a complex, an assembly pathway describes the order in which the subunits assemble for all subunits of the complex. It has been proposed that complexes assemble along an ordered pathway. If the pathway was unordered, testing all possible pathways until the right one occurs by chance would take too much time analogous to Levinthal's paradox of protein folding [80]. There does not necessarily need to be exactly one pathway, but it has been proposed that energetically favorable intermediates exist that ensure fast assembly and reduce the risk of misassembly. Levy et al. have shown that assembly is linked to evolution of protein complexes [59]. As a consequence, if pathways are conserved it seems to be more likely that they are ordered. [60]

An assembly pathway can be represented as dendrogram (see Section 2.1.3). The root corresponds to the final assembly product, inner vertices to subcomplexes and leaves to subunits. The order of assembly is inferred from the leaves to the root.

Experimental determination

Many techniques have been developed to elucidate the assembly of protein complexes. We just name a few and refer to the respective articles for details. Usually, multiple experimental techniques are combined to infer the assembly pathway.

Co-immunoprecipitation (Co-IP) can be used to find stable intermediates [81]. Electrospray mass spectrometry (MS) allows a detailed analysis of the assembly order [82]. Recently developed, complexome profiling identifies intermediates and assembly factors in a bottom-up approach [83, 84].

Computational prediction

Levy et al. [59] and Marsh et al. [60] have predicted the assembly pathway for homomers and heteromers, respectively, based on the number of contacts between subunits of the final complex. Both approaches have used the definition of contacts of the database 3D Complex [50]. They inferred the order of assembly from the highest to lowest number of contacts.

Peterson et al. [61] have adopted the method by Levy et al. [59] and Marsh et al. [60]. Instead of the number of contacts, Peterson et al. have used the buried surface area (BSA). For two subcomplexes, the BSA is computed as the sum of the solvent

accessible surface area (SASA) of the unbound subcomplexes minus the SASA of the bound subcomplexes. Effectively, the BSA is the area of the interface between subunits of the final complex similar to using contacts between subunits of the final structure. The SASA has been computed with Naccess by Hubbard and Thornton[2] [85].

Peterson et al. have applied two different approaches of inferring the assembly order from the BSAs. In the first approach which they termed pairwise BSA, they compute the BSA between all pairs of subunits. In the second approach which they termed subcomplex BSA, they compute the BSA between all possible subcomplexes of two to n subunits where n is the number of all subunits. For subcomplex BSA, multiple subunits or subcomplexes can assemble at one step compared to pairwise BSA where exactly two subunits or subcomplexes assemble at a step. For both approaches, they infer the order of assembly from highest to lowest BSA.

Peterson et al. have developed a new method called Path-LZerD[3]. Path-LZerD performs multi-docking of the subunits predicting the subunits in a complex. Along the multi-docking, Path-LZerD predicts the assembly order. Path-LZerD comes with different parameters for selecting a strategy and scoring function which have been tested by Peterson et al. Because the input of Path-LZerD are the structures of the single subunits, Path-LZerD can be used to predict an assembly pathway where the structure of the final complex is unknown.

According to Peterson et al., computational methods for the prediction of an assembly pathway can be either classified as blind or non-blind. Blind methods do not need the structure of the final assembly product as input for the prediction while non-blind methods do. The method by Levy et al., Marsh et al. and the method using the BSA by Peterson et al. are considered non-blind. For Path-LZerD, there are non-blind as well as blind strategies.

2.5 Respiratory chain

This entire subsection about the respiratory chain is based on Vercellino and Sazanov [31] if not cited differently.

2.5.1 Overview

The respiratory chain refers to the protein complexes and electron transporters taking part in the electron transport chain of cellular respiration. All aerobic organisms making use of cellular respiration have some sort of the respiratory chain. Variations of the respiratory chain and the single protein complexes exist, but core principles and proteins are conserved across species.

The main principle is the chain-like transport of electrons via reduction and oxidation to the final acceptor oxygen. During transport of electrons, a proton gradient is built. The proton gradient is used to produce ATP as energy equivalent to be used by the cell.

In eukaryotes, mitochondrial oxidative phosphorylation comprises four protein complexes located at the inner mitochondrial membrane that participate in the electron transport chain: NADH:ubiquinone oxidoreductase, succinate dehydrogenase, cytochrome bc_1 oxidoreductase and cytochrome c oxidase. In this thesis, following the literature, we will refer to them as (respiratory) complexes I to IV.

[2] http://bioinf.manchester.ac.uk/naccess
[3] https://kiharalab.org/proteindocking/pathlzerd.php

The electrons stem from reduced nicotinamide adenine dinucleotide (NADH) and succinate which are oxidized by complex I and complex II, respectively. Complexes I, III and IV build a proton gradient by different mechanisms. ATP synthase uses this proton gradient to produce ATP and is often referred to as complex V.

For the respiratory chain, it is notable, that one part of each complex is encoded in the mitochondiral deoxyribonucleic acid (DNA) while the other part is encoded in the nuclear DNA. Because of the different places of translation, the complexity and built-in ligands of some complexes, many assisting assembly factors are required for a correct assembly. Furthermore, it has been discovered that the complexes assemble to larger complexes, for example, I_1III_2, III_2IV_1, $I_1III_2IV_1$. Please note that numbers in subscripts stand for the number of copies of a complex.

2.5.2 Complex I

Complex I is the first complex of the respiratory chain. The complex is shaped like an "L" with a membrane arm and a matrix arm. In the matrix arm, NADH is oxidized and quinone reduced. In the membrane arm, four protons are pumped across the membrane. Complex I is the largest complex of the respiratory chain.

Complex I can be divided into four modules: the N and Q module of the matrix arm as well as the P_P and P_D module of the membrane arm. The subunits A12 and S6 are shared between the N and Q module. [86]

2.6 Databases

2.6.1 Protein Data Bank

The PDB is an archive of structural data of large biological molecules that has been first announced in 1971 [87]. The PDB is organized by the worldwide PDB organization[4] [88]. The Research Collaboratory for Structural Bioinformatics (RCSB) PDB[5] is one member of the worldwide PDB [89].

The PDB is the one open-source database for the deposition of structural data of biological macromolecules. The PDB provides tools for validation, search and analysis of structures. Currently, the PDB holds 191.328 biological macromolecules (June 14, 2022).

Each structure uploaded to the PDB is assigned a PDB ID. A PDB ID consists of one number and three alphanumeric characters. In this thesis, we refer to protein structures from the PDB by their name the first and by their PDB ID the following times. A PDB ID followed by one or multiple capital letters refers to the chain of this structure.

Since its announcement in 1971, the PDB used a file format which today is called legacy PDB format. The legacy PDB format is based on punch cards meaning that for each column it is defined which data the column holds. This limits the data, a legacy PDB file can hold. For historic reasons, a legacy PDB file can contain structures up to 99,999 atoms or 62 chains. Structures that contain more than either 99,999 atoms or 62 chains are termed *large structures*. [90, 91]

Because of its limitations, the legacy PDB format has been superseded by the mmCIF format in 2007. In 2012, the development of the legacy PDB format has been ended. Since July, 2019, the PDB only accepts depositions in mmCIF. [91]

[4] https://wwpdb.org
[5] https://rcsb.org/

The mmCIF format is theoretically able to hold structures of any size. mmCIFs are self-organizing meaning they contain meta information describing the file itself. The format can easily be extended allowing adoptions to new experimental features and techniques. [92]

For deposited structures, the asymmetric unit and at least one biological assembly is available. For some structures, asymmetric unit and biological assembly are different from each other, and it depends on the motivation which to use. The asymmetric unit is the smallest portion of a crystal from which the whole crystal can be constructed by symmetry operations, such as rotations and translations. The authors of the PDB entry derive the biological assembly by applying symmetric operations, duplications or deletions to the asymmetric unit in order to produce the complex that is believed to be biologically functional. This means that the biological assembly may contain less, the same number of or more subunits than the asymmetric unit.[6]

2.6.2 DSSP

The original computer program for assigning SSEs to residues has been called Define Secondary Structure of Proteins (DSSP) [93]. DSSP has been re-implemented [94].

DSSP takes a PDB structure as input and assigns an SSE type to each residue based on H-bond energy of placed hydrogens of the backbone. DSSP version 3 is not able to read mmCIFs. We used DSSP version 4.0.0[7] if not mentioned otherwise. We used the assignment of SSEs by DSSP for visualizations of structures in cartoon style if not mentioned otherwise.

2.7 Software tools

Git

Git[8] is a version control system and GitHub[9] a hosting site for version control [95]. Any *commit* to the system can be uniquely identified by its hash code (also called SHA) helping in referencing the state of a software at a given time.

PyMOL

All visualizations of 3D structures in this work are created with PyMOL[10] [96]. The assignments of SSEs is done by DSSP (see Section 2.6.2) if not marked otherwise.

igraph

igraph[11] is a software package for the analysis, manipulation and visualization of graphs. We used *igraph* version 0.9.11 for Python for the computation of the modularity (see Section 2.1.2). [97]

[6] https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/biological-assemblies
[7] https://github.com/PDB-REDO/dssp
[8] https://git-scm.com/
[9] https://github.com/
[10] https://pymol.org/
[11] https://igraph.org/

Leiden algorithm

We used the Python package *leidenalg*[12] for applying the Leiden algorithm (see Section 2.1.2). *leidenalg* is based on a C++ implementation and uses the Python package *igraph*.

Environment for Tree Exploration (ETE)

Environment for Tree Exploration (ETE) [98] is a software for the visualization, analysis and manipulation of trees. ETE is mostly used for phylogenetic trees, but can be applied to dendrograms representing assembly pathways as we do in this work. We used Python's ete3[13] version 3.1.2.

TreeDist

TreeDist[14] [71] is an R package for the computation of metrics such as the CID (see Section 2.1.3) comparing the similarity of trees. TreeDist is mostly used for phylogenetic trees, but can be applied to dendrograms representing assembly pathways as we do in this work.

2.8 Protein Topology Graph Library

2.8.1 Overview

The PTGL is a database of graphs of topologies of proteins and a web server for their analysis. The underlying methods of the PTGL have been developed decades ago [99, 100, 101, 102]. PTGL has first been implemented by May [103, 104] and has been published later [105, 106]. PTGL has been re-implemented by Schäfer [107] and Schäfer et al. [108].

The database and web server are available online[15]. The website offers a search engine for finding the desired graphs or, for example, finding structures similar to a given pattern. The graphs are presented as images and can be downloaded.

The software PTGLtools does all the computations generating the graphs and is publicly available at Github[16]. The main part is PTGLgraphComputation (formerly labeled Visualization of Protein Ligand Graphs (VPLG)) which computes the graphs of topology. PTGLgraphComputation is an object-oriented Java program.

2.8.2 Definition of contacts

Atoms

The positions of the atoms in 3D space is read from a PDB file. Each atom is assigned a sphere. Atoms of proteins and ribonucleic acids (RNAs) are assigned a sphere of 2 Å radius. Atoms of ligands are assigned a sphere of 3 Å radius. An atom contact is defined if two atom spheres overlap. For a contact between two atoms of proteins, the contact can be either backbone–backbone, backbone–sidechain (or vice versa) or sidechain–sidechain.

[12] https://github.com/vtraag/leidenalg
[13] https://pypi.org/project/ete3/
[14] https://ms609.github.io/TreeDist/
[15] https://ptgl.uni-frankfurt.de/
[16] https://github.com/MolBIFFM/PTGLtools

Molecules

Residues, ligands and nucleotides of RNA are treated as *molecules*. A molecule contact is defined if two molecules share an atom contact.

Secondary structure elements

The assignment of residues to SSEs is based on DSSP (see Section 2.6.2) with a few variations. All types of helices are treated as helix regardless of their type. Extended strands and adjacent isolated beta bridges are treated as strand. SSEs are merged if there is one residue between them with a different assignment of secondary structure. SSEs shorter than three residues are neglected.

An SSE contact is defined if two SSEs share contacts of molecules based on the types of the SSEs according to a rule set (see Table 1). An SSE contact is assigned an orientation inferred from the N- to C-terminus: parallel, antiparallel or mixed (see Section 2.8.5).

Table 1: Rule set for the definition of a contact between secondary structure elements (SSEs). For a contact, at least the noted number of contacts of at least one type of atom contacts needs to be present. For some contact types, no rule is defined (-).

		No. of contacts per type of atom contact		
SSE 1	SSE 2	Backbone–backbone	Backbone–sidechain	Sidechain–sidechain
Helix	Helix	-	4	4
Helix	Strand	2	4	4
Strand	Strand	2	3	-

Chains

Polypeptide or RNA chains are identified by a chain ID assigned by the authors of the PDB file. Please note that we use the term chain in the context of implementation or to differentiate specific chains of a PDB file and the term subunit in a biological context or when we generally refer to a protein of a complex which might contain multiple copies of a subunit. A chain contact is defined if two chains share a molecule contact. A chain contact is assigned the number of contacts of molecules.

2.8.3 Graphs of topology

Amino-Acid Graph

We explain Amino-Acid Graphs (AAGs) only shortly, because AAGs are not used in this work. We define an AAG where vertices are amino acids of a chain and edges are spatial contacts between residues (see Section 2.8.2). AAGs apply the most fine-grained level of abstraction, because treating atoms is no abstraction anymore.

Protein and Folding Graph

We define a Protein Graph (PG) where vertices are SSEs and edges are contacts between SSEs (see Section 2.8.2). Each vertex is labeled with the type of SSE: helix, strand or ligand. Each edge is labeled with the orientation between the SSEs:

parallel, antiparallel or mixed. We define six types of PGs that differ in the labeled vertices they include (see Table 2).

Table 2: Overview of graph types of Protein Graphs. Each graph type includes (✓) or excludes (✗) helices, strands and ligands.

Graph type	Vertices include		
	Helix	Strand	Ligand
Alpha	✓	✗	✗
Beta	✗	✓	✗
Albe	✓	✓	✗
Alphalig	✓	✗	✓
Betalig	✗	✓	✓
Albelig	✓	✓	✓

A PG shows the topology of one protein chain. A complex-level PG contains the vertices of all chains of a complex. This is more than just collecting all SSEs of a complex in one graph, because the complex-level PG can contain edges between SSEs of different chains that are not visible in single PGs.

We define a Folding Graph (FG) as a connected component of a PG. This can be the whole PG or as little as a single vertex. An FG can be completely described by a linear notation. A linear notation is a string enumerating all edges of an FG. There are four types of notation types: sequence (SEQ), adjacent (ADJ), reduced (RED) and KEY.

For this work, only the graph visualization of the KEY notation is of relevance. An FG in KEY notation visualizes the topology of the SSEs in a 2D scheme that draws SSEs next to each other if they are structural neighbors. Strands are drawn as arrows and helices as boxes. SSEs are connected by a line from N- to C-terminus. Because of this, a crossing over where two neighboring SSEs have a parallel orientation becomes visible. An example can be seen for 3α,20β-hydroxysteroid dehydrogenase, chain A [6] (2hsdA) (see Figure 4d).

Complex Graph

We define a Complex Graph (CG) where vertices are protein or RNA chains and respectively labeled. Edges are introduced for chain contacts (see Section 2.8.2) and weighted by the number of molecule contacts. Each chain is assigned a molecule ID (ML in the graph visualization). Homologous chains are assigned the same molecule ID.

2.8.4 User interface

On its first execution, PTGLgraphComputation writes a file `PTGLgraphComputation_settings.txt` to the users home directory. The setting's file contains lines of key-value pairs in the format `<key>=<value>`. For example, `PTGLgraphComputation_B_use_mmCIF_parser` is the setting to invoke the mmCIF parser (see Section 2.6.1). All values are initialized as the default values. The user can change a setting by changing the value in the setting's file. Settings by the user will always overwrite the default values during execution of the program.

PTGLgraphComputation is executed from the command line. The user can pass command line arguments along the program call. There is one positional argument, the PDB ID (see Section 2.6.1), and many optional arguments. Optional arguments will always overwrite the default values or values from the setting's file.

2.8.5 Implementation

Computation of contacts

In theory, the distance between each possible pair of atoms needs to be checked to find all contacts of atoms. For N atoms, these are $N * (N-1)/2$ checks. For the largest protein structure of the PDB, capsid of human immunodeficiency virus [9] (3j3q) with 2,440,800 atoms, this means 2,978,751,099,600 checks. Technically, pairs of atoms, where both atoms are part of the same molecule, are not of interest for this question. Nevertheless, regarding the runtime, checking all possible pairs of atoms is not feasible. PTGLgraphComputation applies two methods to skip checking atom pairs that cannot be in contact. The methods are shortly explained here based on Kaden et al. [99] as well as Schäfer and Koch [109]. Both methods speed up the computation by skipping unnecessary checks, but reliably find all residue contacts.

Check of the spheres of molecules A sphere is assigned to each molecule. The center of the sphere is the C_α for residues and the atom with the minimum distance to any other atom for ligands and RNA nucleotides. The radius of the sphere is the maximum distance from its center to any other atom. For protein and RNA nucleotide atoms, 2 Å and for ligand atoms 3 Å are added to account for the contact definition of atoms. As a consequence of the construction of the spheres, a contact can only exist between any pair of atoms of two molecules if the spheres of the molecules overlap. If the spheres do not overlap, the check for atom contacts is skipped for this pair of atoms.

Neighbor skipping If a pair of residues r_n and r_m has a large distance, this means that residues r_{n+1}, r_{n+2}, $r_{n+\ldots}$ sequentially following after r_n must have a large distance to r_m, too. It is possible to skip the check of the spheres of the molecules for these residues. To find the number of residues that can be skipped, we need to find the largest distance between sequential neighbors d_{max} in a pre-processing step.

Let r_n and r_m be residues and $d_{n,m}$ the minimum distance between any pair of atoms of r_n and r_m (see Figure 2). We can skip checking the spheres of the molecules between the next $\lfloor (d_{nm} - threshold - J)/d_{max} \rfloor$ residues after n or m. $threshold$ is the contact threshold and J is a small value to account for rounding errors. For PTGL, J is set to 0.2 Å. Even if the skipped residues formed a straight line to either r_n or r_m, each sequential distance could only be d_{max} at most. This means that of the skipped residues, none could have been in contact with r_m.

Orientation of secondary structure elements

Rationale The orientation of SSEs is determined by computing the *double distance (DD)*. Each residue is assigned an integer identifier incrementing from N- to C-terminus. For SSEs in contact, the identifiers of the residues in contact are compared. For each pair of residues in contact, the sum and the difference of their identifiers are computed. Over all pairs of residues in contact, the maximum sum S_{max}, the

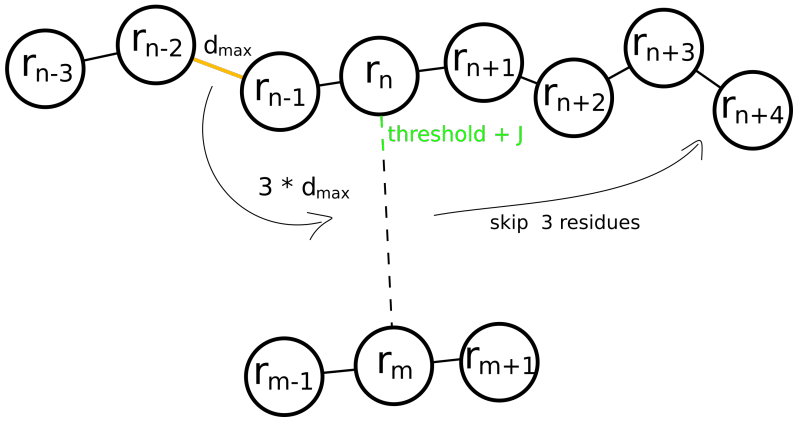


Figure 2: Schematic example of the neighbor skipping. Circles represent residues and are labeled. Peptide bonds between residues are indicated by a straight line. The largest distance between sequential neighbors is marked with an orange line and labeled d_{max}. The distance between residues r_n and r_m is indicated by a dashed line. The green part of the line accounts for the value of $threshold + J$. The arrows illustrate how the computation allows skipping the contact checks of the neighbors after r_n.

minimum sum S_{min}, the maximum difference D_{max} and the minimum difference D_{min} are determined. The DD is defined as $DD = (S_{max} - S_{min}) - (D_{max} - D_{min})$. [99, 109]

The rationale becomes clear when looking at an example. Let us consider two ideal antiparallel SSEs of same length where each residue of one SSE has exactly one contact to a residue of the other SSE (see Fig. 3). The sums of all the residue pairs are the same which results in $S_{max} - S_{min} = 0$. The differences are above zero, because the residue with the lowest identifier in one SSE is in contact with the residue with highest identifier in the other SSE. $D_{max} - D_{min}$ becomes positive resulting in a negative DD. For two ideal parallel SSEs, the $S_{max} - S_{min}$ becomes larger than zero, $D_{max} - D_{min}$ becomes zero and the DD becomes positive.

Generally, a DD of or around zero stands for a mixed, a negative DD for an antiparallel and a positive DD for a parallel orientation. Depending on the types of the involved SSEs, the orientation is assigned based on a rule set for the value of the DD (see Table. 3).

Table 3: Rule set for the assignment of orientations of secondary structure elements based on the Double Difference (DD). A mixed orientation for an interaction of two strands is never assigned (-).

	Strand–Strand	Strand–Helix	Helix–Helix
antiparallel	$DD \leq 0$	$DD \leq -6$	$DD \leq -8$
parallel	$DD \geq 1$	$DD \geq 6$	$DD \geq 8$
mixed	-	$-5 \leq DD \leq 5$	$-7 \leq DD \leq 7$

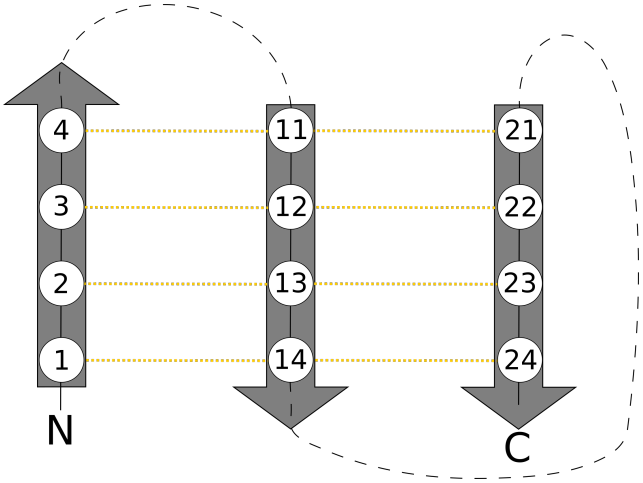


Figure 3: Visualization of an example for the computation of the Double Distance for the assignment of the orientation of secondary structure elements. Residues are depicted as white circles and labeled with a number according to the position in the sequence. A straight black line connecting residues represents peptide bonds. A dashed black line connecting residues represents multiple residues connected by peptide bonds that are not visualized. A yellow dashed line represents residues in spatial contact. The N- and C-terminus are labeled. Arrows represent residues that form a strand.

Implementation In PTGLgraphComputation, all contacts on the level of residues, ligands and RNA are saved in a list of molecule contact infos (MCIs). To compute the DD, the contacting residues of two SSEs need to be iterated through. Iterating through the whole list of all contacts for each pair of contacting SSEs is costly, so in a pre-processing step, the list of MCIs is iterated through to build up a map (see Algorithm 1). The key of the map is the number of the SSE and the value is a list of the MCIs for which one residue is part of the SSE. PTGLgraphComputation iterates through all contacting pairs of SSEs in a nested loop. For each pair of SSEs in contact, PTGLgraphComputation iterates through all MCIs of the respective SSEs. If an MCI is a contact between the pair of SSEs, PTGLgraphCompuation computes the sum and difference of the identifiers of the residues. The maximum and minimum of the sum and the difference of the identifiers of residues is updated if necessary. After iterating all MCIs, PTGLgraphComputation computes the DD.

Algorithm 1 Computation of Double Distance

```
1: iterate through all MCIs and build a map M: SseId → MCI
2: for SSE_A in all SSEs do
3:     for SSE_B in all SSEs do
4:         if contact between SSE_A and SSE_B then
5:             for MCI in M[SSE_A] or M[SSE_B] do
6:                 if MCI is contact between SSE_A and SSE_B then
7:                     compute sum and difference of residues from MCI
8:                     update sum_max, sum_min, diff_max, diff_min if necessary
9:                 end if
10:            end for
11:            DD = (sum_max - sum_min) - (diff_max - diff_min)
12:            assign an orientation based on the DD
13:        end if
14:    end for
15: end for
```

2.9 Data sets

Statistics of contacts

We compiled a data set to investigate the statistics of checked and skipped checks of contacts for different implementations of PTGLgraphComputation. The implementations are sensitive to the number of chains, number of residues and whether a ligand is part of the structure. At the time of application, PTGLgraphComputation was able to only process protein data so we confined the data set to structures that contain only proteins.

We decided to test structures of 1, 3, 28 and 40 chains. Most structures of the PDB contain one, two or three chains, so we covered these typical structures. The focus of our work are complexes that are larger, so we included structures of 28 and 40 chains. Structures of many more chains exist, but we assumed that structures containing 28 or 40 chains are representative of typical enzymes and molecular machines.

For each number of chains, we chose four arbitrary structures. Two structures contained at least one ligand and two did not. Of the two structures containing a ligand and of the two structures without ligand, one had a large number of residues and one a low number in relation to the number of chains.

We note here, that there are three structures of 28 chains that contain a ligand and only one structure of 40 chains that contains a ligand. As a consequence, there is no representation of a structure of 28 chains containing no ligand and a large number of residues and of a structure of 40 chains containing a ligand and a large number of residues. We discuss in the results, that the influence is neglectable. The list of all PDB IDs, whether it contains a ligand, the number of chains, residues and atoms is available in section 3.1.1 (see Table 7).

Runtime comparison

We compiled a data set to investigate the runtime of PTGLgraphComputation. The main criteria was to include structures representative of different levels of difficulty regarding the expected runtime. For structures, the runtime mainly depends on the number of atoms, because of the checks for contacts. Since we tested different implementations that interact with the number of chains, we applied the number of chains as a main criterion.

The data set is based on the PDB from April 18, 2019. For each number of chains for which structures are present, we included a representative structure with an average number of atoms and with maximum number of atoms. Because structures of a low number of chains are overrepresented in the PDB and therefore play an important role for running any application using data from the PDB, we included up to eight representatives for the number of chains from one to 16.

If there is only one structure present for a number of chains, we included this structure. Note that this is the case especially for large structures (see Section 2.6.1). With increasing number of chains, there are less structures available. For example, above 270 chains per structure, there were only structure available with 360, 480, 862, 1,176 and 1,356 chains.

In total, the data set contains 178 structures of 77 different numbers of chains between one and 1,356 (see Table A1).

Prediction of assembly of small protein complexes

Peterson et al. [61] have compiled a data set of 21 protein complexes of three to seven subunits and the complexes' assembly pathways. Peterson et al. have based the assembly pathways on experimental evidence, biological inference, structural inference and the model of assembly.

The pathways are represented in a string format that successively enumerates the subcomplexes. For example, "BG > BGP" means that subunits B and G assemble before P joins the subcomplex. We transformed the pathways to the Newick format (see Section 2.1.3). The Newick format reduces redundancy, because it contains each subunit exactly once instead of possibly repeating it in every subcomplex. For example, "DD' > DD'D" > ADD'D" > AA'DD'D" > AA'A"DD'D"" becomes "(((((D,D'),D"),A),A'),A")". The Newick format also shows more clearly which subunit is added in a step, which can be difficult to retrieve from a long list of subunits of a subcomplex. The full list of all structures and assembly pathways is shown in Table 14.

It is important to note that the order is uncertain or not proposed where more than two subunits or subcomplexes assemble at one step. For example, "AA'BB' > AA'BB'I" means that there is no order proposed for the first subcomplex. This means that there may be a varying number of substeps for complexes of the same number of subunits. We addressed this by using non-binary dendrograms.

2.10 Use cases

2.10.1 Short-chain dehydrogenase/reductase

Short-chain dehydrogenase/reductases (SDRs) are proteins that have been intensively analyzed already since the 70's [110]. The SDR superfamiliy includes over 46,000 deposited sequences and 300 structures in sequence databases and the PDB, respectively [111]. SDRs exhibit many different functions resulting in a subdivision into over 200 families [111]. SDRs have low sequence similarities reported as low as 15 % for some pairs of proteins [110].

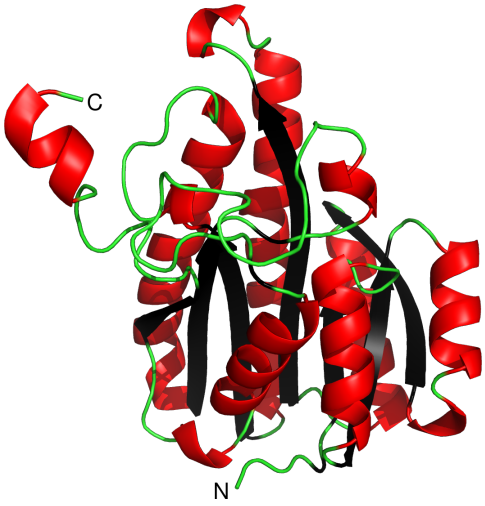
The structure of SDRs is highly conserved. The central conserved element across different classes of SDRs is a Rossmann-fold. The Rossman-fold consists of a sheet of six to seven parallel strands with the topology: 3-2-1-4-5-6-7. As a consequence, there is a large cross-over from strand three to strand four. The cross-over is done by helices which surround the sheet making the SDR fold a α/β fold by definition. [112]

An example of the classic SDR is 2hsdA. 2hsdA is a perfect example for the alternating strands and helices (see Figure 4a) as well as the strand topology (see Figure 4b). The strand topology is represented in the PG (see Figure 4c). The cross-over can easily be spotted in the KEY FG (see Figure 4d).

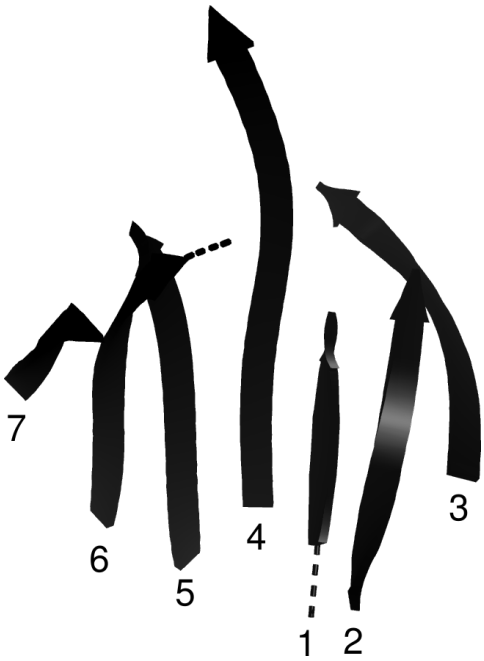
Because of the clear topology of the beta strands that is visible in both the visualization of the structure and PGs, we used SDRs as a use case.

2.10.2 Data of simulations of molecular dynamics

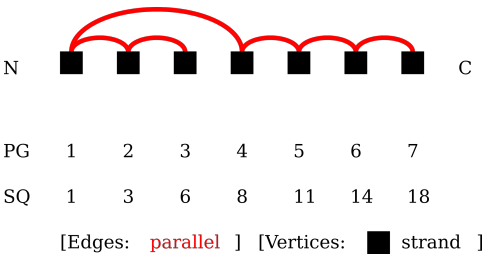
We received MD trajectories (see Section 2.3) from the group of Vivek Sharma at the Department of Physics at the University of Helsinki. Sharma et al. have simulated the dynamics of rCI of *Thermus thermophilus* over 2000 μs. They provided two data sets: one data set without ubiquinone (noQ) and one data set with ubiquinone (Qox).



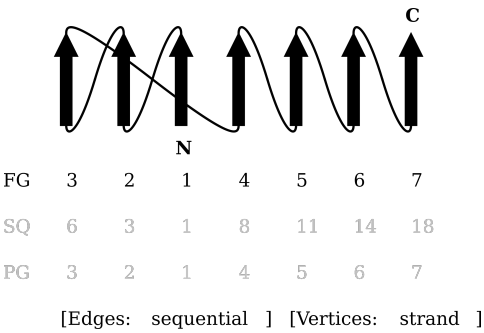
(a) Whole structure in cartoon representation. Helices are depicted as red spirals, strands as black arrows and loop regions as green ribbon.



(b) Structure of sheet in cartoon representation. Strands are depicted as black arrows. Strands are labeled by occurrence in the sequence from N- to C-terminus.



(c) Beta Protein Graph created with PTGLgraphComputation version 3.7.0. Vertices are labeled by occurrence in the Protein Graph (PG) and in the sequence (SQ).



(d) KEY beta Folding Graph created with PTGLgraphComputation version 3.7.0. Strands are labeled by occurrence in the Folding Graph (FG), Protein Graph (PG) and in the sequence (SQ).

Figure 4: Visualization of structure (a,b) and graphs of the topology (c,d) of protein 3α,20β-hydroxysteroid dehydrogenase [6] (PDB: 2hsdA). Where applicable, N- and C-terminus are labeled.

The noQ has been based on a previous MD simulation by Sharma et al. [113]. Djuranekova et al. have modeled missing parts of respiratory complex I from *Thermus thermophilus* [12] (4hea) (see Section 2.10.3) using MODELLER [114] and placed the system in a 1-palmitoyl-2-oleoyl phosphatidylcholine (POPC) membrane using the Orientations of Proteins in Membranes (OPM) database [115] for the orientation of rCI. Djuranekova et al. have surrounded the system with transferable intermolecular potential with 3 points (TIP3) water molecules and Na^+/Cl^- ions to establish physiological conditions. The Qox has been created with the same parameters as the noQ but with the additional ubiquinone placed in its tunnel. [116]

For each data set, we received 2,000 files, one per microsecond. Each file contains the atomic coordinates of the protein subunits and the ligands. The ligands are flavin mononucleotide (FMN), iron-sulfur (Fe-S) clusters and, in the case of the Qox, Ubiquinone. The Fe-S clusters are assigned to subunits and share their chain ID. FMN and ubiquinone are not assigned to a subunit and have their own chain ID, respectively. The full list of subunits and their chain IDs can be found in Table 4.

The files are formatted according to the legacy-PDB style (see Section 2.6.1), but differ in some points from the legacy-PDB format. Because of this, we call it pseudo-legacy-PDB style from now on. The differences are described in detail by Nurhassen [117].

2.10.3 Respiratory complex I of *Thermus thermophilus*

rCI (see Section 2.5.2) of *Thermus thermophilus* consists of 16 subunits. We used the structure 4hea as a use case. We refer to the subunits by their abbreviated gene name for NADH-quinone oxidoreductase (Nqo). The subunit with ID W is not part of the Nqo operon and has been referred to as TTHA1528 [12]. The matching of chain IDs, gene names and modules can be found in Table 4. A visualization of the structure and the position of all subunits can be found in Figure 5a.

2.10.4 Respiratory complexes of *Homo sapiens*

The following structures have been determined by Guo et al. [20].

Respiratory complex I

rCI (see Section 2.5.2) of *Homo sapiens* consists of 45 subunits. 14 subunits are conserved "core subunits" and 31 are supernumerary subunits. The subunits are labeled according to their genes and can be abbreviated by omitting the prefixed "NDU" or "MT-" (see Table 5). [86]

Human respiratory complex I [20] (5xtd) contains all 45 subunits (see Table 5 and Figure 6) and a total of 66,789 atoms at a resolution of 3.7 Å.

Assembly Guerrero-Castillo et al. [84] have investigated the assembly pathway of human rCI using complexome profiling (see Section 2.4). They have proposed an assembly pathway that is based on experimental data for 44 of 45 subunits. They have been able to assign 17 assembly factors to the pathway.

Guerrero-Castillo et al. have proposed a modular assembly in which the known modules Q, N, P_P and P_D assemble into premature modules (pre-module). The pre-modules assemble with each other completing the final rCI. Guerrero-Castillo et al. have proposed two alternative pathways which differ in the order in which

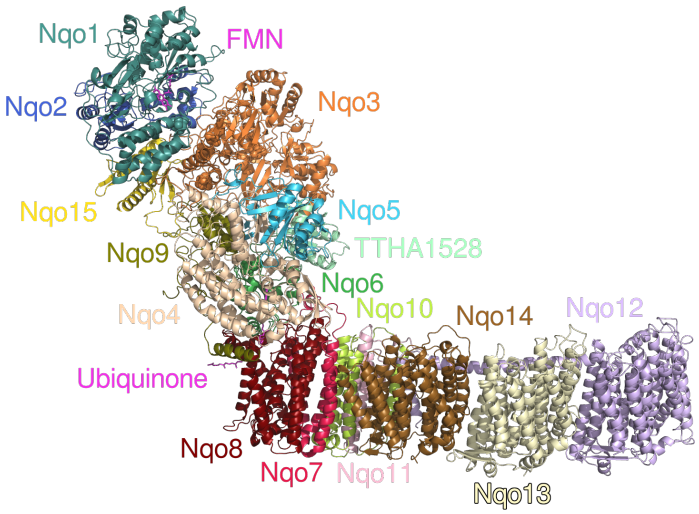
Table 4: Matching of the chain IDs and names of the subunits of respiratory complex I of *Thermus thermophilus* [12] (PDB: 4hea). Chains with an asterisk mark ligands that have been placed in separate chains by Sharma et al. [116]. For each protein chain, the name of the module is given.

Chain ID	Name	Module
1	Nqo1	N
2	Nqo2	N
3	Nqo3	N
4	Nqo4	Q
5	Nqo5	Q
6	Nqo6	Q
9	Nqo9	Q
7	Nqo15	N
W	TTHA1528	Q
A	Nqo7	P_P
J	Nqo10	P_P
K	Nqo11	P_P
L	Nqo12	P_D
M	Nqo13	P_D
N	Nqo14	P_P
H	Nqo8	P_P
F*	Flavin mononucleotide	-
U*	Ubiquinone	-

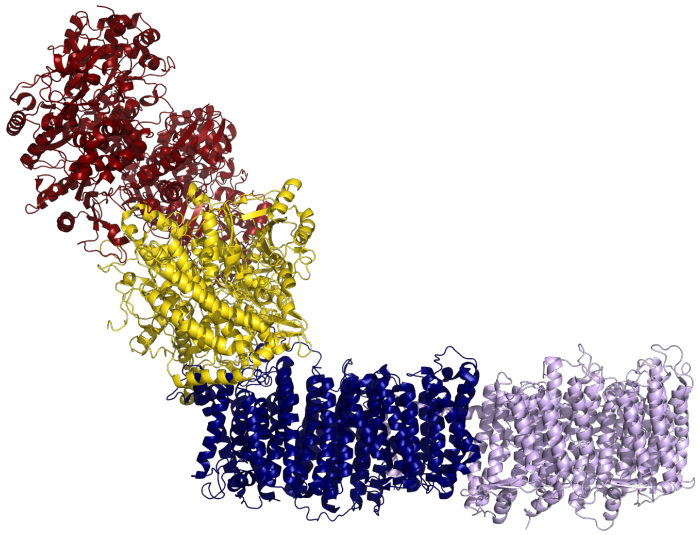
the pre-modules assemble. In one pathway, pre-P_P-b and pre-Q/pre-P_P-a assemble first. In the other pathway, pre-P_P-b and pre-P_D-a assemble first. We refer to the pathways from now on as reference pathway 1 and 2.

We depicted the proposed assembly pathways as dendrograms (see Section 2.4 and Figure 7). Note that the assembly pathways contain steps, in which more than two subunits or subcomplexes assemble. Because of this, the dendrograms are non-binary (see Section 2.1.3).

In the scheme of the proposed assembly pathways (see Figure 7 of Guerrero-Castillo et al. [84]), subunits C1 and C2 are drawn as a single block assembling with subunit ND2. In the text, Guerrero-Castillo et al. mention a "complex composed of ND2, NDUFC1, and NDUFC2" without proposing that C1 and C2 might be assembled before assembly with ND2. Because of this, we do not impose an order on these three subunits, but discuss the possibility of prior assembly of C1 and C2.



(a) Missing parts and ubiquinone modelled by Djurabekova et al. [116]. Each chain is colored individually and labeled. The ligands that are placed in separate chains, ubiquinone and FMN, are colored in magenta.



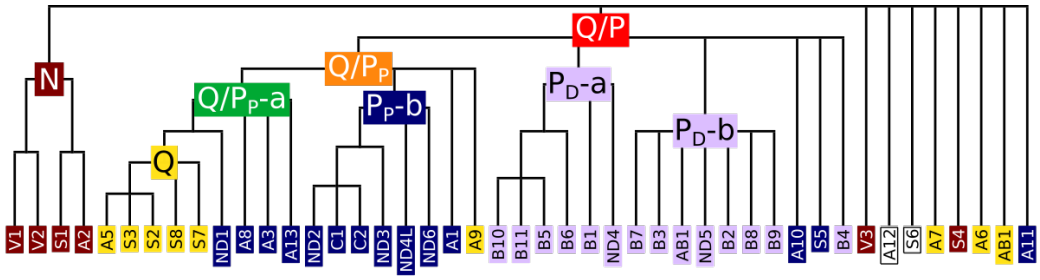
(b) Subunits are colored by modules: N (red), Q (yellow), P_P (dark blue) and P_D (lavender).

Figure 5: Structure of respiratory complex I of *Thermus thermophilus* [12]. Subunits and ligands shown in cartoon style and as balls, respectively.

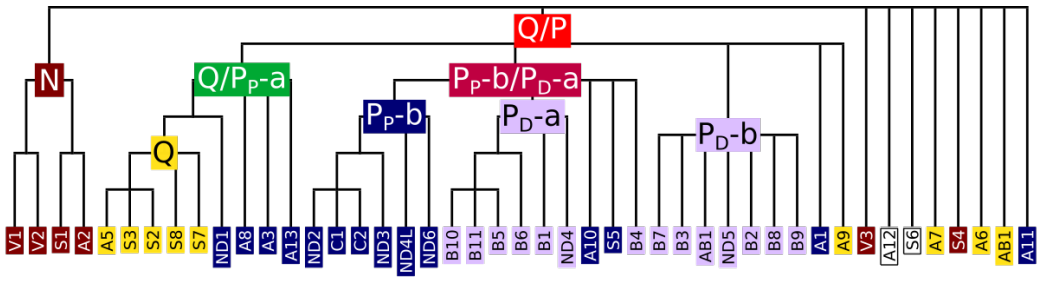
Table 5: Overview of the subunits of respiratory complex I of *Homo sapiens* [20] (PDB: 5xtd). Length is given in number of residues.

Gene name		5xtd		
Full name	Abbreviated name	Chain ID	Length	Module
NDUFA2	A2	F	83	N
NDUFS1	S1	M	687	
NDUFS4	S4	L	118	
NDUFV1	V1	A	431	
NDUFV2	V2	O	212	
NDUFV3	V3	K	33	
NDUFA12	A12	N	143	N/Q
NDUFS6	S6	T	95	
NDUFA5	A5	H	112	Q
NDUFA6	A6	E	113	
NDUFA7	A7	I	95	
NDUFA9	A9	J	337	
NDUFAB1	AB1	G	85	
NDUFS2	S2	Q	430	
NDUFS3	S3	P	208	
NDUFS7	S7	C	156	
NDUFS8	S8	B	176	
NDUFA1	A1	S	70	P_P
NDUFA10	A10	w	320	
NDUFA11	A11	V	140	
NDUFA13	A13	W	138	
NDUFA3	A3	U	83	
NDUFA8	A8	u	169	
NDUFC1	C1	f	47	
NDUFC2	C2	g	119	
MT-ND1	ND1	s	318	
MT-ND2	ND2	i	347	
MT-ND3	ND3	j	115	
MT-ND4L	ND4L	k	97	
MT-ND6	ND6	m	174	
NDUFS5	S5	h	104	
NDUFAB1	AB1	X	85	P_D
NDUFB1	B1	n	56	
NDUFB10	B10	d	171	
NDUFB11	B11	e	97	
NDUFB2	B2	Y	59	
NDUFB3	B3	Z	80	
NDUFB4	B4	o	128	
NDUFB5	B5	a	138	
NDUFB6	B6	b	124	
NDUFB7	B7	v	111	
NDUFB8	B8	c	153	
NDUFB9	B9	p	172	
MT-ND4	ND4	r	459	
MT-ND5	ND5	l	603	

Figure 6: Visualization of the structure of human respiratory complex I [20] (PDB: 5xtd). Residues shown in transparent cartoon style and ligands as sticks or balls. Ligands are colored in magenta and subunits according to their modules: N (red), Q (yellow), P_P (dark blue), P_D (lavender).



(a) Reference pathway 1.



(b) Reference pathway 2.

Figure 7: Proposed assembly pathways by Guerrero-Castillo et al. [84] for human respiratory complex I represented as dendrograms. Leaves that correspond to subunits are labeled with the abbreviated gene names and colored according to the modules they are assigned to: N (red), Q (yellow), P_P (dark blue) and P_D (lavender). The subunits that are shared between the N and Q module are colored white. Inner vertices correspond to subcomplexes and the root to the final protein complex. Selected inner vertices are labeled with the names of the pre-modules.

Respiratory supercomplex $I_1III_2IV_1$

Human respiratory supercomplex $I_1III_2IV_1$ [20] (5xth) contains one copy of complex I, two of complex III and one of complex IV (see Figure 8) as indicated by the numbers in subscript. In total, 5xth contains 80 subunits and 115,642 atoms at a resolution of 3.9 Å.

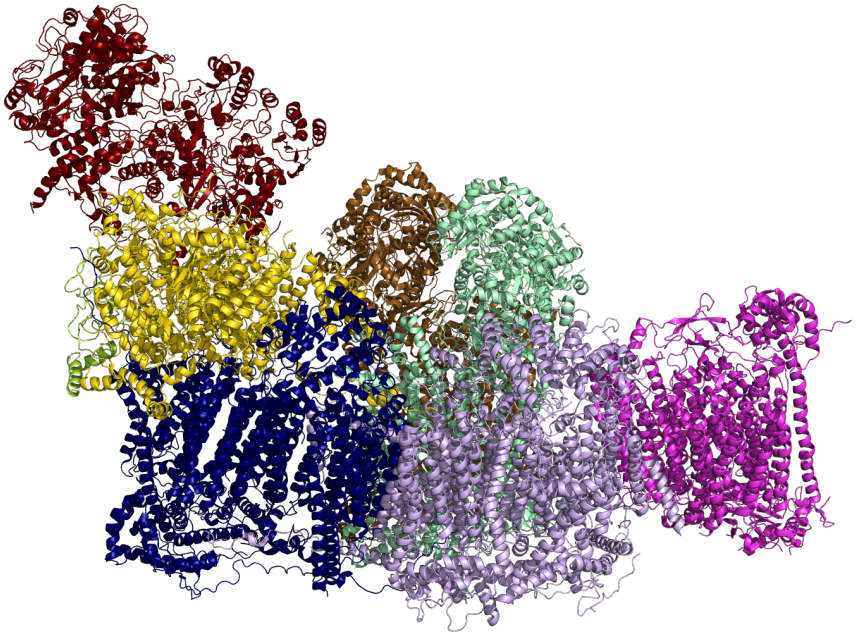
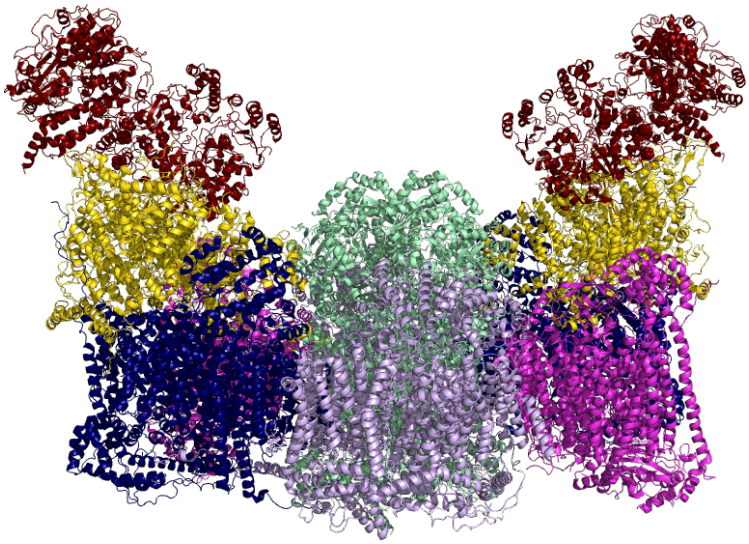


Figure 8: Visualization of structure of respiratory supercomplex $I_1III_2IV_1$ of *Homo sapiens* [20] (PDB: 5xth). Residues shown in cartoon style and ligands as sticks or balls. Complexes are colored individually: III-1 (mint), III-2 (brown) and IV (magenta). The modules of complex I are colored individually: N (red), Q (yellow), P_P (dark blue), P_D (lavender).

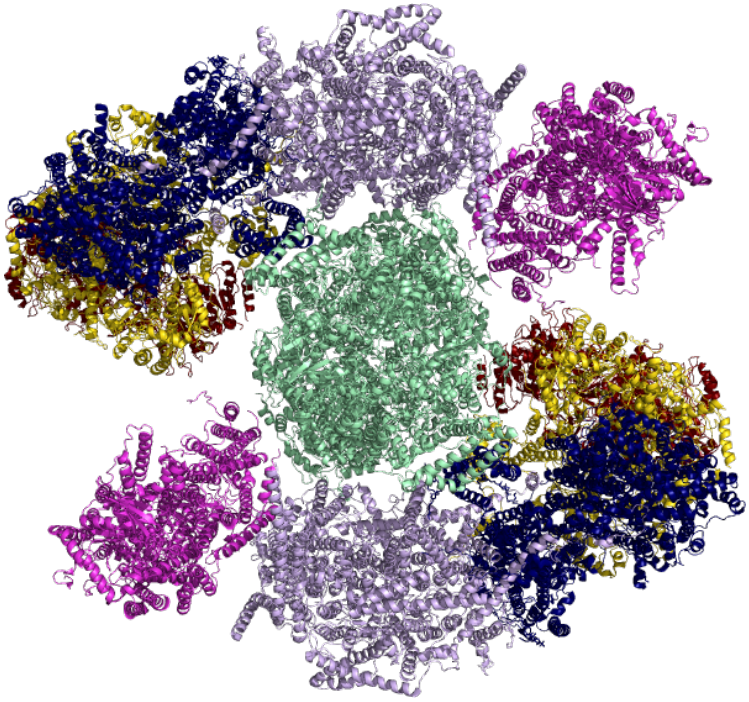
The subunits of rCI are named and abbreviated the same as in 5xtd (see Section 2.10.4) and the chain IDs match, too. The subunits of complex III are abbreviated by omitting the prefixed "UQCR" or "MT-" (see Table 6). The subunits of complex IV are abbreviated by omitting the prefixed "COX" or "MT-CO".

Respiratory megacomplex $I_2III_2IV_2$

Human respiratory megacomplex $I_2III_2IV_2$ [20] (5xti) contains two copies of each complex I, III and IV (see Figure 9). 5xti consists in total of 138 subunits and 196,753 atoms at a resolution of 17.4 Å. The subunits are named and abbreviated as for 5xtd and 5xth.



(a) Aligned x-axis and membrane



(b) Rotated by 90° along the x-axis

Figure 9: Visualization of structure of respiratory megacomplex $I_2III_2IV_2$ of *H. sapiens* [20] (PDB: 5xti). Residues shown in cartoon and ligands as ball-and-stick style. Complexes are colored individually: III (mint) and IV (magenta). The modules of complex I are colored individually: N (red), Q (yellow), P_P (dark blue), P_D (lavender).

Table 6: Overview of the subunits of the two copies of complex III and one copy of complex IV of the respiratory supercomplex $I_1III_2IV_1$ of *Homo sapiens* [20] (PDB: 5xth). Length is given in number of residues. SP stands for single peptide according to Table S2 of Guo et al. [20].

Gene name		5xth	
Full name	Abbreviated name	Chain ID	Length
UQCR10	10	AD	62
UQCR11	11	AG	51
UQCRB	B	AF	106
UQCRC1	C1	AL	446
UQCRC2	C2	AK	416
MT-CYB	CYB	AJ	378
CYC1	CYC1	AH	241
UQCRFS1	FS1	AC	196
UQCRFS1(SP)	FS1(SP)	AB	57
UQCRH	H	AE	74
UQCRQ	Q	AA	81
UQCR10	10	AQ	62
UQCR11	11	AT	51
UQCRB	B	AS	106
UQCRC1	C1	AY	446
UQCRC2	C2	AW	419
MT-CYB	CYB	AV	378
CYC1	CYC1	AU	241
UQCRFS1	FS1	AP	196
UQCRFS1(SP)	FS1(SP)	AO	57
UQCRH	H	AR	74
UQCRQ	Q	AN	81
MT-CO1	1	x	514
MT-CO2	2	y	227
MT-CO3	3	z	261
COX6A2	6A2	3	84
COX4I1	4I1	0	144
COX5A	5A	1	109
COX5B	5B	2	98
COX6B1	6B1	4	75
COX6C	6C	5	73
COX7A1	7A1	6	56
COX7B	7B	7	49
COX7C	7C	8	47
COX8B	8B	9	43

3 Results and discussion

3.1 Implementation

This work aims to analyze the topology of large complexes. To be able to automatically analyze large complexes efficiently, we need feasible runtimes. The runtime will always scale with the size of the input data when analyzing contacts of atoms, but we can aim to achieve a linear or near-linear scaling. In the following subsection, we present our extension of the PTGL that produces more accurate results and is considerably faster in computing graphs of topologies than before.

3.1.1 Computation of contacts

Observations

The check of the spheres of the molecules and neighbor skipping (see Section 2.8.2) have been implemented for single protein chains without ligands. The inclusion of ligands to the computation of contacts and the processing of multiple chains in one execution for the creation of CGs have diminished the potential of the neighbor skipping. The pre-processing of the neighbor skipping to find the maximum distance d_{max} between sequential neighbors has been applied to the whole list of molecules. Thereby, ligands are treated as molecules of the polypeptide chain that are joined in sequence after the last residues at the C-terminus of a chain. Additionally, chains are stitched together meaning the first residue of a chain is treated as joined in sequence with the last residue of the previous chain. This results in a large d_{max}, because molecules are joined sequentially that are not really consecutive in sequence and can have large spatial distances. A large d_{max} results in few skips up to the point where the pre-processing is not worth the saved runtime.

The check of the spheres of the molecules still functions as intended, but for large complexes of many chains, the runtime still becomes quite large demanding a speedup optimized for many chains. We also observed that the spheres of the molecules are often larger than necessary. The reason lies in taking the C_α as midpoint. For most side chains, there are atoms spatially far away from C_α creating a large sphere. This results in unnecessary checks between the pairs of atoms of two molecules.

Centroid for spheres of molecules

The smaller the sphere of a molecule the less spheres overlap where there is no contact possible preventing unnecessary checks of the atoms of two molecules. Ideally, the smallest sphere of a molecule results in skipping the most checks saving runtime. The sphere is smallest if it is centered at the geometric median of the atoms. The geometric median minimizes the maximum distance to any atom. The difference to the centroid of points becomes clear when thinking of a heavily skewed distribution of n points where $n-1$ points are close to each other and one point is far away. The centroid will almost be at the center of the $n-1$ points whereas the geometric median will be approximately halfway between the $n-1$ and the one far away point. It is obvious that for this case, the geometric median produces a smaller and therefore better sphere.

Finding the geometric median in 3D space is not trivial. Algorithms for finding the geometric median in feasible runtimes have been proposed [118]. We decided to use the centroid, because it is easier and faster to compute. Also, the centroid

should not differ too much from the geometric median for atoms in Euclidean space, because they cannot occupy the same spaces.

We implemented a function to compute the centroid and the radius of the sphere of a molecule. The centroid and the radius are saved for each molecule so that they are computed only once. The computation takes extra time, especially in the case of residues where previously simply the C_α has been taken as center. But using the centroid and the maximum distance to any of the molecule's atoms ensures a smaller sphere of the molecule.

This can be seen for the Arginine at position two of chain A in triosephosphate isomerase-phosphoglycolohydroxamate complex [22] (7tim) (see Figure 10). The maximum distance from C_α to any atom is 6.2 Å, because of the long sidechain. The maximum distance from the centroid to any atom is 3.8 Å.

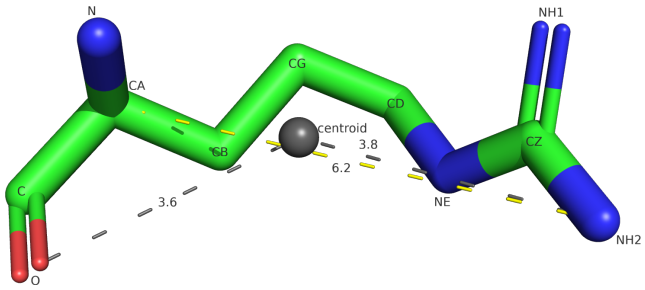


Figure 10: Stick representation of arginine of chain A at position two in triosephosphate isomerase-phosphoglycolohydroxamate complex [22] (PDB 7tim). The atoms are colored and labeled by their type. A grey orb marks the centroid. Distances regarding the centroid are shown as grey and regarding the C_α (CA) as yellow dashed lines. Distances are labeled in Ångström.

Check of the spheres of chains

Similar to the check of the molecules' spheres, we implemented a check of the chains' spheres. We compute the centroid of all atoms of the chain. We compute the radius of the sphere centered in the centroid as the maximum distance between the centroid and any atom. The geometric median would probably be better than the centroid, but we discarded the idea and refer to the previous discussion. We save both the centroid and radius per chain in order to compute them only once. Previously, the contacts between all molecules have been computed by iterating over all molecules of the structures. Instead we iterate chain by chain. For each pair of chains, we check if the chain's spheres plus the contact threshold overlap. If not we can skip checking any pair of molecules of the two chains.

Neighbor skipping optimized for ligands and multi-chain complexes

Previously, all molecules of the structure, i.e. molecules of all chains and all ligands, have been treated as one chain for the neighbor skipping (see Section 2.8.2). Since we iterate through each chain individually for the check of the chain's spheres, we also compute and save the maximum distance of sequential neighbors d_{max} per chain. It is important to note that we exclude ligands and only consider residues as part of the polypeptide and nucleotides as part of the RNA chain.

To enable skipping even when ligands are involved, we make use of the following fact. When computing the distance between two molecules m_1 and m_2, only one of the molecules needs to be part of a chain for neighbor skipping to apply. Let us consider that m_1 is a ligand and m_2 is either a residue or nucleotide of a chain c. Let d_{max} be the maximum distance between sequential neighbors of chain c. We compute the distance between m_1 and m_2. If the distance is larger than d_{max}, we can skip comparing the molecules following after m_2 in the sequence according to neighbor skipping. Note that m_1 does not need to be part of a chain for this to work.

Concluding from this, we can apply neighbor skipping for contact checks within a chain, between chains and between a ligand and a chain. We cannot apply neighbor skipping to contact checks between ligands. This does not pose a problem, because the number of ligands can be expected to be much smaller than the number of molecules that are part of a chain.

From the above, a special conclusion can be drawn for the case of contact checks between chains. We only use neighbor skipping on one chain. This means we can decide which chain we want to use it for. Currently, we apply neighbor skipping on the longer chain, because we expect more skips to happen there. This becomes clear when thinking of the edge case where one chain has length one and can be thought of as a ligand. We apply the neighbor skipping to the other chain and can facilitate the shorter runtime.

For the implementation we can deduct that we have multiple loops (see Algorithm 2). We iterate all intra-chain pairs and all inter-chain pairs differentiating between non-ligand molecules and ligands. These iterations give us all possible contacts between molecules and we facilitate neighbor skipping wherever possible. From now on, we refer to the neighbor skipping optimized for ligands and multi-chain complexes as improved neighbor skipping.

Statistics of contact checks

We tested the check of the chains's spheres and improved neighbor skipping on a data set of proteins (see Section 2.9). We compared the improved neighbor skipping, the check of the chain's spheres and the combination of both to the previous implementation. In the worst case, all possible pairs of residues need to be checked for a contact. The number of skipped checks of pairs of residues and the proportion to the number of possible pairs is the measure of quality. The main comparison of the combination of improved neighbor skipping and check of the spheres of chains to the previous implementation can be found in Table 7. The complete statistics evaluating the methods can be found in Table A2.

For the improved neighbor skipping, we differentiated between skips where both molecules are part of the same chain (intra) and where they are not (inter). For structures with a single protein chain, there are only intra skips possible. Interestingly, the improved neighbor skipping shows slightly less skips than the previous implementation for three structures with one chain: geranyltransferase from *Agrobacterium tumefaciens* (unpublished) (2h8o), L polymerase of vesicular stomatitis virus [14] (5a22) and transcription-associated protein 1 of *Saccharomyces cerevisiae* [18] (5ojs). The differences in proportion to the number of pairs of residues are as little as 0.59 %, 0.13 % and 0.14 %. We thoroughly evaluated the implementation of the improved neighbor skipping and found one case where the previous implementation missed one residue contact, because of a rounding error occurring during the check of the residues' spheres and taking C_α as center of the sphere. As a consequence, we

Algorithm 2 Computation of contacts using extended neighbor skipping

```
1: for c_1 in all chains do                                   ▷ intra-chain contacts
2:     for all pairs of non-ligand molecules of c_1 do
3:         apply neighbor checking
4:     end for
5:     for all pairs of a ligand and a non-ligand molecule of c_1 do
6:         apply neighbor skipping
7:     end for
8:     for all pairs of ligands of c_1 do
9:         do not apply neighbor skipping
10:    end for
11:    for c_2 in all chains that have not been assigned to c_1, yet do   ▷ inter-chain
   contacts
12:        for all pairs of a non-ligand molecule in c_1 and a non-ligand molecule in
   c_2 do
13:            apply neighbor checking
14:        end for
15:        for all pairs of a ligand in c_1 and a non-ligand molecule in c_2 as well as
   all pairs of a non-ligand molecule in c_1 and a ligand in c_2 do
16:            apply neighbor checking
17:        end for
18:        for all pairs of a ligand in c_1 and ligand in c_2 do
19:            do not apply neighbor skipping
20:        end for
21:    end for
22: end for
```

increased J from 0.2 to 0.4 (see Section 2.8.2) fixing this error. This only has a small effect on the neighbor skipping, but might explain the slightly lower number of skips. Because the difference is so small, we did not investigate any further.

Noteworthy is the increase for serine protease EspP N1023D mutant [10] (3slo) from 3 % skips of the previous implementation to 50.16 % skips. 3slo and 5a22 both contain a ligand. It seems that for 3slo, the ligand drastically hampers with d_{max} of the neighbor skipping. This seems not always to be the case with ligands as can be seen with 5a22, but the improved neighbor skipping successfully fixed this limitation.

For the structures with three chains, the improved neighbor skipping clearly outperforms the previous implementation. The percentage of total skips is higher for all structures. The biggest difference occurs for tailspike protein 1 from *Escherichia coli* [13] (4oj5) where the previous implementation applied no skip compared to 76.9 % skips of the improved neighbor skipping. For all four structures, for the previous implementation, this seems to be the negative effect of treating all molecules as one chain. For 4oj5, this may be even worse, because 4oj5 contains a ligand.

Interestingly, compared to structures with one chain, the intra skips alone were consistently higher than the skips of the previous implementation. This emphasizes how negative the effect of treating all molecules as a single chain is, because for the structures with one chain, the number of skips of the previous implementation is slightly higher. Moreover, inter skips are now possible that could not be applied to single chains. The number of inter skips is higher than the number of intra skips for

Table 7: Number of skipped checks for contacts of residues. Structures with an asterisk (*) contain a ligand. Neighbor skipping treating all molecules as one chain (previous implementation) is compared to neighbor skipping treating chains and ligands individually combined with check of chain spheres (combined method). For each method, the percentage of skipped checks for contacts is given.

		Number of ...			Total skips [%]	
PDB ID	Chains	Residues	Atoms	Pairs of residues	Previous method	Combined method
2h8o	1	283	2,468	39,903	30.07	29.48
3slo*		313	2,469	48,828	3.00	50.16
5a22*		2,004	32,188	2,007,006	10.91	10.78
5ojs		3,473	28,407	6,029,128	30.75	30.61
2adv	3	683	5,749	232,903	28.99	61.32
2gvz*		714	5,754	254,541	7.33	57.18
4oj5*		2,272	36,274	2,579,856	0.00	76.90
5x5b		3,139	24,543	4,925,091	0.24	62.18
1pma*	28	5,936	56,321	17,615,080	0.25	90.21
3h6i*		6,159	48,568	18,963,561	8.07	86.87
4r3o		6,243	47,859	19,484,403	18.59	88.51
5da8*		14,030	99,580	98,413,435	10.27	90.03
5im4	40	5,302	40,577	14,052,951	2.55	93.32
5mx2*		5,442	50,447	14,804,961	0.16	85.46
3zlp		6,543	52,731	21,402,153	10.81	93.38
4ro0		8,840	68,787	39,068,380	1.81	93.44

three out of four structures showing the importance of inter skips for multi-chain structures.

For structures with one chain, skips cannot occur because of the check of the chain's spheres. For structures with three chains, the check of the chain's spheres did not take effect. It would be possible in theory if one of possibly three chain contacts is not present. Examining four structures is too little to draw a final conclusion, but it seems plausible that complexes as small as consisting of three chains are likely to be intertwined such that all chains interact with each other preventing skips due to the check of the chain's spheres.

The observations for structures with 28 and 40 chains are comparable which is why we discuss both together. Therefore, we think that the mistake in compiling the data set (see Section 2.9) is neglectable.

Structures containing a ligand allow nearly no skips in the previous implementation with proportions of 0.25 %, 8.07 %, 10.27 % and 0.16 % compared to proportions of 18.59 %, 2.55 %, 10.81 % and 1.81 %. In general, the proportion of skips is low for the previous implementation with a maximum of 18.59 %. The improved neighbor skipping achieves a maximum of 88.35 % and all proportions are above 79 %. Whether a structure contains a ligand or not does not seem to play a role for the improved neighbor skipping.

The proportion of intra skips is less than 2 % for all structures. The majority

of skips are inter skips. This behaviour was expected, because with an increasing number of chains there are more contacts possible between chains than within.

For all structures with 28 and 40 chains, the check of the chain's spheres took effect. The number of intra skips is not affected by the check of the chain's spheres. The number of inter skips is affected, because there are no checks for inter contacts if the chain's spheres do not overlap. Consequently, the number of inter skips is lower for the combined method with proportions between 17.91 % to 61.96 %.

In some cases, the chain sphere-check seems to supersede the improved neighbor skipping. For example, for designed two-component self-assembling icosahedral cage [15] (5im4), 77.85 % of the checks of the residue pairs are skipped because of the chain sphere-check compared to 87.7 % of skipped inter checks for improved neighbor skipping. However, there are also cases where the improved neighbor skipping still contributes to a large number of skips. For example, for photosystem II depleted of the Mn4CaO5 cluster [16] (5mx2), 80.65 % of checks of residue pairs are skipped in inter checks of the improved neighbor skipping. Using the combined method, there still occur 61.96 % skips due to inter neighbor skipping. This means that both methods go hand in hand. The chain sphere-check skips large proportions where it is possible and the neighbor skipping takes effect where it is not possible.

Overall, the total proportion of skips increases for the combined method to up to around 90 %. To give an example, the combined method skips 36,505,280 pairs of residues of MthK gating ring in a ligand-free form (unpublished) (4ro0) in the contact computation of the combined method which accounts for 93.44 % of all residue pairs. In the previous implementation, 706,779 pairs were skipped which accounts for 1.81 %.

We conclude that the improved neighbor skipping is not able to achieve more skips for structures with one chain. With an increasing number of chains, the improved neighbor skipping clearly outperforms the previous implementation. The improved neighbor skipping makes no difference between structures with or without ligands and excels for both. The chain sphere-check takes effect for structures with many chains, such as 28 and 40. The combined method is most effective for these structures and allows skipping the check for a contact in the majority of cases.

Runtime

Skipping checks for contacts saves runtime, but the question arises if this is made up to the runtime of the additional computations, such as the computation of the chain's spheres. We tested the runtime of the combined method against the previous implementation on a data set of representative structures with all numbers of chains and different numbers of atoms (see Section 2.9). We ran all computations on the same machine. We ran non-large structures ten times and large structures between two and eight times to account for fluctuations of the runtime. We measured the runtimes with the Linux program *time* and summed up the times for *user* and *sys*.

The summed up average runtimes for all structures was 876,728 s for the previous implementation and 48,576 s for the combined method. The combined method saved 828,152 s of runtime with an average of 4,652 s per structure. We computed the speedup factor s as $runtime_{combined} * s = runtime_{previous}$ for the average runtimes. A speedup factor less than one means the combined method was slower as the previous implementation and higher than one means it was faster. The speedup factor was less than one for three structures, exactly one for two structures and higher than one for the remaining 173 structures. The average speedup factor is 4.52.

Judging by the speedup factor, the combined method is considerably faster in nearly all cases. Interestingly, the speedup factor increases with the number of chains (see Figure 11). While the speedup factor is around one for structures consisting of a few chains, it rises linearly with an increasing number of chains up to 200. There are not enough structures with more than 200 chains to see whether the speedup increases linearly here, too.

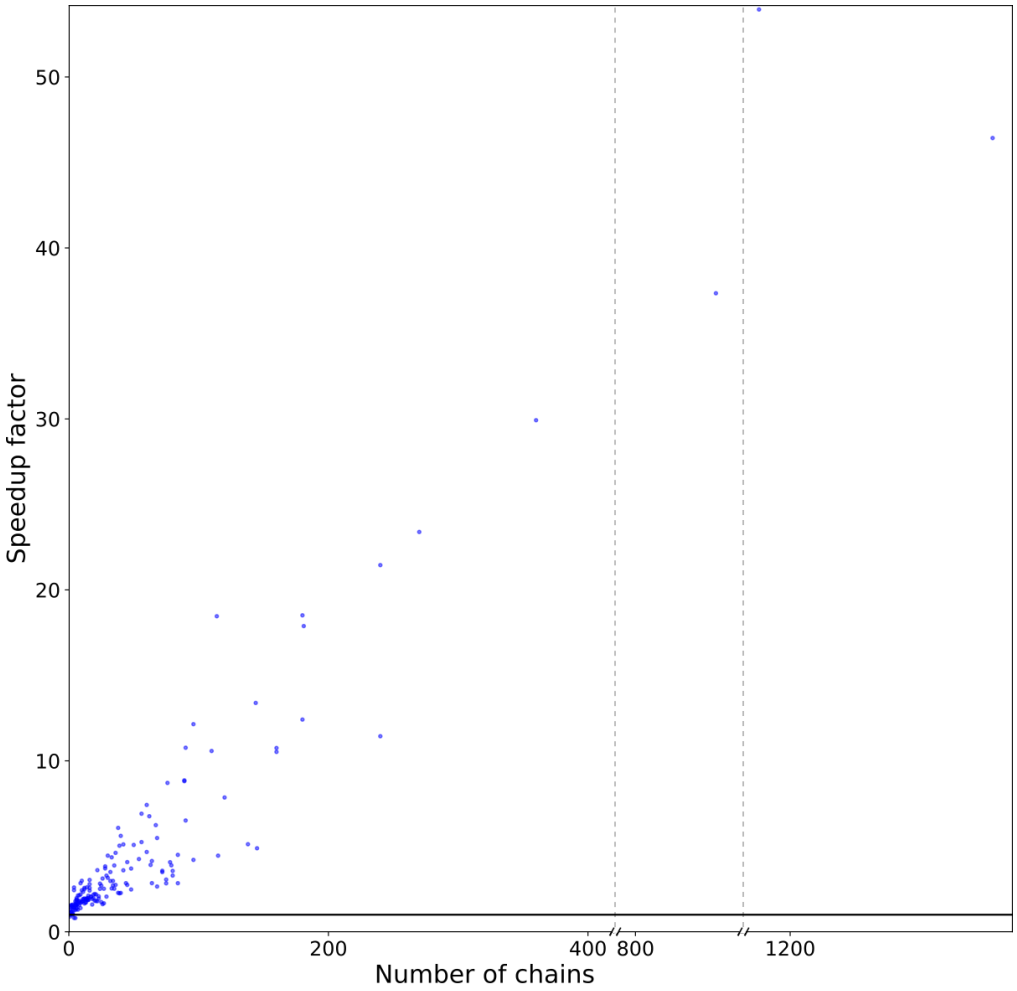


Figure 11: Speedup factor of combination of check of the chain's spheres and improved neighbor skipping versus previous implementation against number of chains. A horizontal black line at 1 marks where both implementation have equal runtimes. Plot created with Matplotlib [119].

The speedup factor is clearly higher for large structures up to 1,356 chains than for structures with a few chains. The highest speedup factor is 53.95 for capsid of human immunodeficiency virus (186 hexamers + 12 pentamers) [9] (3j3y). 3j3y contains 1,176 chains and 2,116,800 atoms and is the second largest structure of the data set regarding the number of chains and atoms. The previous implementation took on average 225,982.5 s and the combined method 4,189 s.

We plotted the runtime per atom in seconds against the number of atoms (see Figure 12). This visualizes that the combined method scales differently with the number of atoms than the previous method. The runtime per atom increases with the number of atoms for the previous implementation. The runtime per atom of

the combined method on the other hand is around the mean of 0.0025 s per atom independent of the number of atoms.

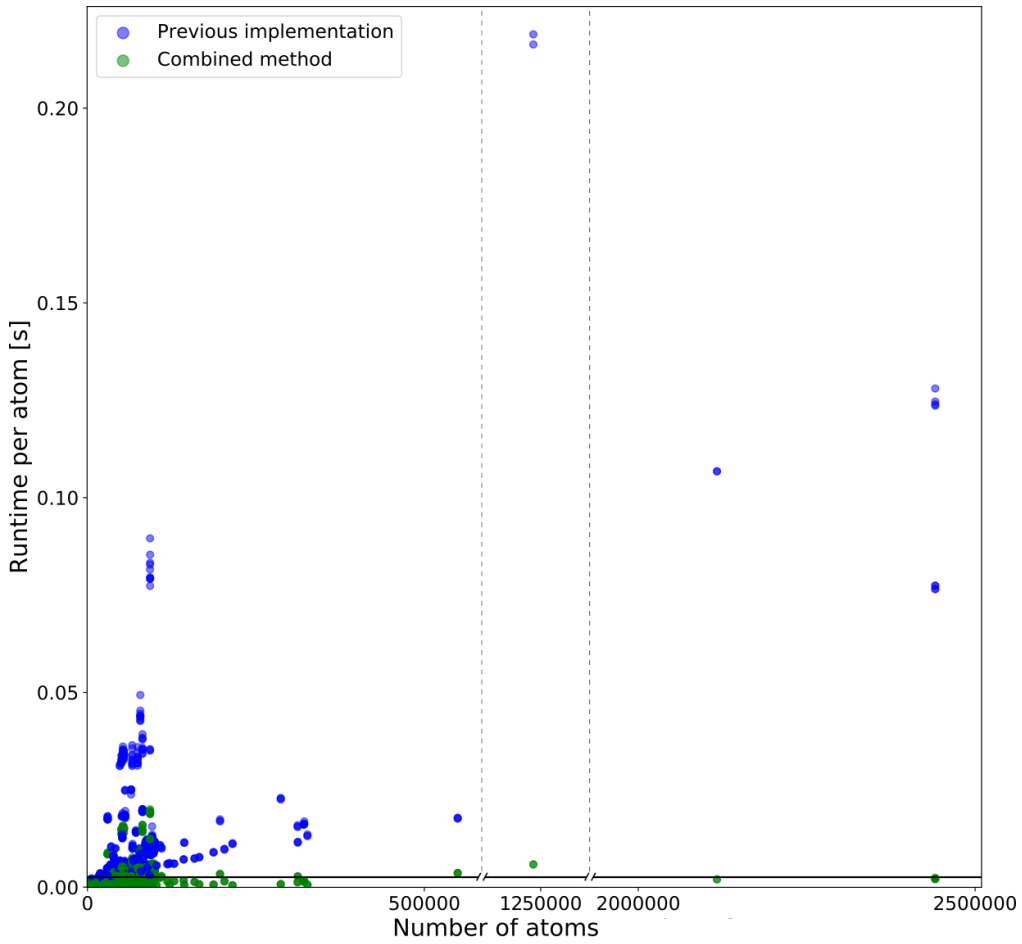


Figure 12: Scatterplot of runtime per atom in seconds against number of atoms for multiple runs of different structures. Results are presented for the previous implementation (blue) and the combination of chain sphere-check and improved neighbor skipping (green). A black horizontal line marks the mean of the combined method at 0.0025. Plot created with Matplotlib [119].

This means that the combined method scales linearly with the number of atoms. Considering that the number of pairs of atoms that in theory need to be checked for a contact scales exponentially, this is a great achievement.

We decided to use the previous implementation for structures with one chain where no chain sphere-check is possible anyway. The combined method can take slightly longer for structures with a few chains, because of the additional computations. Structures with a few chains do not cause long runtimes. For example, structures with two chains took between three and 150.1 s for the combined method. The latter runtime was caused by motor domains from human cytoplasmic dynein-1 in the phi-particle conformation [17] (5nug) which consists of two chains with 46.234 atoms in total and is representative for the maximum number of atoms for structures with two chains. We think that the combined method can be used in general for structures with two or more chains and will not cause relevant longer runtimes for edge cases,

but will save runtime for most structures. The improvements in runtime open up exploring protein complexes in feasible runtimes no matter how large they become.

3.1.2 Orientation of secondary structure elements

Observations for the Double-Distance method

We noticed that the previous computation of the orientation of SSEs can take the majority of computation time for large structures. The computation is done twice per pair of SSEs: once for PGs and once for the complex-level PG, because the orientation is not saved. For small structures, such as 7tim, the computation of the orientation between SSEs for the complex-level PG takes a few seconds which is only a small portion of the total runtime of one to two minutes. For large structures, such as 3j3q, the computation takes 22 hours which is more than half of the total runtime of 43 hours for the combined method (see Section 3.1.1). Computationally determining the orientation of SSEs for the complex-level PG takes longer than parsing the input files, computation of contacts and creation of all other graphs together.

We also noticed wrong orientations, for example, for 17β-hydroxysteroid dehydrogenase [2] (1jtv). 1jtv is a SDR (see Section 2.10.1) which clearly exhibits the α/β fold (see Figure 13a) and the typical strand topology (see Figure 13c). PTGLgraph-Computation assigns an antiparallel orientation to strands six and seven (see Figure 13d). Strand seven is short, consisting of only three residues compared to strand six consisting of eight residues. For short SSEs, it can be difficult to visually impose an orientation. Nevertheless, structurally, strands six and seven clearly have a parallel orientation (see Figure 13b).

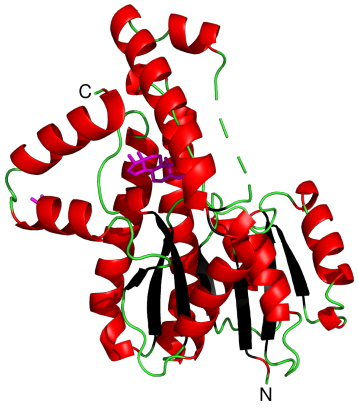
The residue-level contacts between the strands (see Figure 14) differ much from the ideal example (see Figure 3). Notably, the last residue of strand six that has any contacts with strand seven, residue 184, is in contact with all residues of strand seven. This impacts the DD (see Section 2.8.5) of strand six and seven:

$$\begin{aligned} DD &= (S_{max} - S_{min}) - (D_{max} - D_{min}) \\ &= (438 - 434) - (72 - 68) \\ &= 0 \end{aligned}$$

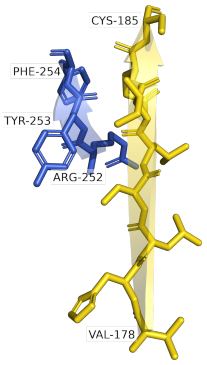
For a DD of zero, an antiparallel orientation is assigned to these strands (see Figure 1). Zero is exactly the threshold at which an antiparallel orientation is assigned. This makes 1jtv an edge case where the assignment of orientations for SSEs using the DD does not work.

Vector method

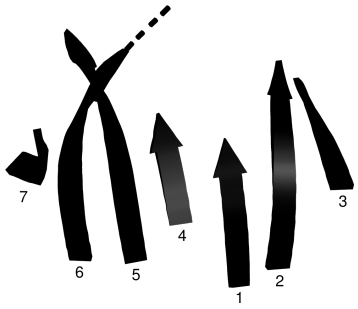
Rationale We wanted to find a method that is faster to compute and more accurate in the assignment of orientations of SSEs. For each SSE, we place a 3D vector that starts at the N-terminal beginning of the SSE and ends at the C-terminal end. We compute the centroids of the first and last c_l residues where c_l is a positive integer that is not zero. SSEs are defined by the arrangement of the backbone atoms. Sidechains can point away from the SSE, such as in all directions for helices. Because of this, we only use the backbone atoms for the computation of the centroid per residue. The start point of the vector is the centroid of the centroids of the c_l N-terminal residues



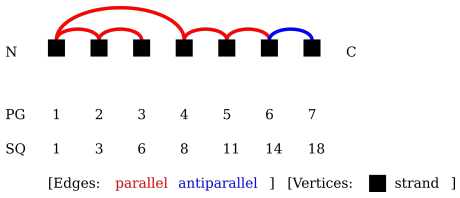
(a) Visualization of the structure in cartoon representation. Helices are depicted as red spirals, strands as black arrows, loop regions as green ribbon and ligands as magenta sticks. Unresolved structures are visualized as dashed ribbons.



(b) Strand six (yellow) and seven (blue) in transparent cartoon representation and stick representation. First and last residue of strand six as well as all residues of strand seven are labeled with their three-letter code and position in sequence.



(c) Visualization of the sheet in cartoon representation. Strands are depicted as black arrows. Strands are labeled by occurrence in the sequence from N- to C-terminus.



(d) Beta Protein Graph created with PTGLgraphComputation version 3.0.0. Vertices are labeled by occurrence in the Protein Graph (PG) and in the sequence (SQ).

Figure 13: Visualization of the structure (a,b,c) and Protein Graph (d) of 17β-hydroxysteroid dehydrogenase [2] (PDB 1jtv). Where applicable, N- and C-terminus are labeled.

and the end point of the c_l C-terminal residues. For SSEs of length l with $l < c_l$, the first and last $l-1$ residues are used for the computation of the centroids. The geometric median would probably be better than the centroid, but we discarded the idea and refer to the previous discussion (see Section 3.1.1).

We determine the angle between the vectors of contacting SSEs. Based on the angle, we assign an orientation. For an angle of 0° to $threshold_{bottom}$° we assign a parallel orientation. For an angle of $threshold_{bottom}$° to $threshold_{top}$° we assign a mixed orientation. For an angle of $threshold_{top}$° to 180° we assign an antiparallel orientation.

Implementation The implementation is a straightforward application of linear algebra. For all contacting SSEs, the angle between the SSEs is computed. Whenever an angle is to be computed, the vectors of the SSEs are computed based on the setting for c_l. The computation of the vector for each SSE is done only once and its

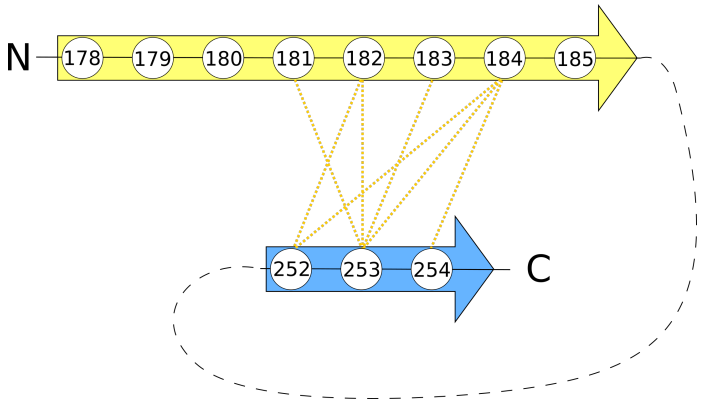


Figure 14: Visualization of residue-level contacts of 17β-hydroxysteroid dehydrogenase [2] (PDB 1jtv). Residues are depicted as white circles and labeled with a number according to the position in the sequence. A straight black line connecting residues represents peptide bonds. A dashed black line connecting residues represents multiple residues connected by peptide bonds that are not visualized. A yellow dashed line represents residues in spatial contact. The N- and C-terminus are labeled. Arrows represent residues that form a strand.

value is saved. This way, a vector for an SSE is only computed when the SSE has at least one contact and the vector is saved for upcoming computations of angles so that it does not need to be computed a second time. The values for c_l, $threshold_{bottom}$ and $threshold_{top}$ can be defined by the user in the settings of PTGLgraphComputation (see Section 2.8.4).

Application We examined 1jtv for differences between the vector and the DD method. As parameters for c_l, we tested 1, 3, 4 and 10. We chose 1 as smallest possible and 10 as large value. Note that for a c_l of 10, most SSEs will be shorter meaning the method will fall back to use all residues for the computation of the centroid except for the first and last one, respectively. The values 3 and 4 account for the number of residues per turn for different helix types. Theoretical α-, 3_{10}, and π–helices have 3.6, 3.0 and 4.4 residues per turn, respectively [120, 121]. The idea behind choosing integer values in accordance with the number of residues per turn for a helix is to centrally place the start and end points of the vector along the length axis of a helix.

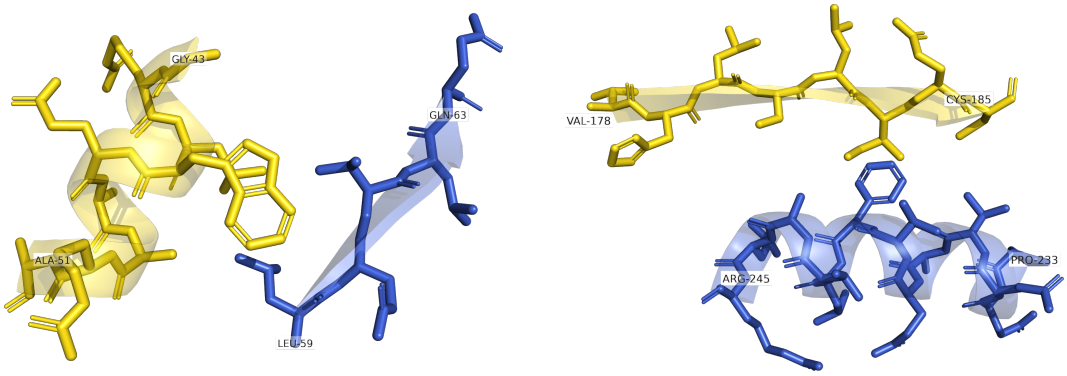
Strands six and seven that are misclassified as antiparallel by the DD method have angles between 29.04° and 30.45° depending on c_l (see Table 8). All angles are close to the ideal value of 0° for a parallel orientation. The vector method is able to correctly classify the orientation of strands six and seven. This enables detecting the SDR fold in 1jtv.

There are more edges that can be interpreted differently using the vector method. Two examples are the edges between a strand and a helix where the DD method assigns a mixed orientation, but the angles can be interpreted as antiparallel. Helix three and strand three have angles between 164.7° and 172.6° depending on c_l (see Table 8). Strand six and helix eleven have angles between 155.65° and 163.2° depending on c_l. Structurally, both orientations can clearly be classified as antiparallel (see Figure 15).

These examples show that the angles between the vectors of the SSEs match

Table 8: Angles between the vectors of chosen secondary structure elements of 17β-hydroxysteroid dehydrogenase [2]. The number (no.) of a secondary structure element refers to its occurrence ordered from N- to C-terminus. The number in brackets refers to the occurrence of the secondary structure element regarding its type: helix or strand. The angles are given in degree for different values of the number of residues taken for the computation of the start and end point of the vector (c_l).

SSE 1		SSE 2		Angles for c_l [°]			
No.	Type	No.	Type	1	3	4	10
5 (3)	Helix	6 (3)	Strand	164.70	172.60	170.18	164.76
14 (6)	Strand	17 (11)	Helix	155.65	159.98	159.06	163.20
14 (6)	Strand	18 (7)	Strand	30.45	29.04	29.97	29.35



(a) Helix three (yellow) and strand three (blue).

(b) Strand six (yellow) and helix eleven (blue).

Figure 15: Visualization of the structure of chosen secondary structure elements of 17β-hydroxysteroid dehydrogenase [2] (PDB 1jtv) in stick and cartoon representation (transparent). First and last residue of each secondary structure element are labeled with its three-letter code and its number in sequence.

angles that can be expected by visual inspection of the structure. Because of this, the vector method is more suitable for the assignment of orientations between SSEs. The only question is what values for the parameters c_l, $threshold_{bottom}$ and $threshold_{top}$ should be chosen.

We investigated all contacts between SSEs of 1jtv, antigen-presenting glycoprotein CD1d1, chain A [7] (3au1A), dihydrodipicolinate synthase, chain A [8] (3denA) and endothiapepsin, chain A [19] (5p4kA). The largest difference of an angle between the four values of c_l was 15.19° between helix 7 and 9 of 3denA. The helices are exceptionally long with a length of eight and ten residues. The angle for $c_l = 10$ differs by 11° to 13° from the other values for c_l. For shorter SSEs, $c_l = 10$ works the same as if it was set to a lower value. It makes sense that $c_l = 10$ only can take effect for SSEs longer than five. Overall, the maximum difference of the angles for the different values of c_l was 4.34° on average with a median of 3.63°.

From the results (see Appendix A.3), we decided to use $c_l = 4$, $threshold_{bottom} = 65$ and $threshold_{top} = 115$ as default parameters for PTGLgraphComputation. These parameters reproduced the assignment of orientations between SSEs from the DD method, but fixed the presented errors. The differences between the angles for

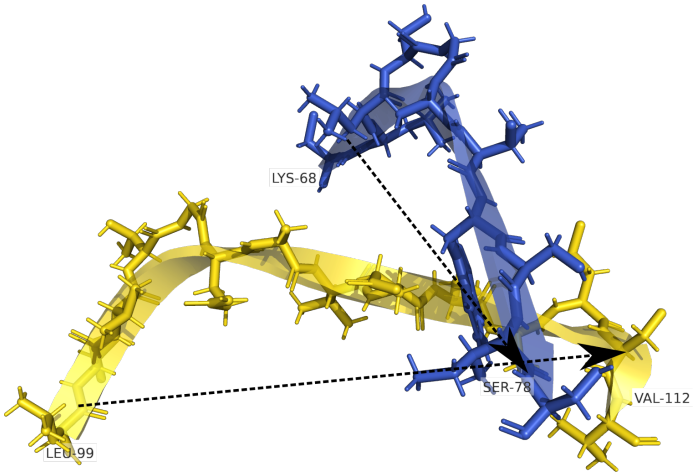
different values of c_l were not too big and notably never caused a different assignment of the orientation based on $threshold_{bottom} = 65$ and $threshold_{top} = 115$.

Note that unlike the DD method, the vector method does not differentiate between helices and strands. The DD method never assigns a mixed orientation to two strands. The vector method assigns mixed orientations to two strands with the same threshold for the angle as for two helices or a strand and a helix. The thresholds are set to prefer parallel and antiparallel orientations for a wider range of degree than mixed orientations, because parallel and antiparallel orientations are relevant for many biological motifs such as the up-and-down barrel. A mixed orientation can be thought of as a contact that clearly is neither parallel nor antiparallel.

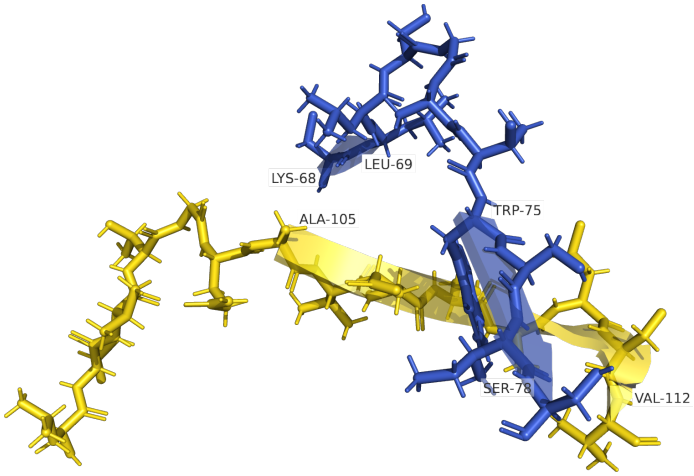
Computing the DD takes a long time, because all MCIs with one of the residues being part of one of the contacting SSEs are iterated through for each contact of SSEs (see Section 2.8.5). The vector method creates a vector once for each SSE that has a contact and computing the angle between vectors is computationally cheap. The runtime stays the same for small structures, such as the use cases for the optimization of the parameters of the vector method. This is expected as for small structures, the DD method does not take much time.

For large structures, such as 3j3q, the runtime decreases immensely from 43 hours to eight and a half hours. The computation of the complex-level PG (see Section 2.8.3) decreases from 22 hours to 90 seconds. It is important to note that this massive decrease is expected, because the vectors are created during computation of the PGs and reused for the computation of the complex-level PG.

A limitation of the vector method in its current implementation may be long, bent SSEs. We experienced such a case in 5p4kA where two of such SSEs are in contact: strand 68-78 and 99-112. Both are exceptionally long with 10 and 13 residues. Both are also exceptionally bent (see Figure 16a). The constructed vectors cannot describe the bend and distort the angle of the SSEs. To overcome this, for long SSEs, multiple vectors may be constructed for different parts of the SSE. However, we noticed that these two long SSEs may be special for the assignment by DSSP. For example, SCOT [122] differently assigns SSEs: strand 68-69, 75-78 and 105-112. PTGLgraphComputation ignores the strand of two residues, because it is too short. The remaining strands are shorter and not bent (see Figure 16b) fixing the unrepresentative vector.



(a) Strands assigned according to DSSP. Black dashed arrows indicate possible vectors for each strand.



(b) Strands assigned according to SCOT.

Figure 16: Visualization of the structure of residues 68-78 and 99-112 of endothiapepsin [19] (PDB 5p4k) in stick and cartoon representation (transparent). First and last residues of each secondary structure element are labeled with its three-letter code and its number in sequence.

3.2 Analysis of dynamics

3.2.1 Introduction

Rationale of the method

MD simulations enable exploring the dynamics of macromolecular systems *in silico* (see Section 2.3). MD simulations produce large amounts of data, because all atoms' coordinates are simulated for each time step. The structures of all time steps can be thought of as time-resolved point clouds. Points correspond to atoms and the resolution of time to the movements of the atoms over simulation time. Analyzing time-resolved point clouds in the biological context of structures is not trivial.

We aim to explore how graphs of topology can assist in the analysis of MD simulation data. We produce graphs of topology for the structural data of each time step. This allows analyzing how the graphs change over time. Graphs abstract the topology of structures and are easier to compare than point clouds.

We did not know beforehand if changes during MD simulations can be captured by PTGL's graphs of topology. If the changes of coordinates are too subtle to affect the contact threshold (see Section 2.8.2), PTGLgraphComputation will not compute different contacts resulting in no differences between the graphs. We decided to use a top-down approach starting on the level of subunits to see at which level differences in the graphs occur.

Collaboration

We tested our rationale on two data sets of Sharma et al. (see Section 2.10.2). The data set is based on 4hea. Sharma et al. discussed the results with us and guided our work.

The following subsection contains work by other group members. Please refer to Nurhassen [117] for a first analysis of the data set and intitial scripts for the analysis. Please refer to Sons [123] for the implementation of the pipeline. Please refer to Wolnitza [124] for the implementation of the heat maps and their detailed analysis. I developed the main ideas, supervised the work and helped programming. In the following, I present an overview of our analysis of dynamics and embed the analysis in a greater context.

3.2.2 Implementation

We implemented a pipeline called *PTGLdynamics*. PTGLdynamics comprises a variety of scripts for different functions. The main input are coordinate files of each time step of an MD simulation. The main output are statistics on changes of contacts and the visualization of these changes of contacts.

PTGLdynamics bases its analysis on the computations of PTGLgraphComputation for each time step. Before PTGLgraphComputation can be run, the input coordinate files are pre-processed. The input coordinate files are parsed from pseudo-legacy-PDB style (see Section 2.10.2) into legacy-PDB style (see Section 2.6.1). For each legacy-PDB file, a DSSP file is computed by DSSP version 2.0.4. PTGLgraphComputation computes all PGs and CGs.

During the implementation of PTGLdynamics, we ended our support for legacy-PDB files by PTGLgraphComputation. A new version of DSSP running on mmCIFs was not available at this time, so we use a version of PTGLgraphComputation that

works with legacy-PDB files. The versioning scheme of PTGLgraphComputation was outdated then and does not help identifying the correct version of PTGLgraph-Computation, so instead we refer to its SHA key of git (see Section 2.7): 57d87d1.

PTGLdynamics generates the following output:

- Line plot of the number of contacts throughout the simulation for all edges
- Line plot of the total number of contacts throughout the simulation
- Statistics of the number of changes of contacts throughout the simulation for all edges
- Structural visualization in PyMOL (see Section 2.7) with heat map showing changes of contacts throughout simulation

PTGLdynamics is invoked from the command line and its execution can be adjusted by command line parameters. Unlike PTGLgraphComputation, PTGLdynamics is a pipeline calling multiple single scripts. As a consequence, the user needs to pass PTGLdynamics all command-line arguments of the underlying scripts so that PTGLdynamics can pass them on. This makes the initial call of PTGLdynamics long comparatively. PTGLdynamics can also be invoked executing single sub-scripts one after another. Notably, the results of single steps of the pipeline are saved and the user can (re-)start the pipeline from any point saving runtime.

3.2.3 Heat-map visualizations

One main goal of PTGLdynamics is the analysis of changes of contacts between subunits and residues throughout the simulation. We define a change of contact if for consecutive time steps, a previously not existing residue contact is established or a previously existing residue contact disappears. In the following, we present different Heat-map visualizations highlighting the number of changes. We explored different approaches of how to measure and count the number of changes. We explain the approaches in detail in this subsection. We provide an overview of the approaches in Table 9. In total, there are eight different approaches for the visualization of changes as heat map per data set (see Section 2.10.2).

In the Heat-map visualizations, we color the structure visualized with PyMOL [96] according to the number of changes. We either color chains or residues. We assign the entity with the lowest and the highest number of changes of contacts a deep blue and a deep red, respectively. Entities of a medium number of changes are colored white. This means we always use the extremes for setting the scale for the colors.

Unlike the other visualizations of structures in this work, the following visualizations are not based on mmCIFs annotated by DSSP (see Section 2.6.2). The mmCIFs produced by the pipeline are post-processed files based on the pseudo-legacy PDB files. They contain exactly what is needed for our purpose, but did not produce correct results when fed to DSSP4. Instead we use the default classification of SSEs by PyMOL [96].

The following visualizations show the structure at 1 μs of simulation time for noQ and Qox, respectively. The position of the subunits can be seen in Figure 5a.

Table 9: Applicable settings for the computation of changes of contacts for the data set without (noQ) and with (Qox) ubiquinone. Changes can be computed based on the Complex Graph and coloring chains (chain-CG), based on residues and coloring chains (chain-res) and based on residues and coloring residues (res-res). Changes can be computed for contacts between chains (inter) or between and within chains (intra-inter). Changes can be treated as absolute changes (abs) or divided by the length of the chain (div). A checkmark (✓) marks applicable and a cross (✗) marks not applicable settings. Settings are not applicable when the division by the lengths of chains cannot be applied (*) or when no contacts within a chain are defined (**).

	noQ				Qox			
	intra-inter		inter		intra-inter		inter	
chain-CG	abs	div	abs	div	abs	div	abs	div
	✗**	✗**	✓	✓	✗**	✗**	✓	✓
	intra-inter		inter		intra-inter		inter	
chain-res	abs	div	abs	div	abs	div	abs	div
	✓	✓	✓	✓	✓	✓	✓	✓
	intra-inter		inter		intra-inter		inter	
res-res	abs	div	abs	div	abs	div	abs	div
	✓	✗*	✓	✗*	✓	✗*	✓	✗*

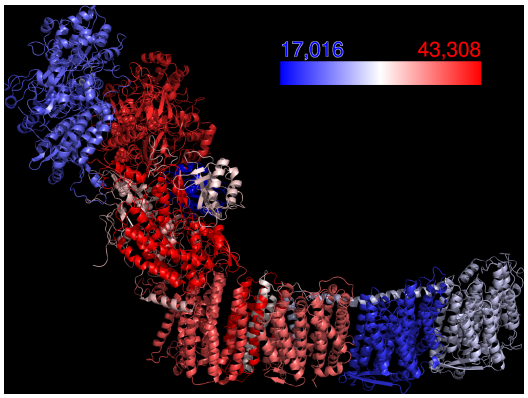
Coloring chains based on Complex Graphs

We started by comparing the CGs of the different time steps. We did not know beforehand whether the simulation leads to changes of contacts in CGs. We quickly noticed that the weights of the edges of the CGs changed during the simulation. We visualized the changes as a heat map on the structure (see Figure 17).

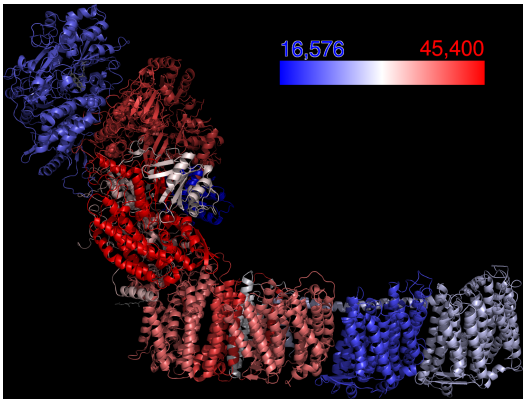
The results of the two data sets do not differ much so the following remarks apply to both. We expected less changes in the membrane than in the matrix arm, because the membrane arm consists mostly of densely packed helices. However, one part of the membrane arm accounts for the most and another for the least number of changes (see Figures 17a and b). In the matrix arm, there is one part that accounts for the most and another for the least number of changes, too. A clear difference between both arms could not be observed.

We counted the absolute number of changes. Long subunits have more residues and therefore more possibilities for contacts and changes of contacts. We created another heat map where we accounted for the length of a subunit by using a normalization dividing the absolute number of changes by the number of residues of a subunit (normalized) (see Figures 17c and d). In the normalized heat map, the N and P_D module have the least changes. Emphasis is put on the subunits Nqo7 and Nqo11 which are the only ones clearly visualized as flexible.

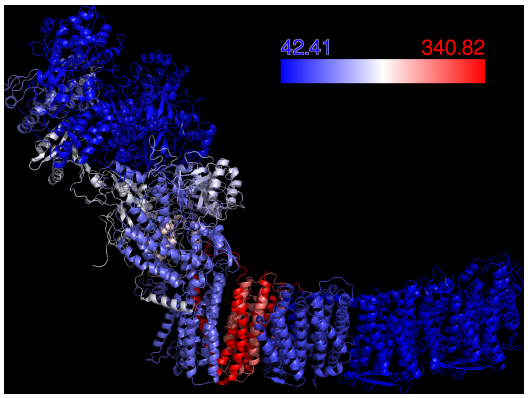
We did not further analyze the changes of contacts coloring chains based on CGs, because they are not accurate. If x contacts appear between two time steps and x different contacts disappear, the change of contacts in the CG would be zero, but we would expect it to be $2x$. To achieve the accurate number of changes, we needed to track the contacts of each residue individually.



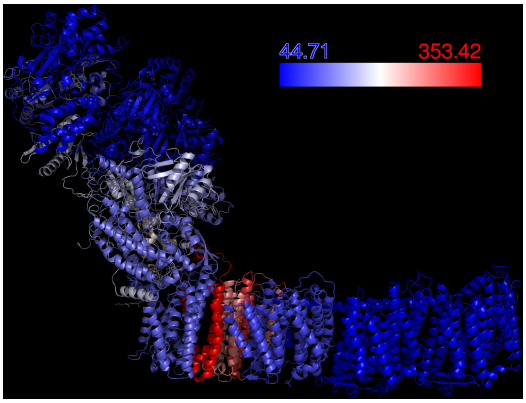
(a) Data set without ubiquinone. Absolute number of contacts.



(b) Data set with ubiquinone. Absolute number of contacts.



(c) Data set without ubiquinone. Number of contacts divided by lengths of chains.



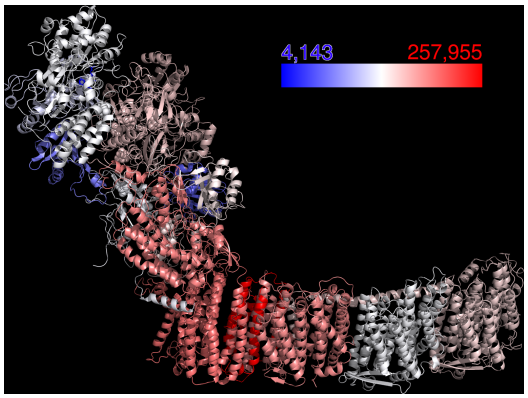
(d) Data set with ubiquinone. Number of contacts divided by lengths of chains.

Figure 17: Heat-map visualization of changes of contacts of MD simulation of respiratory complex I of *Thermus thermophilus*. Changes shown coloring chains and based on Complex Graphs. The structure is depicted in cartoon style. The number of changes of contacts is color-coded from blue to white to red depicting lowest to highest number of changes.

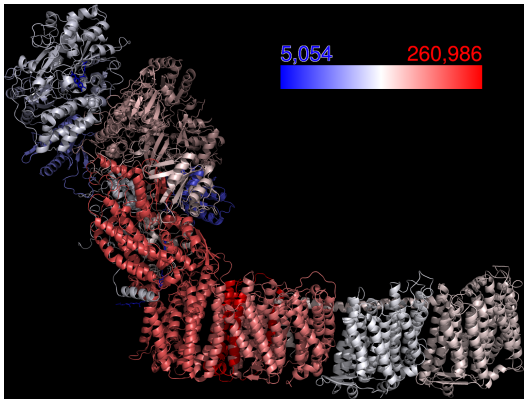
Coloring chains based on residues

Inter-chain changes For each residue, we tracked the contact partners during the simulation and counted whenever a new contact partner appeared or an existing contact partner disappeared. In the CGs, only contacts between subunits are considered, so here we just focused on contacts between residues of different subunits (inter), at first. We visualized the number of changes as a heatmap on the structure (see Figure 18).

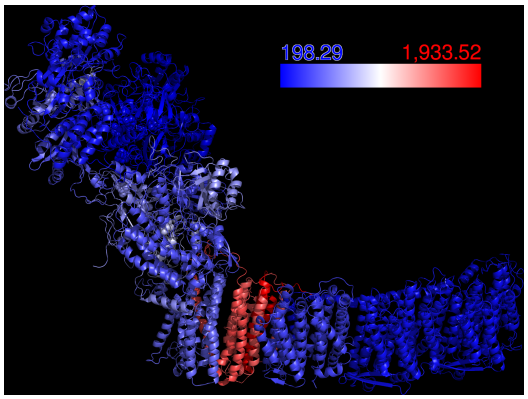
The differences between the data sets are neglectable just like for the heat map coloring chains based on the CGs. The heat map of the absolute number of changes (see Figures 18a and b) differs from the heat map coloring chains based on CGs (see Figures 17a and b). In general, more parts of the complex are neither flexible nor rigid. This means that less extremes of comparably low or high number of changes are present. There do not appear such extreme gradients as, for example, in the membrane part of the heat map coloring chains based on CGs, where rigid subunits are adjacent to flexible subunits. Here, flexible subunits can be found in the P_P



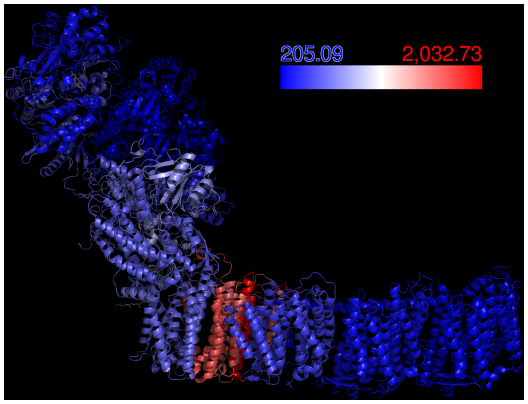
(a) Data set without ubiquinone. Absolute number of contacts.



(b) Data set with ubiquinone. Absolute number of contacts.



(c) Data set without ubiquinone. Number of contacts divided by lengths of chains. Excluded FMN and ubiquinone.



(d) Data set with ubiquinone. Number of contacts divided by lengths of chains. Excluded FMN and ubiquinone.

Figure 18: Heat-map visualization of changes of inter-chain contacts of MD simulation of respiratory complex I of *Thermus thermophilus*. Changes shown coloring chains and based on residues. The structure is depicted in cartoon style. The number of changes of contacts is color-coded from blue to white to red depicting lowest to highest number of changes.

and Q module, such as, FMN that is treated as separate chain (see Section 2.10.2), TTHA1528 and Nqo15.

The normalized heat map allows almost no differentiation between the subunits (see Figures A2a and b). The ligands FMN and ubiquinone are treated as chains of length 1 which means their number of changes is the same for the absolute and normalized number. The number of changes for ubiquinone is 17,257. The highest absolute number of changes for Qox occurs for Nqo10 with 260,989. After dividing by the length of Nqo10, these are 1,631 changes. The difference between the ligands of length 1 and much longer protein chains distorts the normalized numbers.

Since the heat maps do not allow differentiation, we excluded the ligands FMN and ubiquinone from the heat maps (see Figures 18c and d). This leads to more differentiation within the color scale. Interestingly, almost the whole complex is defined as rigid, meaning only a comparably low number of normalized changes occurs. Depicted as flexible are Nqo11, Nqo7 and Nqo10 in descending order of the

number of normalized changes. The heat map is similar to the heat map coloring chains based on the CGs for normalized changes. It is possible that the case of inaccuracy in the number of changes based on CGs does not have as big of an impact as expected. Nevertheless, tracking the number of changes on residues ensures correct results and allows for the inclusion of changes of contacts within a subunit, as well.

Intra- and inter-chain changes The heat maps of the absolute number of changes between chains and within chains (intra) of noQ and Qox (see Figures 19a and b) differ only slightly and will be discussed together. The heat maps of the intra-chain and inter-chain changes differ from the heat map of the inter changes (see Figure 18). In the inter-changes heat map, the largest part of the complex is between little flexible or flexible. In the inter-changes and intra-changes heat map, the largest part of the complex is neither rigid nor flexible. Emphasis is put on a few subunits that are clearly rigid or flexible.

The highest number of changes occurs for Nqo3 of the N module which is clearly visible in the heat map. The second highest number occurs for Nqo12 of the P_D module which is visible as slightly flexible. Low numbers of changes occur for different subunits of the N, Q and P_P module which are depicted as very rigid.

The heat map of the normalized number of inter-chain and intra-chain changes allowed no real differentiation, because of the ligands FMN and ubiquinone (see Figures A2c and d). Excluding the ligands allows for a more meaningful color scale (see Figures 19c and d).

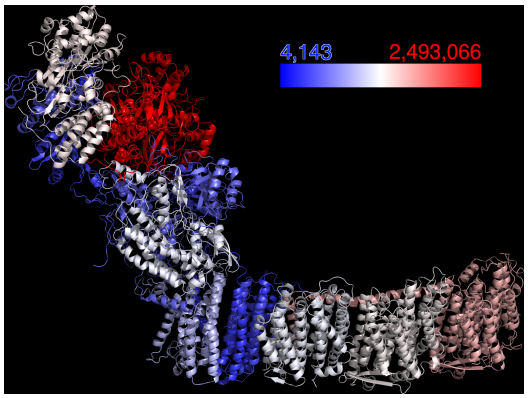
Almost the whole membrane arm is rigid whereas the matrix arm is completely flexible. The flexibility of the membrane arm has been linked to its function as a proton antiporter [12, 125], but this is not detectable in an MD simulation. Over the last decades, transmembrane helices have been identified as more flexible than initially thought [126, 127]. It seems that the flexibility is either not captured by the MD simulation at all or not as well as in the matrix arm. For an MD simulation it makes sense that the helices of the membrane arm are relatively densely packed and less mobile than the matrix arm.

An exception is Nqo11 of the P_P module which is the only subunit of the matrix arm depicted as flexible. For Nqo11, we detected 300,276 changes of contacts in the noQ which is the lowest number except for 4,143 changes for FMN. Nqo11 has 95 residues which is the lowest number of all protein subunits. After normalization, Nqo11 has the third highest number of changes. Considering its length, Nqo11 has a large number of changes of contacts and stands out from the rest of the membrane arm.

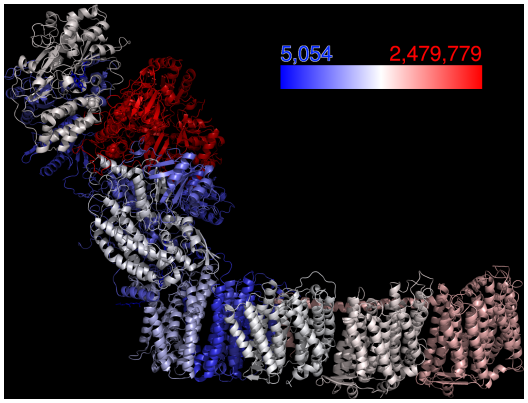
Coloring residues based on residues

Tracking the contact partners of each residue resulted in the exact number of changes of contacts for each chain and enabled us to include intra-chain contacts. As a next step, we also visualized the changes by coloring residues. Instead of coloring subunits, each residue is colored individually. Normalization for the number of residues of a chain cannot be applied here (see Table 9).

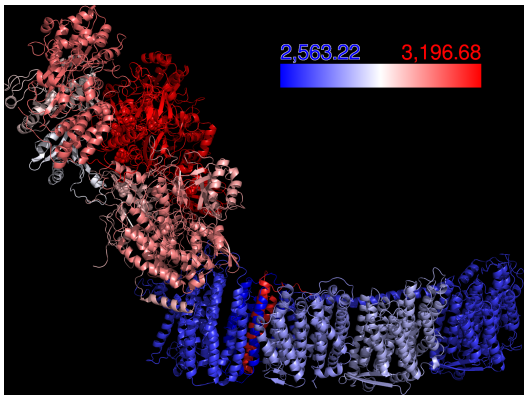
Inter-chain changes We visualized the number of changes of inter-chain contacts as a heat map on the structure (see Figure 20). Note that not all residues are colored blue to white to red in the heat map scheme. We merely count inter-chain contacts.



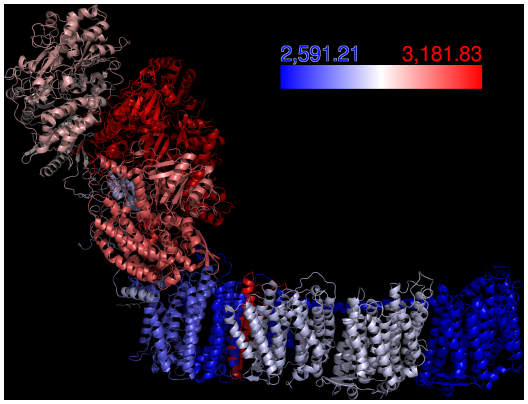
(a) Data set without ubiquinone. Absolute number of contacts.



(b) Data set with ubiquinone. Absolute number of contacts.



(c) Data set without ubiquinone. Number of contacts divided by lengths of chains. Excluded FMN and ubiquinone.



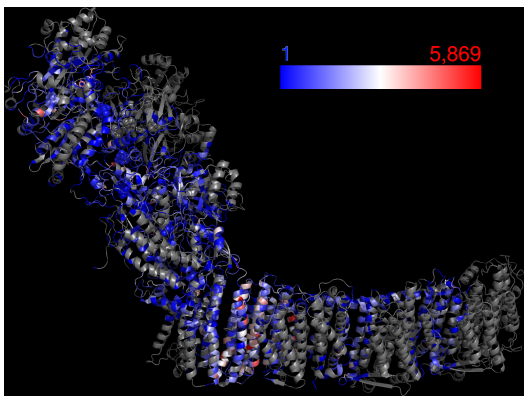
(d) Data set with ubiquinone. Number of contacts divided by lengths of chains. Excluded FMN and ubiquinone.

Figure 19: Heat-map visualization of changes of intra-chain and inter-chain contacts of MD simulation of respiratory complex I of *Thermus thermophilus*. Changes shown coloring chains and based on residues. The structure is depicted in cartoon style. The number of changes of contacts is color-coded from blue to white to red depicting lowest to highest number of changes.

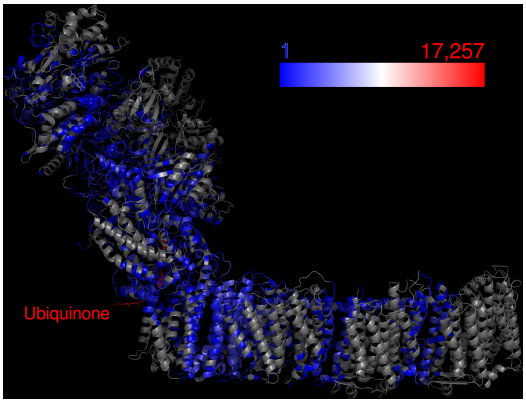
Residues that are buried inside a subunit may have no changes of contacts over the whole simulation. These residues are colored grey.

Most of the residues that have changes of contacts are depicted as rigid, because of the high number of changes for single residues that stretch the color scale. For noQ, these are, for example, residues from the subunits Nqo11, Nqo7, Nqo14 and Nqo10 from the P_P module (see Figure 20c). For Qox, the only molecule depicted as flexible is ubiquinone, which is treated like a residue by PTGL (see Figure 20d).

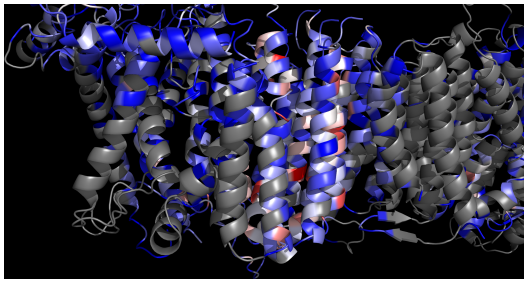
The number of changes for ubiquinone is 17,257 and the second-highest number of changes is 6,418 of residue 59 of subunit Nqo11. For the normalized number of changes, we excluded the ligands, because they stretched the color scale, not allowing differentiation among the other molecules. Here, the case might be different, because the difference really is due to the higher number of changes and not only because of normalization. The heat map directly shows that the highest number of changes of contacts over the simulation occurs for ubiquinone, which might be the desired



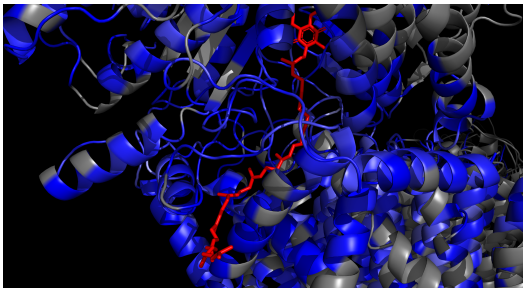
(a)



(b) Ubiquinone labeled in red.



(c) Zoom-in on P_P module.



(d) Zoom in on ubiquinone (red).

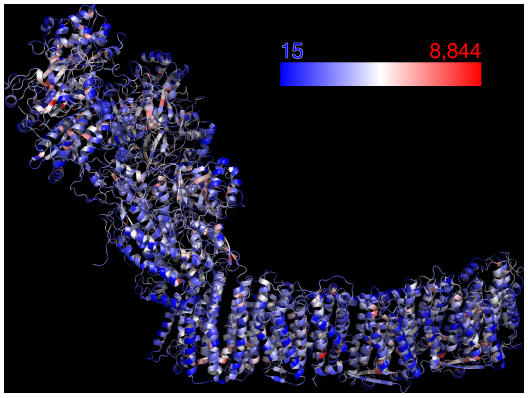
Figure 20: Heat-map visualization of changes of inter-chain contacts of MD simulation of respiratory complex I of *Thermus thermophilus*. Data set without (a and c) and with ubiquinone (b and d). Changes shown coloring residues based on residues. The structure is depicted in cartoon style. The number of changes of contacts is color-coded from blue to white to red depicting lowest to highest number of changes. Residues without changes of contacts are colored grey.

result. However, ubiquinone with its 63 non-hydrogen atoms is a larger molecule than residues of 4 to 14 non-hydrogen atoms. This allows for more contacts and changes of contacts. Moreover, all contacts of ubiquinone are counted as inter-chain contacts, because ubiquinone is treated as a separate chain. For residues, on the other hand, only contacts and their changes to residues of different subunits are counted. We also excluded ligands for the Qox which allowed more differentiation. This resulted in a heat map more similar to the heat map of noQ (see Figure A3).

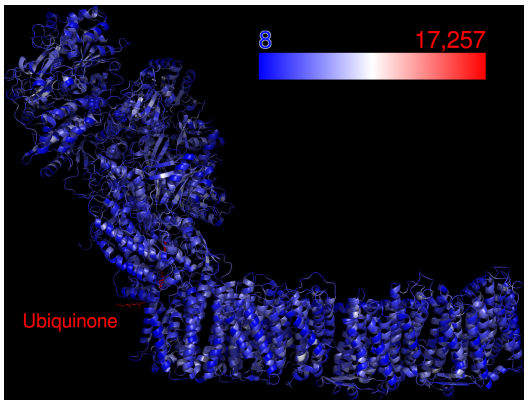
Inter-chain and intra-chain changes We visualized the number of changes of intra-chain and inter-chain contacts as a heat map on the structure (see Figure 21). Opposite to only visualizing inter-chain contacts, here, every residue and ligand is assigned a color from blue to white to red.

For the noQ, most of the complex is depicted rigid or neither rigid nor flexible, and some residues are highlighted as flexible (see Figure 21a). The residue with the highest number of changes of intra-chain and inter-chain contacts is phenylalanin at position 63 of TTTHA1528 (see Figure 21c). This residue is in a loop region which seems to be flexible as many contacts change during simulation.

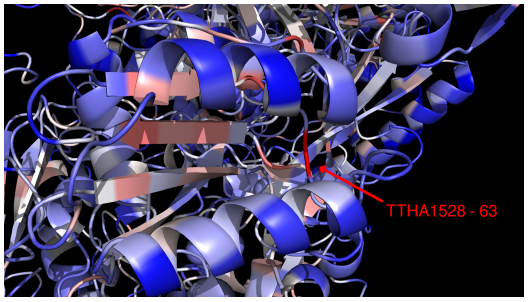
Interestingly, the visualization matches the expectation that residues that are close to other residues generally have a higher number of changes of contacts, for



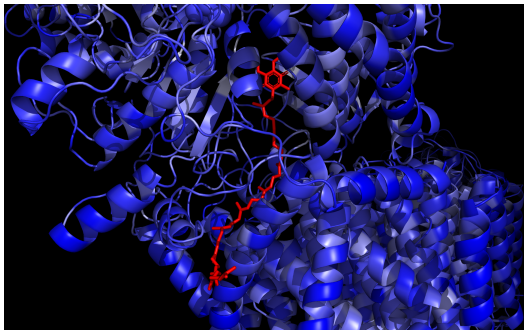
(a)



(b) Ubiquinone labeled in red.



(c) Zoom-in on TTHA1528 phenylalanin at position 63.



(d) Zoom-in on ubiquinone (red).

Figure 21: Heat-map visualization of changes of intra-chain and inter-chain contacts of MD simulation of respiratory complex I of *Thermus thermophilus*. Data set without (a and c) and with ubiquinone (b and d). Changes shown coloring residues based on residues. The structure is depicted in cartoon style. The number of changes of contacts is color-coded from blue to white to red depicting lowest to highest number of changes.

example, for helices. Residues that point towards other SSEs are more likely flexible or neither rigid nor flexible while residues pointing away are more likely rigid. In strands, there are parts which are more flexible or less flexible as well.

For the Qox, ubiquinone is the only molecule or residue depicted as flexible (see Figures 21b and d), just like the inter-chain contacts. A difference is that the gap between ubiquinone and the residue with the second-highest number of changes is lower, so that there are residues that are depicted as neither rigid nor flexible. This is likely, because we now count all contacts and their changes for residues, too, not only for ubiquinone due to the fact that it is placed in it a separate chain.

We excluded ligands during testing. The heat map was similar to the noQ (data not shown). This was expected, because, for example, for both data sets the residue with the most changes is residue 63 of TTHA1528.

Discussion of the approaches We presented a plethora of different approaches (see Table 9) visualizing dynamics within a complex during MD simulation. Different questions can be addressed using heat maps of changes of contacts. Common to all is to highlight the parts of the complex that are flexible so that contacts appear and

disappear more frequently, colored in red, and inflexible parts, colored in blue. The heat maps help to quickly identify outstanding regions.

The heat maps coloring chains based on CGs were produced as a first impression of whether such visualizations can be created by the definition of contacts of PTGL. They showed rigid or flexible parts allowing a quick overview of flexibility in a complex. The normalized heat maps put emphasis on a smaller number of subunits that were depicted as flexible. It is important to note that only inter-chain contacts are counted, which may or may not be desired, depending on the intentions of the visualization. This approach should not be used further, because basing the number of changes on the CGs is inaccurate.

The remaining approaches were based on contacts of residues and reflected the accurate number of changes. The inter-chain heat maps of absolute contacts differed from the heat maps based on the CGs and can be considered as more accurate. Here, the rigid parts were less frequent and more outstanding than the flexible parts. For the normalized heat maps, we excluded the ligands that were placed in separate chains to allow differentiation. Interestingly, the normalized heat maps without ligands were similar to the heat maps based on the CGs: Three subunits of many changes are highlighted as flexible while the rest is rigid. It is unclear whether this is a desired result or an undesired consequence of the normalization.

Basing the heat maps on contacts of residues, allowed us to treat intra-chain contacts, as well. Depending on what is of interest, this provides a more accurate picture of general flexibility. The absolute heat maps make it easy to spot flexible and rigid regions while the majority of the complex is depicted neither rigid nor flexible. The normalized heat map indicated that the membrane arm is less flexible than the matrix arm, with one exception. This could mean that the normalization makes sense only when treating intra-chain and inter-chain contacts.

Basing the heat maps on contacts of residues also allowed us to visualize the changes of contacts coloring residues. The inter-chain heat maps allow the identification of flexible regions in interfaces of subunits. The intra-chain and inter-chain heat maps allow the identification of flexible regions in general. For the Qox, ubiquinon was outstandingly flexible. It might be desirable to normalize against the number of atoms of a molecule to account for the larger size of ligands, such as ubiquinone.

The approaches based on residues can all be useful depending on what the focus of the visualization should be. Then, the visualizations help to quickly identify interesting subunits and molecules, for example, around the ubuquinone tunnel.

3.2.4 Line plots of number of contacts

The heat maps allow instant spotting of regions of low or high changes of contacts. They contain the information of all time steps in one visualization. We created line plots of the number of contacts between subunits to analyze the course of the simulation. This enables spotting interesting time steps for contacting subunits, for example, for Qox at 828 and 1,245 μs for the contacts between subunits Nqo3 and Nqo5 (see Figure 22).

The number of contacts fluctuates around 10 to 15 contacts for consecutive time steps (see Figure 22a). We visualized the mean of respectively 50 time steps with the windowing technique. The mean starts at around 37 and drops with fluctuations to its lowest at around 800 μs. The absolute low point is at 828 μs with an edge weight of eight. The number of contacts increases after that up to the maximum turning point of 48 contacts at 1,245 μs.

There is a difference of 40 contacts between the time steps 828 and 1,245 μs. The difference is caused by a large loop region of Nqo5 that engages in much more contacts at 1,245 μs (see Figures 22b and c). The visualization as a line plot allows the analysis of the course of the edge weight during the simulation and instantly spotting both prominent time points.

3.2.5 Comparison of both data sets

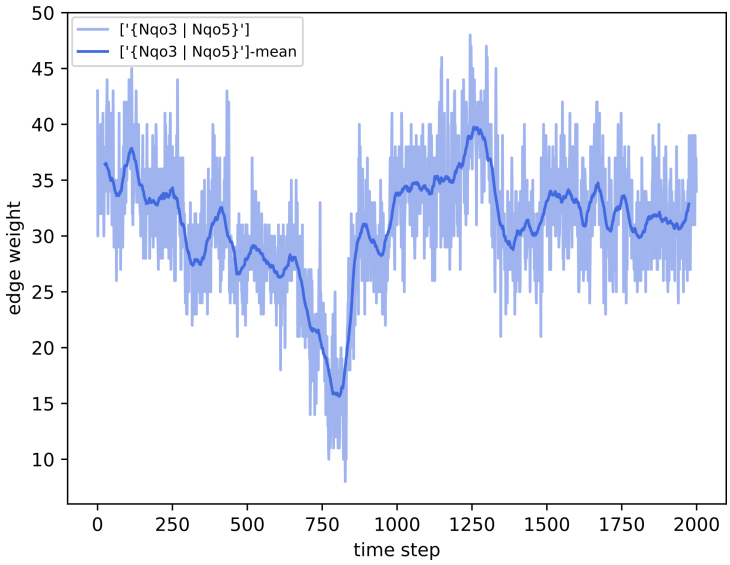
The discussed heat maps and plots included information about one data set. The following plot and heat map shows a direct comparison of noQ and Qox. We implemented scripts for automated comparison of multiple data sets, but did not add them to the standard run of the pipeline, yet.

We implemented a script to visualize the sum of changes per residue of a subunit, for example, Nqo7 (see Figure 23). We compared the two data sets, noQ and Qox. The line plot contains the same information as the heat maps, but enables a direct comparison of both data sets and inspection of the changes of each residue independent of the structure and color.

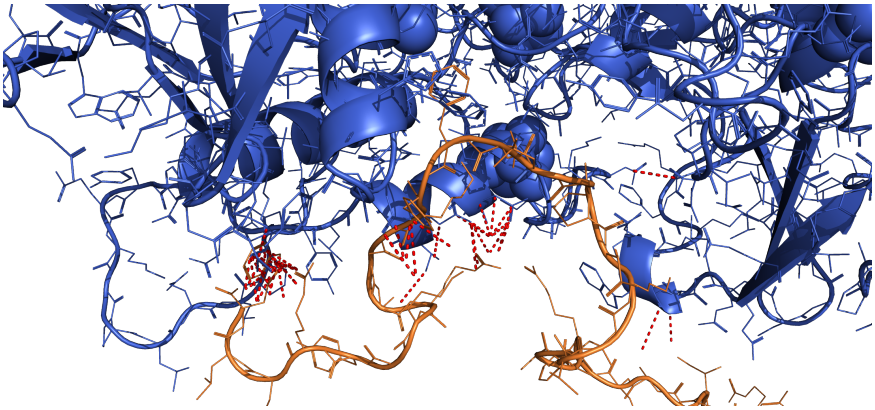
For many residues, the line plot shows a comparable trend, for example, the residues 78 to 114. We can see that for the noQ, there are spikes higher than for Qox, for example, at residues 54, 62 and 69. And there are spikes lower, for example, at the residue 59.

We embedded the comparison of both data sets in a heatmap, too (see Figure 24). We visualized the number of changes coloring residues based on residues. For each residue, we subtracted the number of changes for noQ by the number of changes for Qox. We assigned a color from blue to white for numbers less than zero, white for zero and from white to red for greater than zero. This heat map does not show which residues have low or high number of changes over the simulation within one data set but between the data sets.

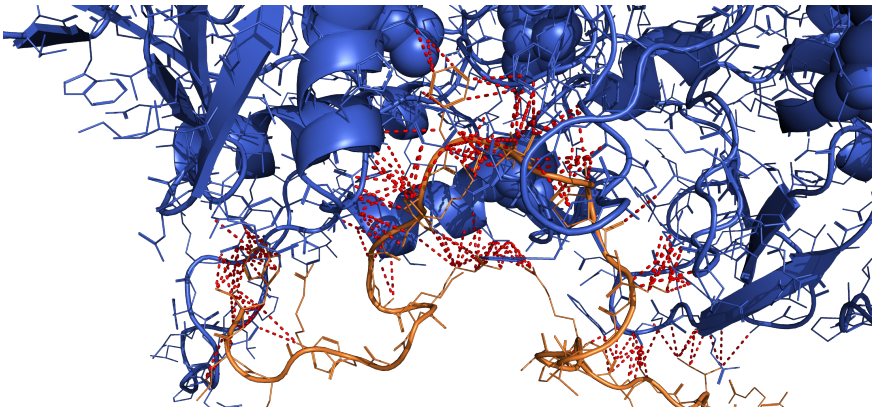
We can see that most residues have comparable numbers of changes (see Figure 24a). A few residues exist, which have a higher number of changes in one data set. The only difference between the simulations is the placement of ubiquinone in its tunnel (see Section 2.10.2). Close to ubiquinone, we can see rather many differences between the data sets (see Figure 24b). There are residues with a higher number of changes in both noQ and Qox



(a) Number of contacts per time step (light blue) and as mean of 50 time steps with windowing technique (dark blue). Plot created with Matplotlib [119].



(b) 828 μs



(c) 1,245 μs

Figure 22: Line plot of the number of contacts of Nqo3 and Nqo5 across the simulation (a). Structural visualization of the interface of Nqo3 (blue) and Nqo5 (orange) (b and c). Structure depicted as cartoon and lines with ligands as balls. Distances between atoms of the subunits are represented as red dashed lines.

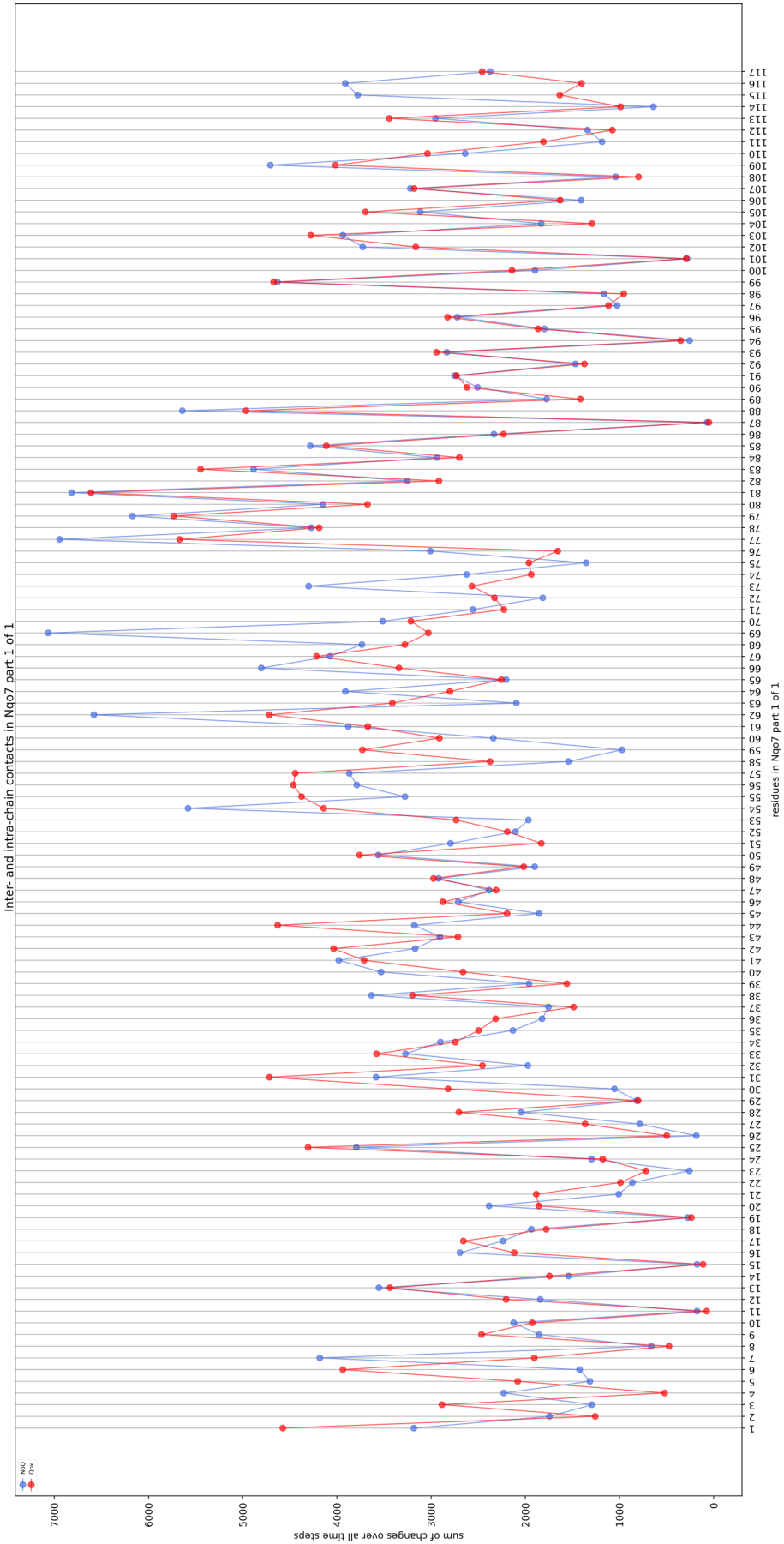
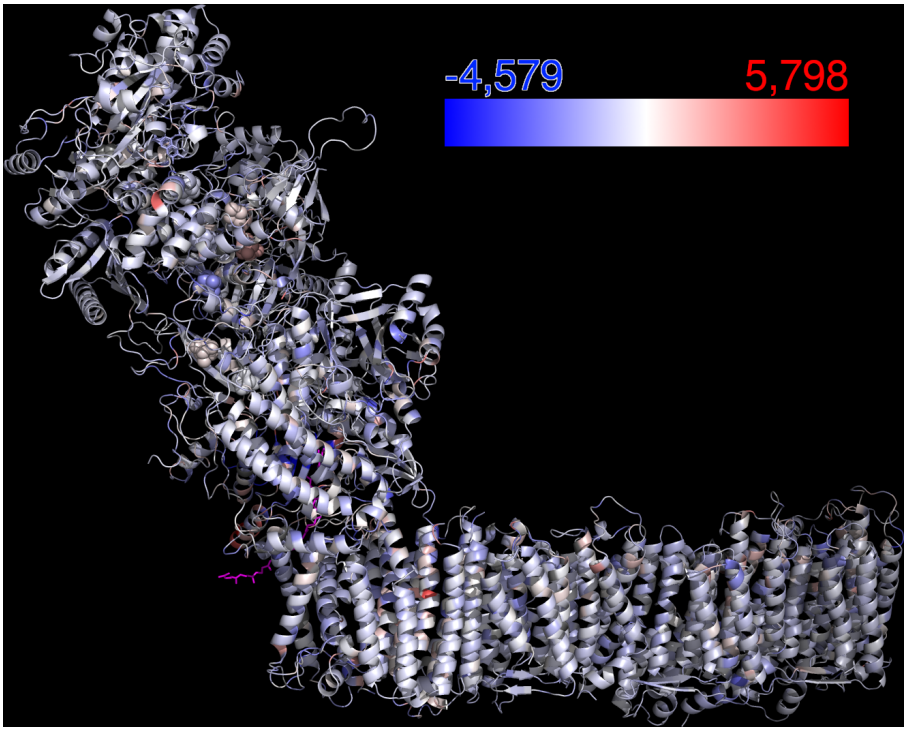
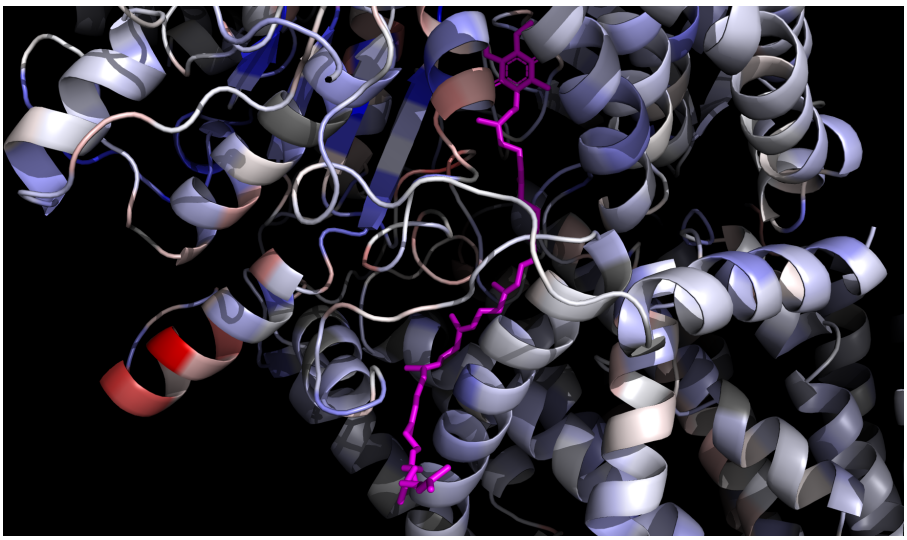


Figure 23: Line plot of the sum of changes over the simulation of the data set without (blue) and with ubiquinone (red) per residue of Nqo7. Plot created with Matplotlib [119].



(a) Whole complex.



(b) Zoom-in on ubiquinone.

Figure 24: Heat-map visualization of changes of intra-chain and inter-chain contacts of MD simulation of respiratory complex I of *Thermus thermophilus*. Comparison of the data set without (noQ) and with ubiquinone (Qox). Changes shown coloring residues based on residues. The structure is depicted in cartoon style. The number of changes of contacts is color-coded from blue to white to red, depicting more changes in noQ, equal changes and more changes in Qox, respectively. Ubiquinone is depicted as sticks in magenta.

3.2.6 Limitations of the pipeline

Implementation

The pipeline reads in and post-processes coordinate files, computes PGs and CGs, counts changes based on CGs and residues' contacts, and visualizes the results in line plots and heat maps. The computation is fully automated with some restrictions. Executing the full pipeline, the program call can become long, because all the parameters need to be passed to the subscripts at once. Using different version of PTGLgraphComputation makes it even more difficult.

The recent version is used for creating the lists of contact partners of residues and getting the lengths of the chains for normalization. This version uses the user's `PTGLgraphComputation_settings.txt` in the home directory while the old version still uses the hidden settings file `.plcc_settings.txt`. It may be confusing for the user which setting file is used for which task.

Running the full pipeline once to create all the results shown here is not possible, at the moment. After some steps, the pipeline needs to be split to apply different settings and continue with the respective output. For example, the intra-chain and inter-chain contacts need to be computed with different settings of PTGLgraphComputation. The absolute and normalized changes are also computed with different calls of the subscript `calculateChanges.py`.

This alone might be acceptable if splitting the pipeline was easy. Currently, either all or no arguments can be passed by PTGLdynamics to a subscript. Ideally, PTGLdynamics only overwrites the settings passed by the user and uses the default values of the pipeline for the rest. Some settings of the pipeline cannot simply be overwritten by the user, for example, some input directories. The pipeline assumes the correct input directories when everything is run at once, but does not allow to pass the correct input directories for some of the subscripts.

Content

PTGLdynamics is based on PTGLgraphComputation concerning what can be parsed and computed. For some molecules, a different interpretation might be desired. For example, placing some ligands in separate chains has a huge impact on inter-chain heat maps. Also, the interpretation of large ligands and their number of contacts in the context of residues' contacts is not trivial. It might be useful to enable different scales of the color scheme to allow better differentiation, for example, by identifying and differently coloring outliers.

PTGLgraphComputation is currently able to handle polypeptide chains, RNA chains and ligands, such as metals or ions. The program lacks an interpretation of membranes. For considering the flexibility of residues of rCI, the membrane might play an important role. The program also lacks an interpretation of water. For rCI, it has been found that water molecules play an important role for function [128]. Treating these cases differently would give a more complete view of what happens with the complex during MD simulation.

The changes of contacts described above throughout the simulation are based on the CGs and residue-wise contacts. We also shortly explored how the PGs of the single subunits change over the course of the simulation. We noticed that the PGs can drastically change over the simulation. One thing we noticed are appearing and disappearing contacts between SSEs. Another thing is that based on the assignment of DSSP, SSEs can become shorter, longer, break up or merge. To analyze changes of

contacts coloring SSEs, we need to track exactly, which residues are part of a certain SSEs to correctly interpret contacts. Currently, the pipeline is not able to do this.

3.3 Normalization of edge weights of Complex Graphs

The edge weights of CGs correspond to the number of contacts of molecules between subunits (see Section 2.8.3). Long chains (either protein or RNA) contain more molecules (either residues or nucleotides) and have therefore more possibilities for contacts of molecules. Edge weights are an expression of how close or intertwined the subunits are spatially. We introduce normalizations to normalize the number of contacts by the number of the molecules of the participating chains in different ways (see Table 10).

Table 10: Name, short name (tag) and equation of all types of normalization of edge weights of Complex Graphs. The equations output the respective normalized weight between two chains i and j, and use the number of contacts between the chains ($c_{i,j}$) and the lengths of the participating chains, l_i and l_j.

Name	Tag	Equation
Absolute	abs	$c_{i,j}$
Additive	add	$\frac{c_{i,j}}{l_i+l_j}$
Multiplicative	mult	$\frac{c_{i,j}}{l_i*l_j}$
Square root	sqrt	$\frac{c_{i,j}}{\sqrt{l_i+l_j}}$
Logarithm	log	$\frac{c_{i,j}}{log_{10}(l_i+l_j)}$
Minimum	min	$\frac{c_{i,j}}{min(l_i,l_j)}$

The normalizations are implemented in PTGLgraphComputation (see Section 2.8) in a class called `ComplexGraphEdgeWeightTypes`. `ComplexGraphEdgeWeightTypes` contains the `enum EdgeWeightType`. Edge weight type refers to the absolute edge weight and all normalizations. For each edge weight type, a function for the computation is deposited as `case` of a `switch-case` block. This enables an easy adjustment and addition of normalizations at one place. The weights are stored as `BigDecimal` allowing an arbitrary precision and lossless computations. By default, we use a precision of 35.

In the following subsections, we apply and evaluate the normalized weights in the context of the detection of modules (see Section 3.4) and the prediction of the assembly pathway (see Section 3.5).

3.4 Prediction of modules

Rationale of the method

Large multimeric protein complexes can be devided into modules. Modules can be seen as functional and structural units and units of the assembly pathway. We applied graph clustering to the CG of a protein complex to investigate how clusters relate to modules.

We used the Leiden algorithm [66] (see Section 2.1.2), as it successfully extends the widely used Louvain algorithm [65]. An important feature of any algorithm for our purpose is the treatment of edge weights, because they carry the information of how extensive an interface between subunit is. We tested different normalizations of the edge weights (see Section 3.3) to account for different lengths of subunits. The algorithm is unsupervised, meaning it does not need the number of requested clusters as input. This way, the detection of modules can be applied without prior knowledge and solely relies on the topology of the subunits of the complex.

The Leiden algorithm is one of many algorithms that clusters by optimizing the modularity of the partition. The underlying assumption of applying such an algorithm for our purpose is that the partition where each cluster corresponds to one module (partition according to the modules) has the optimal modularity. When comparing the different normalizations of edge weights, we favor the normalization where the modularity of the partition according to the modules is the optimal or closest to the optimal modularity. We assume that such a normalization is better fit to identify the modules using a graph clustering algorithm optimizing for the modularity.

Collaboration

The following subsection is based on the work of Zunker [129]. I developed the main idea of identifying modules by graph clustering of CGs and supervised the work. In the following, I present an overview of our detection of modules as well as a deeper analysis and embed the analysis in a greater context.

3.4.1 Complex I of *Thermus thermophilus*

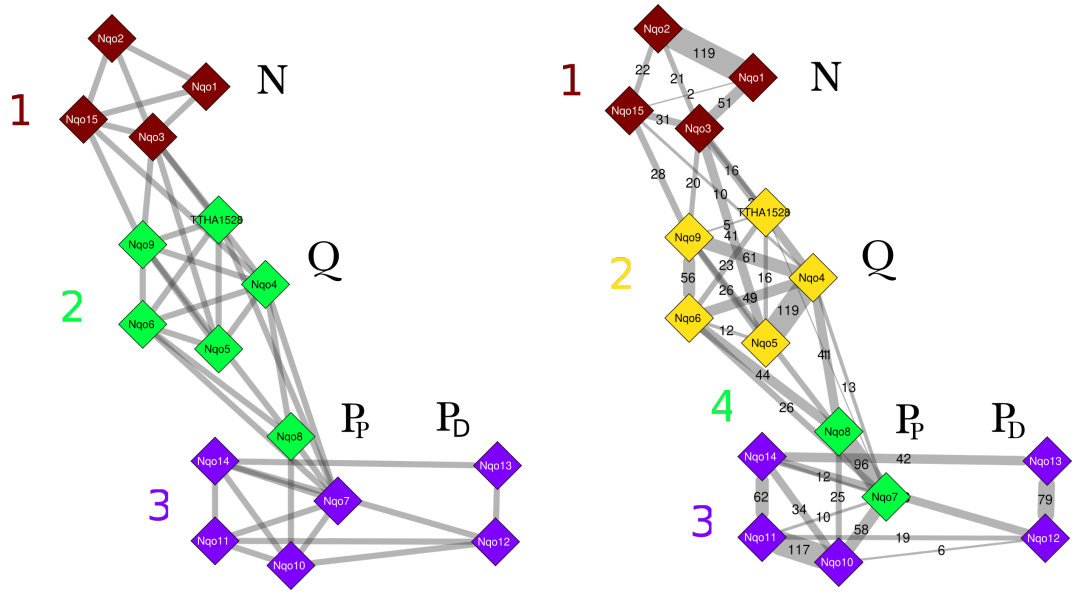
We applied graph clustering to the CG of *Thermus thermophilus* (see Section 2.10.3) using no edge weights, absolute, additive and multiplicative edge weights (see Figure 25). The partition was the same for the absolute, additive and multiplicative weights (see Figure 25b) and will be simply referred to as partition using weights.

Three clusters were assigned using the edges without weights and four clusters using weights. The N module was identified for both using the edges without and with weights. Cluster 2, using the edges without weights, contains all vertices of the Q module and one vertex of the P_P module. Cluster 3 contains the remaining vertices of the P_P module and the two vertices of the P_D module.

The partition using weights correctly identified all the chains belonging to the Q module, because the cluster did not contain the single vertex from the P_P module. However, this vertex forms a separate cluster with another vertex from the P_P module. The remaining vertices are part of cluster 3. Clusters 3 and 4 do not correspond to the P_P and P_D module, but together they contain all vertices of the membrane arm.

The partition using the edges without weights was surprisingly good at identifying one module, almost another module and all but one subunits of the matrix arm. Edge weights of a CG contain the information on the intensity of the contact between two subunits, which is important information for detecting modules in a structure. It seems that the topology of the CG alone is already a good indicator for modules.

The partition using edge weights was even better as it contained two out of four modules correctly. Here, vertices of the membrane arm and of the matrix arm are not mixed together in the same clusters. It seems to be more challenging to detect the clusters corresponding to the membrane arm than to the matrix arm. The two



(a) Partition without edge weights.

(b) Partition for absolute, additive normalized and multiplicative normalized edge weights. The edge label and width correspond to the absolute edge weight.

Figure 25: Visualization of the partition of the Complex Graph of respiratory complex I of *Thermus thermophilus* [12] (PDB 4hea). Vertices correspond to subunits and edges to spatial neighborhoods. Vertices are labeled with the abbreviated gene names of the subunits. Vertices belonging to the same module are grouped together and labeled (N, Q, P_P, P_D). Vertices belonging to the same cluster are colored the same and labeled with numbers.

vertices of the P_D module are assigned to the cluster of the P_P module, and two vertices, Nqo6 and Nqo7, form a separate cluster.

Placing Nqo13 and Nqo12 in the same cluster as Nqo14 is favorable for the algorithm, because of the high edge weights of {Nqo13,Nqo14} and {Nqo12,Nqo14} of 42 and 36, respectively, for absolute weights. Additionally, Nqo12 is connected to Nqo10 and Nqo11 through edges with lower edge weights of 6 and 19. Nqo7 and Nqo8 are placed in a separate cluster, because they are connected by an edge of one of the highest weights of the whole graph. The edge has an absolute weight of 96 which is the fourth highest edge weight in the graph. At the same time, both have edges to vertices of the Q and the P_P module. For the modularity, it is more beneficial to place Nqo7 and Nqo8 in a separate cluster than placing them in the cluster that corresponds to the Q or the P_P module.

The question arises if a clustering algorithm optimizing the modularity can correctly identify the modules. We ensured that the resulting partition exhibits the maximum modularity with the function `partition_opt_modularity` of igraph (see Section 2.7). The modularity of the partition according to the modules is lower than for the partition of the clusterings for all weight types (see Table 11). This means that an algorithm optimizing the modularity will not output a partition corresponding to the modules.

Table 11: Modularities and differences of modularities for different partitions and weight types for respiratory complex I of *Thermus thermophilus* [12] (PDB 4hea).

Edge Weight Type	Predicted Clusters	Modules	Difference
No weight	0.36906	0.29750	0.07156
Absolute	0.45856	0.43416	0.02440
Additive	0.43905	0.43672	0.00233
Multiplicative	0.41805	0.32375	0.09430

The difference between the modularity for the modules and for the clusters is lowest for the additive and largest for the multiplicative normalization. This indicates that edge weights and a suitable normalization help to achieve the goal that a high modularity corresponds to biologically motivated modules. In this case, the additive normalization seems to work best even though the partitions of the absolute edge weight and all normalizations are equal.

3.4.2 Complex I of *Homo sapiens*

We applied graph clustering to the CG of rCI of *Homo sapiens* (see Section 2.10.4) using no edge weights, absolute, additive and multiplicative edge weights (see Figure 26). The subunits S6 and A12 are shared between the N and Q modules (see Section 2.10.4). For the computation of the modularity for the modules, we need a partition. If a non-overlapping matching of the subunits to modules exists which is just unresolved so far, there are four possibilities: Both S6 and A12 are either part of the N or the Q module or each of them is part of one of the modules and vice versa. Because all partitions of all types of edge weights put S6 and A12 to the vertices of the Q module, we consider both as part of the Q module for the computation of the modularity.

The partitions without edge weights, for additive and multiplicative edge weights were not able to reproduce modules. The partitions for additive and multiplicative edge weights differ only in one vertex: ND2 (see Figure 26c and 26d). For both edge weight types, cluster 1 contains all but three vertices from the N and Q module. V1, V2 and V3 form a separate cluster due to high edge weights between them.

Cluster 2 contains vertices only of the P_P module and cluster 5 only of the P_D module. Both do not contain all vertices of the respective modules, because for each module, some vertices are part of cluster 3.

The clustering using the additive and multiplicative weights did not work well. It is remarkable that no vertices of the matrix arm and the membrane arm are mixed. Beside that, the N and Q module are unresolved and the N module is further split. As seen for rCI of *Thermus thermophilus*, the detection of modules belonging to the membrane arm seems to be more difficult by clustering, because the modules are completely unresolved.

The partition without edge weights performed better, because all vertices of the matrix arm are aggregated in cluster 1 without splitting up the N module (see Figure 26a). Cluster 2 contains vertices only of the P_P module and cluster 3 contains all vertices of the P_D module. Cluster 3 contains vertices that should be part of cluster 2 according to the modules.

The partition for absolute edge weights (see Figure 26b) performed best. Three

of four modules are correctly identified: N, Q and P_D. The P_P module is split across the two clusters 3 and 4.

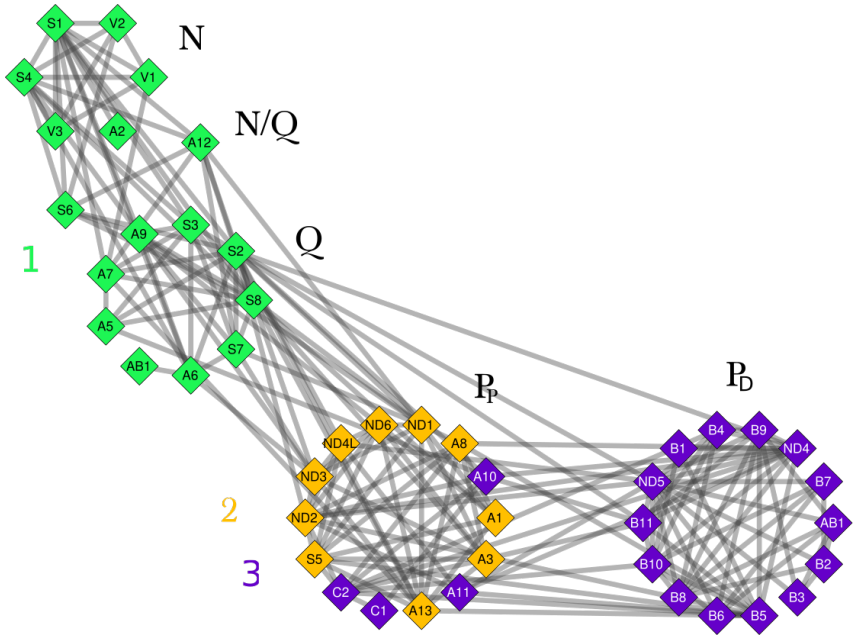
Once again, the difference between the modularity of the partitions according to the modules and the clusters is highest for the multiplicative and in this case lowest for the absolute weight (see Table 12). Ideally, the difference is zero meaning the partition according to the modules produces the highest modularity. The partition for the absolute weight is the best and the difference of the modularities is the lowest. This suggests that the absolute weight is the best choice for identifying modules, so far.

Table 12: Modularity and difference of modularity for different partitions and weight types for human respiratory complex I [20] (PDB 5xtd).

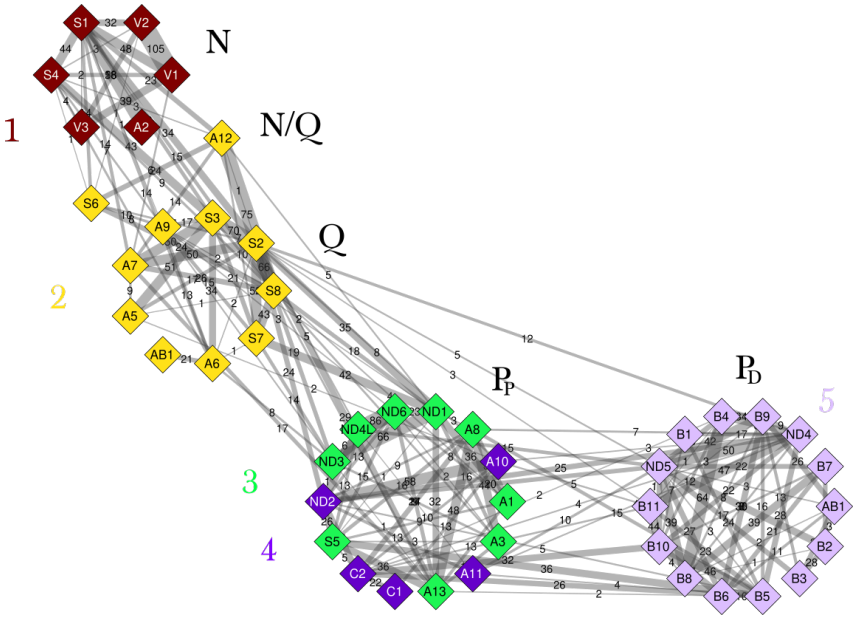
Edge weight type	Predicted clusters	Modules	Difference
No weight	0.46043	0.41349	0.04694
Absolute	0.54159	0.53872	0.00287
Additive	0.57890	0.54698	0.03192
Multiplicative	0.62161	0.57026	0.05135

We computed the modularity of the partitions according to the modules for the different possible assignments of A12 and S6 to the Q and N module (see Table A3). The modularity is highest for the assignment of A12 and S6 to the Q module for all types of edge weight. The biggest difference occurs for the multiplicative normalization with a modularity of 0.57026 for assigning A12 and S6 to the Q module and 0.53668 for assigning A12 to the N and S6 to the Q module. Structurally, it makes sense that both subunits are shared between the two modules (see Figure 27). Considering the modularities justifies assigning both subunits to the Q module for the comparison of partitions and underlines the clustering for the absolute weight where both were assigned to the Q module.

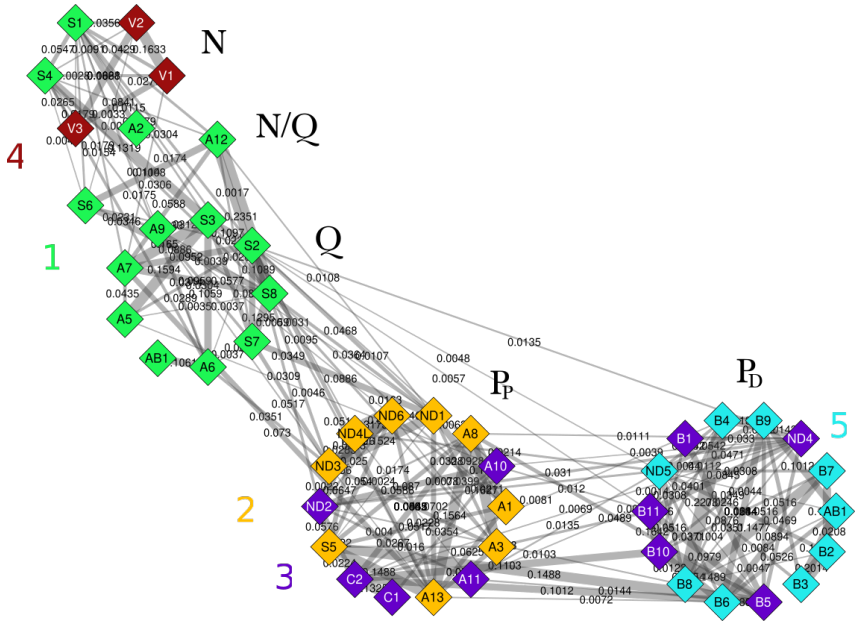
The subunits V1-3 were put in a separate cluster for the additive and multiplicative edge weights (see Figures 26c and 26d). The edges between these vertices have high edge weights and considerably lower edge weights to the vertices of the remaining N module. Splitting the N module in these two parts makes sense, because the submodules can be seen well in the structure (see Figure 27). Using clustering algorithms, it is always important to consider the resolution of clusters. The algorithm may split or merge desirable clusters, in our case the modules N, Q, P_P and P_D. The result may not be wrong only because it is not desired as can bee seen for splitting the N module, which makes sense structurally. The clustering using absolute edge weights identified the N module as a whole. We did not further investigate the resolution of clusters for different algorithms or types of edge weights.



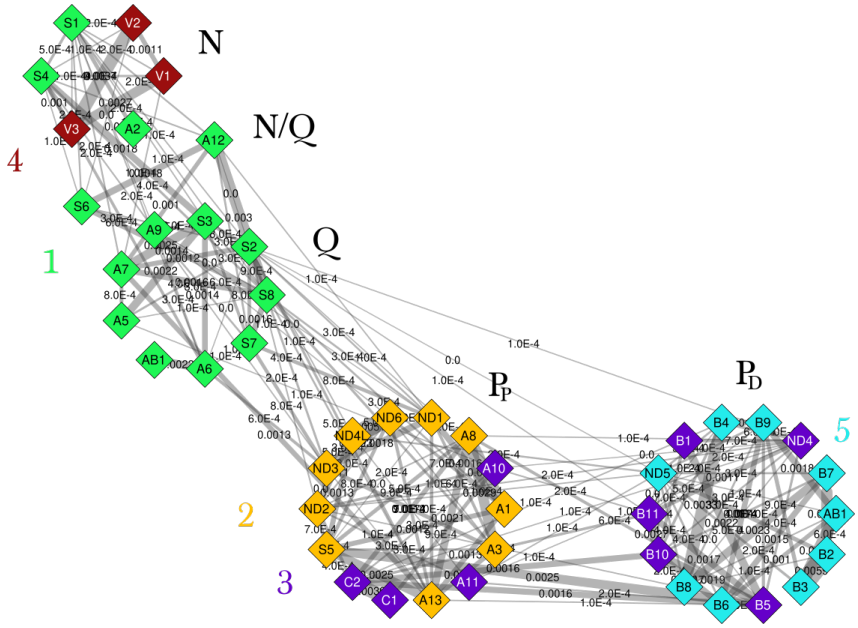
(a) Partition without edge weights.



(b) Partition for absolute edge weights.



(c) Partition for additive edge weights.



(d) Partition for multiplicative edge weights.

Figure 26: Visualization of the partition of the Complex Graph of respiratory complex I of *Homo sapiens* [20] (PDB 5xtd). Vertices correspond to subunits and edges to spatial neighborhoods. Vertices are labeled with the abbreviated gene names of the subunits. Vertices belonging to the same module are grouped together and labeled (N, Q, P_P, P_D). Two vertices are shared between the N and Q module (N/Q). Vertices belonging to the same cluster are colored the same and labeled with numbers. The edge label and width correspond to (b) absolute, (c) additive and (d) multiplicative edge weights.

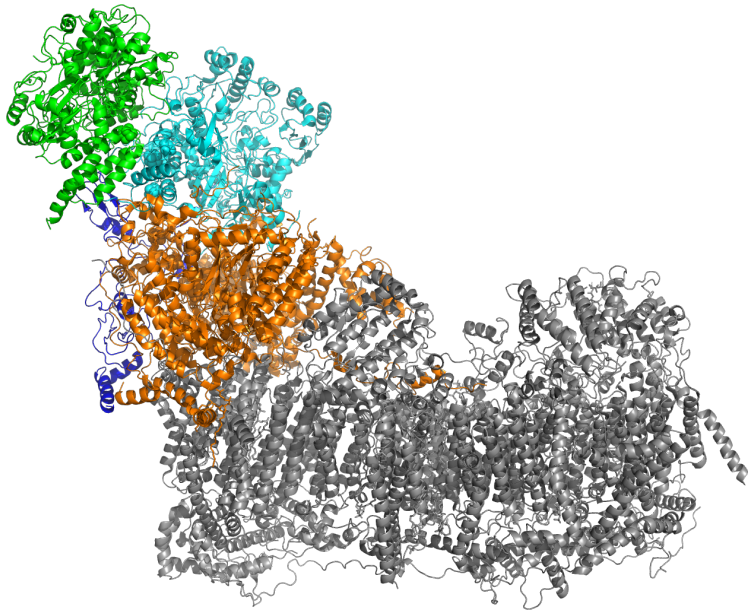


Figure 27: Visualization of the structure of human respiratory complex I [20] (PDB 5xtd). Protein chains are depicted in cartoon style and ligands as ball-and-stick representation. Subunits V1-3 (green) and A2, S1 and S4 (cyan) form the N module. Subunits A12 and S6 (blue) are shared between the N and Q module (orange).

3.4.3 Supercomplex $\mathbf{I_1III_2IV_1}$ of *Homo sapiens*

5xth contains coordinates of multiple complexes, I, III and IV. Moreover, two copies of complex III are present. Complex I can be further split into modules (see Section 2.10.4). In the context of clustering, it is unclear which clusters to expect for 5xth. One expectation is that each cluster corresponds to one complex. In this case, the cluster of complex I would be larger with its 45 subunits compared to 13 subunits of complex IV and 11 of complex III. Another expectation is that complex I is split into clusters corresponding to its modules. For splitting complex I into modules, we treated S6 and A12 as part of the Q module as a consequence of the analysis of the clustering of the CG of complex I (see Section 3.4.2).

We applied graph clustering to the CG of 5xth using no edge weights, absolute, additive and multiplicative edge weights. As for 4hea and 5xtd, we computed the partitions of the optimal modularity additional to the partitions produced by the Leiden algorithm using the function `partition_opt_modularity` of igraph (see Section 2.7). Surprisingly, the partitions of the Leiden algorithm for absolute, additive and multiplicative edge weights did not yield the best modularity. For the absolute weight, one vertex was assigned to a different cluster, for the additive weight three vertices and for the multiplicative six vertices.

The partition with optimal modularity achieved a modularity higher by 0.00058, 0.00019 and $1 * 10^{-16}$ than the partition of the Leiden algorithm for the absolute, additive and multiplicative edge weight, respectively. The differences are extremely small considering the range of the modularity of [-0.5,1]. With such small differences, both partitions and possibly more with comparable modularities might be considered as interesting candidates and could be further investigated. We focused on the partitions with the optimal modularity (see Figure 28).

In the partition using the edges without weights, only complex IV is identified as a cluster, cluster 4 (see Figure 28a). Cluster 5 contains both copies of complex III. Complex I is split across three clusters. The partition of complex I in 5xth equals the partition of 5xtd using the edges without weights. This is not surprising but also not necessarily to be expected. Theoretically, vertices of other complexes can be assigned to clusters of complex I. Aside from that, differences in the structure that can occur between different experiments may lead to different edges. In this case, the only difference between the set of edges is that an edge between A5 and A10 is present for 5xtd and not for 5xth. Despite the difference in the edge and the additional vertices of the other complexes in 5xth, the partition of complex I is the same.

The CG shows that the reason that complex IV is correctly identified is that it is connected only to complex I and only by three edges. The vertices of complex IV are internally well connected. From the two copies of complex III, only complex III-1 is connected to other complexes and only to complex I by four edges. The copies of complex III are connected with each other by twelve edges. Because of this, the copies of complex III can be separated from the other complexes, but seem hard to distinguish from each other using no edge weights.

The partitions for the absolute and additive edge weights are the same (see Figure 28b). Complex IV and both copies of complex III are correctly identified as separate clusters, respectively. This indicates that edge weights help distinguishing the two copies of complex III suggesting that the intra-complex edges have higher edge weights than the inter-complex edges.

Once again, complex I is split in the three clusters, but this time, only A11 is

falsely assigned to the cluster of the P_D module instead of to the P_P module. As for using the edges without weights, the modules of the matrix arm, N and Q, are aggregated in the combined cluster 1. While the partition of complex I using the edges without weights is identical between 5xth and 5xtd, the partitions for absolute and additive edge weights differ. For 5xtd, the clustering using absolute edge weights worked best, as it was able to identify the N, Q and P_D modules. Interestingly, we did not obtain such a good result for complex I of 5xth. Differences in the structure caused differences in the weights of the CG resulting in a different partition. The well-known resolution limit of the modularity might be another factor. We need a deeper analysis of the clustering and the partition to give a conclusive answer.

The partition for the multiplicative edge weights performed worst in identifying complexes or the modules of complex I producing a total of nine clusters when seven were expected. It is the only partition where complex IV is not identified as one cluster but split into two. The copies of complex III are distinguished making the partition better in this aspect than when using the edges without weights.

Complex I is split into five clusters with no cluster corresponding to a module. The vertices V1, V2 and V3 form a separate cluster as we observed for the clustering of 5xtd using additive and multiplicative edge weights. Cluster 8 contains vertices of the P_P and P_D module which is similar to the cluster 3 of the partition using additive and multiplicative edge weights for 5xtd.

We investigated the modularity of the predicted partitions compared to the desired partitions (see Table 13). We investigated both approaches where all clusters correspond to one complex and where complex I is further split into modules. The differences of the modularity between the predicted clusters and the approach focusing on complexes is much higher with values between 0.16647 and 0.18171 compared to the approach of splitting complex I with values between 0.00921 and 0.07046. This makes it much more likely to identify modules as well instead of only complexes when clustering the CG of such a large supercomplex. Ideally, the difference becomes zero which would allow us to identify the desired partition by graph clustering using the modularity. For this goal, the additive edge weight performs best, but only slightly better than the absolute edge weight.

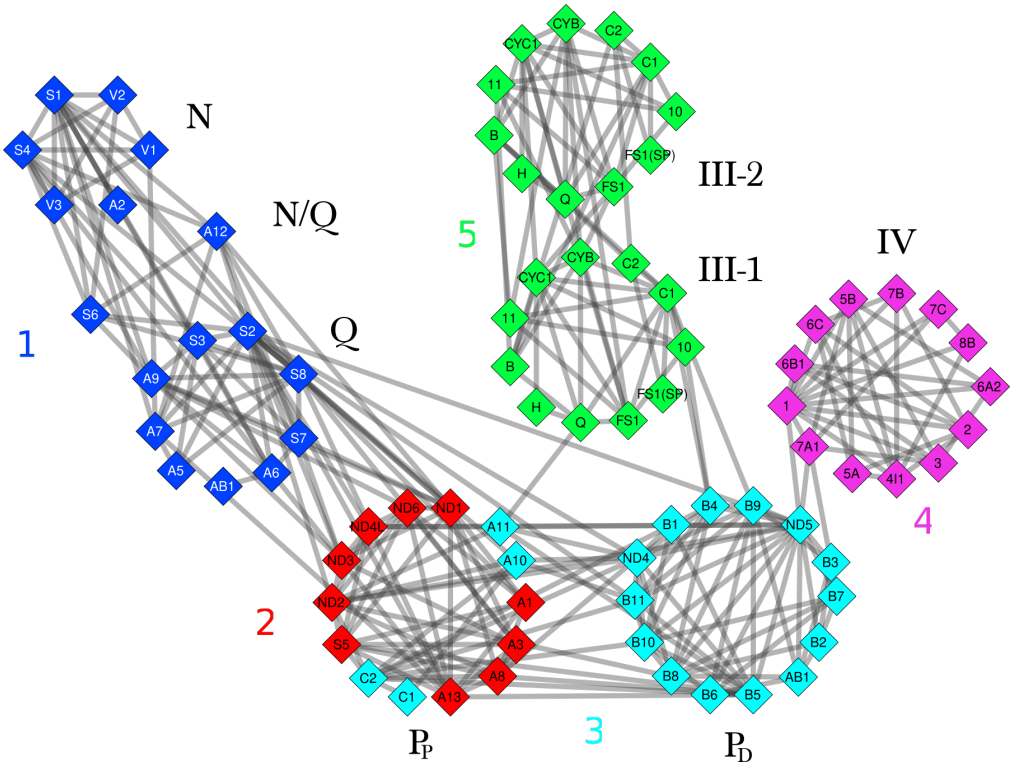
Table 13: Modularity and difference of modularity for different partitions and weight types for human respiratory supercomplex $I_1III_2IV_1$ [20] (PDB 5xth). Modularity of predicted clusters is compared to the partition where each cluster corresponds to one complex and where complex I is additionally split into modules.

	Modularity			Differences	
Edge weight type	Predicted clusters	Complexes	Modules	Complexes	Modules
No weight	0.64018	0.45848	0.56972	0.18171	0.07046
Absolute	0.71120	0.54349	0.70099	0.16771	0.01021
Additive	0.72378	0.54425	0.71457	0.17953	0.00921
Multiplicative	0.7548	0.58833	0.73889	0.16647	0.01590

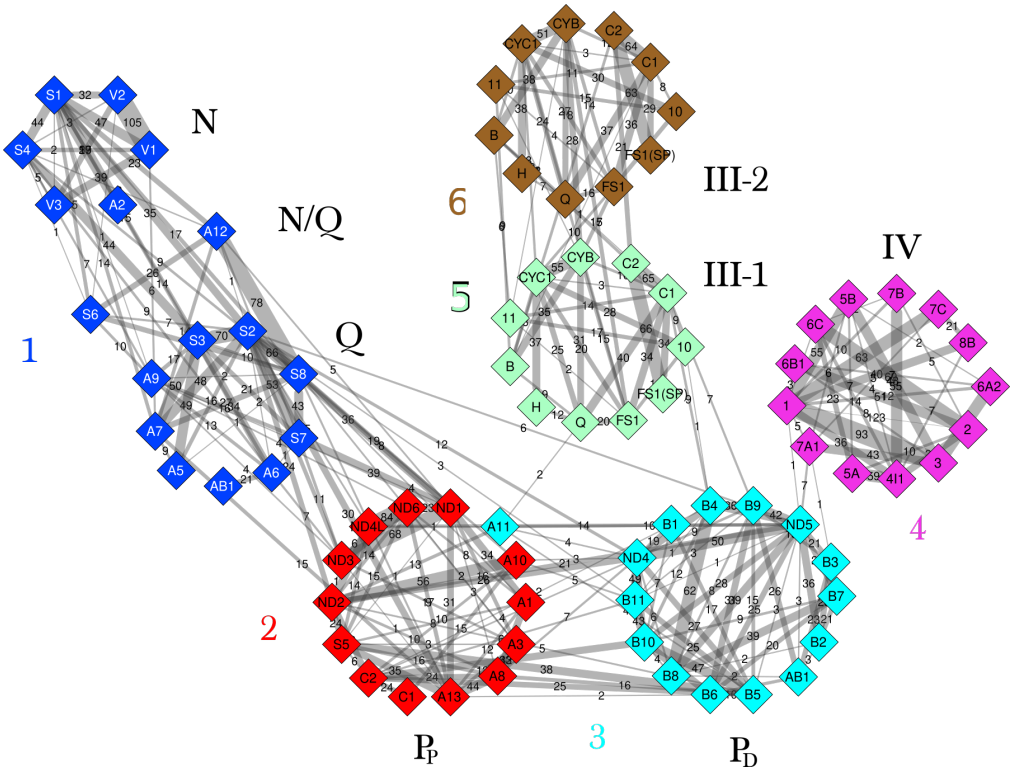
Conclusion

From the partitions and modularities, we draw the conclusion that we can expect clustering algorithms to identify the complexes III and IV as well as the modules of complex I as clusters. This should be taken into account when performing

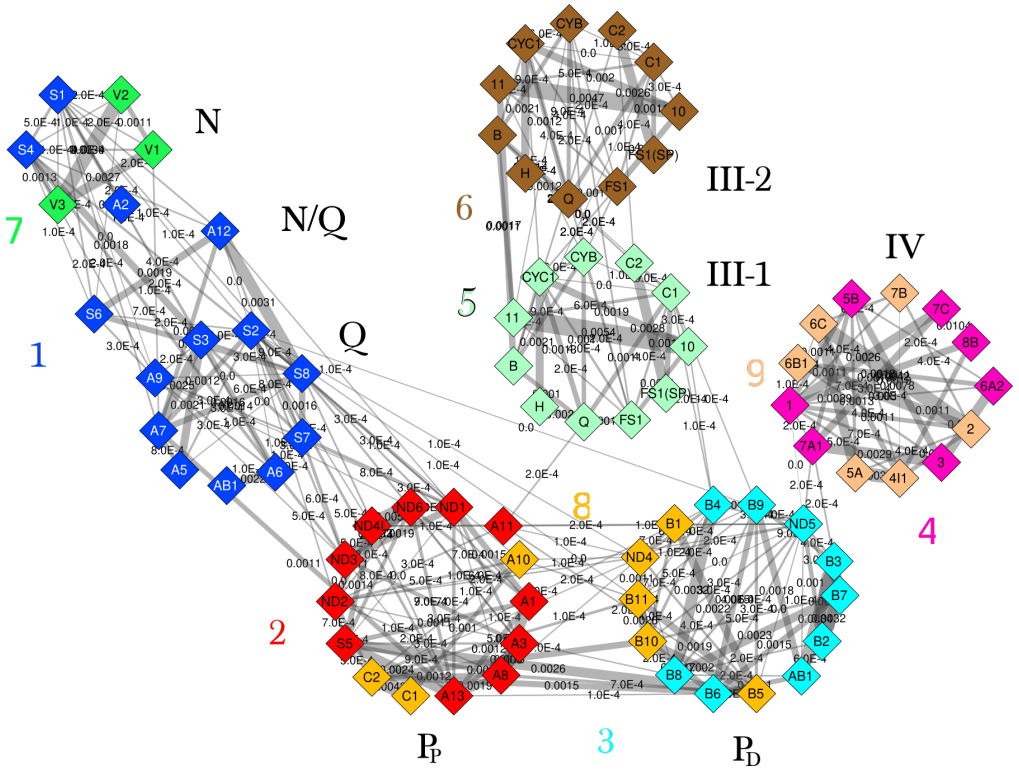
clustering on structures where single complexes or modules are unknown. The partition might contain a mix of clusters that correspond to complexes or to modules of a complex. After all, subunits are building blocks that assemble for different tasks. Different hierarchies such as module and complex are assigned by scientists based on experiments. Breaking it down, graph clustering of a CG identifies clusters that may or may not fit into this hierarchy. Our goal was to investigate if graph clustering is able to identify modules and complexes in principle and how different edge weight types perform. From our limited analysis, we can say that using edge weights improves the performance and that the absolute edge weight consistently produced the partition we deemed best. Another consequence from our analysis is that the Leiden algorithm may fail to find the partition with the optimal modularity, even though only by a small deviation. However, CGs are small enough to compute the partition with the optimal modularity in a few seconds.



(a) Partition without edge weights.



(b) Partition for absolute and additive edge weights.



(c) Partition for multiplicative edge weights.

Figure 28: Visualization of the partition of the Complex Graph of human respiratory supercomplex $I_1III_2IV_1$ [20] (PDB 5xth). Vertices correspond to subunits and edges to spatial neighborhoods. Vertices are labeled with the abbreviated gene names of the subunits. Vertices belonging to the same complex (III-1, III-2 and IV) or module in the case of complex I (N, Q, P_P, P_D) are grouped together and labeled. Two vertices are shared between the N and Q module (N/Q). Vertices belonging to the same cluster are colored the same and labeled with numbers. The edge labels and widths correspond to the (b) absolute and (c) multiplicative edge weights.

3.5 Prediction of the assembly pathway

3.5.1 Agglomerative clustering

Rationale of the method

Protein complexes assemble along an ordered pathway (see Section 2.4). Experimentally determining the assembly pathway demands time-consuming and ressource-consuming experiments. Computational methods can help narrowing down the search space by creating hypotheses of the assembly pathway.

During assembly, subunits form a subcomplex and share an interface. We propose that the order of the assembly can be inferred from the interfaces of the final assembly product. Subunits that share a large interface assemble earlier in the pathway. We use the Euclidean distances formalized in the edge weight of an edge between two subunits in a CG (see Section 2.8.3) as a measure of the interface. We apply a normalization of the edge weights to account for different lengths of subunits (see Section 3.3).

Method

We apply agglomerative clustering (see Section 2.1.4) to predict the assembly pathway of a protein complex. The vertices of a CG are the clusters containing single chains at the start of the clustering. The edge weight, either absolute or normalized, is the measure of similarity of two vertices. Higher edge weights lead to an earlier merging of clusters.

We did not use a typical linkage function, because when modelling edge weights as contacts between subunits they behave differently than could be expected when using typical linkage functions. Linkage functions define which distance is assigned to two non-singleton clusters, depending on all distances of the single members of the respective clusters. In our approach, we merge vertices adjacent to the edge with highest edge weights. After merging, multiple edges may exist between two vertices.

Consider the following example: Vertices A and B are merged, because they have the highest edge weight. There are two edges, {A,C} and {B,C}, with absolute edge weights 1 and 2, respectively. After merging, there are two edges between C and the newly merged cluster, {{A,B},C}, which are merged to one edge with weight 3. If normalized edge weights are applied, the normalization is recalculated after merging. For the lengths of subunits of merged clusters, the summed-up lengths of the individual subunits is used.

In short, merged clusters correspond to subcomplexes. The measure of similarity between subcomplexes is the summed-up number of contacts between the individual subunits of the respective subcomplexes so that the complete interface between the subcomplexes is considered. The output of the agglomerative clustering is a dendrogram representing the assembly pathway (see Section 2.4).

Implementation

We implemented the agglomerative clustering in PTGLgraphComputation (see Section 2.8) in the class `AgglomerativeClustering`. At each step, the merged two clusters are added to an object of the class `ClusteringResult`. The implementation is straightforward following the steps of the described methods (see Algorithm 3). We focused on an implementation that is fast and does not need much memory.

Algorithm 3 Agglomerative clustering of the vertices of a Complex Graph

```
E ← list of edges
while E is not empty do
    Sort E by decreasing edge weight
    merge_edge ← E[0] in case of greedy method
    Add vertices v1 and v2 of merge_edge to clustering result
    for e in E do
        if v1 or v2 adjacent to e then
            Update chain length as sum of chain lengths of v1 and v2
            Merge edge if duplicate
        end if
    end for
end while
```

The time-consuming steps are finding the highest edge weight at each step as well as possibly merging edges and recomputing the normalization after merging two clusters. Because of merging edges, finding the highest edge weight needs to be performed at each step instead of just sorting once and iterating through the list. With overriden functions `compareTo`, `hashCode` and `equals`, we could use Java's pre-implemented and well-performing function `Collections.sort`.

The data structure of the clustering result is the memory-intensive part. We saved memory by using a list as the most basic data structure for this purpose. We simply save the indices of the representatives of the two clusters merged at a step. The representative of a cluster is always the vertex with the lowest ID. For n vertices, we have $n - 1$ steps of agglomerative clustering. For each step, we save two integers. To output the clustering result as a Newick string (see Section 2.1.3), we need to backtrack from the root to the leaves and build the string. The backtracking costs runtime, but runs in linear time and the basic data structure helps reducing the required memory.

We tested the runtime for the largest protein structure deposited in the PDB: 3j3q. For protein 3j3q, we tested the runtime twice on one core of an Intel i5-9500 at 3 GHz. The runtimes for the assembly prediction for all weight types and the output as Newick String were 4,065 and 7,527 ms. Compared to the total runtime of PTGLgraphComputation for protein 3j3q of eight and a half hours (see Section 3.1), the additional runtime is neglectable. Because the runtime is already quite low for the largest structure of the PDB, we did not further investigate runtimes for different structures.

3.5.2 Comparison to state-of-the-art methods

Introduction

We tested our assembly prediction by agglomerative clustering of the vertices of the CG against state-of-the-art methods (see Section 2.4). We computed all CGs of the proteins of the data set (see Section 2.9). If the asymmetric unit was incomplete, we used the biological assembly (see Section 2.6.1). If the asymmetric unit contained multiple copies of the subunits, we ignored the additional copies. We predicted the assembly pathways by agglomerative clustering using additive edge weights in the

CG (see Table 14). We only used additive edge weights, because of the experience from previous results.

Table 14: Overview of the data set of Peterson et al. [61]. For each PDB structure, the number of subunits (#s) is given. For each structure, the assembly pathway proposed in the literature and predicted by agglomerative clustering is described in Newick format. The subunits are referred to by their PDB chain IDs or by their protein names (*). Homologous subunits are indicated by a prime (′) or numbers in superscript. An arrow (←) indicates that the predicted pathway exactly matches the proposed pathway. For non-binary pathways (**), all possible binary pathways are considered to be correct. The predicted assembly pathway is considered as partially correct if at least one subcomplex was correctly predicted.

PDB ID (superseded)	#s	Proposed assembly pathway	Predicted assembly pathway	Correct	No. of correct subcomplexes	Partially correct
1a0r	3	((B,G),P);	←	yes	1/1	yes
1ikn		((A,C),D);	(A,(C,D));	no	0/1	no
1vcb		((A,B),C);	←	yes	1/1	yes
2aze		((A,B),C);	←	yes	1/1	yes
1es7	4	(((A,A′),B),B′);	←	yes	2/2	yes
1gpq		(((A,A′),C),C′);	←	yes	2/2	yes
2e9x		(((B,D),A),C);	((A,C),(B,D));	no	1/2	yes
1kf6		(((C,D),B),A);	←	yes	2/2	yes
2bq1		((E,E′),(I,I′);	←	yes	2/2	yes
2qsp		((A,B),(A′,B′));	←	yes	2/2	yes
3fh6		((A,A′),F,G); **	((F,G),(A,A′));	yes	2/2	yes
1hez	5	((A,B),((A′,B′),E));	←	yes	3/3	yes
1w88		((A,A′,B,B′),I); **	(((A,(B,B′)),A′),I);	yes	3/3	yes
1du3	6	(((((D,D′),D″),A),A′),A″);	←	yes	4/4	yes
1rlb		(((A,A′),(A″,A‴),E),E′);	←	yes	4/4	yes
1s5b		(((((B^1,B^2),B^3),A),B^4),B^5); *	(A,((B^3,(B^1,B^2)),(B^4,B^5)));	no	2/4	yes
3vyt		((C,D),((C′,D′),(E,E′))); *	←	yes	4/4	yes
4hi0		((F,H),(F′,H′)),(G,G′); *	(((F,H),(G,G′)),(F′,H′));	no	3/4	yes
4igc (4yg2)		((((A,A′),C),(D,E)),X); *	(((A,A′),((C,D),X)),E);	no	1/4	yes
3uku	7	(((C,G),(D,F),A),B,E)	((((A,(D,F)),(B,G)),C),E);	no	3/5	yes
4gwp		((A,B,D),C,G),(E,F)); **	((A,D),((B,(C,(E,F))),G));	no	2/5	yes
Total				14/21	45/58	20/21

Some of the assembly pathways proposed in the literature are non-binary. For the comparison, we deemed all assembly pathways correct that agree with the non-binary pathway. For example, for pyruvate dehydrogenase E1 bound to binding domain of E2 [4] (1w88), the proposed pathway was "((A,A′,B,B′),I);". This meant that all possible binary dendrograms, such as "((A,A′),(B,B′)) or "(((A,B),A′),B′)", for the non-binary part "(A,A′,B,B′)" were considered as correct. An incorrect pathway would be, for example, if A and I assemble first, because based on the proposed pathway two copies of A and two copies of B need to assemble before I joins. If multiple copies of a subunit exist, the copies are not differentiated. For example, A and A′ are interchangeable. If necessary we transformed the output and relabeled chain IDs according to the copies.

Peterson et al. [61] have applied three different measurements to evaluate a prediction and the proposed pathway. A straightforward approach was to look for a correct prediction and assign right or wrong. Counting the number of correct subcomplexes allowed more differentiation. For a complex of n subunits, there are $n-2$ subcomplexes in total. For non-binary pathways, it was not intuitive to understand how correct subcomplexes are counted. For example, for the mediator head module bound to the carboxy-terminal domain of RNA polymerase II [11]

(4gwp), Peterson et al. have described the proposed pathway with the string "ABD > ABCDG + EF > ABCDEFG". One could assume that the subcomplexes are "ABD", "ABCDG" and "EF", resulting in a total of three subcomplexes but for seven subunits, there are five subcomplexes according to the formula above. The difference can be seen in dendrograms by resolving non-binary parts (see Figure 29). The step "ABD", for example, accounted for two subcomplexes instead of a single one. In a dendrogram, a subcomplex refers to an inner vertex. The last measurement assigned a pathway whether it is partially correct. A pathway is deemed partially correct if at least one subcomplex is correct.

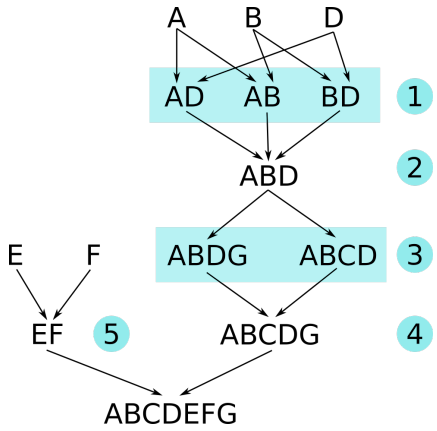


Figure 29: Possible assembly pathways that agree with the proposed assembly pathway of the mediator head module bound to carboxy-terminal domain of RNA polymerase II [11] (PDB 4gwp). The subunits are labeled according to their chain ID. Arrows indicate which subunits assemble from top to bottom. All subcomplexes are enumerated. Cyan boxes group together subcomplexes, where multiple subcomplexes are possible which together count as one subcomplex for the number of subcomplexes.

Results

Correct prediction Our predictions are correct in 14 out of 21 cases. We achieved no correct prediction for any of the two complexes of seven subunits. For the complexes of three to six subunits, we achieved both correct and incorrect predictions with the exception of five subunits where the assembly pathways for both complexes were correct. The success rates, regarding the number of subunits from two to seven, are 75 %, 85.74 %, 100 %, 50 %, 0%. The sample is too small to draw a final conclusion, but it seems that the success rate drops with an increasing number of subunits.

This is not surprising, because the number of possible dendrograms increases with the number of subunits. For example, for three distinct subunits, there are three possible dendrograms while there are 10,395 possible dendrograms for seven distinct subunits (see Section 2.1.3). It is much more likely to predict the correct pathway by chance for smaller complexes.

According to Peterson et al. [61], subcomplex BSA has performed better than pairwise BSA. Subcomplex BSA has achieved 13 correct predictions. From the different tested strategies for Path-LZerD, the low root mean square deviation (RMSD) decoy with the scoring function Shape, and the lowest RMSD with the scoring function OPUS-PSP have performed best, achieving eleven correct predictions.

Number of correct subcomplexes Our predictions were correct for 45 of 58 subcomplexes, overall. The number of subcomplexes increases with the number of subunits per complex. While for three subunits, the pathway is either correct or not resulting in one or zero correct subcomplexes, for seven subunits, there are five subcomplexes where each might be the correct one. This allows for a nuanced analysis of the result for complexes of more than three subunits besides just assigning correct or incorrect. For complexes of three, four and five subunits, most predictions were correct, thus, resulting in the maximum number of correctly predicted subcomplexes. For three subunits, IκBα/NF-κB complex [1] (1ikn) was not correct resulting in zero of one correct subcomplexes. For four subunits, human GINS core complex [5] (2e9x) was not correct, but one of two subcomplexes was correctly predicted. Therefore, 2e9x is the first case for our method, where the measurement of subcomplexes provides further insights than just checking if a pathway is correct or not.

For six subunits, there are three correct predictions resulting in four of four correct subcomplexes. Three of the predictions are not correct, but have two, three and one correct subcomplexes, respectively. The two complexes of seven subunits have three and two of five correctly predicted subcomplexes, respectively.

Peterson et al. have reported 42 subcomplexes as correct for subcomplex BSA. For Path-LZerD, the consensus strategy with the scoring function GOAP has performed best achieving 37 correct subcomplexes.

Partially correct prediction Considering the last measurement, our predictions were partially correct in 20 of 21 cases. The only complex, in which our prediction was not partially correct, is 1ikn, because the only one subcomplex was wrongly predicted. It is not surprising that our method scores almost 100 % considering partial correctness which is a rather weak measurement. One correctly predicted subcomplex per complex would be sufficient to score 100 %, but would still mean a poor performance for larger complexes with more subcomplexes.

Peterson et al. have reported 18 partially correct predictions for subcomplex BSA. Again, the best strategy and scoring function for Path-LZerD has been consensus and GOAP, respectively. Path-LZerD has been partially correct in 17 cases.

Comparison of methods Our prediction of the assembly pathway, using agglomerative clustering of the CG with additive weights, outperformed the other methods in all three measurements: we achieved the most correct predictions, the highest number of correct subcomplexes and the most partially correct predictions. Please note that, we always compared our method to the best performing strategy and scoring function per measurement, if applicable. This means that the gap between the methods may be higher when comparing with a certain strategy and scoring function. For a user of a software, it is usually unknown beforehand which strategy or scoring function would work best.

Peterson et al. have differentiated between strategies that need the structure of the final complex (non-blind) and those that do not (blind) (see Section 2.4). The advantage of blind strategies is that they can be applied when the structure of the final complex is unknown. The consensus strategy is blind while low RMSD decoy, lowest RMSD and subcomplex BSA are non-blind. Our approach can be considered non-blind, because the structure of the complex is required to compute the CG.

Peterson et al. have reported runtimes for eight selected structures of three, four, five and six subunits. The runtimes range between 364.6 and 1,816.4 central

processing unit (CPU) hours on an Intel Xeon-E5 CPU with 126 or 96 GB. For the same structures, our runtimes range from 3 to 15 seconds using one core of an Intel i5-9500 at 3 GHz and by default up to 1 GB of memory. Note that, Path-LZerD can be run in parallel on multiple cores reducing the runtime. For our runtimes, we did not measure the CPU time but the actual runtime. The actual runtime for serialized jobs is greater than or equal to the CPU time, because of time for input, output and waiting time. The difference can be neglected, because of low runtimes in the range of seconds.

With runtimes of multiple days even when parallelized, it becomes unfeasible to run Path-LZerD for large protein complexes of more than seven subunits. We optimized PTGL to run fast even for large structures (see Section 3.1). Predicting the assembly pathway of a much larger structure than eight subunits will be explored in the next subsection (see Section 3.5.3).

Exploring examples

IκBα/NF-κB complex (1ikn) 1ikn is the only structure where our prediction was not partially correct. The proposed pathway is "((A,C),D)", whereas our method predicted that subunits C and D assemble first. We noticed that the additive edge weights are extremely close to each other (see Table 15). We also noticed that some of the residues are missing in the structure (see Figure 30) [130].

Table 15: Absolute, additive and additive corrected for missing residues edge weights of the Complex Graph for IκBα/NF-κB complex [1] (PDB 1ikn).

Vertex 1	Vertex 2	Absolute	Additive	Corrected additive
A	C	27	0.06818	0.06666
A	D	34	0.06814	0.06513
C	D	23	0.06991	0.06479

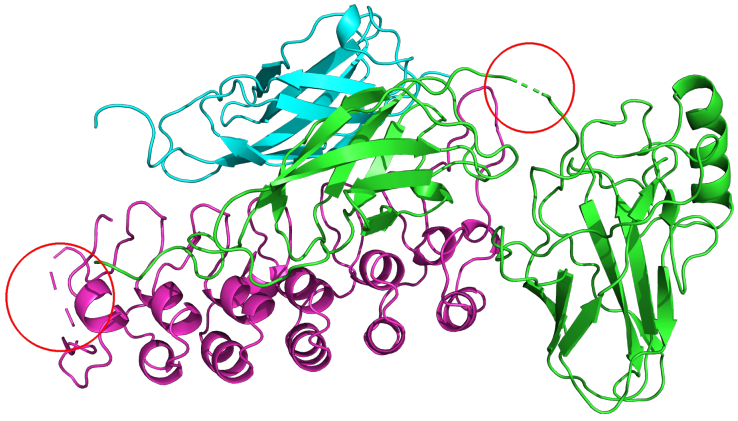


Figure 30: Visualization of the structure of IκBα/NF-κB complex [1] (PDB 1ikn). Protein chains are depicted in cartoon style. Chains A, B and D are colored green, cyan and magenta, respectively. Unresolved residues inside a chain are depicted as dashed lines and marked with a red circle.

When we corrected the additive edge weights with the number of unmodeled residues, the weight between subunits A and C was the highest. This means that

the assembly would be correct if the residues were modeled. However, while some unmodeled residues are clearly in parts, where they would not cause new contacts, some residues at the N- and C-termini of the chains may exhibit new contacts. It is unclear how this would influence the edge weight and, finally, the assembly. This shows again how important complete structures are.

Human GINS core complex (2e9x) 2e9x is the only complex of four subunits, where our prediction was not correct. The proposed pathway is "(((B,D),A),C)", and our prediction is "((A,C),(B,D))". The subcomplex "(B,D)" is correct in that our prediction is partially correct. Because the edge between A and C has the highest edge weight (see Table 16), these two vertices are clustered in the first step making it impossible that A assembles with a subcomplex of B and D. If the vertices B and D are clustered manually in the first step, A has the chance to be added next. Since B and D are merged, the absolute number of contacts between the merged vertex {B,D} and A increases. At the same time, the length is combined and used for the additive normalization, effectively decreasing the edge weight. In the end, the subunits A and C are still predicted to assemble next.

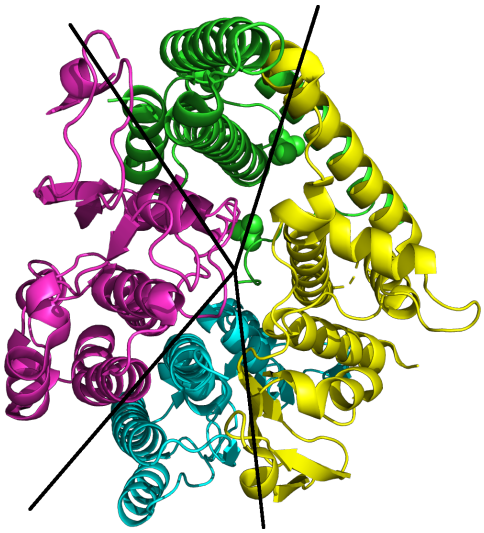


Figure 31: Visualization of the structure of human GINS core complex [5] (PDB 2e9x). Protein chains are depicted in cartoon style and ligands either in ball-and-stick representation. Chains A, B, C and D are colored green, cyan, magenta and yellow, respectively. Black lines approximately cut the complex in its subunits.

When taking a look at the structure (see Figure 31), it became clear what could already be seen in the edge weights of the CG: The subunits arrange in a circular way. This means that each subunit has a strong interface with the left and right neighbor but only a weak interface with the diagonal neighbor. This leads to just a moderate increase of the absolute number of contacts with subunit A, when subunits B and D are merged, because the majority of contacts still comes from the interface of A and D. For complexes with a circular arrangement, it is possible that our method favors balanced dendrograms in contrast to the proposed pathway which is unbalanced. It seems that the rationale of our assembly prediction does not hold for this complex, because the assembly order cannot be inferred from the number of contacts in the final complex.

Table 16: Absolute and additive edge weights of the Complex Graph for human GINS core complex [5] (PDB 2e9x).

Vertex 1	Vertex 2	Absolute	Additive
A	B	3	0.0094
A	C	67	0.203
A	D	56	0.1642
B	C	57	0.1579
B	D	70	0.1882
C	D	6	0.0157
A	{B,D}	59	0.1143

Cholera holotoxin with an A subunit (1s5b) Cholera holotoxin with an A subunit [3] (1s5b) is one of the complexes of six subunits where our prediction was not correct. 1s5b consists of five homologous subunits (B subunits) forming a ring and one additional subunit (A subunit) piercing the ring (see Figures 32a and 32b). 1s5b is an interesting example, because it has been reported that three copies of the B subunit assemble first with subunit A joining afterwards and only then the remaining two copies of the B subunit assemble [61, 131]. In Newick format, the representation of the pathway is: "(((((B^1,B^2),B^3),A),B^4),B^5)". In our prediction, all copies of the subunit B assemble before subunit A joins: "(A,((B^3,(B^1,B^2)),(B^4,B^5)))".

The interfaces of the homopentamer ring are identical (see Figure 32c). With identical interfaces, it seems intuitive that the rest of the complex is predicted to assemble either before or after, but not in between the assembly of the B subunits. However, this is not necessarily the case, because the interface between subunits of B and subunit A increases once subunits of B assemble. It is possible that three copies of subunit B assemble first and consequently the interface between them and subunit A is larger than between them and another copy of B. Three copies of subunit B, the chains E, F and G, have a stronger interface with A than the other two copies of subunit B, the chains D and H (see Table 17).

Table 17: Absolute and additive weights of selected edges of the Complex Graph for cholera holotoxin with an A subunit [3] (PDB 1s5b). Vertices are labeled by the name of the subunit and their chain ID in the case of copies of the B subunit.

Vertex 1	Vertex 2	Absolute	Additive
A	B (D)	2	0.0064
A	B (E)	12	0.0386
A	B (F)	15	0.0482
A	B (G)	17	0.0547
A	B (H)	7	0.0225
B (D)	B (E)	53	0.2573
B (D)	B (H)	53	0.2573
B (G)	B (H)	56	0.2718
A	B_3 (E,F,G)	44	0.0851

In the greedy agglomerative clustering using additive weights, chains E and F are assembled first, joined by D afterwards. Using absolute weights, the order matches

the proposed assembly pathway: chains E and F assemble first joined by G after that. Subunit A is still not added to {E,F,G} neither for absolute nor for additive weights, because the edge weight is too low (see Table 17). For the B_3 subcomplex, the effect of the additive normalization becomes visible. The combined length influences the weight and decreases it much more compared to the other copies of the B subunit.

During visual inspection, we noticed that 17 contacts between subunit A and chain with ID G seemed lower than expected (see Figure 32d). Note that, "contacts" refers to the number of residue contacts. This means that there might be many atom contacts, but if they are all established between two residues, they count as only one residue contact. The helix of subunit A goes right through the ring of B subunits and should have a considerably higher number of contacts with the surrounding subunits. This raised the question of whether the contact threshold of 4 Å (see Section 2.8.2) is suitable to correctly capture the interfaces.

To predict the proposed pathway with our method, the edge weight of {E,F,G}-{A} needs to be higher than any other edge weight after assembly of the chains E, F and G. We tracked the edge weight of {E,F,G}-{A} and competitive edges while increasing the distance threshold in steps of 0.2 Å (see Figure 33). At 4 Å, the default setting, the edge {E,F,G}-{A} has a lower weight than the competitive edges. With the distance threshold increasing, the edge weights increase because more contacts are defined.

The relation between the edge weights stays the same for the additive edge weight (see Figure 33b). This means that the assembly order is independent of the distance threshold. Interestingly, the relation between the edge weights changes for the absolute edge weights (see Figure 33a). At 6 Å, the absolute edge weight of {E,F,G}-{A} is even or greater than all the competitive edge weights for the first time. From 6.6 to 6.8 Å, the absolute weight jumps from 169 to 195 while the competitive edge weights increase by less than 20. At 7 Å, the absolute weight of {E,F,G}-{A} is higher by 11 than any of the competitive edge weights.

With a threshold of 7 Å and using absolute edge weights, finding the proposed assembly pathway with our method is possible. Interestingly, the number of contacts rises differently for different interfaces, when increasing the distance threshold. However, a threshold of 7 Å is so large that contacts exist with residues that are spatially behind another residue. It remains unclear if this makes sense in general. Optimizing parameters to get the desired outcome is generally not a good idea, because the user cannot know what parameters to apply beforehand. Nevertheless, we could show that our method, in principle, could find the correct assembly pathway even for an example this hard.

Mediator head module bound to the carboxy-terminal domain of RNA polymerase II (PDB 4gwp) 4gwp is one of the two complexes of seven subunits, for which our method could not predict the correct assembly pathway. In the proposed assembly pathway, subunits A, B and D assemble in no specific order followed by the joining of subunits C and G in no specific order. Subunits E and F assemble separately and join the other subunits in the last step. In our prediction, two subcomplexes are correct: the assembly of A and D as well as the assembly of E and F. In the next steps, first subunit C and then B join the subcomplex {E,F}.

In the complete structure (see Figure 34e), the helix bundle formed by the subunits A, B and D can be seen. It makes sense that the subunits assemble, because they are congregated. One question is why subunit B was not predicted to assemble

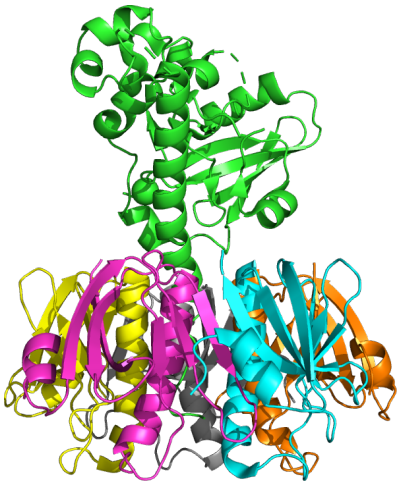
with subcomplex {A,D} and another why subunit C was predicted to assemble with subcomplex {E,F}.

We visualized the course of the predicted assembly (see Figure 34 a-e). The subunits E and F assemble first which makes sense looking at their extensive interface (see Figure 34a). However, the next step is already problematic: C assembles with the subcomplex of E and F. This cannot be explained with the interface between these subunits (see Figure 34b). The only region in contact is a helix of subunit C and a loop of subunit E. Despite an extensive local interface (see Figure 34f), the subunits do not interact globally. Note that, we discuss the assembly based on the final structure and cannot include the formation of substructures during the assembly process that differ from the final structure.

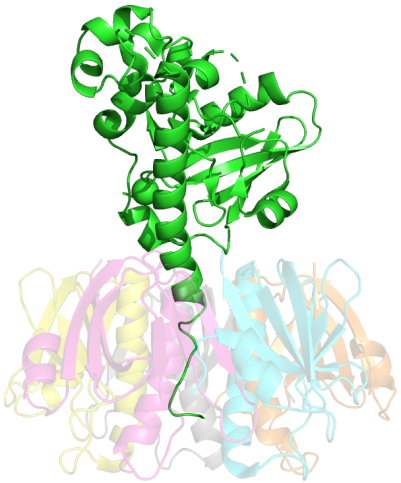
The small interface between subunit C and E is enough to make the edge weight the highest one (see Table 18). To get the prediction right, subunit C could either assemble with the subcomplex {A,B,D}, which is not yet assembled at this point in time of the assembly prediction, or with subunit G. The edge weight between C and {E,F} is higher than between C and G for the absolute and additive weight. This is interesting, because usually the additive weight of subcomplexes decreases due to the increasing lengths of the participating subunits. From the experience with 1s5b, we tested increasing the contact threshold to 7 Å. The edge weights flipped such that, now, the weight between the subunits C and G was higher, at least for the additive normalization. At a contact threshold of 7 Å and when using the additive edge weight, the assembly prediction became correct fixing not only the case of subunits C and G but also of A, B and D.

Table 18: Absolute and additive weights of selected edges of the Complex Graph for the mediator head module bound to carboxy-terminal domain of RNA polymerase II [11] (PDB 4gwp). Edge weights are shown for a contact threshold of 4 and 7 Ångström (Å). Vertices are labeled by the chain ID of the subunit. One edge indicated by a dashes instead of weights that is not existing is included to highlight that the edge is not present.

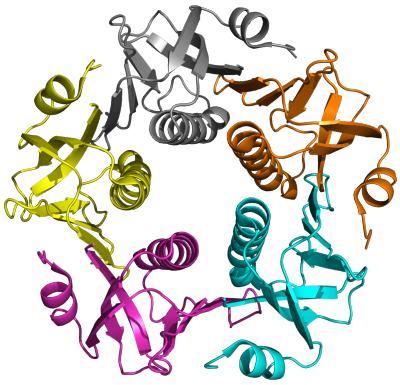
		4 Å		7 Å	
Vertex 1	Vertex 2	Absolute	Additive	Absolute	Additive
C	E	34	0.0778	153	0.3501
C	F	-	-	-	-
C	{E,F}	34	0.0526	153	0.2368
C	G	14	0.0395	126	0.3559



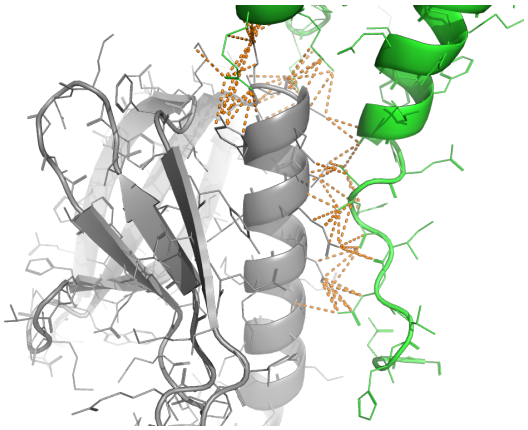
(a) Whole complex.



(b) Whole complex with transparent ring.

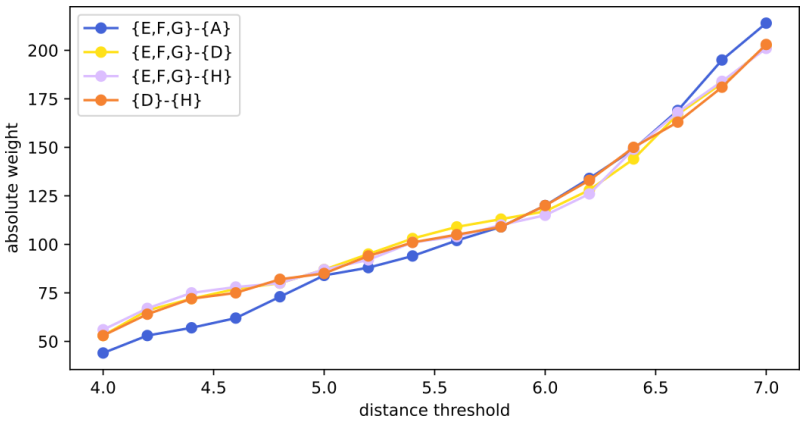


(c) Ring of B subunits.

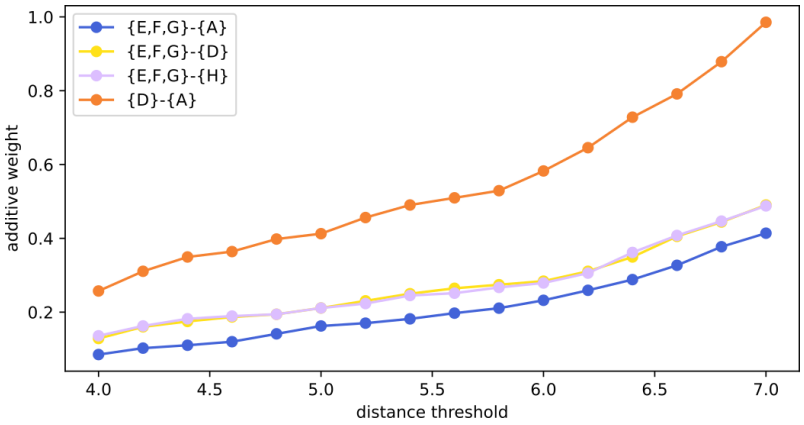


(d) Interface between chains A and G. Atoms shown in line style. Orange dashed lines depict distances between the subunits less than or equal to 4 Å.

Figure 32: Visualization of the structure of cholera holotoxin with an A subunit [3] (PDB 1s5b). Protein chains are depicted in cartoon style. Chains A, D, E, F, G and H are colored green, cyan, magenta, yellow, grey and orange, respectively.

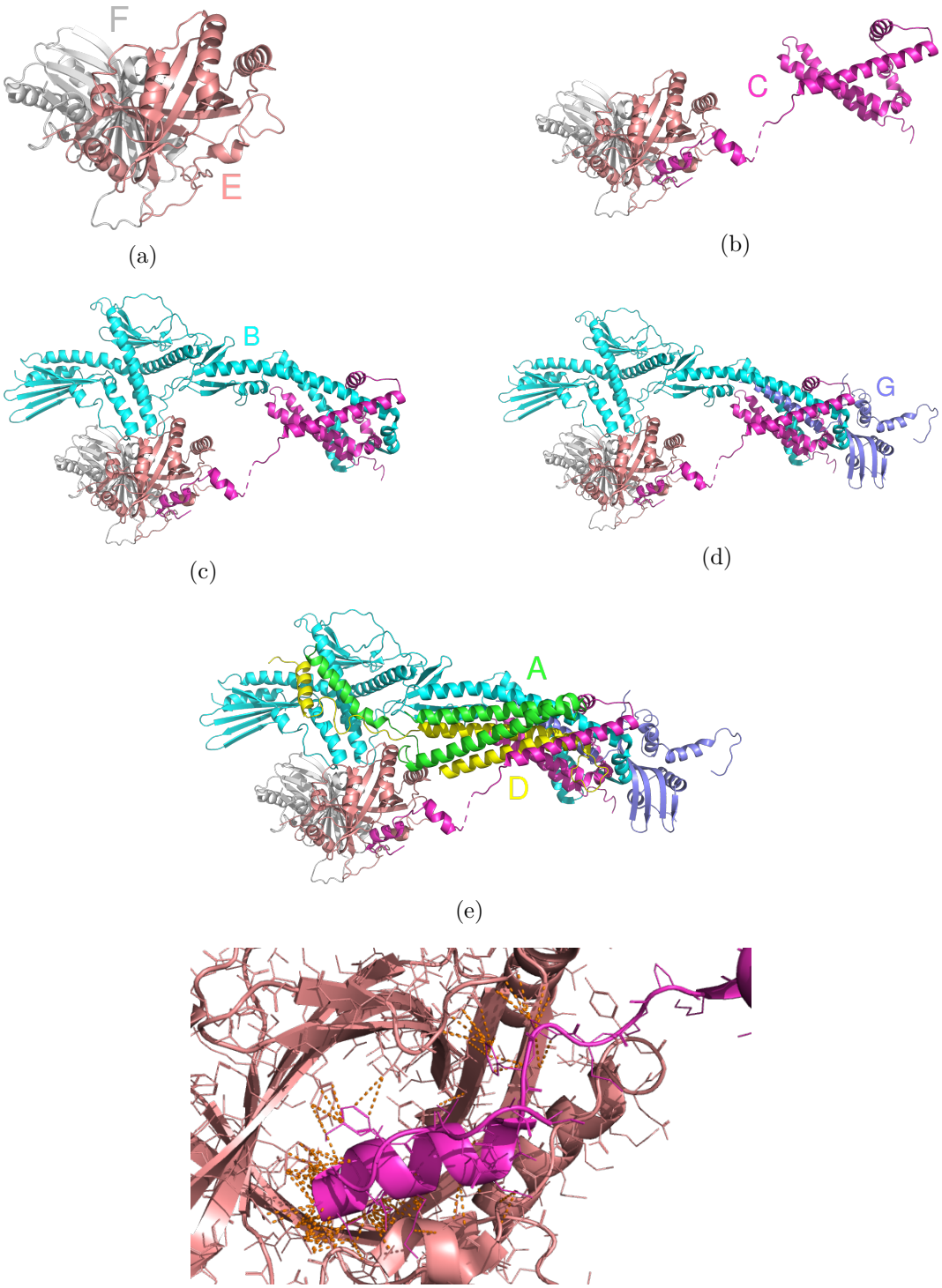


(a) Absolute edge weights.



(b) Additive edge weights.

Figure 33: Plot of the edge weights of selected edges of the Complex Graph of cholera holotoxin with an A subunit [3] (PDB 1s5b) against an increasing distance threshold. The vertices are labeled according to their chain IDs. For merged vertices, all single chain IDs are enumerated. Distance threshold is given in Ångström.



(f) Interface between subunits C and E. Atoms shown in line style. Orange dashed lines depict distances between the subunits less than or equal to 4 Å.

Figure 34: Visualization of the structure of the mediator head module bound to carboxy-terminal domain of RNA polymerase II [11] (PDB 4gwp). Protein chains are depicted in cartoon style. Chains A, B, C, D, E, F and G are colored green, cyan, magenta, yellow, grey and salmon, respectively. The subfigures a-e show the course of a predicted assembly. In each subfigure a-e, the subunits that recently joined the assembly are labeled.

Conclusion

Predicting the assembly pathway by agglomerative clustering of the CG using additive edge weights gave promising results. For the data set of 21 complexes, 14 pathways were predicted correctly. The method achieved higher accuracies than the methods BSA and Path-LZerD for different strategies and scoring functions we compared with for all three measurements.

However, we doubt that the three measurements Peterson et al. [61] applied are all equally suitable to rate methods for the assembly prediction against each other. A correct prediction becomes harder the more subunits a complex contains. A partial prediction on the other hand becomes easier the more subunits a complex contains. In our opinion, the number of correctly predicted subcomplexes works best. The number of correctly predicted subcomplexes still has drawbacks. For example, for 1w88, a method scores all subcomplexes if it simply adds subunit I in the end (see Table 14). The order of the other subunits does not matter, because there is no order proposed for them. So if the assembly is predicted to initiate with subunit I, the method scores zero correct subcomplexes. In such a case, the measurement acts in a binary way just like assigning correct or partially correct to the pathway.

Computing the CG of a complex and performing the agglomerative clustering is extremely fast compared to the docking approach of Peterson et al. and takes only a few seconds per complex. Path-LZerD can be run blindly without the structure of the final complex. The fast runtime of the agglomerative clustering makes up for the fact that a CG can only be computed for a resolved complex, because the complex could be computationally predicted beforehand. We showed that completely modeled structures are important and may influence the outcome. Modeling missing residues may improve the prediction. We showed that increasing the contact threshold helped in certain cases correcting the prediction, but want to point out that more insights are needed.

3.5.3 Respiratory complex I of *Homo sapiens*

We used 5xtd (see Section 2.10.4) as a case study. We computed the CG (see Figure A4) and predicted the assembly pathway by agglomerative clustering of the vertices of the CG (see Figure 35). We computed an assembly pathway for each edge weight type (see Section 3.3). We compared our predictions to two reference pathways that have been proposed by Guerrero-Castillo et al. [84] (see Section 2.10.4).

In the following, we refer to a dendrogram based on a given edge weight type of the CG as, for example, the absolute dendrogram. The subcomplexes of the reference assembly pathways are labeled according to the structural modules even if the module is not completely assembled at a given step. We differentiate between the completely assembled modules of the structure (N, Q, P_P and P_D) and the subcomplexes of the reference assembly pathway by prefixing "pre-" for subcomplexes of the assembly pathway where ambiguity exists. For example, we refer to the N subcomplex in the proposed assembly pathway as pre-N, but do not change the label of the P_D-a subcomplex, because the postfix "-a" already indicates that the P_D-a subcomplex does not refer to the complete module. We labeled inner vertices of the predicted dendrograms with the names of the modules they correspond to, but this does not mean that all subunits of the module are present in the labelled subcomplex. We did not consider that the subunits with chain IDs G and X are copies of subunit AB1,

because of the additional effort and neglectable influence on the number of existing dendrograms and the CID.

Quantitative analysis

We compared the predicted assembly pathways using the different edge weight types with the reference pathways using the nRF distance and CID (see Section 2.1.3). The nRF distances between a prediction and reference 1 or 2 are the same for all edge weight types, which allows no differentiation between the reference pathways. The nRF distance is lowest for the additive dendrogram with a distance of 0.8276. The next highest distance occurs for the square root dendrogram with a distance of 0.9310. The dendrograms of the remaining edge weight types have equal distances of 0.9655. The gap between the additive and the other dendrograms is considerably higher, making this prediction the best considering the nRF distance.

Table 19: The normalized Robinson-Foulds (nRF) distance and clustering information distance (CID) between predicted assembly pathways for different edge weight types and reference pathways. A left arrow (←) indicates that the value is the same for reference 1 and 2.

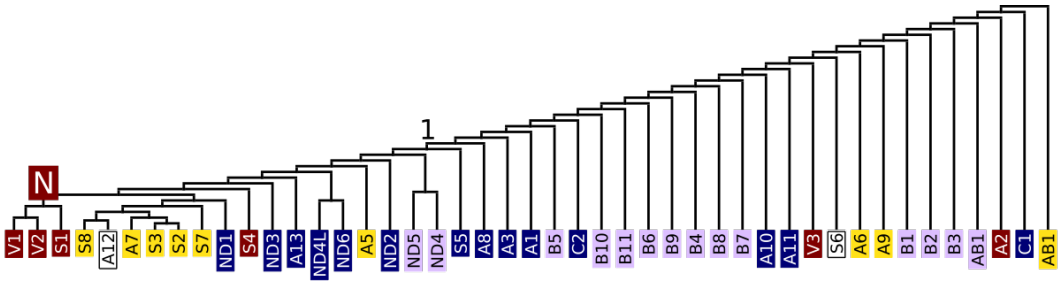
	nRF		CID	
	Ref. 1	Ref. 2	Ref. 1	Ref. 2
Absolute	0.9655	←	0.8704	0.8732
Additive	0.8276	←	0.6666	0.6777
Multiplicative	0.9655	←	0.7173	0.7164
Square Root	0.931	←	0.7639	0.7720
Logarithm	0.9655	←	0.8833	0.8903
Minimum	0.9655	←	0.7774	0.784

The CID allows differentiating between the reference pathways, because the distances are different when comparing to reference 1 or 2. All differences comparing to reference 1 are lower than comparing to reference 2 for a given edge weight type. This may indicate that our assembly prediction tends to favor reference 1. However, the differences are subtle, changing a digit at the second or third decimal place.

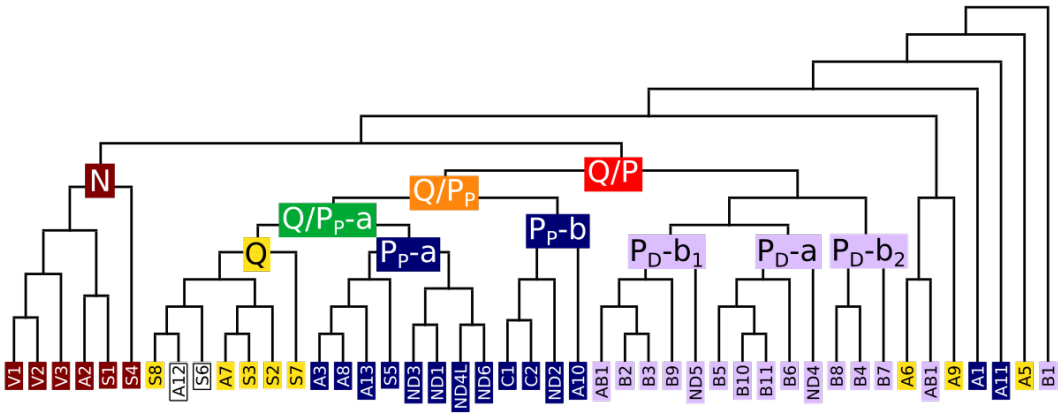
The lowest difference of 0.6666 occurs between the additive dendrogram and reference 1. The highest difference occurs between the logarithm dendrogram with a distance of 0.8833 compared to reference 1 and 0.8903 compared to reference 2. Again, the additive dendrogram has a large gap to the other distances and can be considered the best prediction.

It is remarkable that we only encountered four different nRF distances for six different edge weight types, and that no differentiation is possible between reference 1 and 2. The nRF distance can only adapt $n - 2$ distinct values for n leaves. The nRF distances assign one of the highest possible distances for five of six edge weight types. The CID is much more sensitive allowing differentiation between all edge weight types and the reference pathways.

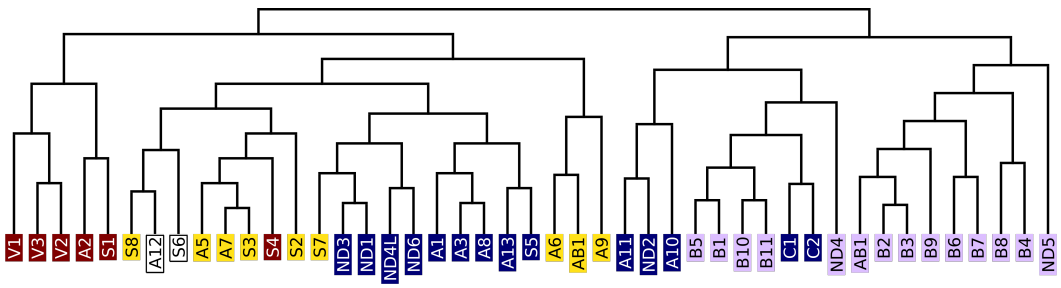
Using the distance metrics, we were able to identify the best performing edge weight type, but we could not evaluate the quality of the predicted pathway. Considered naively, the lowest CID of 0.6666 is closer to the maximum of 1 than to the minimum of 0. This depends, however, on the landscape of the CID.



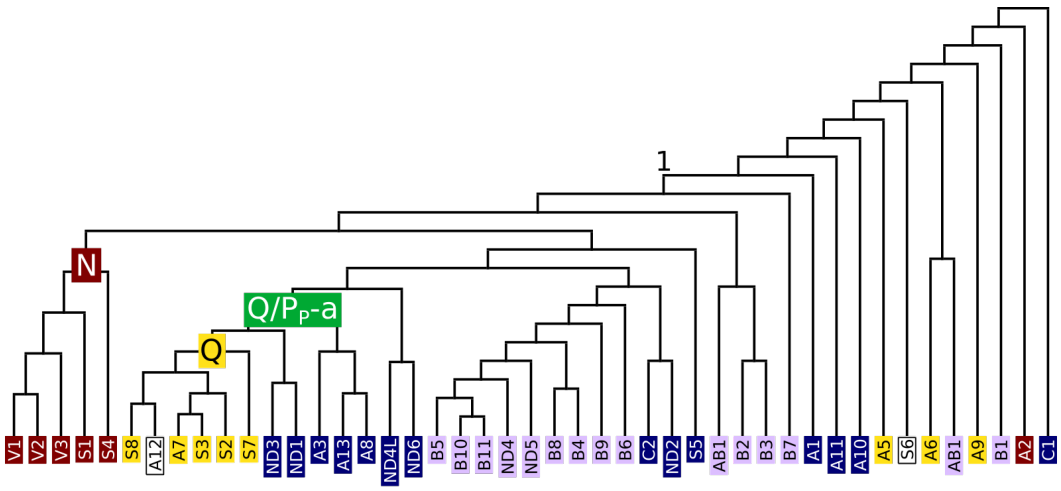
(a) Absolute edge weight.



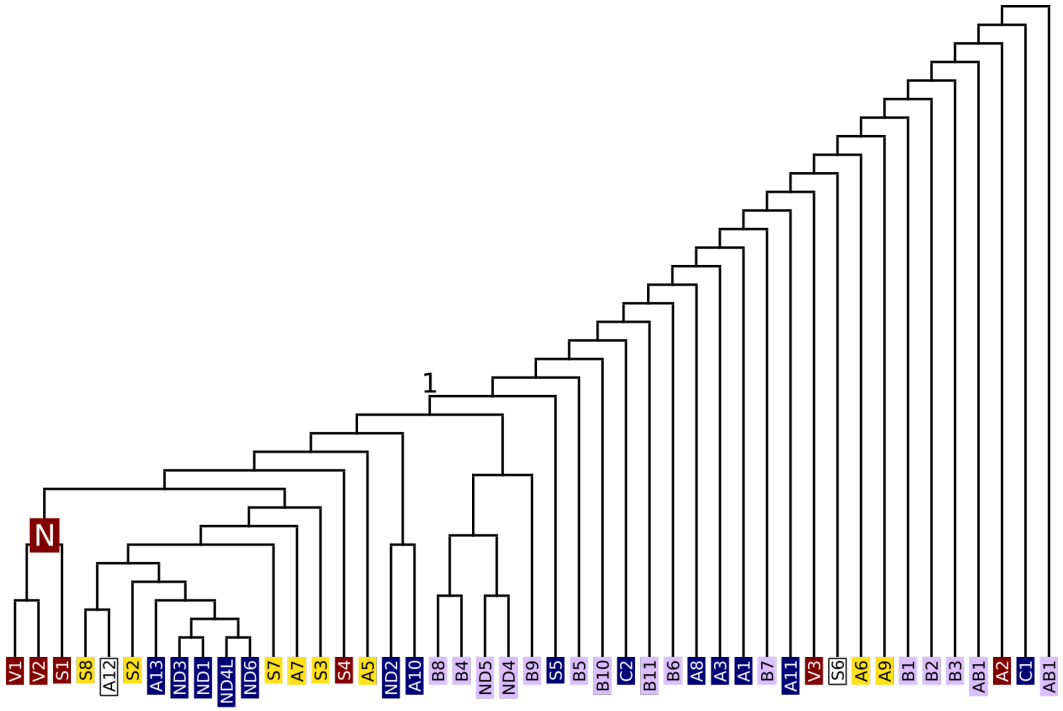
(b) Additive edge weight.



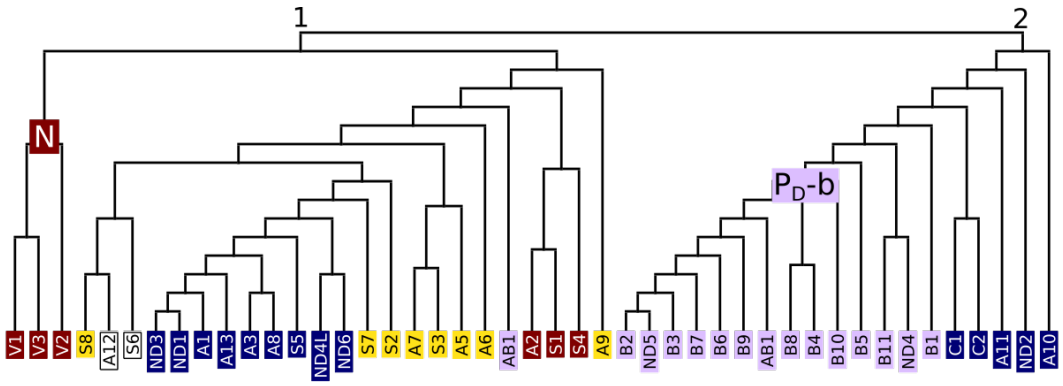
(c) Multiplicative edge weight.



(d) Square root edge weight.



(e) Logarithm edge weight.



(f) Minimum edge weight.

Figure 35: Predicted assembly pathways using different edge weight types for human respiratory complex I [20] (PDB 5xtd) represented as dendrograms. Leaves that correspond to subunits are labeled with the abbreviated gene names and colored according to the modules they are assigned to: N (red), Q (yellow), P_P (dark blue) and P_D (lavender). The subunits that are shared between the N and Q module are colored white. Inner vertices correspond to subcomplexes and the root to the final protein complex. Selected inner vertices are labeled.

Evaluation of the value of the clustering information distance

Random dendrograms To put the value of 0.6666 into perspective, we generated random dendrograms with the leaf labels of 5xtd and compared these dendrograms to the reference pathways using the nRF distance and the CID. We used the method by Furnas [72] (see Section 2.1.3) to generate random dendrograms, because the method draws uniformly from all existing dendrograms with the leaf labels of 5xtd. By drawing uniformly, the whole landscape of possible dendrograms is equally represented. We implemented Furnas' method in PTGLgraphComputation in a new class `RandomTreeGenerator`.

We generated three million random dendrograms. Checking for duplicates either demands high runtimes comparing each pair of dendrograms or large memory using a data structure to organize already seen dendrograms. In theory, $3.9 * 10^{66}$ different dendrograms are possible (see Section 2.1.3). We did not check for duplicates, because of the demands for resources and the low probability of duplicates.

We computed the nRF distance and the CID between each random dendrogram and each of the reference pathways. There are only four different values of the nRF distance comparing the random dendrograms to reference 1 or 2: 1.0, 0.9655, 0.931 and 0.8966. This shows the low sensitivity of the nRF distance that has been reported [71].

There are random dendrograms with a higher nRF distance than any of the predicted dendrograms and random dendrograms with the same distance as the absolute, multiplicative, square root, logarithm and minimum dendrogram. The additive dendrogram has a lower nRF distance than any of the random dendrograms. Because of the low sensitivity of the nRF distance, we did not use it for further analyses.

The CID on the other hand has a much better sensitivity with 583,770 distinct values. The minimum, maximum and median of the CIDs vary only slightly when comparing to reference 1 and 2 (see Table 20). The mean is the same. Because of the similarity of the two, we focus on the discussion of reference 1 for simplicity.

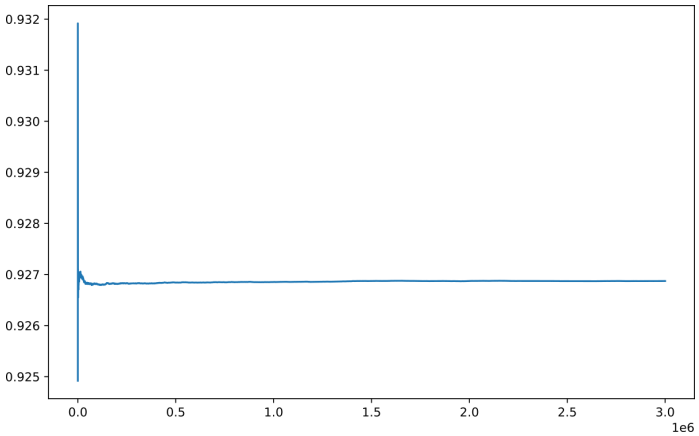
Table 20: Statistics of the clustering information distances of three million random dendrograms compared to reference 1 and 2. The clustering information distance is a value between zero and one. A value of zero indicates equal dendrograms.

	Minimum	Maximum	Mean	Median
Reference 1	0.8169	0.9687	0.9269	0.9283
Reference 2	0.8161	0.9682	0.9269	0.9282

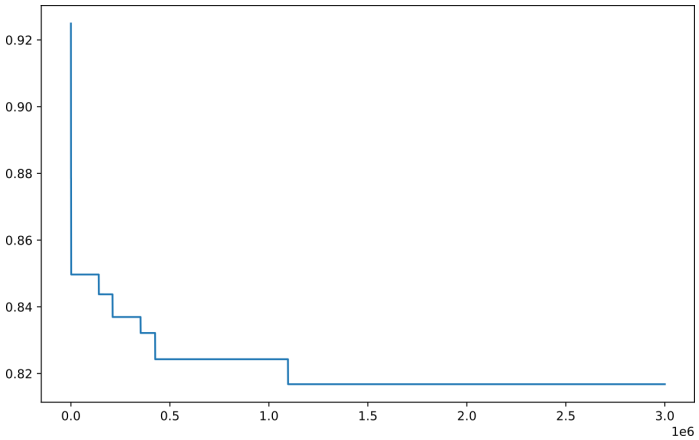
Three million random dendrograms are only a small fraction of $3.9 * 10^{66}$ different dendrograms. We used two approaches to monitor if the number of generated random dendrograms is meaningful for investigating the landscape of the CID or if we should generate more dendrograms. The first approach consecutively computes the mean and the second the minimum. Thus, we computed the mean and minimum for all n dendrograms using the dendrograms from d_0 to d_i for the dendrogram d_i. For the first approach, the idea is to stop when the mean does not change anymore. For the second approach, the idea is to stop when the gaps between newly encountered minima become larger.

The first approach was of limited use considering how many random dendrograms

are enough (see Figure 36a). The consecutive mean ranges from 0.9249 to 0.9319. This is only a small window of values. The consecutive mean reaches the mean of all values, 0.9269, at dendrogram 270 the first time. At dendrogram 13,573, there is a local high point of 0.9271. After that, the consecutive mean drops below the mean of all values and approaches it from there on. According to this approach, a few ten thousand random dendrograms would have been enough, because the consecutive mean did not change considerably anymore.



(a) Consecutive mean plotted on the y-axis.



(b) Consecutive minimum plotted on the y-axis.

Figure 36: Consecutive statistics of the values of the clustering information distance comparing three million random dendrograms to reference 1. The number of the dendrogram is plotted on the x-axis divided by one million (1e6).

We found the second approach more useful to decide when to stop generating dendrograms (see Figure 36b). The vertical lines of the plot show how much lower the CID of the new minimum is and the horizontal lines how long the gaps are until a new minimum is encountered. As expected, the gaps are extremely small at the beginning with 2, 1, 36 and 16 required new dendrograms, for example, until a new minimum is encountered. The gaps become larger quite fast with, for example, 139,544 required dendrograms for a new minimum. For the last 1.9 million dendrograms, no new dendrogram achieved a CID less than the minimum of 0.8169. The gap before that is 671,191 dendrograms. The CID decreased by 0.0075 from the second-last to the last

minimum. If the newly encountered CID always decreased by 0.0075 from there on, we would need to find a new minimum 20 times in order to encounter a dendrogram with the same CID as the additive dendrogram compared to reference 1. We decided to stop at three million random dendrograms.

We refer to the dendrogram with the highest and lowest CID to reference 1 as worst and best random dendrogram, respectively. The worst random dendrogram (see Figure 37a) shows no aggregation of modules as opposed to reference 1 (see Figure 7a). Most of the subunits assemble successively with the largest subcomplex which creates a staircase-like topology (see Section 2.1.3). Interestingly, both the worst random dendrogram and the absolute dendrogram exhibit a staircase-like topology. The worst random dendrogram has a CID of 0.9687 to reference 1 which is higher by 0.0983 than the CID of the absolute dendrogram. The larger difference is most likely caused by the different arrangement of the subunits, because the topology looks similar. In the absolute dendrogram, subunits of the same module often assemble one after the other, while in the worst random dendrogram subunits assemble independently of their assignment to modules.

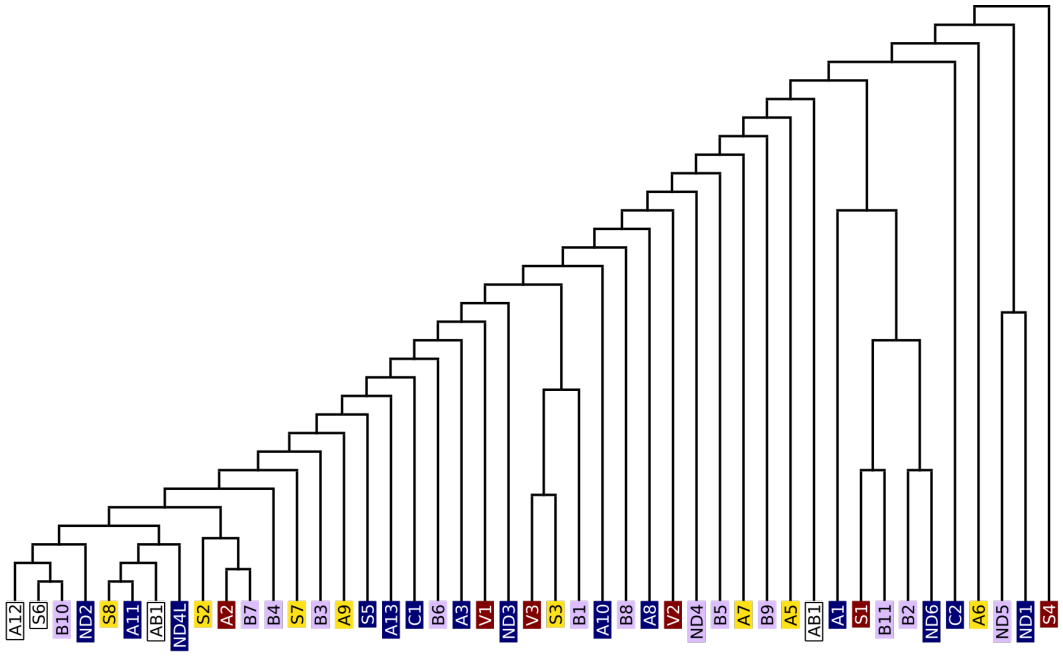
The best random dendrogram (see Figure 37b), on the other hand, shows a more modular topology. Subunits do not assemble with the largest subcomplex but assemble in subcomplexes that later merge with other subcomplexes. The subunits are not as unordered as in the worst random dendrogram considering the assignment to modules. In fact, certain features of the reference pathway 1 can be observed. For example:

- Subunits V1 and V2 assemble, and S1 is added in the next step
- Subunits C2, ND4L, ND3 and ND6 form a subcomplex
- Subunits A5 and S2 assemble, and S7 is added in the next step

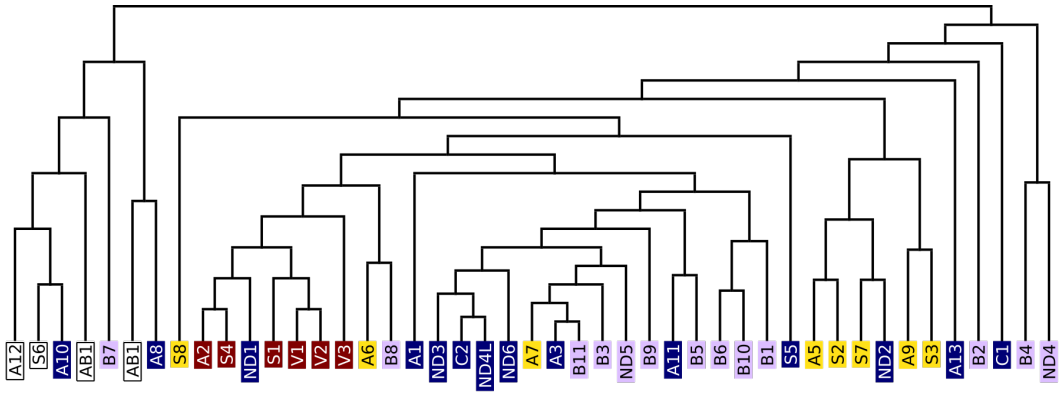
The worst random dendrogram does not share many similarities with reference 1 while the best random dendrogram contains features of reference 1. We concluded that the CID is a suitable measurement to assess how good a prediction is compared to the reference. The CID of the best random dendrogram to reference 1 is higher by 0.1502 than the CID of the additive dendrogram. Thus, none of the three million random dendrograms comes even near the additive dendrogram considering the CID.

The standard deviation of the CIDs of the random dendrograms is 0.0129. The CID of the additive dendrogram is twenty times the standard deviation less than the mean of the CIDs of the random dendrograms. We concluded that the additive dendrogram is a prediction of the assembly much closer to the reference than can be expected by chance. Even for a much larger data set of random dendrograms an equally or better performing dendrogram than the additive dendrogram cannot be expected.

Comparing non-binary and binary dendrograms We considered another property of the comparison using the CID: the reference pathway is non-binary in contrast to the predicted or generated pathways. We transformed the non-binary dendrogram of reference 1 to a binary dendrogram by imposing a binary order for each non-binary part. We imposed the binary order by greedily clustering the vertices (see Section 2.1.4) using additive edge weights (see Figure 38). We implemented an interactive agglomerative clustering, in which the user is presented with the table of



(a) Highest clustering information distance to reference 1.



(b) Lowest clustering information distance to reference 1.

Figure 37: Visualization of randomly generated dendrograms representing the assembly pathway of human respiratory complex I. Leaves that correspond to subunits are labeled with the abbreviated gene names and colored according to the modules they are assigned to: N (red), Q (yellow), P_P (dark blue) and P_D (lavender). The subunits that are shared between the N and Q module, and the two duplicates of subunit AB1 that can be assigned either to the Q or P_P module are colored white. Inner vertices correspond to subcomplexes and the root to the final protein complex.

all edges sorted by weight at each step and can choose which edge should be used for clustering. We refer to the dendrogram as greedy binary reference 1.

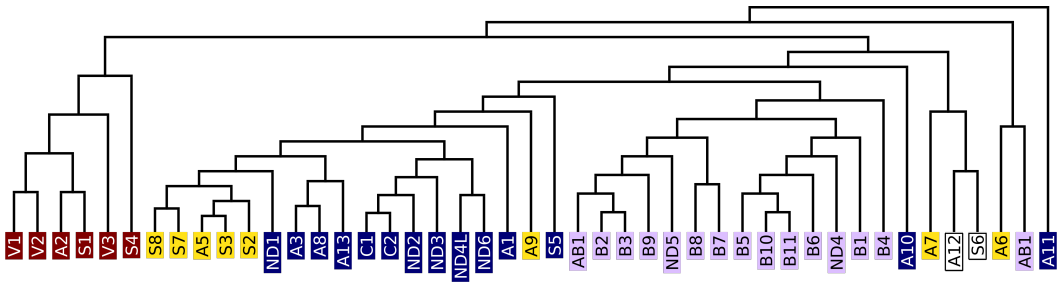


Figure 38: One proposed assembly pathway by Guerrero-Castillo et al. [84] for human respiratory complex I represented as dendrogram. Leaves that correspond to subunits are labeled with the abbreviated gene names and colored according to the modules they are assigned to: N (red), Q (yellow), P_P (dark blue) and P_D (lavender). The subunits that are shared between the N and Q module are colored white. Inner vertices correspond to subcomplexes and the root to the final protein complex. We imposed an order to steps, where more than two subunits or subcomplexes have been proposed to assemble creating a binary dendrogram.

The greedy binary reference 1 has a CID of 0.4362 to reference 1. It is possible that the greedy binary reference is not the binary dendrogram inferred from reference 1 that achieves the lowest CID because of the greedy approach. Still, we can see that an obviously very well matching dendrogram has a CID far from zero. This is probably a side effect of comparing a binary dendrogram to a non-binary dendrogram. If we aim for a CID of around 0.4362, because lower values cannot be achieved, the additive dendrogram with a CID of 0.6666 seems to have performed better than the value initially suggested.

Worst interactive dendrogram With the ability to interactively choose an edge for the agglomerative clustering, we produced a dendrogram by choosing the edge with the lowest edge weight at each step for the additive weight (see Figure 39). We refer to this dendrogram as worst interactive dendrogram.

The worst interactive dendrogram exhibits a staircase-like topology just like the worst random and the absolute dendrogram. In the worst interactive dendrogram, not all subunits assemble with the one largest subcomplex but assemble in three modules. For two modules, there is a perfect staircase-like topology. Regarding the structural modules of rCI, some parts seem to be ordered such as the subcomplex of B11, ND5 and AB1, which all belong to the P_D module.

The worst interactive dendrogram has a CID of 0.9308 to reference 1. This CID is 0.0379 smaller than the CID of the worst random dendrogram and 0.0604 higher than of the absolute dendrogram. However, an interactively created dendrogram is always bound to the edges that exist in the CG. The randomly generated dendrograms, on the other hand, are detached from the CG and can have subcomplexes with vertices that are not connected in the CG. We can infer from the high CID to the reference of the worst interactive dendrogram that greedily choosing the highest edge weight during the agglomerative clustering relates to the reference assembly pathway.

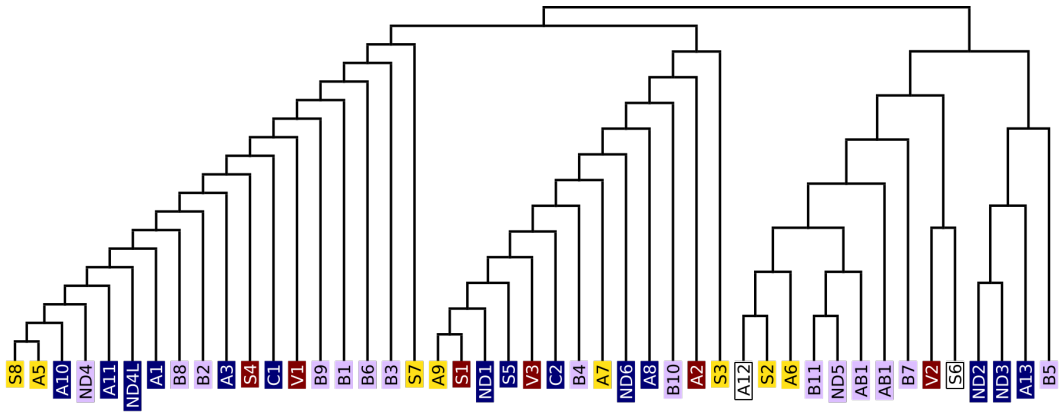


Figure 39: Assembly pathway of respiratory complex I of *Homo sapiens* [20] created by choosing edge with least additive weight in interactive agglomerative clustering. Leaves that correspond to subunits are labeled with the abbreviated gene names and colored according to the modules they are assigned to: N (red), Q (yellow), P_P (dark blue) and P_D (lavender). The subunits that are shared between the N and Q module are colored white. Inner vertices correspond to subcomplexes and the root to the final protein complex.

Contacts transitive by ligands The following passage is based on work by Möller [132]. I developed the idea of contacts transitive by ligands (CTLs) and supervised the work. In the following, I present the results of applying CTLs to the prediction of the assembly pathway.

The PTGL treats ligands as parts of the subunit the authors have assigned the ligand to. Contacts between a ligand and another subunit are added to the number of contacts of subunits in a CG. Biologically, this is questionable, because ligands are separate molecules and not part of the subunit. Moreover, the assignment of ligands to subunits is up to the authors and may be unreasonable. For example, a ligand may be assigned to a subunit that is spatially distant. This would lead to undesired contacts in the CG between this and other subunits because of the ligand. Neglecting contacts with ligands is no real solution. Ligands may be embedded between subunits, and completely neglecting the contacts causes an undesired representation of the interface of subunits.

The concept of CTLs tries to include contacts of ligands in a more desirable way. The goal is that the edge weights in CGs only refer to contacts between residues or nucleic acids, but include contacts of ligands implicitly. If a ligand is embedded between subunits, it mediates the contact between the molecules of the different subunits. Therefore, a CTL is a contact between molecules mediated by a ligand. A ligand mediates a contact between two atoms of different molecules if a ligand atom has a contact to both the atoms of the molecule.

We predicted the assembly pathway for 5xtd using CTLs (see Figure A5). The CID to reference 1 is 0.6938 and thus higher than without using CTLs. Möller has shown that the ligands are assigned well to subunits by the authors of the structure [132]. This may be the reason why including CTLs does not improve the prediction. We did not further investigate the pathway using CTLs, because the CID was lower without CTLs.

Conclusion We investigated the landscape of the CID with the use of randomly generated dendrograms. The additive dendrogram has a much lower CID to reference 1 than expected by chance. We showed that the greedy binary reference 1 does not achieve a CID near zero. Thus, the additive dendrogram is not that far away from the minimum reachable value as initially thought. We concluded that the additive dendrogram can be considered a prediction close to the reference. We showed that the CID can be used to measure the quality of a prediction compared to a reference and that choosing the edge with the highest normalized weight during agglomerative clustering is a suitable method for predicting the assembly pathway.

Qualitative analysis

Staircase-like versus modular topology The reference clearly exhibits a modular topology (see Figure 7). There are non-binary parts of the dendrogram, which may be resolved as staircase topology, for example, AB1, B2, B3, B7-9 and ND5. But even if these non-binary parts were resolved as a staircase topology, overall the dendrogram would exhibit a modular topology, because pre-N, pre-Q, Q/P_P-a, P_P-b, P_D-a, and P_D-b assemble separately.

The absolute and the logarithm dendrogram (see Figures 35a and 35e) exhibit a staircase-like topology. For both, there are modular parts, but from inner vertex 1, respectively, on to the root, there is a staircase topology. The square root dendrogram (see Figure 35d) exhibits a weaker staircase-like topology. From inner vertex 1 on to the root, with the exception of A6 and AB1, there is a staircase topology.

The minimum dendrogram is difficult to classify with our informal definitions. There are parts exhibiting a staircase topology, such as, inner vertex 1, with the exception of the subcomplexes {B4,B8}, {B11, ND4} and {C1,C2}. The assembly of the whole complex is divided into two modules, inner vertices 1 and 2, that assemble during the last step. Interestingly, all but one of the leaves under inner vertex 2 belong to the P_D module and five of 14 subunits of the P_P module. Thus, inner vertex 1 has leaves of only the membrane arm. Structurally, the two modules can be seen really well (see Figure 40).

The additive dendrogram (see Figure 35b) only exhibits a staircase topology in small parts and the multiplicative dendrogram (see Figure 35c) almost none at all.

The different normalizations of the edge weights have a huge impact on the topologies of the dendrograms. If no normalization is applied, which is the definition of the absolute weight, the topology of the dendrogram is mostly staircase-like. We concluded from this that the normalizations help produce a modular topology which is desired in correspondence with the reference.

The reason for more modular topologies can be seen in the multiplicative dendrogram. If two clusters are merged in the agglomerative clustering, the lengths of the subunits are added. Next, the edge weights are recalculated and the summed-up length is multiplied with the length of the other subcomplex. The factor in the denominator decreases the edge weights after two clusters are merged.

For example, in the first step of the greedy agglomerative clustering using multiplicative weights, subunits B2 and B3 with an edge weight of 0.0059 are merged. The weight of the edge between AB1 and B3 decreases from 0.0034 to 0.0022 after recalculation. As a consequence, mostly subcomplexes with the same or a similar number of subunits assemble. This example also shows how greedily choosing the edge of highest weight changes the course of the agglomerative clustering. The multiplicative edge weight clearly favors a modular topology.

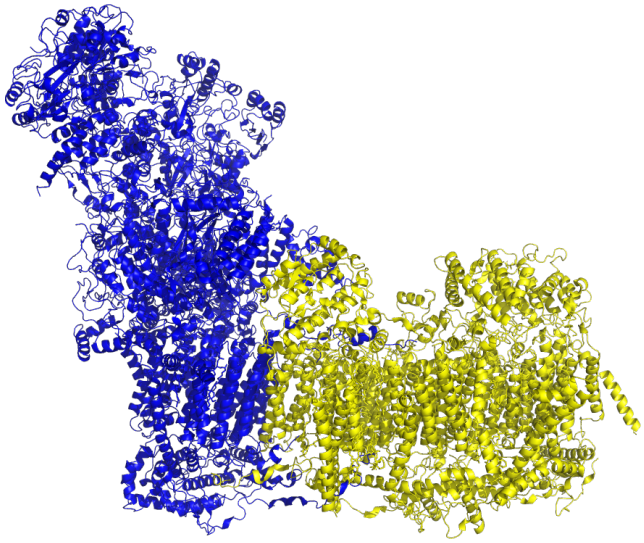


Figure 40: Visualization of the structure of human respiratory complex I [20] (PDB 5xtd). Residues shown in cartoon style and ligands as ball-and-stick model. The structure is colored according to the inner vertices 1 (blue) and 2 (yellow) of the dendrogram of the assembly prediction using minimum edge weights.

The effect seems to be lower for the additive edge weight. This is because the chain lengths are summed up and not multiplied in the denominator. But still, the lengths increase with each merge as opposed to the minimum weight, for example.

Pre-N module In the reference, V1, V2, S1 and A2 of the N module assemble pair-wise as subcomplexes {V1,V2} and {S1,A2}. The two other subunits of the N module, V3 and S4, join the assembly in the last step along with other subunits in no particular order. The pair {V1,V2} is assembled in the absolute, additive, square root and logarithm dendrograms. In the multiplicative dendrogram, V2 first assembles with V3. In the minimum dendrogram, V1 first assembles with V3. In this regard, the multiplicative and the minimum dendrogram are not in accordance with the reference.

In the additive, multiplicative and minimum dendrogram, S1 and A2 assemble. In the minimum dendrogram, the subcomplex {S1,A2} is multiple steps away from the subcomplex {V1,V2}. In the additive dendrogram, V3 joins V1 and V2 before the subcomplex {A2,S1} joins. After that, S4 joins the subcomplex which produces the final N module. Subunit V3 joins in a different order than in the reference, but still the additive dendrogram is most similar to the reference.

It is important to note that in the last step of the reference pre-N, pre-Q/pre-P and eight more subunits assemble in no proposed order. This means that, if V3 and S4 are predicted to assemble with the rest of the N module, this agrees with the reference. The additive dendrogram is the only dendrogram, where the subunits of N module assemble before subunits of another module are added. This means, the subunits of the N module are not scattered across the dendrogram as opposed to, for example, the multiplicative dendrogram, where subunit S4 assembles with subunits of the Q and later the P_P module, before it joins the pre-N module. Such a pathway does not match the reference.

Pre-Q module In the reference, A5, S3 and S2 assemble in no proposed order joined by S8 and S7 in no proposed order producing the pre-Q module. After that, subunits of the P_P-a module join. The subunits A7, A6 and AB1 of the Q module join the assembly in the last step. The subunits A12 and S6 that are shared between the N and Q module join in the last step as well.

Interestingly, subunit A12 assembles with S8 in all predictions. This means, for A12, none of the predictions matches the reference. Subunits A12 and S8 have an extensive interface that stretches almost across the length of the whole subunits (see Figure 41). Loop and helix regions align next to each other resulting in 75 residue contacts.

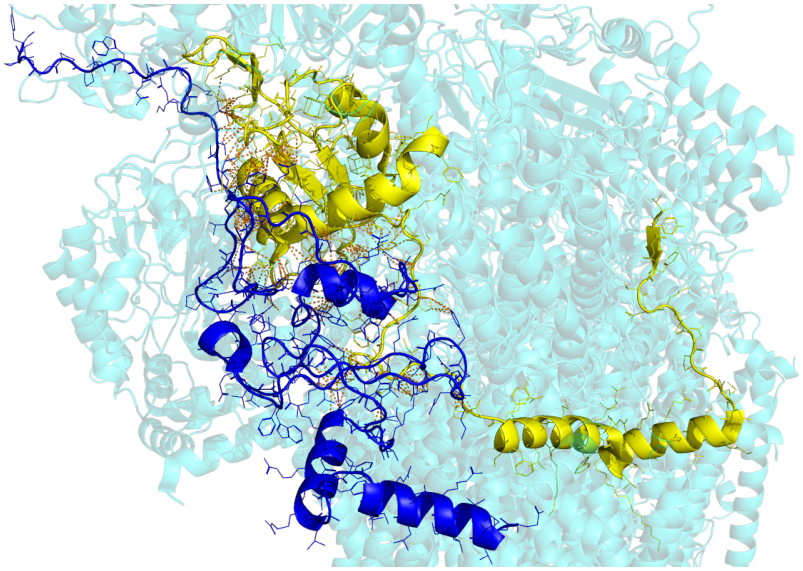


Figure 41: Visualization of the structure of human respiratory complex I [20] (PDB 5xtd). Residues are shown in cartoon style and ligands as ball-and-stick model. The subunits A12 and S8 are colored blue and yellow, respectively. The rest of the complex is colored cyan and depicted transparent. Distances between A12 and S8 less than four Angstrom are depicted as dashed orange lines.

The absolute weight between A12 and S8 is the third highest, and the additive weight the second highest of all edge weights. Because of this high edge weight, the subunits are assembled before S8 is assembled with the other subunits of the Q module in the greedy agglomerative clustering. This raises the question if S8 assembles with the subcomplex {A5,S3,S2} and possibly S7 if these subunits assembled first. The vertices in the neighborhood are well-connected (see Figure 42).

There are two ways for S8 to assemble according to the reference. In each case, subunits A5, S2 and S3 assemble first (Q1). Next, S8 joins either directly or after S7. Both ways compete with the edge between S8 and A12 with an absolute weight of 75 and an additive weight of 0.2351. For both ways, the edge between A12 and S8 has a higher additive and lower absolute weight (see Table 21).

When {Q1, S7} assembles first, the absolute weight between {Q1, S7} and S8 becomes considerably higher than between A12 and S8. This is not unexpected, because it is likely that the absolute weight is higher when more and more subunits are assembled, which leads to the staircase topology for the absolute weight. The additive weight normalizes for the chain lengths. It is expected that the additive weight is less for S8 and {Q1, S7} consisting of four subunits. Maybe the lengths

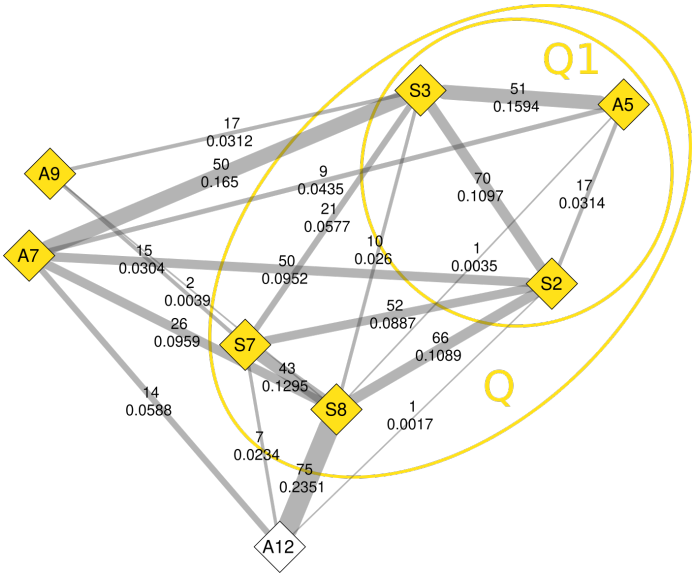


Figure 42: Visualization of the neighborhood of A12 in the Complex Graph of respiratory complex I of *Homo sapiens* [20] (PDB 5xtd). Only selected vertices and edges are shown. Vertices correspond to subunits and edges to spatial neighborhoods. Vertices are labeled with the abbreviated gene names of the subunits and colored according to modules: N (yellow) and N/Q (white). Edges are labeled with the absolute and additive edge weight. The width of edges corresponds to the additive weight. The vertices, assembling first for the pre-Q module (Q1) and next (Q), are labeled and encircled.

of the single subunits of a subcomplex still have an impact too big for the additive normalization. Even a non-greedy approach cannot assemble the pre-Q module according to the reference.

We investigated how the contacts can be differentiated by the type of secondary structure into the classes: SSE-SSE, SSE-loop and loop-loop. Note that, the order is of no interest meaning SSE-loop is the same as loop-SSE. The high number of contacts between A12 and S8 is mainly attributed to loop-loop contacts as opposed to the contacts between {Q1,S7} and S8 which are mainly attributed to SSE-loop contacts (see Table 21). Loop regions are more flexible and contacts may be less reliable in the sense whether they consistently exist or are just resolved for this state of the complex. The results indicate that contacts involving SSEs are more meaningful in the context of the prediction of the assembly pathway.

In the additive, multiplicative and minimum dendrogram, A12 and S6 assemble with subunits of the Q module. This suggests both subunits are more likely part of the Q than of the N module. This is the same result as for the prediction of modules (see Section 3.4).

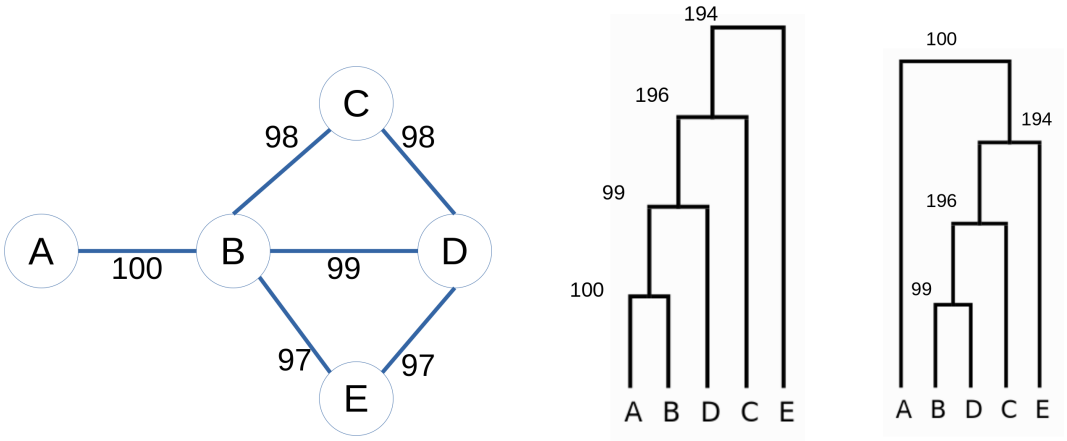
Large Interface Score We defined the Large Interface Score (LIS) for two reasons. First, to help with cases, where the greedy approach does not find the desired assembly order, such as A12 and S8. Second, to have a measure of quality for a dendrogram, for which no reference is available.

We define the LIS as $\sum_v \frac{w_i}{l_i}$, where v are all inner vertices, w_i is the edge weight of the edge that is merged during the agglomerative clustering leading to v_i and

Table 21: Absolute and additive weights of selected edges of the Complex Graph for human respiratory complex I [20] (PDB 5xtd). Absolute edge weights are differentiated in classes according to whether a secondary structure element (SSE) is involved: SSE-SSE, SSE-loop and loop-loop. Vertices are labeled by the abbreviated gene names of the subunits.

Vertex 1	Vertex 2	Absolute				Additive
		Total	SSE-SSE	SSE-loop	loop-loop	
S8	A12	75	3	33	39	0.2351
{S2,S3,A5}	S8	77	18	39	20	0.0832
{S2,S3,S7,A5}	S8	120	18	64	38	0.1109

l_i is the number of leaves under v_i. Thus, the LIS increases when large interfaces expressed as high edge weights are merged early in the agglomerative clustering (see Figure 43). The LIS can be computed for each edge weight type independent from the edge weight type that was used for the agglomerative clustering. For example, the additive dendrogram can be assigned a LIS based on the absolute weights of the CG.



(a) Complex Graph (b) Greedy (c) Optimum

Figure 43: Example for the Large Interface Score (LIS).

The case of A12 and S8 (see Figure 42) is similar to the example for the LIS (see Figure 43a). The LIS cannot be trivially computed for A12 and S8, because the sub-dendrogram of the additive and reference dendrogram differ not only in the order of the subunits but in the composition. The LIS of two dendrograms can only be compared between dendrograms that are based on the same CG.

We computed the LISs for all edge weight types for the additive and greedy binary dendrogram (see Table 22). The LISs are higher for the additive dendrogram than for the greedy binary dendrogram for all weight types. This means that the LIS cannot be used to globally refine the dendrogram making it more similar to the reference.

Table 22: Large interface scores of additive and greedy binary dendrogram for different edge weight types.

Edge weight type	Additive dendrogram	Greedy binary dendrogram
Absolute	792.6	657.4
Additive	1.71	1.37
Multiplicative	0.025	0.0205
Square root	34.6	27.8
Logarithm	125.49	102.81
Minimum	5.95	4.88

pre-Q/P_P-a module In the reference, pre-Q assembles with ND1. Next, subcomplex {Q, ND1} and subunits A3, A8 and A13 assemble in no proposed order. Two aspects are important for the comparison with the predicted pathways: the subunits of the P_P module, ND1, A3, A8 and A13 assemble and they join the pre-Q module.

The additive and the square root dendrogram are the only dendrograms where A3, A8 and A13 assemble. In the other dendrograms, the subunits are separated. For example, in the absolute dendrogram, A3 joins the subcomplex right after A8, but A13 is assembled much earlier.

Subunit ND1 is separated from A3, A8 and A13 and assembles with other subunits of the P_P module for all but the absolute and square root dendrogram. For example, in the additive dendrogram, ND1 assembles with ND3, ND4L and ND6 before joining the subcomplex of A3, A8 and A13 forming P_P-a. In the absolute dendrogram, ND1 and in the square root dendrogram, subcomplex {ND1,ND3} join a subcomplex of S8, A12, A7, S3, S2 and S7, which contains all but one subunit of the pre-Q module.

Considering both aspects, the square root dendrogram is most similar to the reference, with the additive dendrogram being second.

P_P-b module In the reference, ND2, C1 and C2 assemble. It is unclear if C1 and C2 assemble prior in the reference (see Section 2.5.2). Next, subunit ND3 joins the subcomplex. The subcomplex {ND2, C1, C2, ND3} and the subunits ND4L and ND6 assemble in no proposed order forming the P_P-b complex.

The subunits C1 and C2 assemble in the additive, multiplicative and minimum dendrogram. Only in the additive dendrogram, ND2 joins the complex of C1 and C2. In the multiplicative dendrogram, for example, the subcomplex {C1,C2} assembles with a subcomplex of the P_D module composed of B1, B5, B10 and B11.

In the additive dendrogram, subunit A10 joins the subcomplex {C1, C2, ND2} differing from the reference. The subunits ND4L and ND6 assemble in all dendrograms. In the additive dendrogram, the subcomplex of ND4L and ND6 assembles with a subcomplex of ND3 and ND1. This subcomplex assembles with a subcomplex of A3, A8, A13 and S5. So there is a subcomplex of ND4L and ND6, but the subcomplex and subunit ND3 are in a different spot compared to the reference. The additive dendrogram is still the most similar to the reference, because there is no subcomplex of C1, C2 and ND2 in the other dendrograms.

P_D-a module In the reference, subunits B5, B10 and B11 assemble in no order joined by B6 afterwards. The subcomplex {B5, B10, B11, B6} and subunits B1 and ND4 assemble in no proposed order forming P_D-a.

Only in the additive and the square root dendrogram, subunits B5, B10 and B11 assemble. The order is the same for both dendrograms: first B10 and B11 assemble. The multiplicative, logarithm and minimum dendrogram share some similarities with the reference dendrogram, because two of three subunits assemble directly or right after another, but the third subunit is separated.

In the additive dendrogram, B6 assembles with the subcomplex of B5, B10 and B11. In the next step, ND4 joins the subcomplex. However, subunit B1 is separated and joins in the last of all steps of assembly of rCI. Nevertheless, the additive dendrogram agrees with the reference in all but one subunit regarding assembly of the correct subunits and the order.

P_D-b module In the reference, subunits B2, B3, B7-9, AB1 and ND5 assemble forming the P_D-b module without an order proposed among them. Seven subunits, assembling without proposed order, makes this the second-highest number of children of an inner vertex. Comparing the reference to the predictions, only the composition of subunits but not the order can be discussed.

In the absolute and logarithm dendrogram, most of the subunits are scattered across the dendrogram. Some subunits are close to each other, but because of the staircase-like topology, they do not form a separate subcomplex, but successively join the largest existing complex. In the square root dendrogram, there is a subcomplex of B2, B3 and AB1, which joins a large assembly and is joined by B7 after that. The subunits B8, B9 and ND5 are scattered over the remaining dendrogram.

In the multiplicative and minimum dendrogram, there is a subcomplex of all subunits of the P_D-b module and two additional subunits: B4 and B6. In the additive dendrogram, there is a subcomplex of AB1, B2, B3, B9 and ND5 (P_D-b_1) and a subcomplex of B4, B7 and B8 (P_D-b_2). Both subcomplexes assemble after P_D-b_1 assembles with P_D-a.

The multiplicative, minimum and, aside from splitting the module, additive dendrogram contain a subcomplex of the subunits of the reference. For all three, the subcomplex additionally contains subunits B4 and B6 disagreeing with the reference. In the reference, B6 is part of the P_D-a module and B4 joins the subcomplex after the assembly of P_D-b.

Assembly of modules into the final complex In reference 1, pre-Q/P_P-a, P_P-B and subunits A1 and A9 assemble without proposed order forming pre-Q/P_P. Next, pre-Q/P_P, P_D-a, P_D-b and subunits A10, S5 and B4 assemble in no proposed order forming the module pre-Q/pre-P. In the final step, pre-N, pre-Q/pre-P, V3, A12, S6, A7, S4, A6, AB1 and A11 assemble in no proposed order. Because of the single subunits, this is the step with the highest number of assembling subcomplexes in no proposed order.

All dendrograms contain a subcomplex similar to the pre-N module. The absolute and the logarithm dendrograms show no further modules besides the pre-N module because of the staircase-like topology. The minimum dendrogram contains a subcomplex similar to the P_D-b module. The minimum dendrogram contains no further modules, but instead assembles into two large subcomplexes (vertices 1 and 2).

The square root dendrogram contains a subcomplex similar to the pre-Q module. The subcomplex assembles with ND1, ND3, A3, A8 and A13. The resulting subcom-

plex is similar to the pre-Q/P_P-a complex. Further subcomplexes cannot be clearly identified and the dendrogram becomes staircase-like towards the root.

The multiplicative dendrogram exhibits a modular topology, but not all of the subcomplexes clearly correspond to modules of the reference. The dendrogram contains a subcomplex similar to the pre-Q module and one similar to the P_P-a module. Both assemble forming a subcomplex similar to the pre-Q/P_P-a module. A subcomplex corresponding to the P_P-b module cannot be clearly identified, because, for example, the subunits C1 and C2 assemble with subunits of the P_D module. This is why the P_D-a module cannot be clearly assigned to vertex 1. The dendrogram contains a subcomplex similar to the P_D-b module. No vertex corresponds to the pre-Q/pre-P module, because the pre-N module assembles with the pre-Q/P_P module.

Considering the assembly of modules, the additive dendrogram is most similar to reference 1. There is a subcomplex similar to the pre-Q/P_P-a module that assembles with a subcomplex similar to the P_P-b module. The P_D-b module is split across two subcomplexes one of which assembles with a subcomplex similar to the P_D-a module. With the assembly of the pre-Q/pre-P_P and the pre-P_D module, a subcomplex similar to the pre-Q/pre-P module is formed. The pre-N and the pre-Q/pre-P module assemble followed by single subunits producing the final complex similar to the reference 1. The single subunits do not match exactly, but A6 and A11 match between the additive and the reference 1 dendrograms.

Conclusion

We discussed the dendrograms of the agglomerative clusterings for the different edge weight types. We only discussed reference 1, because the CID suggested that the predictions are more similar to reference 1. We deduced from the staircase-like topology of the absolute dendrogram that normalizations of the edge weights are necessary. The logarithm and minimum dendrogram were more similar to the absolute dendrogram than to the reference. The square root dendrogram contained inner vertices that can be compared to modules of the reference, but showed a staircase-like topology towards the root. The multiplicative one showed a rather modular topology and similarities with the reference in many cases.

The additive dendrogram was most similar to the reference. All modules of the assembly of reference 1 could be assigned to inner vertices of the additive dendrogram. From the assigned modules, most subunits were correctly assembled. Together with the results from the analysis of the CIDs we concluded that the additive normalization worked best for the agglomerative clustering of 5xtd predicting the assembly pathway.

3.5.4 Respiratory super- and megacomplexes of *Homo sapiens*

We extended the use case of 5xtd to 5xth and 5xti (see Section 2.10.4). Just like with the detection of modules (see Section 3.4), we wanted to investigate how the agglomerative clustering performs for multi-complex structures. We addressed the question if the agglomerative clustering produces stable results, for example, for the prediction of the assembly of rCI if the complex is encompassed by subunits of other complexes. We computed the CGs (see Figures A6 and A8) and predicted the assembly pathways using agglomerative clustering.

We discuss only the additive dendrograms (see Figure 44), because the additive dendrogram of 5xtd performed best considering the comparison to the reference using the CID and visual inspection. We present the dendrograms for the other edge

weight types in the Appendix (see Figures A7 and A9). We compare the predictions only to reference 1, because of the learnings from 5xtd.

In the additive dendrogram of 5xth (see Figure 44a), complex I and IV assemble separately. The two copies of complex III assemble partly together. Two large parts of III-1 and III-2 assemble separately before assembling together (pre-III_2). UQCR10 and UQCR11 of III-1 assemble with UQCRB of III-2 joined by UQCRH of III1-1. This subcomplex joins the pre-III_2 subcomplex. Only in the multiplicative (see Figure A7b) and logarithm (see Figure A7d) dendrograms, the copies of complex III assemble separately. In the additive dendrogram, complex I assembles with III_2 and this subcomplex assembles with subcomplex IV.

In the additive dendrogram of 5xti (see Figure 44b), the copies of the complexes I and IV assemble separately. The copies of complex III assemble mostly separately before assembling together. Only subunit UQCRB joins after assembly of pre-III_2. Complex I-1 assembles with III_2. Complex I-2, IV-2 and IV-1 join sequentially.

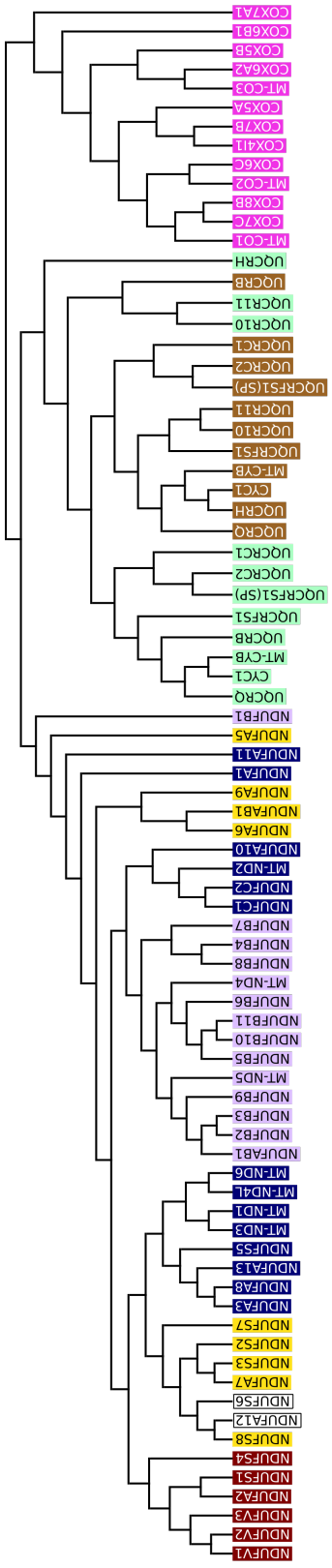
We considered complex I isolated from the rest of the predicted assembly pathway and computed the CID to reference 1 (see Table 23). In the absolute dendrogram of 5xti, the subunits of complex I are scattered across the dendrogram. The CID cannot be computed in this case, because no connected dendrogram with the same leaf labels as complex I can be retrieved. In the logarithm dendrogram of 5xti, complex I-1 is scattered, because AB1 assembles with complex I-2. We did not compute the CID between CI-2 and the reference, because we used complex I-1 for the other edge weight types and wanted to retain comparability.

Table 23: Clustering information distance of predicted assembly pathways for different edge weight types of human respiratory supercomplex $I_1III_2IV_1$ (PDB 5xth) and human respiratory megacomplex $I_2III_2IV_2$ [20] (PDB 5xti) to reference pathway 1. Values of respiratory complex I (PDB 5xtd) given for comparison. A dash indicates that complex I cannot be isolated from the dendrogram and the clustering information distance cannot be calculated, because its subunits are scattered across the dendrogram.

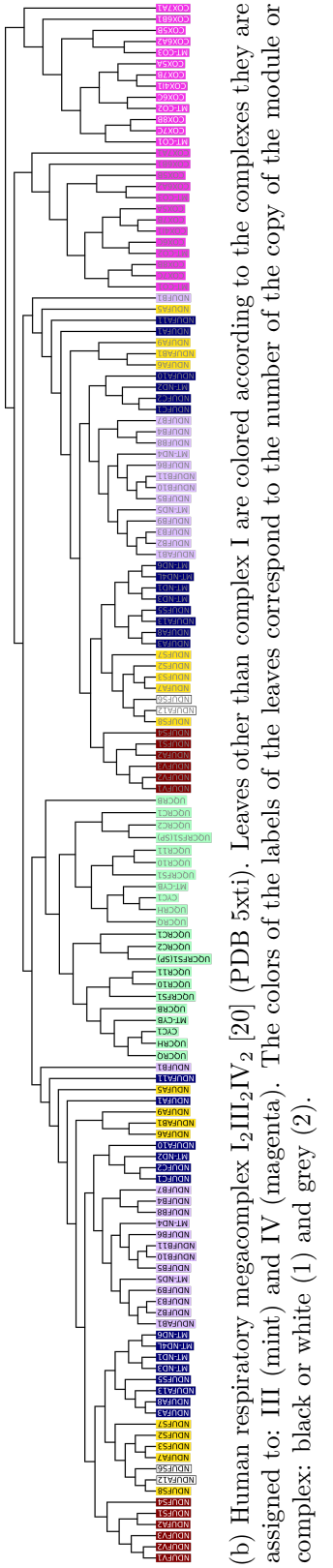
Edge weight type	5xtd	5xth	5xti
Absolute	0.8704	0.8168	-
Additive	0.6666	0.6996	0.672
Multiplicative	0.7173	0.7076	0.687
Square root	0.7639	0.7352	0.7783
Logarithm	0.8833	0.8785	-
Minimum	0.784	0.7163	0.7743

The additive dendrogram of 5xtd remains the dendrogram with the overall lowest CID to reference 1. Interestingly, the CID is lower for 5xth than for 5xtd per edge weight type except for the additive edge weight. For example, the CID for the minimum weight is 0.784 for 5xtd and 0.7163 for 5xth. For 5xti, there are examples where the CID is higher or lower than for 5xth or 5xtd.

In the computation of the CID, we neglected that the copies of AB1 are interchangeable just like for 5xtd. The resolution of 5xti of 17.4 Å is far too high to draw final conclusions with respect to every atom or residue contact. We assume that contacts are plausible on a large scale for discussing the assembly pathway of the agglomerative clustering.



(a) Human respiratory supercomplex $I_1III_2IV_1$ [20] (PDB 5xth). Leaves other than complex I are colored according to the complexes they are assigned to: III-1 (mint), III-2 (brown) and IV (magenta).



(b) Human respiratory megacomplex $I_2III_2IV_2$ [20] (PDB 5xti). Leaves other than complex I are colored according to the complexes they are assigned to: III (mint) and IV (magenta). The colors of the labels of the leaves correspond to the number of the copy of the module or complex: black or white (1) and grey (2).

Figure 44: Predicted assembly pathways of respiratory supercomplex and megacomplex of *Homo sapiens* [20] using additive edge weights represented as dendrogram. Leaves correspond to subunits and are labeled with the gene names. Leaves of complex I are colored according to the modules they are assigned to: N (red), Q (yellow), P_P (dark blue), P_D (lavender) and N/Q (white). Inner vertices correspond to subcomplexes and the root to the final protein complex.

We conclude that the agglomerative clustering is able to differentiate between different complexes for multi-complex structures depending on the edge weight type used. The additive edge weights do not perform ideally in this matter, because the copies of complex III do not assemble separately. When looking at complex I only, for neither 5xth nor 5xti independent of the edge weight type, the prediction is closer to the reference than for 5xtd using additive edge weights. This suggests that the agglomerative clustering should be applied separately on subcomplexes that are known to assemble separately.

4 Conclusion and outlook

Conclusion

The aim of this work was to analyze protein complexes up to large sizes computationally, because they are almost impossible to analyze manually. But they also pose problems for computational analyses because of long runtimes. We implemented a fast contact check and a computation of orientations between SSEs using vectors. Both enabled us to run even large complexes in a feasible time of a few minutes.

The feasible runtimes enabled us to implement a pipeline that computes the contacts and CGs for all steps of an MD simulation. We compared different approaches to count contacts and changes of contacts. We visualized changes of contacts during the simulations as heat maps put onto the structure and as line plots. The visualizations helped spotting interesting time steps and structural regions.

We introduced five normalizations for edge weights of a CG. We normalized the number of residue contacts by the lengths of the participating subunits. We compared the normalizations against each other in the following methods.

We applied graph clustering to CGs using the normalizations to identify structural and functional modules of a protein complex. As use cases we examined rCI of *Thermus thermophilus*, rCI of *Homo sapiens* and the respiratory supercomplex of *Homo sapiens*. The clustering using no normalization performed best, successfully identifying, for example, three out of four modules of rCI of *Homo sapiens*.

We applied agglomerative clustering to the vertices of a CG to predict the assembly pathway of a protein complex. We compared our method to two state-of-the-art methods on a data set of 21 protein complexes with known assembly pathways. Our approach was correct in more cases for all three defined measurements of quality. It ran in seconds to minutes compared to a runtime of multiple days of Path-LZerD even when parallelized.

We predicted the assembly pathways of rCI, the respiratory supercomplex and the respiratory megacomplex of *Homo sapiens*, using the different normalizations of the edge weights. We compared the predictions to a proposed assembly pathway of rCI that has been experimentally determined. We quantified the difference between our predictions and the reference using the nRF distance and the CID. The CID proved to be more sensitive and therefore fit for the analysis. We examined the landscape of the CID using randomly generated dendrograms. We discussed in detail the differences between the predicted pathways and the reference. We concluded that the predicted pathway using the additive edge weight performed considerably well, as it achieved a much better CID than expected at random and contained important features of the reference pathway, such as the modular assembly.

We used dendrograms defined as non-binary and binary trees to represent assembly pathways. Dendrograms structure the assembly pathway by grouping the subunits of the subcomplexes. Dendrograms are versatile allowing both the assembly of two or an arbitrary number of subunits per step. We represented dendrograms in the Newick format, which is human-readable and can be handled by many software tools. We showed advantages of the Newick format compared to a string representation of the steps of the assembly pathway as, for example, in Peterson et al. [61]. Most importantly, dendrograms represented as Newick strings are defined and can be interpreted and manipulated by software. Dendrograms can be visualized by existing software tools and the visualizations are easy to read. We conclude that dendrograms are a suitable representation of assembly pathways.

Currently, ligands are either counted as part of a subunit or treated as a separate subunit. Both approaches have strengths and weaknesses. In the heatmaps of PTGLdynamics, the ligands that are treated as a separate subunit distort the range of the coloring scheme. In the agglomerative clustering, contacts of ligands are counted as contacts between protein subunits. We tested a different approach with the implementation of CTLs. The assembly prediction did not benefit from the inclusion of CTLs. Still, we need a way to treat ligands that is motivated by the role they play in protein complexes.

The normalizations have proven to be necessary for the assembly prediction, because the absolute weight did not produce convincing results. For the detection of modules, the absolute weight performed best. It is possible that different methods require different normalizations. We believe that normalizing the number of contacts by the length of the participating subunits makes sense and should be further explored. We tested intuitive normalizations, but in the end, more elaborate normalizations could produce better results.

We showed how graphs, representing the topology of protein complexes, can be used for analyses of time-resolved data of dynamics, predict structural and functional modules, and predict the assembly pathway. We tried different approaches for each method, validated the results and compared them to references from the literature. We showed the strengths of the graph-theoretic methods and that they can be reliably applied for research.

Outlook

The current implementation of PTGL is considerably fast. Nevertheless, the implementation should be optimized wherever possible to ensure feasible runtimes for even larger structures and for the increasing number of structures deposited to the PDB. We suggest to identify the parts of the code that take longest and investigate how they could be optimized. We showed that the computation of contacts is already mostly optimized, but also seemingly small parts of the software, such as the assignment of orientations to contacts of SSEs, can take longer than necessary.

We showed that the vector method produces desirable results for the assignment of orientations to SSEs. Finding the best fitting line through all backbone atoms of an SSE in 3D space may capture the orientation of an SSE even better. In general, we need more research to identify the best parameters such as the threshold of the angles. To allow for different thresholds, the angles between SSEs could be saved in the database of PTGL. A user could then choose the desired thresholds and assign the orientations based on them. A differentiation between helices and strands might be favorable, as well.

We presented an example, where the SSE assigned by DSSP was unusually long and bent. This interfered with the correct assignment of an orientation to the contacts of the SSE. A solution would be to break up long SSEs for the computation of an angle. Using a different tool for the assignment of SSEs, such as SCOT [122], might also be advisable.

The pipeline PTGLdynamics allows automatic analyses of MD simulation data. The program call should be simplified to be more user-friendly. The user interface could be extended by a setting's file. A setting's file would allow moving arguments of the program call to a file without the need to pass the arguments with each call. The setting's file would also allow providing more specific settings that can currently only be applied by changing the code of the software, such as some paths to directories.

The pipeline could be extended by several features. Adding the possibility to automatically investigate and compare two data sets of MD simulations might open up the possibility for interesting analyses. Comparing the changes of SSEs and PGs is not trivial, because the assignment of residues to SSEs may change with each step, but would be quite interesting. More graph-theoretic approaches could be applied and mapped back to the structure such as the betweeness centrality of subunits or residues to find subunits or residues that are important for the stability of the protein.

Tracking the contact partners of each residue made it possible to see the flexibility of each residue during simulation. The information could be used to investigate the flexibility per residue type. It might be interesting to normalize the number of changes for the typical number of changes of the residue type. This way, outstanding residues per type could be identified.

We showed that graph clustering is able to identify modules within a complex. We learned that the Leiden algorithm finds the optimum or a comparably good partition, but may fail to find the optimum partition for large CGs. We suggest to use algorithms that ensure finding the optimum partition and noticed that the runtime for this is feasible for the typical sizes of CGs. It might be interesting, however, to investigate not only the best but also comparably good partitions. From these partitions, a consensus partition could be derived and the vertices could be assigned a score based on how certain the vertex is placed in this partition. The analysis may be repeated with more recent structures, such as respiratory complex I from *Thermus thermophilus* [21] (6y11) instead of 4hea, because they might be more complete or have a better resolution.

The partition using absolute and additive edge weights differed between 5xtd and 5xth. We suggest a detailed analysis to find the reason. In general, we need more use cases to draw a final conclusion on which normalization performs best or excels for which case.

For complexes, where the modules consist of only a few subunits, the clustering may fail to reproduce this, because it merges modules into one cluster, for example, as seen with the RNA polymerase II [129]. Such graphs are just not fit to find the modules with graph clustering, because of the resolution limit of modularity. Applying the graph clustering on a lower level of abstraction might help. One approach could be dividing the subunits into domains which become the vertices of the graph.

The partition of a CG could be used to predict the turnover rate of subunits or groups of subunits. We hypothesize that groups of subunits, for example, corresponding to a cluster in the CG, are replaced together. The definition of a cut of a graph could be used to identify subunits that are loosely connected to the remaining complex. This could be investigated using rCI as a use case for which different turnover rates have been reported [133].

The agglomerative clustering of the vertices of a CG performed best compared to two state-of-the-art methods on a data set of 21 protein complexes. The prediction was not correct for all cases and could be further improved. In the style of subcomplex BSA, we could consider all possible interfaces even involving multiple proteins at one step instead of greedily merging two subcomplexes. The effect of important parameters, such as the type of normalization and the contact threshold, should be investigated. The goal is to present the user the best set of parameters that most likely results in the correct prediction.

The data set of Peterson et al. [61] proved to be useful for the comparison of methods for the assembly prediction. The data set could be further extended to create a benchmark data set for computational assembly prediction. We suggest adding complexes of more than seven subunits, for example, up to rCI with 45 subunits. This poses the problem that the assembly pathway for larger complexes might not be as reliably verified as for small complexes. We also discussed the strengths and weaknesses of the three applied quality measurements for the comparison of predictions. We suggest a deeper investigation which might yield a more consistent and reliable quality measurement.

We discussed the case of 1s5b which is an assembly of five B subunits and an A subunit. We propose to integrate concentrations of subunits in some form. Stochastically, there is a higher chance that two B subunits encounter each other in the beginning. Over the course of the assembly, the number of available B subunits decreases making it more probable that an A subunit assembles with the existing subcomplex of B subunits. This might help in the case of 1s5b, where three B subunits assemble first before an A subunit joins. The idea marks a step in the direction of integrating rates and kinetics.

For the comparison of our prediction of the assembly pathway of rCI to a reference, we used the CID. The CID has proven to be a good choice, but according to Smith "[s]ubtracting the similarity score from the information content of all splits in the 'correct' tree" [71] might be better if one dendrogram is considered correct. The reference dendrogram might be considered correct for this purpose.

We created random dendrograms to investigate the landscape of the CID. We used a method by Furnas [72], which uniformly draws from all possible dendrograms. This gives an unbiased view on the landscape of the CID, but this might not be desired. Assembly pathways are usually modular as can be seen for rCI, so we might want a bias towards dendrograms with a low depth. A naive approach would perform an agglomerative clustering and choose two random clusters at each step. This yields symmetric dendrograms, which have a low depth, with a higher probability. The approach by Orsini et al. [134] starts from the original dendrogram and applies random changes preserving different graph properties defined by the user. Applying the method to dendrograms representing assembly pathways requires a lot of theoretical work that is out of the scope of this thesis, but may yield interesting insights.

We presented the LIS as a quality measure for a dendrogram representing an assembly pathway. We could not show that the LIS is higher for a dendrogram of rCI based on the reference than for a predicted dendrogram. Nevertheless, the LIS is a quality measure that can be applied to guide the assembly pathway and might be useful for examples where no reference pathway exists. A possibility to make use of the LIS for refining the dendrogram of a prediction is, for example, simulated annealing. The greedily predicted dendrogram could be changed optimizing the LIS yielding an overall better prediction in the end.

Even when no complete and reliable reference pathway is available for a protein complex, there is often prior knowledge on parts of the assembly. For example, it might be known that two subunits assemble first or two subunits definitely do not assemble before a specific subcomplex is formed. A good computational assembly prediction should be able to take prior knowledge in consideration. The interactive agglomerative clustering we implemented can be used to apply prior knowledge by choosing the edge of highest weight that complies with the prior knowledge. To

reduce manual steps, prior knowledge could be read in as pre-conditions that the agglomerative clustering will comply with.

There are two major aspects that are untreated in the agglomerative clustering of the CG of the final structure: dynamics during the assembly and assembly factors. Once structures of assembly intermediates have been resolved, the assembly pathway for the intermediates could be predicted and used as prior knowledge for the prediction of the assembly of the final structure. Prior knowledge of assembly factors could be applied by inserting new edges between subunits that are connected via an assembly factor during assembly.

Both the detection of modules and the prediction of the assembly pathway are based on the contacts defined by PTGL. In the assessment of the assembly prediction, we saw that the numbers of contacts of SSE-SSE, SSE-loop and loop-loop may vary between interfaces. We are sure that contacts involving SSEs are more reliable and should be favored. We propose investigating interfaces for the classes of contacts and testing weighting contacts involving SSEs differently.

Our main use cases were complexes from the respiratory chain. We propose investigating more use cases to gain a broader view of how the agglomerative clustering performs. Candidates are the assembly of plant complex I [135] or photosystem II [136, 137].

We used dendrograms as a representation of assembly pathways. The visualizations are easy to read, but could be further improved upon. Biologists usually visualize assembly pathways as a flow chart with cartoons of the structures, for example, in Guerrero-Castillo et al. [84]. In dendrograms, the leaves are nicely visible, but subcomplexes are more interesting as they correspond to the steps of the assembly pathway. In flow charts with cartoons of the structures, the subcomplexes can be spotted more nicely, because the cartoons of the structures become bigger towards the final product. An automatic visualization of dendrograms as flow charts with cartoons of the structure might be beneficial.

PTGLgraphComputation could be extended to treat more classes of molecules. For example, lipids [138] and water [139, 140] play a role as contact partners for rCI. Zunker [129] has implemented the treatment of RNA and has analyzed RNA-protein complexes. Next, DNA could be included, too. Because of cryo-EM, an increasing number of structures contains carbohydrates, which could be included, as well [141].

We showed how CGs can be analyzed and used to gain insights on protein complexes. The field of graph theory is wide and more approaches could be applied. For example, graph properties, metrics and centralities could be investigated in the context of CGs. We think that this would provide a deeper understanding of the topology of protein complexes. This may lead to a classification like CATH and SCOP for SSEs or extend the work of 3D Complex.

We showed that graphs of topology are a useful abstraction of protein structures. We expect that biologists greatly benefit from graph-theoretic methods and applications that can visualize, analyze and guide experiments. Much work is still needed to place these methods and applications on validated foundation and provide reliable and user-friendly tools.

Appendix

A.1 Data set for runtime comparison

Table A1: Data set for runtime comparison in four columns. For each structure, the PDB ID, number of atoms (#a) and number of chains (#c) is given.

PDB ID	#a	#c	PDB ID	#a	#c	PDB ID	#a	#c	PDB ID	#a	#c
3slo	2,469	1	3cmv	71,769	8	5jxt	40,421	23	5luf	74,555	62
2l5y	2,470	1	5xlo	15,421	9	5lcw	72,097	23	5tcr	128,658	63
5uad	2,470	1	4gyv	15,742	9	4gxu	38,436	24	4v9g	38,108	64
5w1r	25,559	1	5ue6	43,091	9	1hto	97,896	24	5gjr	142,753	64
5ojs	28,407	1	2pff	71,870	9	2wss	54,974	25	4wz7	35,169	67
5a22	32,188	1	3sja	15,883	10	3iyl	81,010	25	3jc1	92,106	68
5dg5	4,719	2	5j5h	17,290	10	2jes	39,286	26	4v7o	158,904	68
1tm0	4,720	2	3qlv	59,290	10	1o1f	76,897	26	4v8k	50,862	72
3hf1	4,720	2	2vyc	62,912	10	2x53	59,769	27	5b5n	51,893	72
3vkh	45,976	2	5o5k	18,930	11	5gai	98,069	27	5ler	36,539	75
5nug	46,234	2	5nzr	18,980	11	3h6i	48,568	28	5lfb	36,600	75
4w8f	84,824	2	3jc7	40,309	11	5da8	99,580	28	5j8k	55,850	76
2gvz	5,754	3	3jbl	80,134	11	5xtc	38,864	29	4v96	118,740	78
3g82	5,758	3	4wxy	22,179	12	1o1c	85,947	29	3jc9	78,216	79
1r5k	5,759	3	4mhh	22,796	12	4wjg	47,822	30	5leg	41,355	80
4oj6	35,718	3	5u6y	87,208	12	1o18	92,745	30	5gup	109,982	80
4oj5	36,274	3	2vdc	89,640	12	4wl1	57,535	32	5gpn	75,545	84
5vlj	49,791	3	1vf7	23,667	13	3oaa	99,605	32	4yuu	92,765	84
1lbi	8,875	4	5uz9	24,377	13	5afu	55,985	33	5j4z	64,743	89
3l49	8,876	4	4ayb	54,903	13	4cr3	80,171	33	5j7y	64,743	89
3f6x	8,876	4	5vsw	57,656	13	4y8g	50,035	34	4v7g	102,534	90
5j8v	73,619	4	5vj6	26,472	14	5mpa	67,917	34	4wiz	203,250	90
4uwe	81,603	4	3gzt	26,550	14	5lzp	46,591	35	4nwr	88,858	96
4uwa	81,643	4	5jzw	83,432	14	5a5b	83,782	35	5iv7	312,210	96
4lw5	9,416	5	5jzh	90,803	14	3f9k	46,776	36	6ek5	165,959	110
2y7y	9,467	5	2wzp	27,662	15	3whe	88,188	36	5cod	48,030	114
4afg	9,498	5	3rhw	29,212	15	3wu2	54,074	38	3jc8	107,640	115
3j6q	42,580	5	5wq9	56,354	15	5a9q	95,921	38	4v46	79,720	120
4au6	43,709	5	5wq8	56,474	15	5ws5	52,584	39	5xti	196,753	138
5lki	55,419	5	1kiu	29,673	16	5nw4	66,335	39	4v6b	187,090	144
4tle	11,463	6	3ab4	30,986	16	5mx2	50,447	40	5iv5	549,576	145
4tlb	12,061	6	1gq2	71,535	16	4ro0	68,787	40	4v98	121,990	160
5t0d	12,076	6	5cx1	74,604	16	3von	51,491	42	6ekc	321,799	160
1sfy	12,079	6	2o01	29,863	17	4u0g	81,782	42	5mq3	143,640	180
5g4f	69,215	6	5fj9	38,697	17	1yce	28,145	44	4ctg	311,940	180
2vkz	85,965	6	5xtb	27,980	18	5lc5	51,763	45	5vlz	96,897	181
2uv8	85,968	6	3zif	94,394	18	5xtd	66,834	45	4ctf	321,060	240
3hmj	88,836	6	5fl7	30,138	19	3j6d	65,999	48	4udf	326,520	240
5msk	11,552	7	5vhh	55,299	19	3j9q	99,695	48	5v74	215,283	270
5jzc	11,941	7	2a06	33,910	20	5mdx	78,374	50	5mq7	286,920	360
5vy9	38,744	7	3kic	83,540	20	5xnm	92,896	54	5y6p	1,234,811	862
3ala	39,110	7	5a1y	39,977	21	3hf9	90,499	56	3j3y	2,116,800	1,176
5ybb	15,465	8	5g04	65,502	21	3unb	99,292	56	3j3q	2,440,800	1,356
4zxa	16,282	8	4qrm	22,798	22	5mpp	70,619	60			
5c1b	69,393	8	4qiw	51,091	22	4y5z	86,982	60			

A.2 Complete statistics of skipped checks of contacts

Table A2: Number of skipped checks for contacts of residues. Structures with an asterisk (*) contain a ligand. Three methods are compared: neighbor skipping treating all molecules as one chain (previous implementation), neighbor skipping treating chains and ligands individually (improved neighbor skipping) and chain sphere-check together with improved neighbor skipping (combined method). For each method, there is the total number of skipped contact checks. For improved neighbor skipping, there is the number of skips within a chain (intra) and between different chains or ligands (inter). For the combined method, the number of skipped intra contacts is omitted, because it is the same as for improved neighbor skipping. A left arrow indicates that the numbers are the same as for improved neighbor skipping. For each column showing skipped checks of contacts, the absolute number is given and the percentage of the number of pairs of residues in brackets.

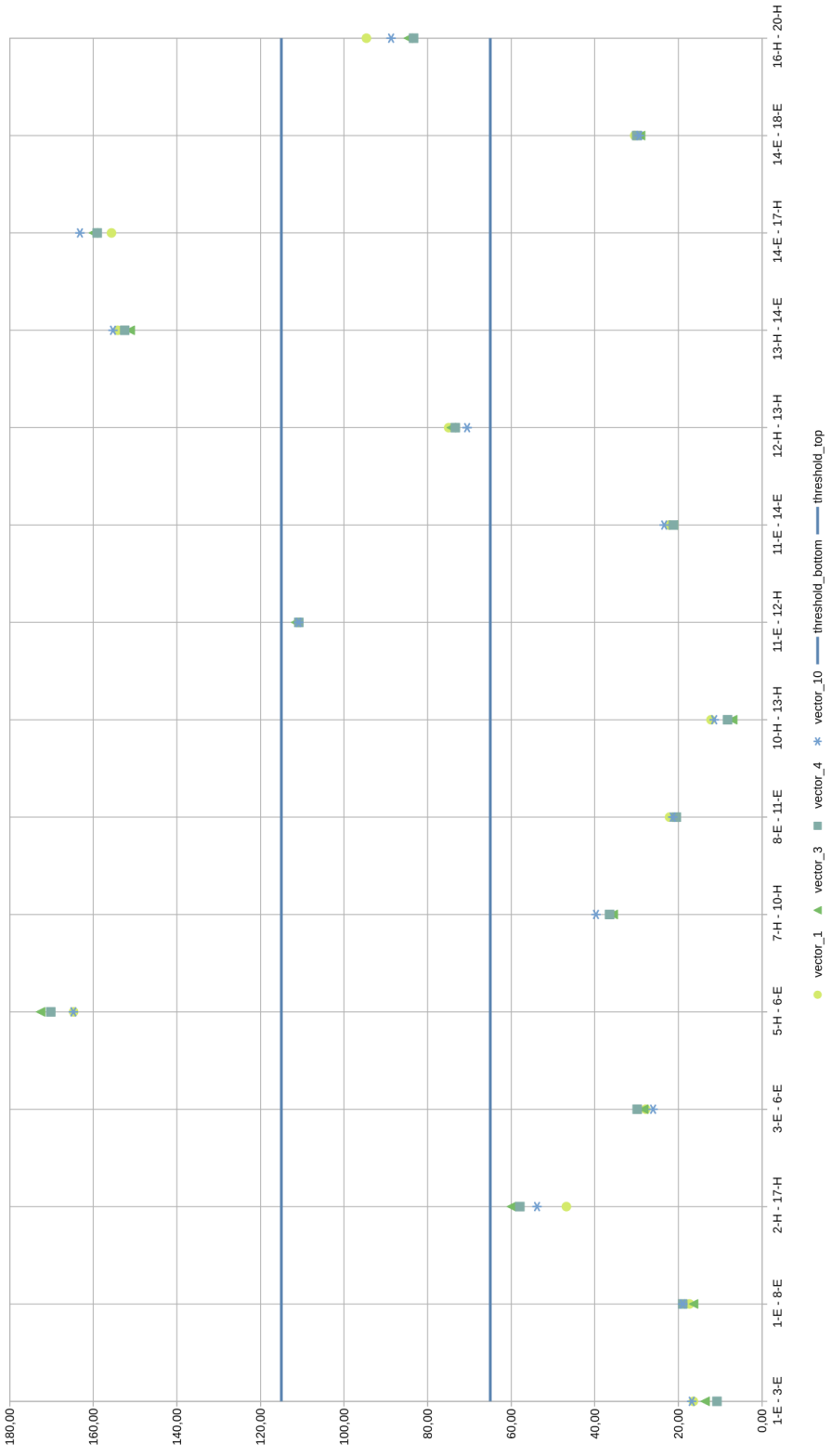
	Number of ...				Previous implementation	Improved neighbor skipping			Combined method		
PDB ID	Chains	Residues	Atoms	Pairs of residues	No. skipped total (%)	No. skipped intra (%)	No. skipped inter (%)	No. skipped total (%)	No. skipped inter (%)	No. skipped chain sphere (%)	No. skipped total (%)
2h8o	1	283	2,468	39,903	12,000 (30.07)	11,763 (29.48)	0 (0)	11,763 (29.48)	←	0 (0)	←
3slo*		313	2,469	48,828	1,466 (3)	24,493 (50.16)	0 (0)	24,493 (50.16)	←	0 (0)	←
5a22*		2,004	32,188	2,007,006	219,038 (10.91)	216,435 (10.78)	0 (0)	216,435 (10.78)	←	0 (0)	←
5ojs		3,473	28,407	6,029,128	1,854,176 (30.75)	1,845,586 (30.61)	0 (0)	1,845,586 (30.61)	←	0 (0)	←
2adv	3	683	5,749	232,903	67,529 (28.99)	78,760 (33.82)	64,053 (27.5)	142,813 (61.32)	←	0 (0)	←
2gvz*		714	5,754	254,541	18,647 (7.33)	31,671 (12.44)	113,863 (44.73)	145,534 (57.18)	←	0 (0)	←
4oj5*		2,272	36,274	2,579,856	0 (0)	600,692 (23.28)	1,383,288 (53.62)	1,983,980 (76.9)	←	0 (0)	←
5x5b		3,139	24,543	4,925,091	11,727 (0.24)	904,820 (18.37)	2,157,635 (43.81)	3,062,455 (62.18)	←	0 (0)	←
1pma*	28	5,936	56,321	17,615,080	44,856 (0.25)	259,397 (1.47)	14,638,680 (83.1)	14,898,077 (84.58)	5,515,231 (31.31)	10,116,344 (57.43)	15,890,972 (90.21)
3h6i*		6,159	48,568	18,963,561	1,531,109 (8.07)	199,745 (1.05)	14,792,295 (78)	14,992,040 (79.06)	5,598,659 (29.52)	10,675,980 (56.3)	16,474,384 (86.87)
4r3o		6,243	47,859	19,484,403	3,621,671 (18.59)	300,083 (1.54)	16,189,295 (83.09)	16,489,378 (84.63)	8,656,115 (44.43)	8,289,745 (42.55)	17,245,943 (88.51)
5da8*		14,030	99,580	98,413,435	10,102,472 (10.27)	1,653,716 (1.68)	79,686,576 (80.97)	81,340,292 (82.65)	25,208,755 (25.62)	61,738,655 (62.73)	88,601,136 (90.03)
5im4	40	5,302	40,577	14,052,951	358,341 (2.55)	91,652 (0.65)	12,324,011 (87.7)	12,415,663 (88.35)	2,082,273 (14.82)	10,940,830 (77.85)	13,114,755 (93.32)
5mx2*		5,442	50,447	14,804,961	22,989 (0.16)	491,866 (3.32)	11,940,596 (80.65)	12,432,462 (83.97)	9,172,971 (61.96)	2,987,872 (20.18)	12,652,709 (85.46)
3zlp		6,543	52,731	21,402,153	2,314,267 (10.81)	188,974 (0.88)	18,392,424 (85.94)	18,581,398 (86.82)	3,832,173 (17.91)	15,965,124 (74.6)	19,986,271 (93.38)
4ro0		8,840	68,787	39,068,380	706,779 (1.81)	458,672 (1.17)	33,891,261 (86.75)	34,349,933 (87.92)	8,402,602 (21.51)	27,644,006 (70.76)	36,505,280 (93.44)

A.3 Inspection of use cases for parameters of vector mode

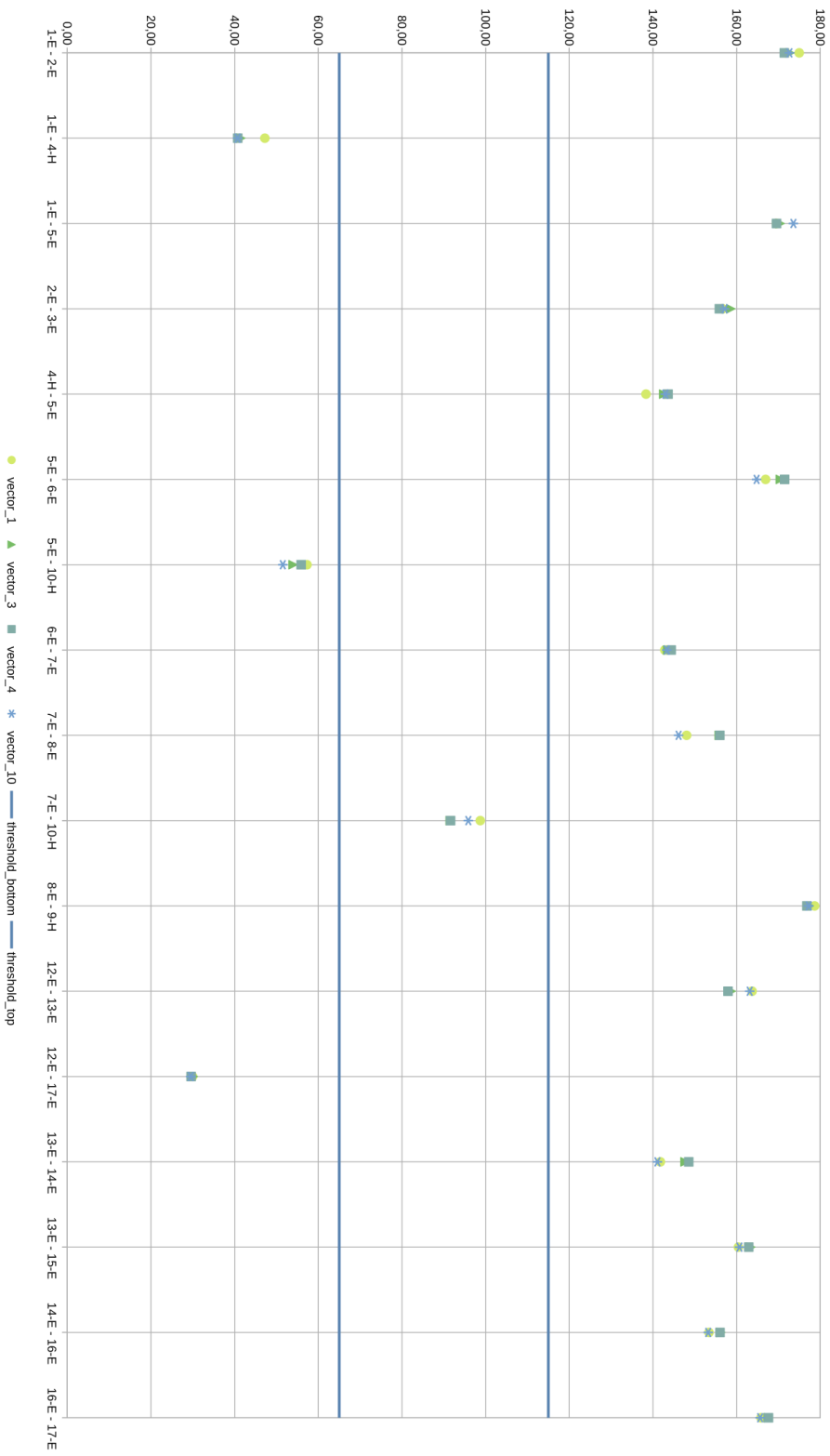
In this subsection, we present the angles of the vector mode (see Section 3.1.2) for all contacts of SSEs of different proteins. The contacts of SSEs may differ from the current version of PTGLgraphComputation, because the results are based on a previous version of PTGLgraphComputation. This does only affect whether or not there is a contact between two SSEs, but not the angles. The results are still valid for considering suitable parameters for the vector method.

We present the angles for the following structures:

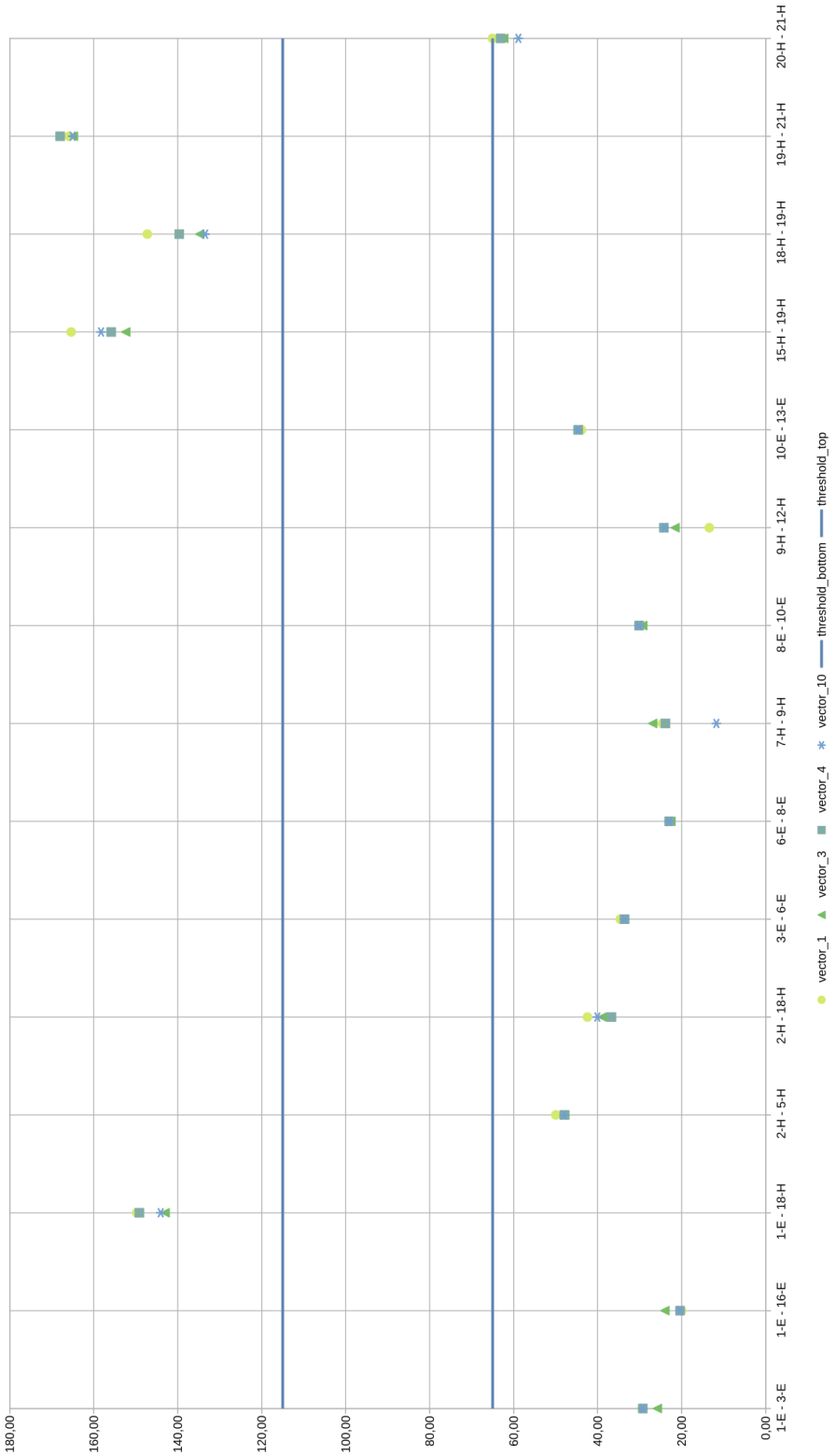
- 1jtv (see Figure A1a)
- 3au1A (see Figure A1b)
- 3denA (see Figure A1c)
- 5p4kA (see Figure A1d)



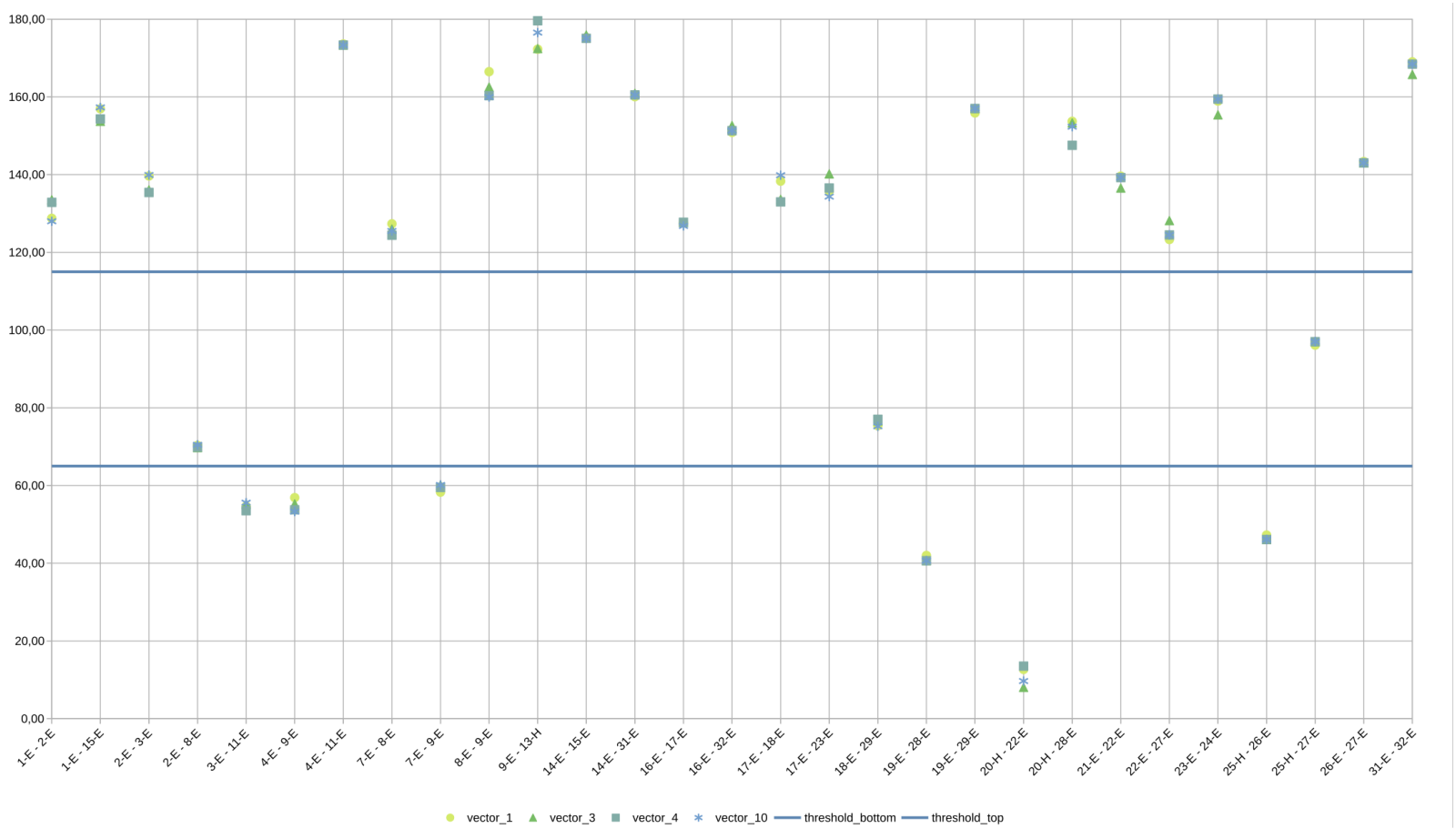
(a) 17β-hydroxysteroid dehydrogenase [2] (PDB: 1jtv)



(b) Antigen-presenting glycoprotein CD1d1 [7] (PDB: 3au1)



(c) Dihydrodipicolinate synthase [8] (PDB: 3den)



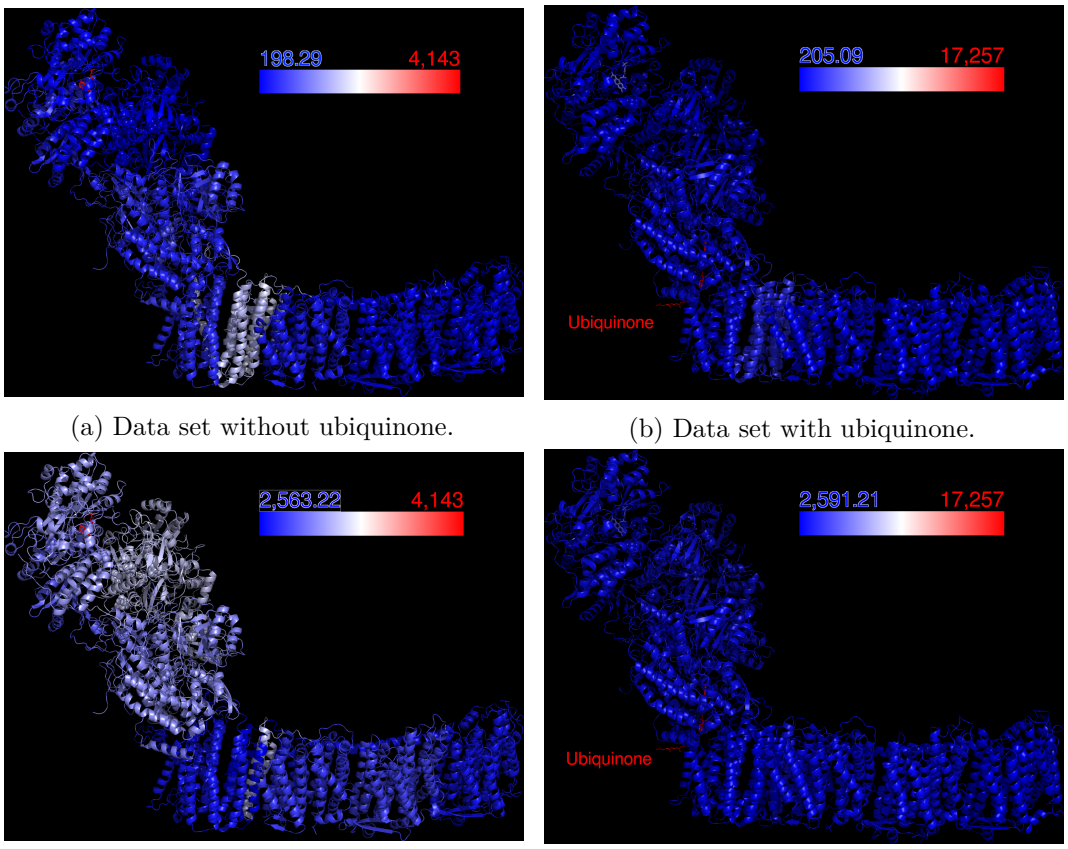
(d) Endothiapepsin [19] (PDB: 5p4k)

Figure A1: Plot of the angles of the vectors of the secondary structure elements (y-axis) of different use cases for all contacting secondary structure elements (x-axis). Angles are shown for values 1, 3, 4 and 10 for c_l (vector 1 to 10). The default thresholds for differentiating between parallel and mixed (threshold_bottom) and mixed and antiparallel (threshold_top) are marked by blue lines.

A.4 Additional output of PTGLdynamics

In this subsection, we present additional heat maps produced with PTGLdynamics (see Section 3.2). We present the following heat maps:

- Chain-level based on residues including ligands (see Figure A2)
- Inter-chain residue-level based on residues excluding ligands (see Figure A3)



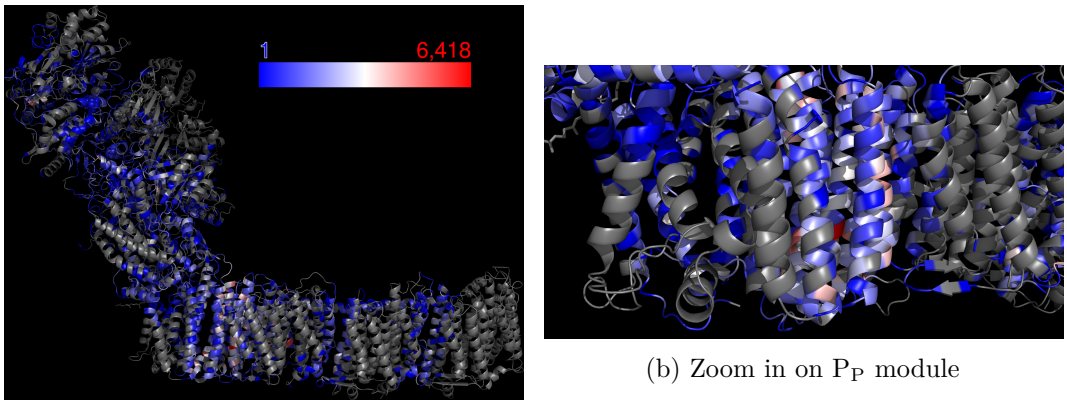
(a) Data set without ubiquinone.

(b) Data set with ubiquinone.

(c) Data set without ubiquinone.

(d) Data set with ubiquinone.

Figure A2: Heat map visualization of changes of contacts of MD simulation of respiratory complex I of *Thermus thermophilus*. Inter-chain (a and b) and inter- and intra-chain changes (c and d) shown on chain-level and based on residues. The number of contacts is divided by the length of the chain. The structure is depicted in cartoon style. The number of changes of contacts is color coded from blue over white to red depicting lowest to highest number of changes. Ubiquinone is labeled in red (b and d).



(a)

(b) Zoom in on P_P module

Figure A3: Heat map visualization of changes of contacts of MD simulation of respiratory complex I of *Thermus thermophilus*. Inter-chain changes shown on residue-level and based on residues. The structure is depicted in cartoon style. The number of changes of contacts is color coded from blue over white to red depicting lowest to highest number of changes. The ligands FMN and ubiquinone are excluded.

A.5 Prediction of modules

Table A3: Modularity of partitions according to modules for human respiratory complex I [20] (PDB: 5xtd). All variants of assigning A12 and S6 to the Q and N module and all edge weight types are presented.

Edge Weight Type	A12 assigned to	S6 assigned to	Modularity
No weight	Q	Q	0.41349
	N	N	0.41220
	Q	N	0.41336
	N	Q	0.40307
Absolute	Q	Q	0.53872
	N	N	0.51915
	Q	N	0.52812
	N	Q	0.51761
Additive	Q	Q	0.54698
	N	N	0.51795
	Q	N	0.53056
	N	Q	0.51504
Multiplicative	Q	Q	0.57026
	N	N	0.53986
	Q	N	0.55096
	N	Q	0.53668

A.6 Prediction of the assembly pathway

A.6.1 Human respiratory complex I (PDB: 5xtd)

Complex Graph

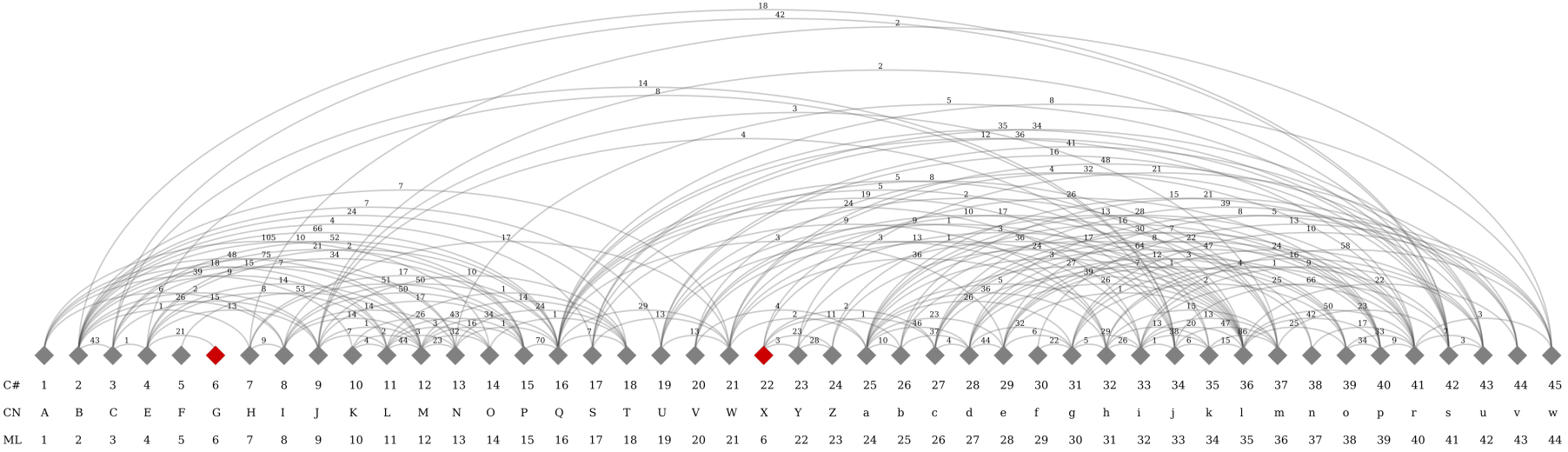


Figure A4: Complex Graph of respiratory complex I of *Homo sapiens* [20] (PDB: 5xtd). Vertices correspond to subunits and edges to spatial neighborhoods. Edges are weighted with the number of residue contacts. Beneath each vertex, the chain number (C#), the chain name (CN), the molecule identifier (ML) and the molecule name (MN) is given. Vertices of the same color are homologous and grey vertices have no homologue.

Additive dendrogram using contacts transitive by ligands

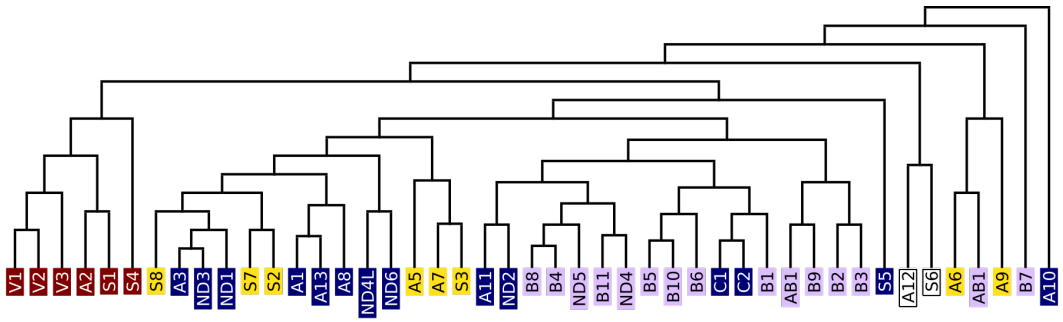


Figure A5: Assembly pathway of respiratory complex I of *Homo sapiens* using additive weights and contacts transitive by ligands visualized as dendrogram. Leaves correspond to subunits, are labeled with the abbreviated gene names and colored according to the modules they are assigned to: N (red), Q (yellow), P_P (dark blue) and P_D (lavender). The subunits that are shared between the N and Q module are colored white. Inner vertices correspond to subassemblies and the root to the final protein complex.

A.6.2 Human respiratory supercomplex $I_1III_2IV_1$ (PDB: 5xth)

Complex Graph

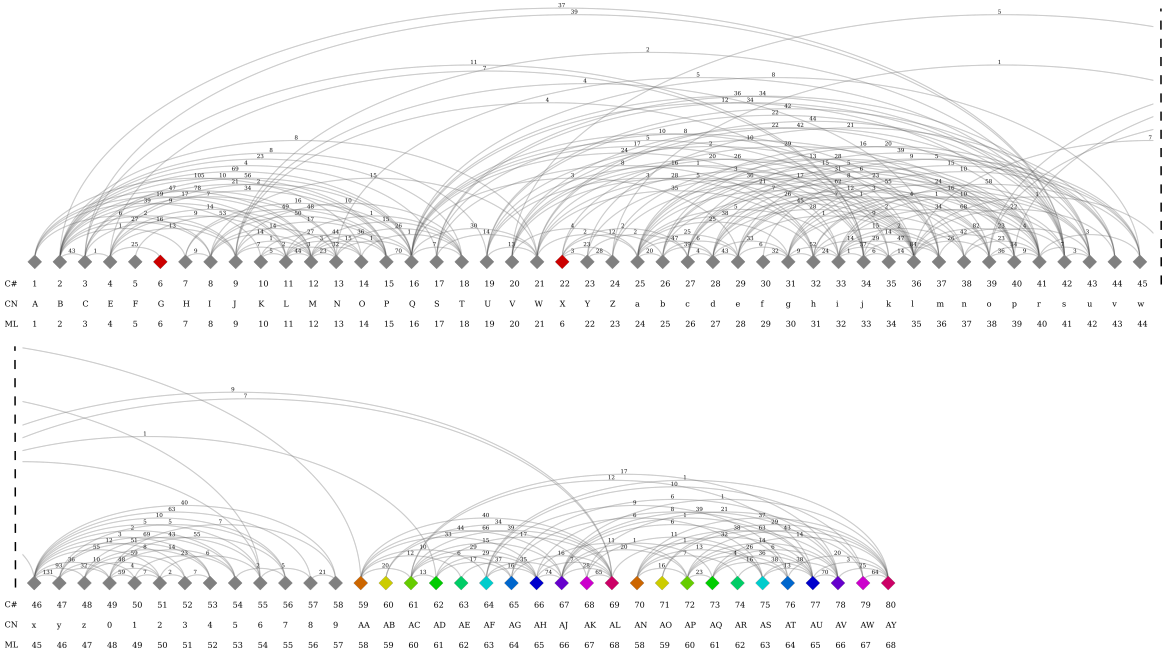
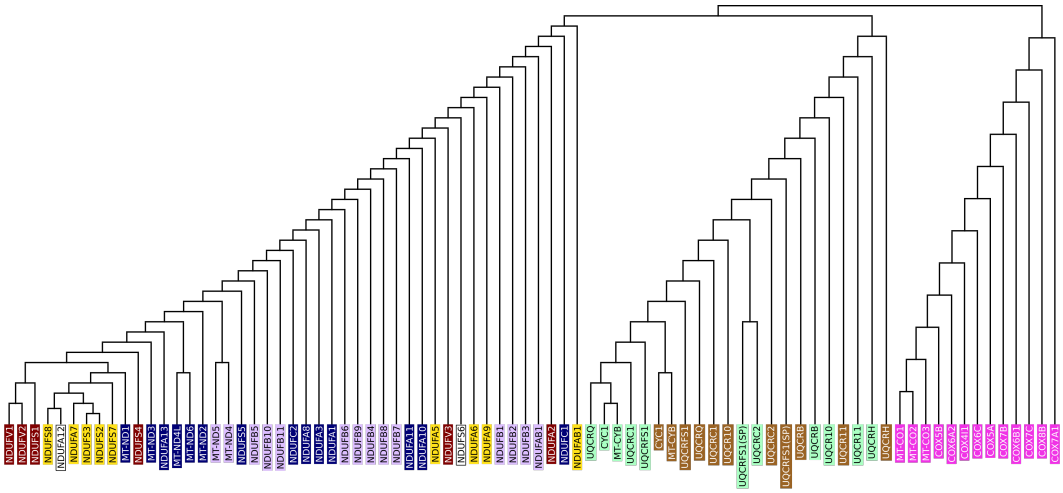
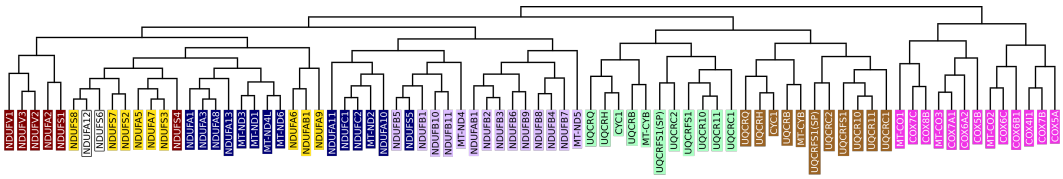


Figure A6: Complex Graph of respiratory supercomplex $I_1III_2IV_1$ of *Homo sapiens* [20] (PDB: 5xth). Vertices correspond to subunits and edges to spatial neighborhoods. Edges are weighted with the number of residue contacts. Beneath each vertex, the chain number (C#), the chain name (CN), the molecule identifier (ML) and the molecule name (MN) is given. Vertices of the same color are homologous and grey vertices have no homologue. Complex Graph is split vertically at dashed line.

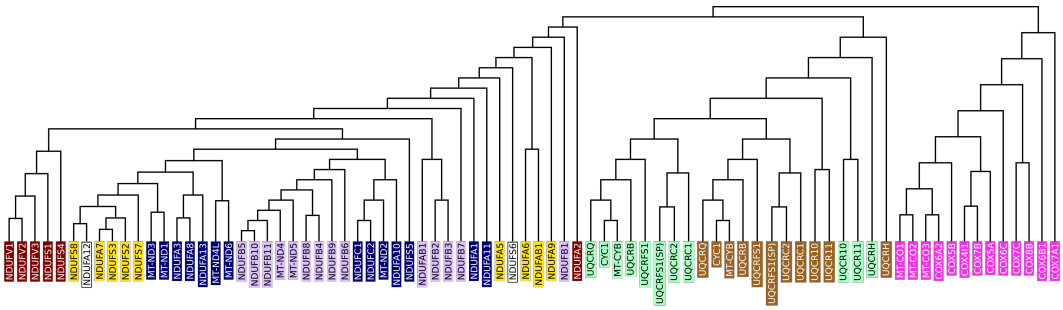
Dendrograms of other edge weight types than additive



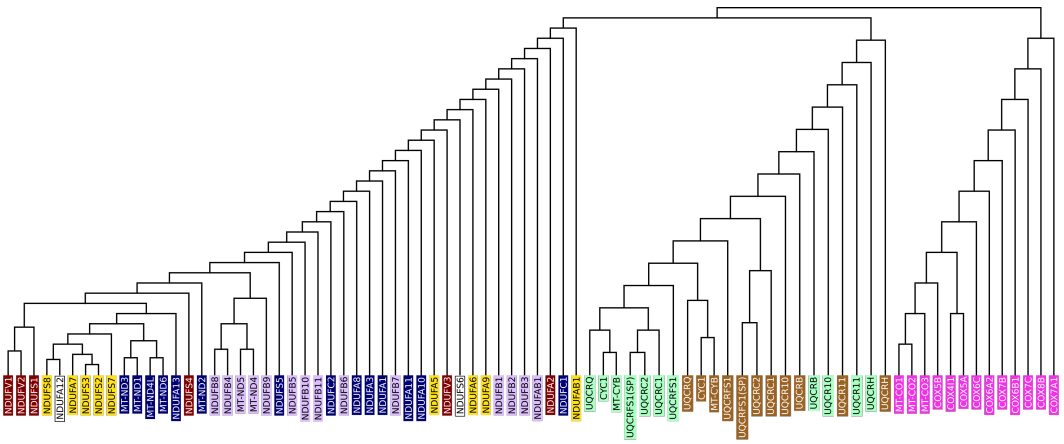
(a) Absolute



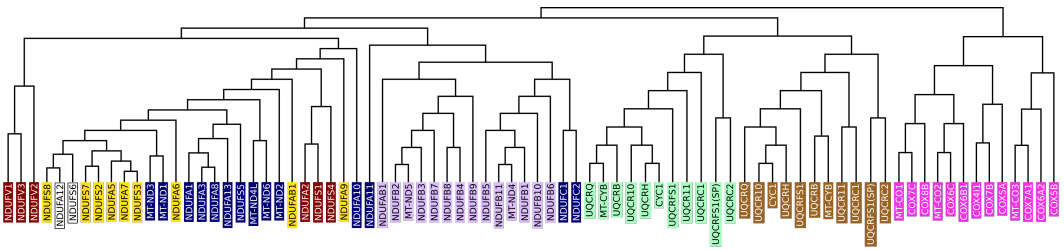
(b) Multiplicative



(c) Square root



(d) Logarithm



(e) Minimum

Figure A7: Predicted assembly pathways using different edge weight types other than additive for respiratory supercomplex of *Homo sapiens* [20] (PDB: 5xth) represented as dendrograms. Leaves correspond to subunits, are labeled with the abbreviated gene names and colored according to the complexes or modules they are assigned to: N (red), Q (yellow), P_P (dark blue), P_D (lavender), III-1 (mint), III-2 (brown) and IV (magenta). The subunits that are shared between the N and Q module are colored white. Inner vertices correspond to subcomplexes and the root to the final protein complex. Selected inner vertices are labeled.

A.6.3 Human respiratory megacomplex $I_2III_2IV_2$ (PDB: 5xti)

Complex Graph

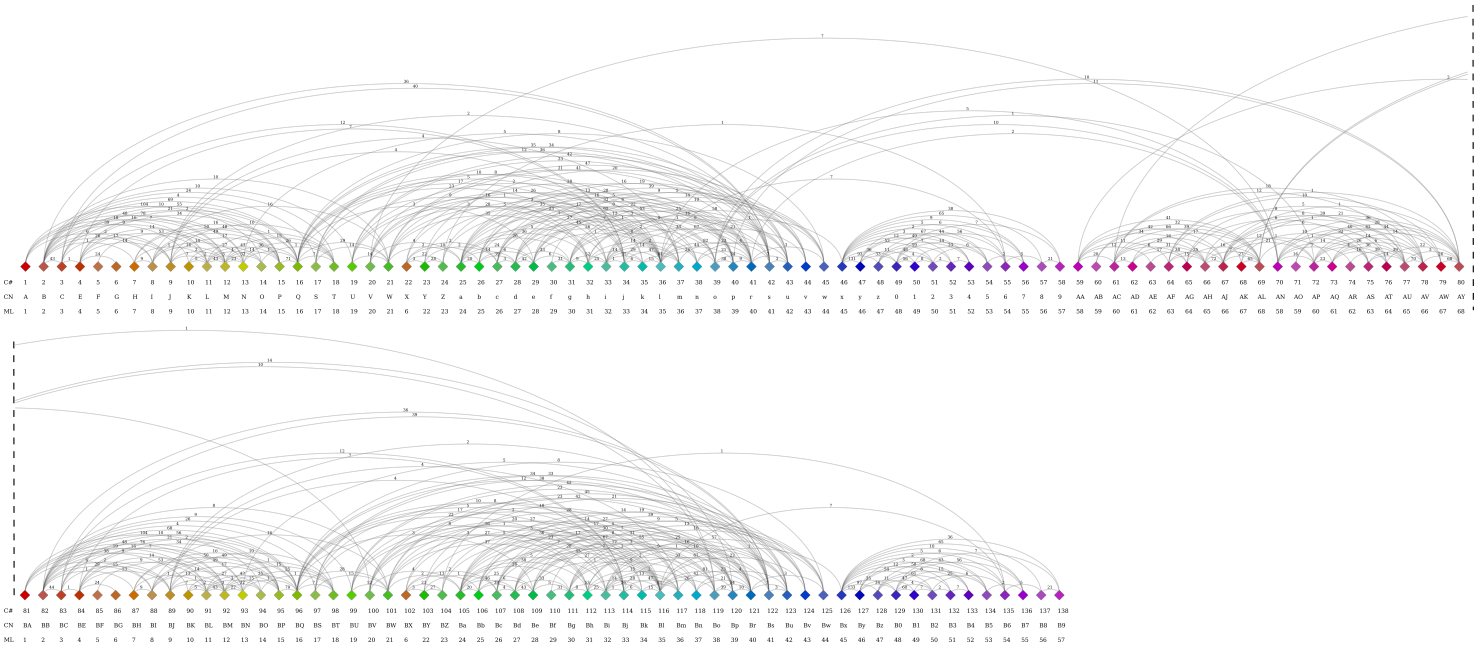
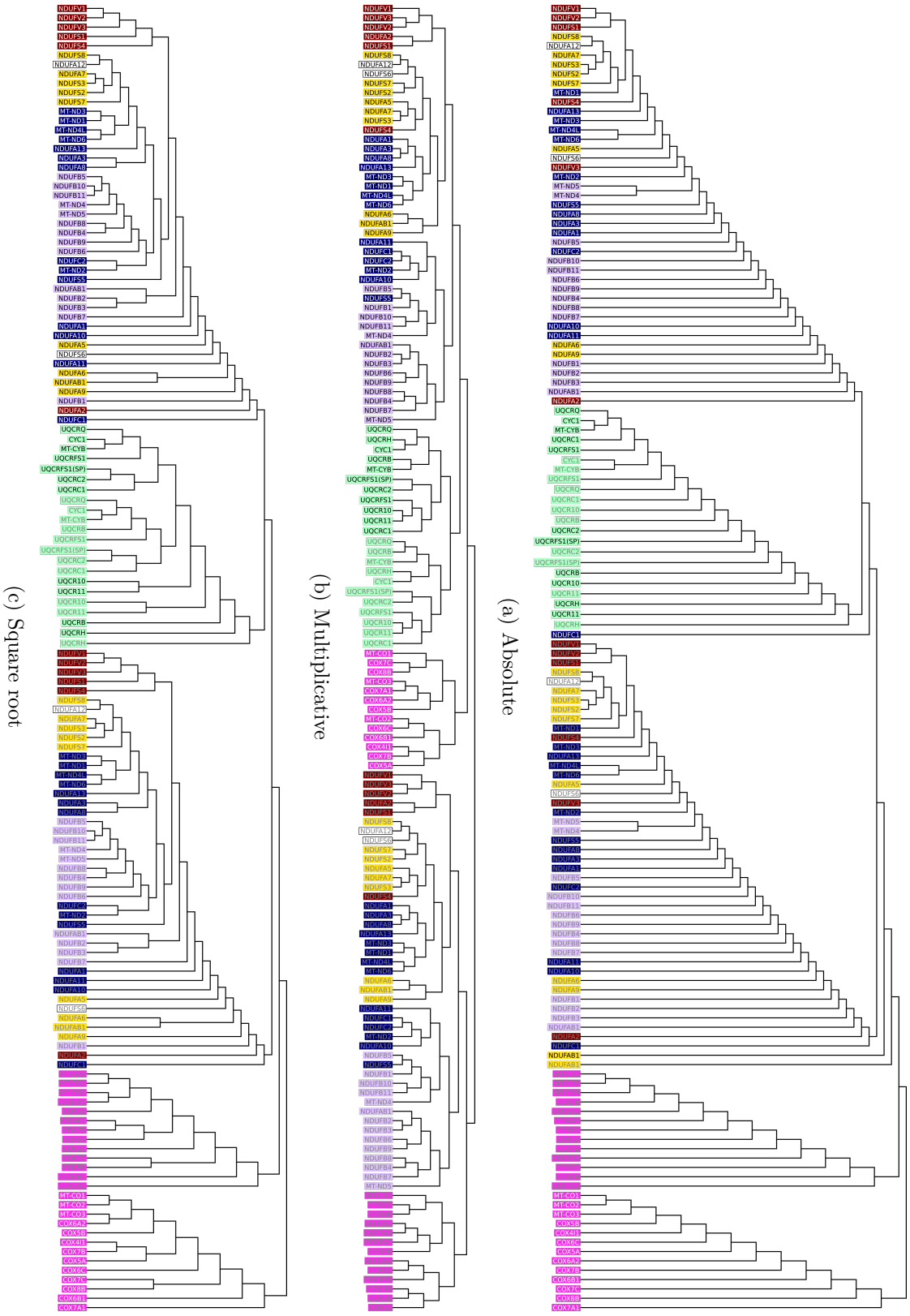
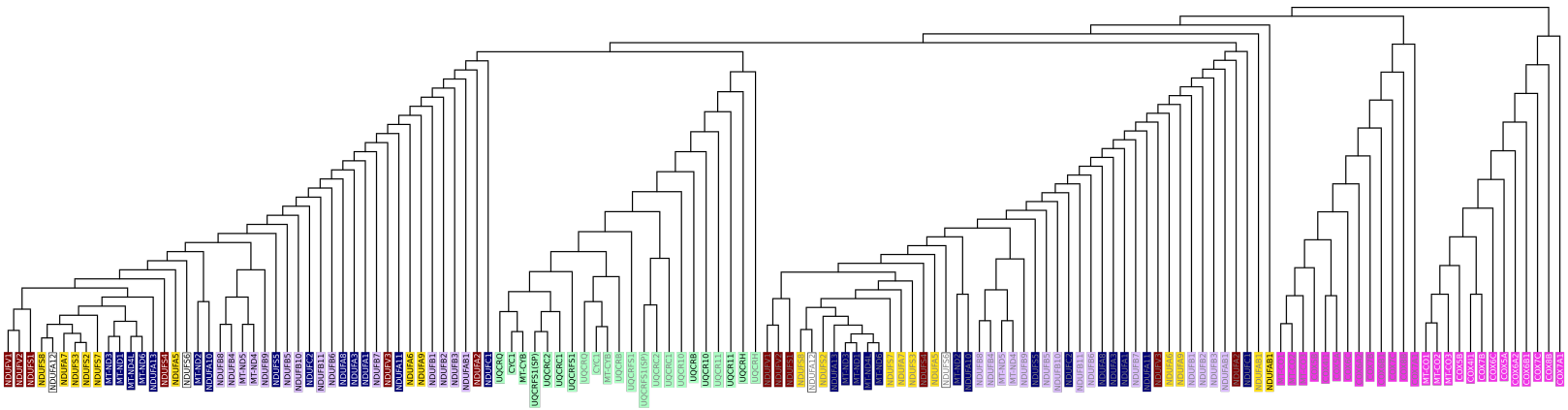


Figure A8: Complex Graph of respiratory megacomplex $I_2III_2IV_2$ of *Homo sapiens* [20] (PDB: 5xti). Vertices correspond to subunits and edges to spatial neighborhoods. Edges are weighted with the number of residue contacts. Beneath each vertex, the chain number (C#), the chain name (CN), the molecule identifier (ML) and the molecule name (MN) is given. Vertices of the same color are homologous and grey vertices have no homologue. Complex Graph is split vertically at dashed line.

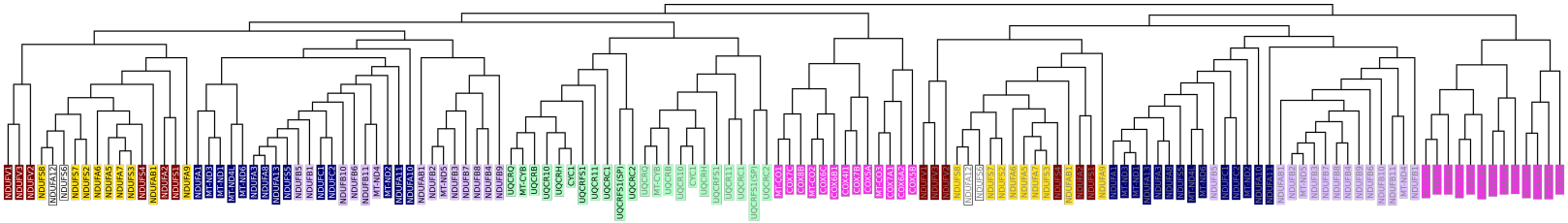
Dendrograms of other edge weight types than additive



(a) Absolute (b) Multiplicative (c) Square root



(d) Logarithm



(e) Minimum

Figure A9: Predicted assembly pathways using different edge weight types other than additive for human respiratory megacomplex [20] (PDB: 5xti) represented as dendrograms. Leaves correspond to subunits, are labeled with the gene names and colored according to the complexes or modules they are assigned to: N (red), Q (yellow), P_P (dark blue), P_D (lavender), III (mint) and IV (magenta). The colors of the labels of the leaves correspond to the number of the copy of the module or complex: black or white (1) and grey (2). The subunits that are shared between the N and Q module are colored white. Inner vertices correspond to subcomplexes and the root to the final protein complex.

Bibliography

[1] T. Huxford, D. Huang, S. Malek, and G. Ghosh. The Crystal Structure of the IκBα/NF-κB Complex Reveals Mechanisms of NF-κB Inactivation. *Cell*, 95(6):759–770, 1998.

[2] A. Gangloff, R. Shi, V. Nahoum, and S. Lin. Pseudo-symmetry of C19-steroids, alternative binding orientations and multispecificity in human estrogenic 17β-hydroxysteroid dehydrogenase. *The FASEB Journal*, 17(2):274–276, 2003.

[3] C. J. O'Neal, E. I. Amaya, M. G. Jobling, R. K. Holmes, and W. G. J. Hol. Crystal Structures of an Intrinsically Active Cholera Toxin Mutant Yield Insight into the Toxin Activation Mechanism. *Biochemistry*, 43(13):3772–3782, 2004.

[4] R. A. W. Frank, C. M. Titman, J. V. Pratap, B. F. Luisi, and R. N. Perham. A Molecular Switch and Proton Wire Synchronize the Active Sites in Thiamine Enzymes. *Science*, 306(5697):872–876, 2004.

[5] K. Kamada, Y. Kubota, T. Arata, Y. Shindo, and F. Hanaoka. Structure of the human GINS complex and its assembly and functional interface in replication initiation. *Nature Structural & Molecular Biology*, 14(5):388–396, 2007.

[6] D. Ghosh, Z. Wawrzak, C. M. Weeks, W. L. Duax, and M. Erman. The refined three-dimensional structure of 3α, 20β-hydroxysteroid dehydrogenase and possible roles of the residues conserved in short-chain dehydrogenases. *Structure*, 2(7):629–640, 1994.

[7] T. Mallevaey, C. A. J., J. P. Scott-Browne, M. H. Young, L. C. Roisman, D. G. Pellicci, O. Patel, J. P. Vivian, J. L. Matsuda, J. McCluskey, D. I. Godfrey, P. Marrack, J. Rossjohn, and L. Gapin. A molecular basis for NKT cell recognition of CD1d-self-antigen. *Immunity*, 34(3):315–326, 2011.

[8] F. G. Pearce, R. C. J. Dobson, A. Weber, L. A. Lane, M. G. McCammon, M. A. Squire, M. A. Perugini, G. B. Jameson, C. V. Robinson, and J. A. Gerrard. Mutating the Tight-Dimer Interface of Dihydrodipicolinate Synthase Disrupts the Enzyme Quaternary Structure: Toward a Monomeric Enzyme. *Biochemistry*, 47(46):12108–12117, 2008.

[9] G. Zhao, J. R. Perilla, E. L. Yufenyuy, X. Meng, B. Chen, J. Ning, J. Ahn, A. M. Gronenborn, K. Schulten, C. Aiken, and P. Zhang. Mature HIV-1 capsid structure by cryo-electron microscopy and all-atom molecular dynamics. *Nature*, 497(7451):643–646, 2013.

[10] T. J. Barnard, J. Gumbart, J. H. Peterson, N. Noinaj, N. C. Easley, N. Dautin, A. J. Kuszak, E. Tajkhorshid, H. D. Bernstein, and S. K. Buchanan. Molecular basis for the activation of a catalytic asparagine residue in a self-cleaving bacterial autotransporter. *Journal of Molecular Biology*, 415(1):128–142, 2012.

[11] P. J. J. Robinson, D. A. Bushnell, M. J. Trnka, A. L. Burlingame, and R. D. Kornberg. Structure of the Mediator Head module bound to the carboxy-terminal domain of RNA polymerase II. *Proceedings of the National Academy of Sciences*, 109(44):17931–17935, 2012.

[12] R. Baradaran, J. M. Berrisford, G. S. Minhas, and L. A. Sazanov. Crystal structure of the entire respiratory complex I. *Nature*, 494(7438):443–448, 2013.

[13] C. Chen, P. Bales, J. Greenfield, R. D. Heselpoth, D. C. Nelson, and O. Herzberg. Crystal structure of orf210 from e. coli o157: H1 phage cba120 (tsp1), a putative tailspike protein. *PloS One*, 9(3):e93156, 2014.

[14] B. Liang, Z. Li, S. Jenni, A. A. Rahmeh, B. M. Morin, T. Grant, N. Grigorieff, S. C. Harrison, and S. P. J. Whelan. Structure of the l protein of vesicular stomatitis virus from electron cryomicroscopy. *Cell*, 162(2):314–327, 2015.

[15] J. B. Bale, S. Gonen, Y. Liu, W. Sheffler, D. Ellis, C. Thomas, D. Cascio, T. O. Yeates, T. Gonen, N. P. King, and D. Baker. Accurate design of megadalton-scale two-component icosahedral protein complexes. *Science*, 353(6297):389–394, 2016.

[16] M. Zhang, M. Bommer, R. Chatterjee, R. Hussein, J. Yano, H. Dau, J. Kern, H. Dobbek, and A. Zouni. Structural insights into the light-driven auto-assembly process of the water-oxidizing mn4cao5-cluster in photosystem ii. *Elife*, 6:e26933, 2017.

[17] K. Zhang, H. E. Foster, A. Rondelet, S. E. Lacey, N. Bahi-Buisson, A. W. Bird, and A. P. Carter. Cryo-EM Reveals How Human Cytoplasmic Dynein Is Auto-inhibited and Activated. *Cell*, 169(7):1303–1314, 2017.

[18] L. M. Diaz-Santin, N. Lukoyanova, E. Aciyan, and A. C. M. Cheung. Cryo-EM structure of the SAGA and NuA4 coactivator subunit Tra1 at 3.7 angstrom resolution. *Elife*, 6:e28384, 2017.

[19] J. Schiebel, S. Krimmer, K. Röwer, A. Knörlein, X. Wang, A. Park, M. Stieler, F. Ehrmann, K. Fu, N. Radeva, M. Krug, F. Huschmann, S. Glöckner, M. Weiss, U. Mueller, G. Klebe, and A. Heine. High-Throughput Crystallography: Reliable and Efficient Identification of Fragment Hits. *Structure*, 24(8):1398–1409, 2016.

[20] R. Guo, S. Zong, M. Wu, J. Gu, and M. Yang. Architecture of Human Mitochondrial Respiratory Megacomplex $I_2III_2IV_2$. *Cell*, 170(6):1247–1257, 2017.

[21] J. Gutiérrez-Fernández, K. Kaszuba, G. S. Minhas, R. Baradaran, M. Tambalo, D. T. Gallagher, and L. A. Sazanov. Key role of quinone in the mechanism of respiratory complex I. *Nature Communications*, 11(1):1–17, 2020.

[22] R. C. Davenport, P. A. Bash, B. A. Seaton, M. Karplus, G. A. Petsko, and D. Ringe. Structure of the triosephosphate isomerase-phosphoglycolohydroxamate complex: an analog of the intermediate on the reaction pathway. *Biochemistry*, 30(24):5821–5826, 1991.

[23] G. J. Mulder. Sur la composition de quelques substances animales. *Bulletin des Sciences Physiques et Naturelles en Neerlande*, pages 104–119, 1838.

[24] H. B. Vickery and C. L. A. Schmidt. The history of the discovery of the amino acids. *Chemical Reviews*, 9(2):169–318, 1931.

[25] C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

[26] P. Tompa. Intrinsically disordered proteins: a 10-year recap. *Trends in Biochemical Sciences*, 37(12):509–516, 2012.

[27] S. J. Singer and G. L. Nicolson. The fluid mosaic model of the structure of cell membranes: Cell membranes are viewed as two-dimensional solutions of oriented globular proteins and lipids. *Science*, 175(4023):720–731, 1972.

[28] R. J. Ellis. Protein Misassembly. *Molecular Aspects of the Stress Response: Chaperones, Membranes and Networks*, 594:1–13, 2007.

[29] F. K. Friedman and S. Beychok. Probes of Subunit Assembly and Reconstitution Pathways in Multisubunit Proteins. *Annual Review of Biochemistry*, 48(1):217–250, 1979.

[30] E. Antonini and E. Chiancone. Assembly of Multisubunit Respiratory Proteins. *Annual Review of Biophysics and Bioengineering*, 6(1):239–271, 1977.

[31] I. Vercellino and L. A. Sazanov. The assembly, regulation and function of the mitochondrial respiratory chain. *Nature Reviews Molecular Cell Biology*, 23(2):141–161, 2022.

[32] F. Distelmaier, W. J. H. Koopman, L. P. van den Heuvel, R. J. Rodenburg, E. Mayatepek, P. H. G. M. Willems, and J. A. M. Smeitink. Mitochondrial complex I deficiency: from organelle dysfunction to clinical disease. *Brain*, 132(4):833–842, 2009.

[33] B. Chance, W. D. Bonner Jr, and B. T. Storey. Electron transport in respiration. *Annual Review of Plant Physiology*, 19(1):295–320, 1968.

[34] C. I. Branden and J. Tooze. *Introduction to Protein Structure*. Garland Publishing, Inc., 2nd edition, 1998.

[35] I. Sillitoe, N. Bordin, N. Dawson, V. P. Waman, P. Ashford, H. M. Scholes, C. S. M. Pang, L. Woodridge, C. Rauer, N. Sen, M. Abbasian, S. Le Cornu, S. D. Lam, K. Berka, I. H. Varekova, R. Svobodova, J. Lees, and C. A. Orengo. CATH: increased structural coverage of functional space. *Nucleic Acids Research*, 49(D1):D266–D273, 2021.

[36] A. Andreeva, D. Howorth, C. Chothia, E. Kulesha, and A. G. Murzin. SCOP2 prototype: a new approach to protein structure mining. *Nucleic Acids Research*, 42(D1):D310–D314, 2014.

[37] A. Andreeva, E. Kulesha, J. Gough, and A. G. Murzin. The SCOP database in 2020: expanded classification of representative family and superfamily domains of known protein structures. *Nucleic Acids Research*, 48(D1):D376–D382, 2020.

[38] N. K. Fox, S. E. Brenner, and J. Chandonia. SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1):D304–D309, 2014.

[39] J. Chandonia, L. Guan, S. Lin, C. Yu, N. K. Fox, and S. E. Brenner. SCOPe: improvements to the structural classification of proteins–extended database to facilitate variant interpretation and machine learning. *Nucleic Acids Research*, 50(D1):D553–D559, 2022.

[40] I. Michalopoulos, G. M. Torrance, D. R. Gilbert, and D. R. Westhead. TOPS: an enhanced database of protein structural topology. *Nucleic Acids Research*, 32(suppl_1):D251–D254, 2004.

[41] S. Shi, B. Chitturi, and N. V. Grishin. ProSMoS server: a pattern-based search using interaction matrix representation of protein structures. *Nucleic Acids Research*, 37(suppl_2):W526–W531, 2009.

[42] A. Stivala, M. Wybrow, A. Wirth, J. C. Whisstock, and P. J. Stuckey. Automatic generation of protein structure cartoons with Pro-origami. *Bioinformatics*, 27(23):3315–3316, 2011.

[43] G. Faure, J. Andreani, and R. Guerois. InterEvol database: exploring the structure and evolution of protein complex interfaces. *Nucleic Acids Research*, 40(D1):D847–D856, 2012.

[44] M. Giurgiu, J. Reinhard, B. Brauner, I. Dunger-Kaltenbach, G. Fobo, G. Frishman, C. Montrone, and A. Ruepp. CORUM: the comprehensive resource of mammalian protein complexes—2019. *Nucleic Acids Research*, 47(D1):D559–D563, 2019.

[45] S. Orchard, M. Ammari, B. Aranda, L. Breuza, L. Briganti, F. Broackes-Carter, N. H. Campbell, G. Chavali, C. Chen, N. Del-Toro, M. Duesbury, M. Dumousseau, E. Galeota, U. Hinz, M. Iannuccelli, S. Jagannathan, R. Jimenez, J. Khadake, A. Lagreid, L. Licata, R. C. Lovering, B. Meldal, A. N. Melidoni, M. Milagros, D. Peluso, L. Perfetto, P. Porras, A. Raghunath, S. Ricard-Blum, B. Roechert, A. Stutz, M. Tognolli, K. van Roey, G. Cesareni, and H. Hermjakob. The MIntAct project—IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Research*, 42(D1):D358–D363, 2014.

[46] B. H. M. Meldal, H. Bye-A-Jee, L. Gajdoš, Z. Hammerová, A. Horáčková, F. Melicher, L. Perfetto, D. Pokornỳ, M. R. Lopez, A. Türková, E. D. Wong, Z. Xie, E. B. Casanova, N. del Toro, M. Koch, P. Porras, H. Hermjakob, and S. Orchard. Complex Portal 2018: extended content and enhanced visualization tools for macromolecular complexes. *Nucleic Acids Research*, 47(D1):D550–D558, 2019.

[47] S. Kikugawa, K. Nishikata, K. Murakami, Y. Sato, M. Suzuki, M. Altaf-Ul-Amin, S. Kanaya, and T. Imanishi. PCDq: human protein complex database with quality index which summarizes different levels of evidences of protein complexes predicted from H-invitational protein-protein interactions integrative dataset. In *BMC Systems Biology*, volume 6, pages 1–14. BioMed Central, 2012.

[48] D. Szklarczyk, A. L. Gable, K. C. Nastou, D. Lyon, R. Kirsch, S. Pyysalo, N. T. Doncheva, M. Legeay, T. Fang, P. Bork, L. J. Jensen, and C. von Mering. The STRING database in 2021: customizable protein–protein networks, and

functional characterization of user-uploaded gene/measurement sets. *Nucleic Acids Research*, 49(D1):D605–D612, 2021.

[49] Y. Lo, C. Lin, and J. Yang. PCFamily: a web server for searching homologous protein complexes. *Nucleic Acids Research*, 38(suppl_2):W516–W522, 2010.

[50] E. D. Levy, J. B. Pereira-Leal, C. Chothia, and S. A. Teichmann. 3D Complex: A Structural Classification of Protein Complexes. *PLoS Computational Biology*, 2(11):e155, 2006.

[51] W. Humphrey, A. Dalke, and K. Schulten. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14:33–38, 1996.

[52] R. Veevers and S. Hayward. Methodological improvements for the analysis of domain movements in large biomolecular complexes. *Biophysics and Physicobiology*, 16:328–336, 2019.

[53] S. Bougueroua, R. Spezia, S. Pezzotti, S. Vial, F. Quessette, D. Barth, and M. . P. Gaigeot. Graph theory for automatic structural recognition in molecular dynamics simulations. *The Journal of Chemical Physics*, 149(18):184102, 2018.

[54] N. Zaki, D. Efimov, and J. Berengueres. Protein complex detection using interaction reliability assessment and weighted clustering coefficient. *BMC Bioinformatics*, 14(1):1–9, 2013.

[55] J. Wang, J. Zhong, G. Chen, M. Li, F. Wu, and Y. Pan. ClusterViz: A Cytoscape APP for Cluster Analysis of Biological Network. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(4):815–822, 2014.

[56] T. Nepusz, H. Yu, and A. Paccanaro. Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods*, 9(5):471–472, 2012.

[57] G. D. Bader and C. W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1):1–27, 2003.

[58] Y. Qi, F. Balem, C. Faloutsos, J. Klein-Seetharaman, and Z. Bar-Joseph. Protein complex identification by supervised graph local clustering. *Bioinformatics*, 24(13):i250–i268, 2008.

[59] E. D. Levy, E. B. Erba, C. V. Robinson, and S. A. Teichmann. Assembly reflects evolution of protein complexes. *Nature*, 453(7199):1262–1265, 2008.

[60] J. A. Marsh, H. Hernández, Z. Hall, S. E. Ahnert, T. Perica, C. V. Robinson, and S. A. Teichmann. Protein Complexes Are under Evolutionary Selection to Assemble via Ordered Pathways. *Cell*, 153(2):461–470, 2013.

[61] L. X. Peterson, Y. Togawa, J. Esquivel-Rodriguez, G. Terashi, C. Christoffer, A. Roy, W. Shin, and D. Kihara. Modeling the assembly order of multimeric heteroprotein complexes. *PLoS Computational Biology*, 14(1):e1005937, 2018.

[62] M. E. J. Newman. *Networks*. Oxford University Press, 2010.

[63] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[64] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[65] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[66] V. A. Traag, L. Waltman, and N. J. Van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):1–12, 2019.

[67] F. Lapointe and P. Legendre. The Generation of Random Ultrametric Matrices Representing Dendrograms. *Journal of Classification*, 8(2):177–200, 1991.

[68] J. Felsenstein. The Number of Evolutionary Trees. *Systematic Zoology*, 27(1):27–33, 1978.

[69] J. B. Phipps. Dendrogram topology: nomenclature. *Canadian Journal of Botany*, 53(18):2047–2049, 1975.

[70] D. F. Robinson and L. R. Foulds. Comparison of Phylogenetic Trees. *Mathematical Biosciences*, 53(1-2):131–147, 1981.

[71] M. R. Smith. Information theoretic Generalized Robinson–Foulds metrics for comparing phylogenetic trees. *Bioinformatics*, 36(20):5007–5013, 2020.

[72] G. W. Furnas. The Generation of Random, Binary Unordered Trees. *Journal of Classification*, 1(1):187–233, 1984.

[73] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: an overview, II. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(6):e1219, 2017.

[74] J. Arora, K. Khatter, and M. Tushir. Fuzzy c-Means Clustering Strategies: A Review of Distance Measures. In M. N. Hoda, N. Chauhan, S. M. K. Quadri, and P. R. Srivastava, editors, *Software Engineering*, pages 153–162. Springer Singapore, 2019.

[75] K. Miyaguchi. Direct imaging electron microscopy (EM) methods in modern structural biology: Overview and comparison with X-ray crystallography and single-particle cryo-EM reconstruction in the studies of large macromolecules. *Biology of the Cell*, 106(10):323–345, 2014.

[76] M. A. González. Force fields and molecular dynamics simulations. *École thématique de la Société Française de la Neutronique*, 12:169–200, 2011.

[77] B. R. Brooks, C. L. Brooks III, A. D. Mackerell Jr, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. CHARMM: The biomolecular simulation program. *Journal of Computational Chemistry*, 30(10):1545–1614, 2009.

[78] J. C. Phillips, D. J. Hardy, J. D. C. Maia, J. E. Stone, J. V. Ribeiro, R. C. Bernardi, R. Buch, G. Fiorin, J. Hénin, W. Jiang, R. McGreevy, M. C. R. Melo, B. K. Radak, R. D. Skeel, A. Singharoy, Y. Wang, B. Roux, A. Aksimentiev, Z. Luthey-Schulten, L. V. Kalé, K. Schulten, C. Chipot, and E. Tajkhorshid. Scalable molecular dynamics on CPU and GPU architectures with NAMD. *The Journal of Chemical Physics*, 153(4):044130, 2020.

[79] S. Páll, A. Zhmurov, P. Bauer, M. Abraham, M. Lundborg, A. Gray, B. Hess, and E. Lindahl. Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS. *The Journal of Chemical Physics*, 153(13):134110, 2020.

[80] C. Levinthal. How to fold graciously. In J. T. P. DeBrunner and E. Munck, editors, *Mossbauer Spectroscopy in Biological Systems: Proceedings of a Meeting Held at Allerton House*, pages 22–24. University of Illinois Press, Urbana, IL, 1969.

[81] K. A. Kennedy, E. G. Gachelet, and B. Traxler. Evidence for Multiple Pathways in the Assembly of the *Escherichia coli* Maltose Transport Complex. *Journal of Biological Chemistry*, 279(32):33290–33297, 2004.

[82] H. Hernandez and C. V. Robinson. Determining the stoichiometry and interactions of macromolecular assemblies from mass spectrometry. *Nature Protocols*, 2(3):715–726, 2007.

[83] H. Heide, L. Bleier, M. Steger, J. Ackermann, S. Dröse, B. Schwamb, M. Zörnig, A. S. Reichert, I. Koch, I. Wittig, and U. Brandt. Complexome profiling identifies tmem126b as a component of the mitochondrial complex i assembly complex. *Cell Metabolism*, 16(4):538–549, 2012.

[84] S. Guerrero-Castillo, F. Baertling, D. Kownatzki, H. J. Wessels, S. Arnold, U. Brandt, and L. Nijtmans. The Assembly Pathway of Mitochondrial Respiratory Chain Complex I. *Cell Metabolism*, 25(1):128–139, 2017.

[85] S. J. Hubbard and J. M. Thornton. Naccess. *Computer Program, Department of Biochemistry and Molecular Biology, University College London*, 1993.

[86] C. Wirth, U. Brandt, C. Hunte, and V. Zickermann. Structure and function of mitochondrial complex I. *Biochimica et Biophysica Acta (BBA)-Bioenergetics*, 1857(7):902–914, 2016.

[87] Protein Data Bank. *Nature New Biology*, 233:223, 1971.

[88] H. Berman, K. Henrick, and H. Nakamura. Announcing the worldwide Protein Data Bank. *Nature Structural & Molecular Biology*, 10(12):980–980, 2003.

[89] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[90] T. N. Bhat, P. Bourne, Z. Feng, G. Gilliland, S. Jain, V. Ravichandran, B. Schneider, K. Schneider, N. Thanki, H. Weissig, J. Westbrook, and H. M. Berman. The PDB data uniformity project. *Nucleic Acids Research*, 29(1):214–218, 2001.

[91] P. D. Adams, P. V. Afonine, K. Baskaran, H. M. Berman, J. Berrisford, G. Bricogne, D. G. Brown, S. K. Burley, M. Chen, Z. Feng, C. Flensburg, A. Gutmanas, J. C. Hoch, Y. Ikegawa, Y. Kengaku, K. Eugene, G. Kurisu, Y. Liang, D. Liebschner, L. Mak, J. L. Markley, N. W. Moriarty, G. N. Murshudov, M. Noble, E. Peisach, I. Persikova, B. K. Poon, O. V. Sobolev, E. L. Ulrich, S. Velankar, C. Vonrhein, J. Westbrook, M. Wojdyr, M. Yokochij, and J. Y. Young. Announcing mandatory submission of PDBx/mmCIF format files for crystallographic depositions to the Protein Data Bank (PDB). *Acta Crystallographica Section D: Structural Biology*, 75(4):451–454, 2019.

[92] P. E. Bourne, H. M. Berman, B. McMahon, K. D. Watenpaugh, J. D. Westbrook, and P. M. D. Fitzgerald. [30] Macromolecular crystallographic information file. In *Macromolecular Crystallography Part B*, volume 277 of *Methods in Enzymology*, pages 571–590. Academic Press, 1997.

[93] W. Kabsch and C. Sander. Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features. *Biopolymers*, 22(12):2577–2637, 1983.

[94] R. P. Joosten, T. A. H. Te Beek, E. Krieger, M. L. Hekkelman, R. W. W. Hooft, R. Schneider, C. Sander, and G. Vriend. A series of PDB related databases for everyday needs. *Nucleic Acids Research*, 39(suppl_1):D411–D419, 2010.

[95] J. D. Blischak, E. R. Davenport, and G. Wilson. A quick introduction to version control with Git and GitHub. *PLoS computational biology*, 12(1):e1004668, 2016.

[96] Schrödinger, LLC. The PyMOL Molecular Graphics System, Version 2.4.0. Software.

[97] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems(5):1695, 2006.

[98] J. Huerta-Cepas, F. Serra, and P. Bork. ETE 3: Reconstruction, Analysis, and Visualization of Phylogenomic Data. *Molecular Biology and Evolution*, 33(6):1635–1638, 2016.

[99] F. Kaden, I. Koch, and J. Selbig. Knowledge-based Prediction of Protein Structures. *Journal of Theoretical Biology*, 147(1):85–100, 1990.

[100] I. Koch, F. Kaden, and J. Selbig. Analysis of Protein Sheet Topologies by Graph Theoretical Methods. *Proteins: Structure, Function, and Bioinformatics*, 12(4):314–323, 1992.

[101] I. Koch, T. Lengauer, and E. Wanke. An Algorithm for Finding Maximal Common Subtopologies in a Set of Protein Structures. *Journal of Computational Biology*, 3(2):289–306, 1996.

[102] I. Koch and T. Lengauer. Detection of distant structural similarities in a set of proteins using a fast graph-based method. In *ISMB Proceedings*, pages 167–178, 1997.

[103] P. May. Eine webbasierte Datenbank-Applikation für Proteinstruktur-Topologien. Master's thesis, Technische Fachhochschule Berlin, 2003.

[104] P. May. Protein Structure Analysis using Contact Maps and Secondary Structure. Phd thesis, Freie Universität Berlin, 2007.

[105] P. May, S. Barthel, and I. Koch. PTGL—a web-based database application for protein topologies. *Bioinformatics*, 20(17):3277–3279, 2004.

[106] P. May, A. Kreuchwig, T. Steinke, and I. Koch. PTGL: a database for secondary structure-based protein topologies. *Nucleic Acids Research*, 38(suppl_1):D326–D330, 2010.

[107] T. Schäfer. Application of Graph Theory to the Topologies of Proteins and Hodgkin Lymphoma. Phd thesis, Freie Universität Berlin, 2016.

[108] T. Schäfer, A. Scheck, D. Bruneß, P. May, and I. Koch. The new protein topology graph library web server. *Bioinformatics*, 32(3):474–476, 2016.

[109] T. Schäfer, P. May, and I. Koch. Computation and Visualization of Protein Topology Graphs Including Ligand Information. In S. Böcker, F. Hufsky, K. Scheubert, J. Schleicher, and S. Schuster, editors, *German Conference on Bioinformatics 2012*, volume 26 of *OpenAccess Series in Informatics (OASIcs)*, pages 108–118, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum füer Informatik.

[110] H. Jörnvall, B. Persson, M. Krook, S. Atrian, R. Gonzalez-Duarte, J. Jeffery, and D. Ghosh. Short-chain dehydrogenases/reductases (SDR). *Biochemistry*, 34(18):6003–6013, 1995.

[111] B. Persson, Y. Kallberg, J. E. Bray, E. Bruford, S. L. Dellaporta, A. D. Favia, R. G. Duarte, H. Jörnvall, K. L. Kavanagh, N. Kedishvili, M. Kisiela, E. Maser, R. Mindnich, S. Orchard, T. M. Penning, J. M. Thornton, J. Adamski, and U. Oppermann. The SDR (short-chain dehydrogenase/reductase and related enzymes) nomenclature initiative. *Chemico-Biological Interactions*, 178(1-3):94–98, 2009.

[112] K. L. Kavanagh, H. Jçrnvall, B. Persson, and U. Oppermann. The SDR superfamily: functional and structural diversity within a family of metabolic and regulatory enzymes. *Cellular and Molecular Life Sciences*, 65:3895–3906, 2008.

[113] V. Sharma, G. Belevich, A. P. Gamiz-Hernandez, T. Róg, I. Vattulainen, M. L. Verkhovskaya, M. Wikström, G. Hummer, and V. R. I. Kaila. Redox-induced activation of the proton pump in the respiratory complex I. *Proceedings of the National Academy of Sciences*, 112(37):11571–11576, 2015.

[114] N. Eswar, B. Webb, M. A. Marti-Renom, M. S. Madhusudhan, D. Eramian, M. Shen, U. Pieper, and A. Sali. Comparative protein structure modeling using Modeller. *Current Protocols in Bioinformatics*, 15(1):5–6, 2006.

[115] M. A. Lomize, I. D. Pogozheva, H. Joo, H. I. Mosberg, and A. L. Lomize. Opm database and ppm web server: resources for positioning of proteins in membranes. *Nucleic Acids Research*, 40(D1):D370–D376, 2012.

[116] A. Djurabekova, O. Haapanen, and V. Sharma. Proton motive function of the terminal antiporter-like subunit in respiratory complex I. *Biochimica et Biophysica Acta (BBA)-Bioenergetics*, 1861(7):148185, 2020.

[117] I. Nurhassen. Bioinformatics analysis of dynamics of large protein structures focusing on topological changes based on graph-theoretical modeling. Master's thesis, Johann Wolfgang Goethe-University, 2021.

[118] M. B. Cohen, Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford. Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 9–21, New York, NY, USA, 2016. Association for Computing Machinery.

[119] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[120] D. J. Barlow and J. M. Thornton. Helix Geometry in Proteins. *Journal of Molecular Biology*, 201(3):601–619, 1988.

[121] M. N. Fodje and S. Al-Karadaghi. Occurrence, conformational features and amino acid propensities for the π-helix. *Protein Engineering, Design and Selection*, 15(5):353–358, 2002.

[122] T. Brinkjost, C. Ehrt, O. Koch, and P. Mutzel. SCOT: Rethinking the classification of secondary structure elements. *Bioinformatics*, 36(8):2417–2428, 2020.

[123] V. Sons. Bioinformatik analysis of molecular dynamics of protein complexes using graph-theoretical methods. Bachelor's thesis, Johann Wolfgang Goethe-University, 2021.

[124] A. Wolnitza. Development and bioinformatic analysis of heatmap visualisation of structural dynamics in bacterial respiratory complex I based on time-series molecular dynamics data. Bachelor's thesis, Johann Wolfgang Goethe-University, 2021.

[125] L. A. Sazanov. A giant molecular proton pump: structure and mechanism of respiratory complex I. *Nature Reviews Molecular Cell Biology*, 16(6):375–388, 2015.

[126] D. Langosch and I. T. Arkin. Interaction and conformational dynamics of membrane-spanning protein helices. *Protein Science*, 18(7):1343–1358, 2009.

[127] Z. Cao and J. U. Bowie. Shifting hydrogen bonds may produce flexible transmembrane helices. *Proceedings of the National Academy of Sciences*, 109(21):8121–8126, 2012.

[128] O. Haapanen and V. Sharma. Role of water and protein dynamics in proton pumping by respiratory complex I. *Scientific Reports*, 7(1):1–12, 2017.

[129] M. Zunker. Bioinformatical investigation of subunits of protein complexes using graph-theoretic approaches. Master's thesis, Johann Wolfgang Goethe-University, 2021.

[130] T. Huxford, D. Huang, S. Malek, and G. Ghosh. Validation report of IκBα/NF-κB complex (PDB 1ikn). https://files.rcsb.org/pub/pdb/validation_reports/ik/1ikn/1ikn_full_validation.pdf. Accessed: 2020-11-29.

[131] S. J. Hardy, J. Holmgren, S. Johansson, J. Sanchez, and T. R. Hirst. Coordinated assembly of multisubunit proteins: Oligomerization of bacterial enterotoxins *in vivo* and *in vitro*. *Proceedings of the National Academy of Sciences*, 85(19):7109–7113, 1988.

[132] C. M. Möller. Bioinformatic exploration of contacts transitive by ligands applied to assembly prediction of protein complexes. Bachelor's thesis, Johann Wolfgang Goethe-University, 2022.

[133] K. Szczepanowska, K. Senft, J. Heidler, M. Herholz, A. Kukat, Klaus Hö, Zwicker, Sergio Guerrero-Castillo, Linda Baumann, Johanna Kauppila, Anastasia Rumyantseva, Stefan Müller, Christian K. Frese, Ulrich Brandt, Jan Riemer, Ilka Wittig, and Aleksandra Trifunovic. A salvage pathway maintains highly functional respiratory complex I. *Nature Communications*, 11(1):1–18, 2020.

[134] C. Orsini, M. M. Dankulov, P. Colomer de Simón, A. Jamakovic, P. Mahadevan, A. Vahdat, K. E. Bassler, Z. Toroczkai, M. Boguná, G. Caldarelli, S. Fortunato, and D. Krioukov. Quantifying randomness in real networks. *Nature Communications*, 6(1):1–10, 2015.

[135] N. Subrahmanian, C. Remacle, and P. P. Hamel. Plant mitochondrial Complex I composition and assembly: A review. *Biochimica et Biophysica Acta (BBA)-Bioenergetics*, 1857(7):1001–1014, 2016.

[136] P. J. Nixon, F. Michoux, J. Yu, M. Boehm, and J. Komenda. Recent advances in understanding the assembly and repair of photosystem II. *Annals of Botany*, 106(1):1–16, 2010.

[137] J. Zabret, S. Bohn, S. K. Schuller, O. Arnolds, M. Möller, J. Meier-Credo, P. Liauw, A. Chan, E. Tajkhorshid, J. D. Langer, R. Stoll, A. Krieger-Liszkay, B. D. Engel, T. Rudack, J. M. Schuller, and M. M. Nowaczyk. Structural insights into photosystem II assembly. *Nature Plants*, 7(4):524–538, 2021.

[138] V. Zickermann. Personal communication.

[139] D. N. Grba and J. Hirst. Mitochondrial complex I structure reveals ordered water molecules for catalysis and proton translocation. *Nature Structural & Molecular Biology*, 27(10):892–900, 2020.

[140] K. Parey, J. Lasham, D. J. Mills, A. Djurabekova, O. Haapanen, E. G. Yoga, H. Xie, W. Kühlbrandt, V. Sharma, J. Vonck, and V. Zickermann. High-resolution structure and dynamics of mitochondrial complex I—Insights into the proton pumping mechanism. *Science Advances*, 7(46):eabj3221, 2021.

[141] J. Agirre. Strategies for carbohydrate model building, refinement and validation. *Acta Crystallographica Section D: Structural Biology*, 73(2):171–186, 2017.

Acknowledgements

With this work, I finish nine years of study at the Goethe-University. I have done my Bachelor's and Master's Thesis at the group of Molecular Bioinformatics and now my Doctoral Thesis, too. This work would not have been possible without Prof. Dr. Ina Koch who gave me the chance and guided me. Thank you very much for being understanding and for providing a friendly atmosphere in the group.

I am extremely grateful to Dr. Jörg Ackermann for all the discussions we had no matter if on research, politics or good books. I would like to express sincere thanks to Brigitte Scheidemantel-Geiß for all the administrative support over all the years and the good cooperation, especially during Covid crisis. I am also grateful for the technical support by Norbert "Nodi" Dichter and his patience with all the problems I caused on the computer cluster. I want to acknowledge the RBI for technical support and want to mention Thomas Schönberger and Marko for helping me with my computer.

Many thanks to the MolBI group in general and especially to Zhen Song and Jens Rieser with whom I shared an office. I want to thank my fellow PhD students for the great discussions and valuable feedback. Big thanks to Florian Gisdon for the feedback on my work and thesis. I had the pleasure of working with many students and want to thank them for their passion for our projects. Special thanks to Mariella Zunker for enduring the Covid crisis together, and proofreading my thesis.

Lastly, I want to thank my family for always supporting me during my education. I wish to mention Benjamin Wolf, Silvia Wolf and Jürgen Ruhl who mean so much to me. I wished Karin Ruhl were still here and want to include her in the acknowledgements for all the support over my whole life.

This thesis marks the end of my education and I want to thank my fellow students from the study for the Bachelor's and Master's degree for the great time. Sincere thanks to all my friends that kept me going. I want to mention Dominik Hansen and Maren Meier for whom I am really grateful, because they were always there for me.

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die Doktorarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Referenzen auf andere Veröffentlichungen, wörtlich oder dem Sinn nach, sind angegeben.

Ich habe die Grundsätze der guten wissenschaftlichen Praxis nach bestem Wissen und Gewissen beachtet.

Die Doktorarbeit wurde keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

Ort, Datum Vorname Nachname