Bioinformatische Anwendung von *Graphlets* zur Analyse von Proteinstrukturtopologien zur Analyse von Proteinen
Rohfassung

Ben Haladik

9. Februar 2016

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

TODO: Zitationen einfügen: schwierigkeit der Analyse, bereits bekannte Methoden

Dei Ähnlickeit von Proteinen zu bestimmen ist eine große Herausforderung. In der Zeit als wenige Strukturen bekannt waren wurde die strukturelle Ähnlichkeit von Proteinen noch von Experten visuell bewertet, aber mittlerweile befinden sich die Strukturen von über 100000 biologischen Makromolekülen in der PDB. Der Vergleich dieser gigantischen Anzahl von Strukturdaten erfordert effiziente Analysemethoden. Solche Analysen können tiefe Einblicke in die ferne evolutionäre Verwandschaft von Proteinen liefern und sie helfen bei der Bestimmung der Funktion eines Proteins. Sie sind im Bereich des *Drug-Design* hoch interessant, denn Strukturdaten liefer Informationen über m ögliche Liganden, die ein Protein binden kann und damit auch über mögliche Ziele von Medikamenten bei der Bekämpfung von Krankheiten. Deshalb sind strukturbasierte Ähnlichkeitsanalysen in der pharmakologischen Forschung von zentraler Bedeutung. Das Problem hierbei ist, dass die Berechnung der Ähnlichkeit von 3D-Strukturen algorithmisch ein schwieriges Problem darstellt; lange Berechnungszeiten sind die Regel und es ist schwierig herauszufinden, ob der gefundene Ähnlichkeitswert für zwei Proteine nur ein lokales Optimum darstellt. Deshalb wird versucht, von der 3D-Darstellung zu abstrahieren, ohne zentrale Strukturinformationen zu verlieren. *Dali* [4]verwendet beispielsweise Distanzmatrizen, die die Abstände einzelner Residuen zueinander speichern, anstatt die Koordinaten jedes Atoms abdsspzuspeichern. Der Strukturvergleich findet dann als Vergleich dieser Distanzmatrizen statt.

Auch Graphen eignen sich, um Strukturdaten in einer leichter zu analysierenden Form abzuspeichern. Denn Graphen können genau wie Distanzmatrizen Informationen über räumliche Nähe aufbewahren. Die PTGL [2] speichert Proteinstrukturtopologien als Graphen ab. So werden zentrale Informationen über die Struktur eines Proteins aufbewahrt während die Größe der Daten maßgeb-

lich reduziert wird. Diese Darstellung hat den weiteren Vorteil, dass Graphen zu den meistuntersuchten mathematischen strukturen der letzten Jahre gehören. Soziale Netzwerke, Interaktionen von Proteinen in Zellen, das Internet: All diese Dinge lassen sich als Graphen darstellem und werden als solche untersucht. Doch auch die Ähnlichkeit von Graphen zu bestimmen ist ein schwieriges Problem. Es leitet sich vom *Graph-Isomorphismus-Problem* ab. Es ist nicht klar, ob dieses Problem NP-vollständig ist. In den letzten Jahren wurde *Feature*-basierte Methoden erforscht, die diese Problem reduzieren. Anstatt direkt zwei Graphen miteinander zu vergleichen, werden *Features* verglichen, so dass einfache Datenpunkte anstatt von komplexen Graphen verglichen werden. Solche Methoden wurden häufig erfolgreich angewandt. Diese Anwendungen fanden immer auf großen Netzwerken statt. Nun stellt sich die Frage, ob sich solche Methoden auch für deutlich kleinere Netzwerke eignen. in der vorliegenden Arbeit wird eine solche *Feature*-basierte Methode - der *Graphlet*-Algorithmus auf verschiedene Proteingraphen angewandt.

1.2 *State of the Art*

TODO: Methoden nennen, weitere Zitationen

Es gibt bereits einige Methoden, um Proteinstrukturen miteinander zu vergleichen. *Hasegawa et al* [1] liefern einen umfangreichen Vergleich verschiedener Methoden, bei denen Strukturen auf unterschiedlichen Abstraktionsstufen betrachtet und verglichen werden. So können Proteinstrukturen dreidimensional, zweidimensional und eindimensional betrachtet und verglichen werden.

3D-Methoden versuchen zunächst mittels Sequenzalignment einen Bereich in den zu vergleichenden Proteinen zu finden, in dem sich beide Proteine sehr ähnlich sind. Dieser Bereich fungiert gewissermaßen als *Anker* für das weitere Alignment. In den weiteren Schritten werden die Proteine so positioniert, dass die Distanzen in dem alignierten Bereich minimal sind. Von diesem *Template* ausgehend, werden die Distanzen zwischen den weiteren Residuen der Proteine berechnet und meist mittels *Root-mean-square-deviation* bewertet.

2D-methoden versuchen Kontakte zwischen Residuen oder Sekundärstrukturen zu vergleichen. Diese Kontakte werden beispielsweise als graphen oder Distanzmatrizen dargestellt. Der Vergleich zwischen zwei Proteinstrukturen wird dann beispielsweise als Verlgeich zweier Distanzmatrizen durchgeführt.

1D-Methoden nutzen Strukturprofile zur Darstellung von Proteinen. In Strukturprofilen repräsentieren einzelne Buchstaben Eigenschaften von Residuen und die Konformation des Protein-*Backbone* an der entsprechenden Stelle. So können schnelle *String*-Algorithmen genutzt werden, um Strukturen zu suchen und zu vergleichen.

0D-Methoden reduzieren die 3D-Struktur am stärksten. Die gesamte Struktur wird durch eine Zahl beschrieben, die sich aus der Struktur berechnen lässt. Sie erlauben sehr schnelle Suchen in Datenbanken, haben aber das Problem, dass sie keinen Vergleich von Teilstrukturen ermöglichen.

Die Methoden stellen alle einen Versuch dar, die Ähnlichkeit von Proteinen zu beziffern. Sie werden angewendet, um entferent homologe Proteine aufzuspüren und in Datenbanken eine Ähnlichkeitssuche zu ermöglichen. Dies findet vor allem im pharmakologischen Bereich Anwendung.

Interessant ist, dass unter den von *Hasegawa et al* vorgestellten 1D-Methoden keine wirklich analog zur Analyse mit *Graphlets* funktioniert. Andere 1D-Methoden versuchen die Polypeptidkette als *String* darzustellen und damit die Konformationsänderung des *Backbone* zu beschreiben. Im Gegensatz dazu zählt der *Graphlet*-Algorithmus die *Graphlets* unabhängig von ihrer Position im Graphen. Somit repräsentiert der *Graphlet*-Vektor an jeder Stelle eine globale Eigenschaft des Graphen, anstatt die Veränderung von einer Sekundärstruktur zur nächsten zu beschreiben.

Des weiteren sind die meisten Methoden, die Proteinstrukturen vergleichen, *Template*-basiert. Um den Suchraum einzuschränken muss eine Struktur ausgewählt werden, die als Vorlage für den Vergleich mit anderen Strukturen dient. Dementsprechend ändert sich das Ergebnis des Vergleichs auch in Abhängigkeit der gewählten Vorlage.

1.3 Ziele

Ziel dieser Arbeit war zunächst die Erweiterung der Funktionalität des `graphletAnalyser`. Hierzu gehört eine funktionierende Datenbankanbindung, so dass die *Graphlet*-Vektoren für alle verschiedenen Graphtypen korrekt in die PTGL eingetragen werden können. Die Suche nach markierten *Graphlets* sollte so implementiert werden, dass sie auf Graphen mit beliebigen Markierungen angewandt werden kann. Deshalb wurde ein Algorithmus entwickelt und implementiert, der aus einem Alphabet von Knotenmarkierungen alle Worte berechnet, die markierte 2- und 3-*Graphlets* repräsentieren - der *Graphlet*-Worte-Algorithmus. Schlussendlich sollte im durch Fallstudien überprüft werden, ob und inwiefern sich *Graphlets* eignen, um die Ähnlichkeit von Proteinstrukturtopologien zu untersuchen. Dies wurde mit unterschiedlichen Metriken getestet.

1.4 Aufbau der Arbeit

Zunächst wird im Kapitel *Materialien und Methoden* PLCC vorgestellt - das Programm, mit dem die Graphen der PTGL erzeugt werden. Es folgt eine Kurzbeschreibung der PTGL selbst, sowie eine Beschreibung des *Graphlet*-Algorithmus und des *Graphlet*-Worte-Algorithmus. Weiterhin wird das Programm `graphletAnalyser`vorgestellt, welches den *Graphlet*-Algorithmus im-

plementiert. Anschließend werden verschiedene Metriken vorgestellt, mit denen die erhaltenen *Graphlet*-Vektoren verglichen werden. Die Ergebnisse dieses Vergleichs werden im Teil Ergebnisse und Diskussion vorgestellt. Hierbei werden die verschiedenen Metriken verglichen und es wird überprüft inwiefern die Suche nach markierten *Graphlets* zu diesem Ergebnis beiträgt.

Kapitel 2

Materialien und Methoden

Um die Proteinstrukturtopologien aus der PTGL zu vergleichen wurde das Programm `graphletAnalyser` genutzt und erweitert. Es wurde bereits 2013 von *Tatiana Bakirova* im Rahmen ihrer Diplomarbeit im Arbeitskreis *Molekulare Bioinformatik* geschrieben. Die ursprüngliche Funktionalitt wurde erweitert. Hierbei wurden Funktionen zur Analyse von Komplexgraphen, Aminosäuregraphen und den Sekundärstrukturgraphen implemetiert. Diese Graphen stammen allesamt aus der PTGL (Protein Topology Graph Library) von Tim Schäfer.

2.1 Die PTGL

Die Protein Topology Graph Library ist 2009 von *May et al.* [2] entwickelt worden. Ausgehend von der Tatsache, dass sich Proteinstrukturtopologien als räunliche Beziehungen von SSEs untereinander definieren lassen, verwendet die PTGL *Graphen*, um Proteinstrukturtopologien darzustellen. Hierbei stellen die Knoten des Graphen die SSEs eines Proteins dar. Sie werden dem jeweiligen SSE entsprechend markiert. Knoten, die α-Helices repräsentieren werden mit einem H markiert, β-Faltblätter mit einem E. Weiterhin ermöglicht die PTGL die Darstellung von Liganden, ([5]) denen mit L markierte Knoten zugeordnet werden. Um die räumliche Nachbarschaft von Sekundärstrukturen und Liganden untereinander darstellen zu können werden ungerichtete Kanten zwischen Knoten gezogen wenn die entsprechenden Elemente benachbart sind. Jede Polypeptidkette eines Proteins wird dann als *Proteingraph* dargestellt. Die Zusammenhangskomponenten eines Proteingraphen werden als Faltungsgraphen bezeichnet, weil sie typischerweise eine unabhängige Faltungseinheit darstellen.

Die Berechnung dieser Graphen erfolgt unter Verwendung der entsprechenden PDB und DSSP Dateien. Um den Graphen einer Polypeptidkette zu berechnen werden aus der DSSP-Datei die Sekundärstrukturelemente (SSEs) des Proteins ausgelesen. Für jedes Paar von SSEs wird die Anzahl der räumlichen Kontakte ihrere Residuen berechnet. Wenn die Anzahl dieser Kontakte

einen gewissen Grenzwert überschreitet, wird die angenommen, dass diese SSEs räumlich benachbart sind und die jeweiligen Knoten werden durch eine Kante verbunden.

Komplexgraphen werden ebenfalls in dieser Arbeit untersucht. Ihre Berechnung erfolgt analog zur Berechnung der Proteingraphen. Komplexgraphen können einen vollständigen Proteinkomplex darstellen. In einem solchen Graphen werden zusätzlich die Nachbarschaften von SSEs unterschiedlicher Polypeptidketten einbezoogen. Ein Komplexgraph setzt sich also aus mehreren Proteingraphen zusammen. Hier wird jedem Knoten zusätzlich zum SSE die Zugehörigkeit zu einer Polypeptidkette zugeordnet.

Aminosäuregraphen werden ebenfalls untersucht und analog zu Proteingraphen berechnet. Hier werden keine SSEs betrachtet. Stattdessen repräsentiert jeder Knoten eine Aminosäure eines Proteins. Die Knoten werden entsprechend der chemischen Eigenschaften der Aminosäuren markiert. Knoten, die saure oder basische Residuen darstellen werden mit einem c markiert. Ein p markiert Knoten für polare Residuen, die weder sauer noch basisch sind. Für unpolare Aminosäuren wird ein h verwendet. Auch Liganden können in Aminosäuregraphen dargestellt werden. Ihre Knoten werden durch ein ? markiert.

Für die Berechnungen dieser Arbeit wurde eine lokale Datenbank erstellt, die das gleiche Schema wie die PTGL verwendet.

2.2 Der *Graphlet*-Algorithmus

Motivation Die PTGL [5] ermöglicht also die Darstellung von Proteinstrukturtopologien als Graphen. Um aus diesen Graphen weitere Informationen zu gewinnen, ist es sinnvoll, sie untereinander vergleichen zu können. Ein solcher Vergleich ist jedoch ein schwieriges Problem: Gesucht ist eine Funktion $f : (G, G') \to \mathbb{R}$, die für zwei Graphen G und G' deren Ähnlichkeit zueinander beziffert. Es gibt diverse Möglichkeiten diesesd Problem zu bearbeiten, von denen jedoch keine einfach ist. Eine Möglichkeit ist, die Suche nach größten gemeinsamen isomorphen Teilgraphen in G und G', oder man versucht eine Editierdistanz zu berechnen - also herauszufinden, wie viele Operationen (Hinzufügen oder Entfernen von Knoten und Kanten) nötig sind um G in G' zu überführen. Diese beiden genannten Methoden erfordern jedoch aufwändige Berechnungen. Deshalb werden Methoden verwendet, die *Topologische Charakteristiken* berechnen und dies in polynomieller Laufzeit bewerkstelligen. Der Vorteil hierbei ist, dass die (aufwändige) Berechnung dieser Charakteristiken nur einmal pro Graph erfolgen muss. Die Charakteristika können dann als Vektoren verglichen werden und man spart sich die Berechnungen, die man sonst für alle Paare von Graphen G, G' durchführen muss.

Beschreibung des Algorithmus *Graphlets* sind kleine induzierte Teilgraphen eines größeren ungerichteten Graphen. *N. Shervashidze* stellte diese Me-

thode als Vergleichsschema für Graphen 2009 zum ersten Mal vor. (Literaturverweis einfügen). Folgendes Bild zeigt alle *Graphlets* der Größe 4:

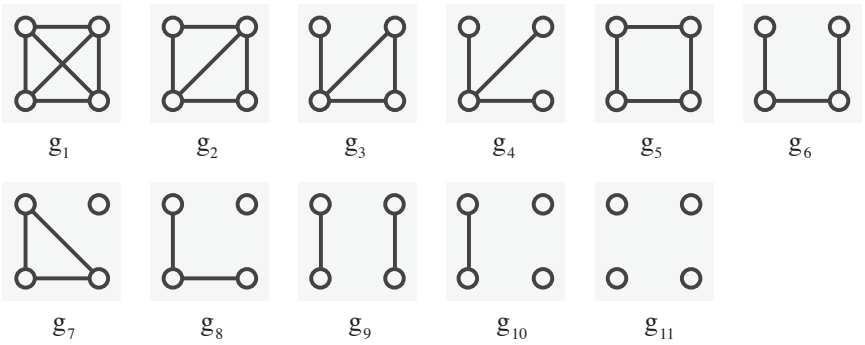

Abbildung 2.1: Graphlets der Größe 4 (*Shervashidze et al.*)

Um ein *Graphlet* der Größe k zu finden, besucht der Algorithmus alle Euler-Wege der Länge k, in dem gegebenen Graphen. Für jeden dieser Wege überprüft er, für alle Paare von Knoten v, w ob es eine Kante $e = v, w$ gibt, die nicht zu dem besuchten Euler-Weges gehgört. Je nachdem, welche Kanten hierbei gefunden werden, wird der Zähler für das entsprechende *Graphlet* erhöht. Der Algorithmus zählt hier aber nur alle zusammenhängenden *Graphlets*. Er verwendet die folgenden Gewichtungsvektoren:

Graphlet-Gewichtungsvektoren

$$w_2 := \left(\frac{1}{2}\right) \tag{2.1a}$$

$$w_3 := \left(\frac{1}{6}, \frac{1}{2}\right) \tag{2.1b}$$

$$w_4 := \left(\frac{1}{24}, \frac{1}{12}, \frac{1}{4}, 1, \frac{1}{8}, \frac{1}{2}\right) \tag{2.1c}$$

$$w_5 := \left(\frac{1}{120}, \frac{1}{72}, \frac{1}{48}, \frac{1}{36}, \frac{1}{28}, \frac{1}{20}, \frac{1}{14}, \frac{1}{10}, \frac{1}{12}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, \frac{1}{12}\frac{1}{12}, \frac{1}{4}\frac{1}{4}, \frac{1}{2}, 1, \frac{1}{2}, 1\right) \tag{2.1d}$$

Jede Stelle eines Vektors w_i ist mit einem *Graphlet* assoziiert. Da der Algorithmus alle Euler-Wege einer Länge i in dem Graphen abläuft, sind in den Vektoren Brüche eingetragen, wobei der Zähler für die Anzahl der Euler-Wege der Länge i steht. Dies stimmt natürlich nicht für die sogenannten Stern-*Graphlets*

(g_4 in 2.1 g_{19},g_{20} und g_{21} in 5.3). Da diese keinen Euler-Weg der Länge 4 bzw. 5 enthalten werden sie anders gezählt. (beispiel mit Pseudocode einfügen?)

2.2.1 Der *Graphlet*-Worte-Algorithmus

In der letzten Version von `graphletAnalyser` war es bereits möglich markierte *Graphlets* mit 2 und 3 Knoten in Proteingraphen zu zählen. Diese Funktionalität wurde im Rahmen dieser Arbeit verallgemeinert, so dass der Nutzer beliebige Alphabete angeben kann. Der Algorithmus erhält das Alphabet $\sum = \{\sigma_i : i \in \mathbb{N}\}$ der Knotenmarkierungen. Aus diesem Alphabet berechnet er Worte w, die zur Repräsentation der markierten *Graphlets* genutzt werden. Hierbei können verschiedene Worte das gleiche *Graphlet* repräsentieren. Im Falle von 2-*Graphlets* repräsentieren die zwei Worte (σ_i, σ_j) und (σ_j, σ_i) das gleiche markierte *Graphlet* mit den Knotenmarkierungen σ_i, σ_j. Worte, die das gleiche *Graphlet* repräsentieren, werden im Folgenden als *äquivalente Graphlet-Worte* bezeichnet.

Die Berechnung der äquivalenten *Graphlet*-Worte der Länge 2 ist trivial. Aus dem Alphabet $\sum$ werden alle Worte $w = (\sigma_i, \sigma_j)$ berechnet, wobei Spiegelungen nicht mit ausgegeben werden, da zwei Worte (σ_j, σ_i) und (σ_j, σ_i) äquivalente *Graphlet*-Worte sind.

Die Berechnung aller äquivalenten *Graphlet*-Worte der Länge 3 ist komplizierter, da sie für zwei verschiedene *Graphlets* berechnet werden müssen. Für das *Graphlet* g_1 sind alle Worte äquivalent zueinander, die zyklische Vertauschungen voneinander sind. Für das *Graphlet* g_2 sind Worte äquivalent zueinander, die Spiegelungen voneinander sind (siehe Abbildung 5.1).

Der Algorithmus besteht aus 3 *for*-Schleifen, die über das Alphabet iterieren. In der äußersten Schleife werden Worte hinzugefügt, in denen alle Buchstaben gleich sind. In der zweiten Schleife wird jeweils der nächste Buchstabe des Alphabets betrachtet. Für jedes Paar a, b von Buchstaben über einem Alphabet $\sum$ sind die Mengen der äquivalenten Worte für das *Graphlet* g_1 $P_{3-Kreis} = \{\}$ und $P_{3-Pfad} = \{aaa, aba, aab\}$ für das *Graphlet* g_2

TODO: Pseudocode, Beschreibung der Mengen, Beweis

$$p_2 := (HH, HE,$$
(2.2a)

$$p_{3-Weg} := (HHH, HEH, HHE, HEE, EHE, HEL, LHE, ELH, HLH, HHL, HLL, LHL, EEE, ELE, EEL,$$
(2.2b)

$$p_{3-Kreis} := (HHH, HEH, HEE, HEL, LEH, HLH, HLL, EEE,$$
(2.2c)

(2.2d)

Die Vektoren a_2, a_{3-Weg} und $a_{3-Kreis}$ beschreiben die Worte für *Graphlets*

in AA-Graphen

(2.3a)

$$a_{3-Weg} := (hhh, hph, hhp, hpp, php, hpa, ahp, pah, hp?, ?hp, p?h, hah, hha, haa, aha, ha?, ?ha, a?h, h?h, hh?, h??$$

(2.3b)

$$a_{3-Kreis} := (hhh, hph, hpp, hpa, aph, h$$

(2.3c)

2.3 `graphletAnalyser`

Das Programm `graphletAnalyser` implementiert die oben beschriebenen Algorithmen und kann die Resultate der Berechnungen in verschiedenen Formaten lokal auf dem Rechner des Nutzers speichern und sie in einer *Postgres-ql*-Datenbank ablegen, die mit PLCC erstellt wurde. `graphletAnalyser` ist in C++ geschrieben und nutzt die *Boost-Graph-Library* zur internen Darstellung der Graphen.

Das Programm wird über die Konsole gestartet und erhält als *Input* eine oder mehrere GML-Dateien. Der Nutzer kann weiterhin über Parameter festlegen, ob markierte *Graphlets* berechnet werden sollen, ob die Resultate in einer mit PLCC erstellten Datenbank gespeichert werden sollen und ob er die vordefinierten Knotenmarkierungen von SSE-Graphen, Komplexgraphen oder Aminosäuregraphen nutzen möchte.

Weiterhin können in der Konfigurationsdatei nutzerdefinierte Alphabete von Knotenmarkierungen eingegeben werden, die bei der Berechnung markierter *Graphlets* genutzt werden können.

Das Programm bestand bereits vor dem Beginn dieser Arbeit. Es wurden Fehler im Code beseitigt und einige Funktionen hinzugefügt, die im folgenden kurz beschrieben werden.

Das Einlesen von Komplexgraphen und Aminosäuregraphen ist implementiert worden. Die entsprechenden Alphabete sind im Programmcode vordefiniert und können vom Nutzer über Parameter ausgewählt werden.

Nutzerdefinierte Knotenmarkierungen können nun in der Konfigurationsdatei angegeben werden. Der Nutzer kann ein Alphabet von Knotenmarkierungen und ein *Label* unter dem diese Knotenmarkierungen in den GML-Dateien abgelegt sind angeben. Für dieses Alphabet werden alle äquivalenten *Graphlet*-Worte durch den *Graphlet*-Worte-Algorithmus berechnet. Diese werden dann bei der Berechnung der markierten *Graphlets* im Graphen unter dem vorgegebenen *Label* gesucht und gezählt. Es können also beliebige Alphabete und *Labels* angegeben werden, so lange die Markierungen der Knoten nicht mehr als einen Buchstaben enthalten.

Die Datenbankanbindung wurde um Funktionen zum Speichern von Aminosäuregraphen und Komplexgraphen erweitert. Das Speichern von Vektoren markierter *Graphlets* wurde implementiert.

2.4 Scoring

2.4.1 Relative *Graphlet*-Häufigkeiten-Distanz

N. Pržulj et al. haben *Graphlets* bereits in verschiedensten Zusammenhänen auf biologische Daten wie Protein-Protein-Interaktionsnetzwerke [3] angewandt. Als Maß für die Ähnlichkeit von Netzwerken nutzen sie die Relative-*Graphlet*-Häufigkeiten-Distanz (RGF) $D(G, H)$. Diese Metrik berechnet den Abstand zwischen zwei Graphen G und H als logarithmierte Differenz der normalisierten Anzahl der *Graphlets* in G und H. Sie ist folgendermaßen definiert:

Sei $N_i(G)$ die Anzahl der *Graphlets* von Typ $i \in 1, ..., 29$ und
$T(G) = \sum_{i=1}^{29} N_i(G)$ die Anzahl der *Graphlets* in G, beziehungsweise H

Dann ist die Relative-*Graphlet*-Häufigkeiten-Distanz $D(G, H)$ für zwei Graphen G und H definert als:

$$D(G, H) := \sum_{i=1}^{29} |F_i(G) - F_i(H)| \tag{2.4a}$$

$$mit F_i(G) := -log(\frac{N_i(G)}{T(G)}) \tag{2.4b}$$

Diese Metrik lässt sich analog zur euklidischen Distanz, oder einer beliebigen anderen Vektornorm auffassen, denn sie berechnet für jeden Index i die Differenz zwischen den Stellen x_i, y_i zweier Vektoren x und y. Sie verwendet die normalisierten *Graphlet*-Vektoren unter der Annahme, dass die Ähnlichkeit zweier Netzwerke sich aus der Ähnlichkeit lokaler Substrukturen ableiten lässt [3]. Somit können Netzwerke, die ähnliche Substrukturen habe, sich aber in ihrer Größe stark unterschieden, immer noch als ähnlich erkannt werden. Weiterhin wurde gezeigt, [3] dass diese Metrik auch bei verrauschten Daten noch sehr gut funktioniert. Hierbei ist jedoch zu beachten, dass sie bisher vor allem für sehr große Netzwerke mit mehreren Tausend Knoten und Kanten verwendet wurde. Diese Größe kann von Aminosäuregraphen erreicht werden, wenn sie große Proteine modellieren. Proteingraphen und Komplexgraphen sind aber deutlich kleiner.

2.4.2 Modifizierter Jaccard-Index

Der Jaccard-Index ist im eigentlichen Sinne ein Maß, um die Ähnlichkeit von gleichmächtigen Mengen zu bewerten. Für zwei Mengen A, B berechnet sich der Jaccard-Index $D_{Jac}(A, B)$ folgendermaßen:

$$D_{Jac}(A,B) := \frac{\sum_{x\ inA \wedge x \in B} 1}{\sum_{x \in A \vee x \in B} 1}$$

Dementsprechend sind zwei Mengen A, B gleich, wenn gilt $D_{Jac} = 1$ und disjunkt, wenn gilt $D_{Jac} = 0$. Mit ihm wird die relative Anzahl der Elemente beider Mengen berechnet. Um dieses Maß in sinnvoller Weise auf *Graphlet*-Vektoren zu übertragen wurde ein zusätzlicher Faktor $k \in \mathbb{R}$ mit $k \in [0,1]$ eingeführt, der als Präzisionsfaktor zu verstehen ist. Die Defintion des modifizierten Jaccard-Index $D_{Jac-m}(v,w)$ für zwei Vektoren v, w lautet also:

$$D_{Jac-m}(v,w) := \frac{\sum_{i=1}^{n} x_i}{\sum_{x \in A \vee x \in B} 1}$$

Hierbei gilt:

$$x_i = \begin{cases} 1 & \text{if} \quad v_i >= w_i \times k \wedge w_i >= v_i \times k \\ 0 & \text{else} \end{cases}$$

In dieser modifizierten Variante werden zwei Vektoren v, w als gleich angesehen, wenn sich v_i, w_i für alle i höchstens um den Faktor k unterschieden. Dies steht im Gegensatz zur RGF, die - analog zu einer Vektornorm - die Abstände zwischen zwei *Graphlet*-Vektoren misst.

2.5 Datensätze

2.5.1 Fallstudien

Anfangs wurden 20 verschiedene Proteine aus verschiedenen CATH-Klassen zum Vergleich ausgewählt. Die Proteine wurden so gewählt, dass die drei CATH-klassen mit vielen Sekundärstrukturelementen *mainly-alpha*, *mainly-beta* und *alpha-beta* vertreten sind. Weiterhin wurde die Auswahl so getroffen, dass es für jede Stufe der Hierarchie mindestens zwei Proteine gibt, die auf der entsprechenden Stufe in die gleiche Klasse eingeordnet werden. Die Proteine werden im folgenden kurz beschreiben.

4-Helix-*Bundles*

Das Cytochrom B562 von *E. coli* (Bild einfügen) ist ein Protein, das für Elektronentransport zuständig ist. Strukturell wird es als 4-Helix-*Bundle* eingeordnet. Zu diesem Protein gibt es zwei PDB-Einträge: 1QPU und 256B. Es gibt von diesem Protein auch eine Hem-bindende Variante, die oxidiert ist. Sie hat die PDB-ID 1QQ3. All diese Strukturdaten werden von CATH als sehr ähnliche Strukturen mit hoher Sequenzidentität eingeordnet. Um die Parameter und Metriken für den Vergleich zu testen, bietet es sich an, diese Proteine mit weiteren Proteinen aus der selben homologen Superfamilie zu vergleichen und diese dann mit Proteinen aus anderen Homologen Superfamilien zu vergleichen. Die

entsprechenden Proteine werden händisch entsprechend der Einordnung durch CATH gewählt. Proteine die neu designt wurden könnten sich auch anbieten. Inwiefern eine Untersuchung an ihnen sinnvoll ist, muss aber noch überprüft werden. Weitere Proteine aus der Homologen Superfamilie, die sich eignen sind:
2XL6: Cytochrom C aus Alcaligenes Xylosoxidans, gebunden mit NO
2YL1: s.o. nur andere Variante mit anderem Bindungspartner.
Diese beiden müssten strukturell ähnlicher sein, als die weiter oben genannten Varianten von Cytochrom B562.
Die oben genannten Proteine werden von CATH alle in die *Topologie* der 4-Helix-*Bundles* (Hemerythrin Untereinheit A) eingeordnet. Sie sind alle Teil der selben homologen Superfamilie.

Zum Vergleich werden weiterhin 1rxq und 2qe9 aus der Homologen Superfamilie *dinb family like domain*

Wachstumshormone

Sie können mit Proteinen aus der Topologie *Growth Hormone, Chain A* verglichen werden. Denn beide werden von CATH in die gleiche Architektur *Up-down bundle* eingeordnet. Proteine aus dieser Topologie sind:
1HGU: Menschliches Wachstumshormon
1M4R: Rekombinantes menschliches Interleukin 22
1D9C: Rinder Interferon Gamma
Die oben genannten Proteine aus der Topologie der Wachstumshormone haben eine Sequenzidentität zwischen 35 und 60 Prozent. Sie scheinen sich gut für die untersuchung zu eignen. Das Protein mit der PDB-ID 1PV6 liegt in der selben Topologie, wird aber in eine andere Superfamilie eingeordnet.

Aldolasen

Die Aldolasen 7TIM, 1NEY und 2V5L werden durch das gleich Gen in Hefe codiert. Da sie eine β-*Barrel*-Struktur besitzen, sollten sie eine niedrigere *Graphlet*-Distanz zueinander aufweisen, als zu allen anderen Strukturen.

Zusammengefasst

PDB-ID	CATH-Code
2UTG	1.10.210.10
1VIB	1.10.287.120
1QPU	1.20.120.10
1QQ3	1.20.120.10
2XL6	1.20.120.10
2YL1	1.20.120.10
1rxq	1.20.120.450
2qe9	1.20.120.450
2RD9	1.20.120.450
1M4R	1.20.1250.10
1HGU	1.20.1250.10
1PV6	1.20.1250.20
1AWR	2.40.100.10
1D9C	1.20.1250.10
1NGL	2.40.128.20
1N7V	2.105.10.10
7TIM	3.20.20.70
1NEY	3.20.20.70
2V5L	3.20.20.70
1V3Z	3.30.70.100
1A17	1.25.40.10
1ar0	3.10.450.50

2.5.2 Datensatz 1

Für die ersten Tests wurden verschiedene Proteine entsprechend ihrer CATH-Klassifizeirugn zusammengestellt. Die Proteine wurden so gewählt, dass die 3 CATH-Klassen gleich stark vertreten sind, in denen sich viele Sekundärstrukturelemente befinden. Dementsprechend wurden Proteine aus der CATH-Klasse 4 ignoriert. Diese Klasse enthält nur Proteine mit wenigen Sekundärstrukturelementen.

Es befinden sich je 5 Vertreter der drei CATH-Klassen 1 (*mainly-alpha*), 2 (*mainly-beta*) und 3 (*alpha-beta*) in diesem Datensatz. Diese wurden so gewählt, dass je ein Paar innerhalb der selben Klasse ein direkter struktureller Nachbar mit eine Sequenzidentität von mehr als 95%, sowie ein Nachbar mit einer Sequenzidentität von meher als 35% befindet, um zu überprüfen, ob die *Graphlet*-Vektoren von Aminosäuregraphen eine Korrelation mit der Sequenzidentität aufweisen können. Weiterhin wurde je ein Vertreter der selben Topologie aus einer anderen Superfamilie, sowie der selben Architektur mit anderer Topologie gewählt.

Da die *Graphlet*-Vektoren globale Eigenschaften eines Graphen beschreiben, wurde bei diesem Datensatz darauf geachtet, dass alle Strukturen aus genau einer Polypeptidkette mit genau einer Domäne bestehen.

PDB-ID	CATH	Klassifizierung
1qpu	1.20.120.10	Elektronentransport
1qq3	1.20.120.10	Elektronentransport
1cgn	1.20.120.10	Elektronentransport
1he9	1.20.120.260	Toxin (Exoenzym)
3gf9	1.20.900.10	Endocytose
1exs	2.40.128.20	Lipid bindendes Protein
1ngl	2.40.128.20	Transport Protein
1qqs	2.40.128.20	Zucker bindendes Protein
3slo	2.40.128.130	Protein Transport
1wjx	2.40.280.10	RNA-bindendes Protein
5chy	3.40.50.2300	Signaltransduktion
2id9	3.40.50.2300	Signal Protein
3i42	3.40.50.2300	unbekannte Funktion
1d4o	3.40.50.1220	Oxidoreductase
2w0i	3.40.20.10	Transferase

Vergleich von Proteinen ähnlicher Größe

Da *Graphlets* in den Experimenten von Pruzlj et al und Shervashidze et auf große Graphen angewandt wurden, bietet es sich an, sie auch auf große Proteinkomplexe anzuwenden und bei kleinen Proeinen den Aminsosäuregraphen zu verwenden. So kann man (hoffentlich) das Problem der dünn besetzten Graphen umgehen und für verschiedene Größen von Graphen relevante Ergebnisse erzielen. Dafür muss ein Datensatz aus größen Proteinkomplexen mit bekannter struktureller Ähnlichkeit zusammengestellt werden.

Vergleich nach EC-Termen

EC-Terme beschreiben die enzymatische Aktivität von Biomolekülen. Analog zu GO-Termen ermöglichen sie eine Klassfizierung von Biomolekülen entsprechend iherer biolgoischen Funktion, ohne jedoch den gesamten Prozess, oder das Zellorganell, in den das Molekül involviert ist, zu beschreiben. Dem Dogma folgend, dass die Struktur eines Proteins seine Funktion bestimmt, liegt die Vermutung nahe, dass Proteine mit ähnlicher Funktion auch eine ähnliche Struktur besitzen müssen. Der Vergleich anhand von EC-Termen ist interessant, weil strukturell ähnliche Domänen unterschiedliche enzymatische Aktivitäten besitzen können. Wenn Proteine mit gleicher Domäne und unterschiedlichen enzymatischen Aktivitäten richtig eingeordnet werden, spricht das für die Genauigkeit des Algorithmus. Es bietet sich ein Vergleich von Mannose-6-phosphat Isomerasen (MPhI) und 5-dehydro-4-deoxy-D-glucuronat Isomerasen (ddGI) an. Es gibt Enzyme mit diesen Funktionen in diversen CATH-Superfamilien. Sie sind interessant für die Analyse, weil sich Vertreter beider Enzymklassen teilweise die selbe Superfamilie teilen, während es gleichzeitig Superfamilien gibt, in denen nur eine der beiden enzymatischen Funktionen vorkommt.

PDB-ID	Chain	CATH-Code	EC-Code
1PMI	A	2.60.120.10	5.3.1.8
1X8M	A02	2.60.120.10	5.3.1.17
1YWK	????	nicht da	5.3.1.17

Kapitel 3

Ergebnisse

3.1 Fallstudien - Datensatz 1

Die paarweisen RGF-Distanzen der Proteine befinden sich in den Tabellen 5.2 5.2 und 5.2. Die paarweisen Tanimoto-Koeffizienten der Strukturen befinden sich in den Tabellen 5.2, 5.2 und 5.2. Hierbei wurden die Zellen, die die 4 besten Bewertungen für das Protein der entsprechenden Zeile enthalten grün eingefärbt. Hierbei gilt, dass das Grün umso dunkler ist, je stärker die Ähnlichkeit bewertet wird.

Der Vergleich mittels RGF zeigt für die Aminosäuregraphen die stärksten Ähnlichkeiten immer innerhalb der entsprechenden CATH-Klassen. Sowohl die Proteine aus der *mainly-alpha*-Klasse, als auch die Proteine der *mainly-beta*-Klasse haben die besten Ähnlichkeitswerte mit Proteinen der gleichen Klasse. Innerhalb der Klasse der *alpha-beta*-Proteine, gibt es mit 2id9 und 1d4o zwei Proteine, denen eine größere Ähnlichkeit zu Proteinen der *mainly-alpha*-Klasse attestiert wurde. Bei 1d4o fällt auf, dass der niedrigste Wert mit 7,091 deutlich höher ist, als die besten Werte aller anderen Proteine. 2id9 hat laut der RGF-Distanz die größte Ähnlichkeit zu 1he9. Bis auf diese beiden Außnahmen lässt sich jedoch eine große starke Korrelation mit den CATH-Klassen erkennen. Alle anderen Proteine haben mindestens die zwei kleinsten zwei RGF-Distanzen zu Vertretern aus der selben CATH-Klasse.

Dies gilt für die Aminosäuregraphen. Die RGF-Distanzen der Proteingraphen zueinander zeigen ein weniger klares Bild. Für die *mainly-alpha*-Klasse und die der *mainly-beta*-Klasse befinden sich die Proteine mit den kürzesten Distanzen immer noch in der selben Klasse. Dies lässt sich für die Proteine der *alpha-beta*-Klasse aber nicht mehr behaupten. Hier haben 2ID9, 3I42 und 2w0I die kürzesten Distanzen zu Proteinen anderer Klassen.

Bei den Komplexgraphen ist die Korrelation zwischen der RGF-Distanz und der Zugehörigkeit zur CATH-Klasse noch geringer. Die Tabelle 5.2 zeigt nur für die Proteine 1QQ3, 1HE9, 1EXS und 1QQS die kürzeste Distanz zu einem

Vertreter der gleichen Klasse. Es fällt jedoch auf, dass besonders häufig die Proteine der *alpha-beta*-Klasse 5CHY, 2ID9 und 3I42 als ähnlich zu anderen bewertet werden.

Der Vergleich der Tanimoto-Koeffizienten zeigt ein ähnliches Bild, wie der Vergleich der RGF-Distanzen. Bei den Aminosäuregraphen zeigt sich, dass innerhalb der *mainly-alpha*-Klasse wieder die paarweisen Ähnlichkeiten der *mainly-alpha*-Proteine am größten sind. Dies gilt bis auf eine Ausnahme auch für die *mainly-beta*-Proteine. Das Protein mit der PDB-ID 1NGL wird als strukturell ähnlichstes Protein zu 2W0I bewertet. In der Klasse der *alpha-beta*-Proteine gibt es mit 2ID9 und 1D4O wieder zwei Ausreißer, die die größten paarweisen Ähnlichkeiten nicht zu Vertretern der eigenen Klasse haben. Für 2ID9 wird 1HE9 als ähnlichstes Protein angegeben und 1D4O wird 1QQS zugeordnet.

Die paarweisen Tanimoto-Koeffizienten der Proteingraphen zeigen - wie schon bei den RGF-Distanzen - eine geringere Korrelation mit der Zugehörigkeit zu den CATH-Klassen, als die Koeffizienten der Aminosäuregraphen. Es haben zwar wieder mindestens 3 Vertreter jeder Klasse ihren nächsten Nachbarn in der gleichen Klasse, aber es gibt auch einige Proteine, die ihren nächsten Nachbarn außerhalb der eigenen Klasse haben. Hierzu gehören 1D4O, 2W0I (beide *alpha-beta*) und 3GF9 (*mainly-alpha*). Es fällt auf, dass wieder die Proteine mit den PDB-DIs 2ID9 und 3I42 besonders häufig als strukturell ähnlich zu vielen anderen Proteinen bewertet werden.

Für die KOmplexgraphen zeigt die Tabelle wieder eine hohe Ähnlichkeite unter den ersten 3 Proteinen 1QPU, 1QQ3 und 1CGN. Auch innerhalb der Klasse *alpha-beta* sind 3 Proteine mit der höchsten paarweisen Ähnlichkeit bewertet worden. Die geringe Anzahl von stark bewerteten Ähnlichkeiten innerhalb der *mainly-beta*-Klasse ist sehr auffällig. 1QQS und 3SLO sind das einzige Paar mit *beta*-Topologie, dessen Ähnlichkeit als groß bewertet wurde.

Kapitel 4

Diskussion und Ausblick

4.1 Diskussion

4.1.1 Datensatz 1

4.1.2 Datensatz 2

4.2 Ausblick

4.2.1 Verbesserung des Scoring

4.2.2 Optimierung der Laufzeit von `graphletAnalyser`

4.2.3 Ähnlichkeitssuche

4.2.4 *Graphlets* im Faltungsraum

Es bieten sich Paper an, die in Liisa Holm *Advances and Pitfalls of Protein Structure Prediction* besprochen wurden. Das Problem ist, dass `graphletAnalyser` Vektoren erzeugt - also eindimensionale Daten. Somit eignen sich Methoden, die 2D- oder 3D-Daten miteinander alignieren schlecht zum Vergleich. *Holm* erwähnt in ihrer Arbeit aber auch Autoren, die Vergleiche von eindimensionalen Daten durchgeführt haben. Dies sind folgende:

Liu X, Zhao YP, Zheng WM: CLEMAPS: multiple alignment of protein structures based on conformational letters

Sierk ML, Pearson WR: Sensitivity and selectivity in protein structure comparison

Sacan A, Toroslu IH, Ferhatosmanoglu H: Integrated search and alignment of protein structures

Friedberg I, Harder T, Kolodny R et al. Using an alignment of fragment strings for comparing protein structures

Tung CH, Huan JW, Yang JM: Kappa-alpha plot derived structural alphabet and BLOSUM-like substitution matrix for rapid search of protein structure

database

Yang JA: Comprehensive description of protein structures using protein shape code

Kapitel 5

Anhang

5.1 Bildverzeichnis

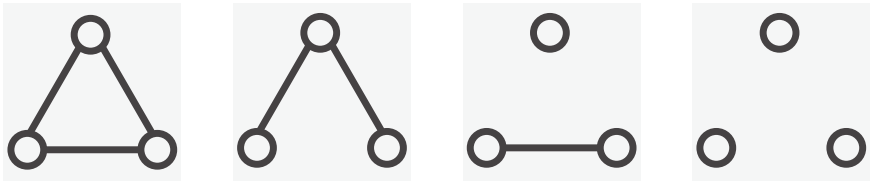

Abbildung 5.1: Graphlets der Größe 3 (*Shervashidze et al.*)

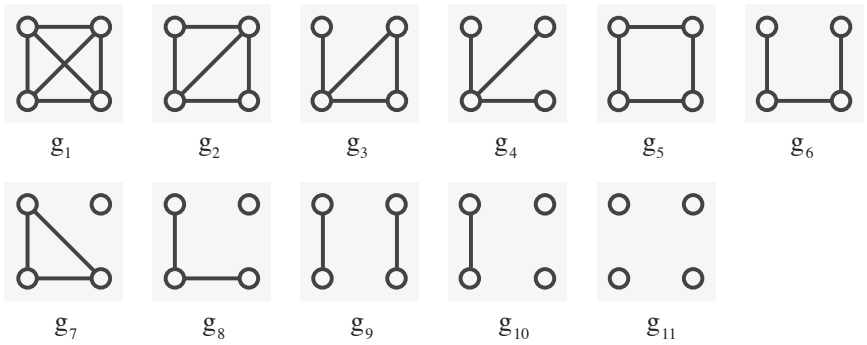

Abbildung 5.2: Graphlets der Größe 4 (*Shervashidze et al.*)

5.2 Tabellenverzeichnis

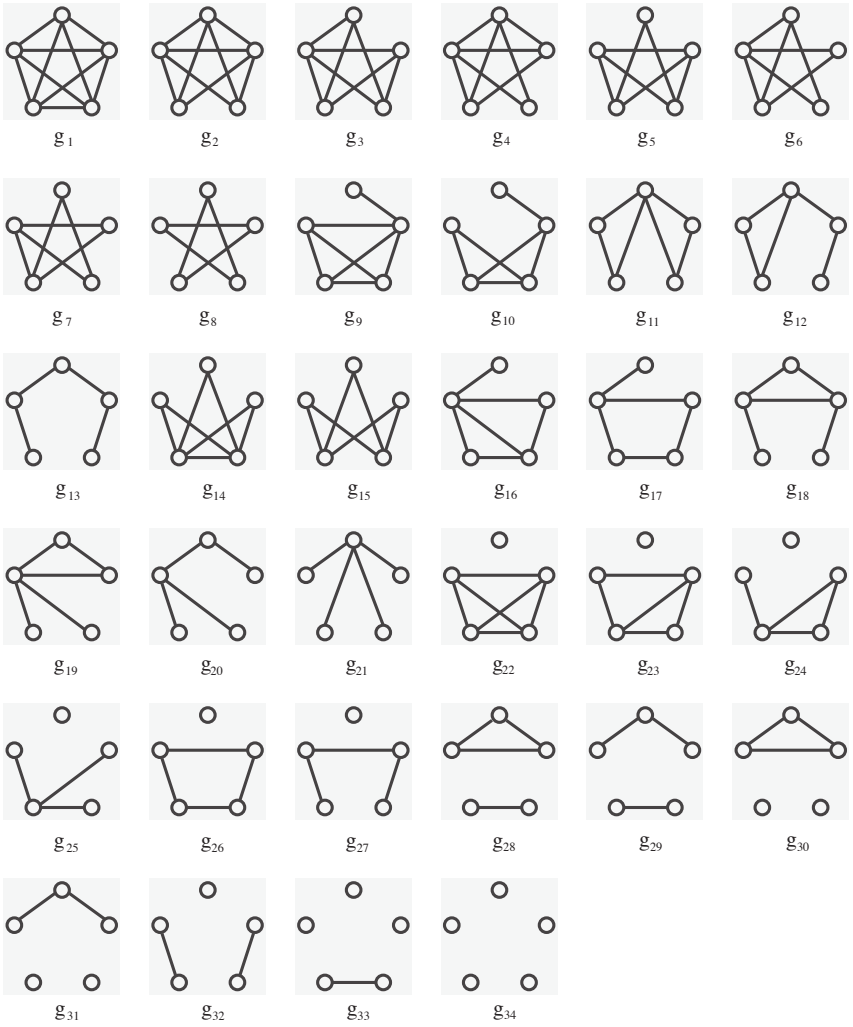

Abbildung 5.3: Graphlets der Größe 5 (*Shervashidze et al.*)

PDB-ID	2utg	1vib	1qpu	1qq3	2xl6	2yl1	1rxq	2qe9	2rd9	1m4r	1hgu	1pv6	1awr	1d9c	1ngl	1n7v	7tim	1ney	1v3z	1a17	1ar0	2v5l
2utg	X	8.915	8.626	8.150	9.759	13.62	6.408	8.128	7.384	5.158	4.944	3.570	11.67	4.392	7.387	13.02	9.171	8.737	8.025	2.490	9.329	6.253
1vib	8.915	X	15.54	15.89	16.89	21.08	15.22	16.80	16.23	13.39	13.26	12.32	17.13	12.55	11.80	17.87	17.44	17.07	13.78	7.943	16.33	14.27
1qpu	8.626	15.54	X	1.451	3.655	5.632	6.272	8.187	7.539	5.410	7.849	7.087	11.01	7.053	9.864	12.54	8.261	8.829	8.822	9.192	9.692	6.462
1qq3	8.150	15.89	1.451	X	3.614	5.651	6.064	7.554	7.252	5.087	7.413	6.776	11.02	6.721	9.918	12.56	7.787	8.354	8.892	9.531	9.216	6.019
2xl6	9.759	16.89	3.655	3.614	X	4.518	5.106	5.523	5.482	5.514	8.147	8.655	8.374	6.685	8.091	9.919	5.950	6.459	6.255	10.30	7.054	4.716
2yl1	13.62	21.08	5.632	5.651	4.518	X	7.737	7.197	7.589	8.855	11.36	11.82	11.53	10.34	12.29	12.94	8.343	8.923	10.59	14.75	10.14	8.137
1rxq	6.408	15.22	6.272	6.064	5.106	7.737	X	2.271	1.563	1.846	4.457	4.428	8.105	2.685	6.737	10.11	4.059	4.769	5.299	8.066	5.910	3.043
2qe9	8.128	16.80	8.187	7.554	5.523	7.197	2.271	X	2.024	4.006	6.167	5.959	8.708	4.649	8.068	10.83	4.470	5.180	6.065	9.575	6.728	4.542
2rd9	7.384	16.23	7.539	7.252	5.482	7.589	1.563	2.024	X	3.205	5.325	5.829	7.137	3.853	6.677	9.126	3.049	3.785	4.747	9.046	5.086	3.123
1m4r	5.158	13.39	5.410	5.087	5.514	8.855	1.846	4.006	3.205	X	3.657	3.784	8.468	1.837	6.243	10.41	5.063	5.650	5.068	6.336	5.972	2.922
1hgu	4.944	13.26	7.849	7.413	8.147	11.36	4.457	6.167	5.325	3.657	X	4.692	9.337	3.187	5.776	10.59	6.059	5.603	6.003	7.045	6.636	3.742
1pv6	3.570	12.32	7.087	6.776	8.655	11.82	4.428	5.959	5.829	3.784	4.692	X	12.09	4.568	8.081	13.67	7.678	8.304	8.510	4.926	9.202	5.753
1awr	11.67	17.13	11.01	11.02	8.374	11.53	8.105	8.708	7.137	8.468	9.337	12.09	X	7.762	7.577	2.196	4.751	4.070	3.972	11.87	3.395	6.506
1d9c	4.392	12.55	7.053	6.721	6.685	10.34	2.685	4.649	3.853	1.837	3.187	4.568	7.762	X	6.366	9.743	5.605	5.246	4.468	5.626	6.116	3.637
1ngl	7.387	11.80	9.864	9.918	8.091	12.29	6.737	8.068	6.677	6.243	5.776	8.081	7.577	6.366	X	8.064	6.805	6.711	4.658	8.473	5.325	4.582
1n7v	13.02	17.87	12.54	12.56	9.919	12.94	10.11	10.83	9.126	10.41	10.59	13.67	2.196	9.743	8.064	X	6.405	5.675	5.484	13.53	4.509	7.987
7tim	9.171	17.44	8.261	7.787	5.950	8.343	4.059	4.470	3.049	5.063	6.059	7.678	4.751	5.605	6.805	6.405	X	0.919	3.869	11.02	2.858	3.619
1ney	8.737	17.07	8.829	8.354	6.459	8.923	4.769	5.180	3.785	5.650	5.603	8.304	4.070	5.246	6.711	5.675	0.919	X	3.500	10.59	2.683	3.646
1v3z	8.025	13.78	8.822	8.892	6.255	10.59	5.299	6.065	4.747	5.068	6.003	8.510	3.972	4.468	4.658	5.484	3.869	3.500	X	8.286	2.693	3.435
1a17	2.490	7.943	9.192	9.531	10.30	14.75	8.066	9.575	9.046	6.336	7.045	4.926	11.87	5.626	8.473	13.53	11.02	10.59	8.286	X	10.56	7.612
1ar0	9.329	16.33	9.692	9.216	7.054	10.14	5.910	6.728	5.086	5.972	6.636	9.202	3.395	6.116	5.325	4.509	2.858	2.683	2.693	10.56	X	3.679
2v5l	6.253	14.27	6.462	6.019	4.716	8.137	3.043	4.542	3.123	2.922	3.742	5.753	6.506	3.637	4.582	7.987	3.619	3.646	3.435	7.612	3.679	X

PDB-ID	1qpu	1qq3	1cgn	1he9	3gf9	1exs	1ngl	1qqs	3slo	1wjx	5chy	2id9	3i42	1d4o	2w0i
1qpu	X	1.139	1.296	6.159	5.656	10.52	11.00	12.35	12.45	11.66	7.517	6.128	6.625	7.326	8.572
1qq3	1.139	X	1.310	5.980	5.645	9.964	10.38	11.61	11.84	11.10	6.967	5.776	6.039	7.491	8.059
1cgn	1.296	1.310	X	6.093	5.737	9.987	10.44	11.55	11.81	11.16	6.801	5.663	5.929	7.091	7.918
1he9	6.159	5.980	6.093	X	2.289	8.032	6.813	11.03	10.87	8.405	3.974	2.445	3.452	10.93	4.807
3gf9	5.656	5.645	5.737	2.289	X	9.544	8.479	12.36	12.19	10.03	5.135	3.370	4.359	12.08	6.018
1exs	10.52	9.964	9.987	8.032	9.544	X	4.463	4.041	3.138	2.802	4.773	7.086	5.541	7.374	4.530
1ngl	11.00	10.38	10.44	6.813	8.479	4.463	X	7.716	6.865	3.764	4.718	6.054	5.065	9.755	4.031
1qqs	12.35	11.61	11.55	11.03	12.36	4.041	7.716	X	1.674	5.340	8.148	10.20	8.612	7.226	7.404
3slo	12.45	11.84	11.81	10.87	12.19	3.138	6.865	1.674	X	4.750	7.488	9.994	8.600	7.680	6.997
1wjx	11.66	11.10	11.16	8.405	10.03	2.802	3.764	5.340	4.750	X	5.264	7.522	5.699	8.450	4.384
5chy	7.517	6.967	6.801	3.974	5.135	4.773	4.718	8.148	7.488	5.264	X	2.600	2.817	8.667	1.497
2id9	6.128	5.776	5.663	2.445	3.370	7.086	6.054	10.20	9.994	7.522	2.600	X	2.447	10.24	3.657
3i42	6.625	6.039	5.929	3.452	4.359	5.541	5.065	8.612	8.600	5.699	2.817	2.447	X	9.544	2.817
1d4o	7.326	7.491	7.091	10.93	12.08	7.374	9.755	7.226	7.680	8.450	8.667	10.24	9.544	X	8.970
2w0i	8.572	8.059	7.918	4.807	6.018	4.530	4.031	7.404	6.997	4.384	1.497	3.657	2.817	8.970	X

Tabelle 5.1: Distanzen der Aminosäuregraphen, gemessen mit RGF

PDB-ID	1qpu	1qq3	1cgn	1he9	3gf9	1exs	1ngl	1qqs	3slo	1wjx	5chy	2id9	3i42	1d4o	2w0i
1qpu	X	0.806	0.606	0.697	3.131	1.159	2.195	2.197	1.226	1.049	1.014	0.811	1.098	0.628	0.972
1qq3	0.806	X	0.405	1.504	1.567	1.703	0.883	0.883	1.396	1.270	0.869	0.405	0.693	0.988	1.779
1cgn	0.606	0.405	X	0.810	1.432	1.633	1.193	1.193	1.513	1.339	1.060	0.811	1.098	1.011	1.516
1he9	0.697	1.504	0.810	X	4.492	3.253	1.386	1.386	4.357	1.291	1.135	1.099	1.386	3.599	2.506
3gf9	3.131	1.567	1.432	4.492	X	2.523	2.815	2.817	1.596	2.056	1.816	1.917	2.034	3.122	4.130
1exs	1.159	1.703	1.633	3.253	2.523	X	1.324	1.324	2.968	0.696	0.885	1.492	1.610	3.102	3.820
1ngl	2.195	0.883	1.193	1.386	2.815	1.324	X	0.002	0.787	2.629	2.279	0.788	0.286	0.980	2.722
1qqs	2.197	0.883	1.193	1.386	2.817	1.324	0.002	X	0.787	2.631	2.281	0.788	0.285	0.980	2.722
3slo	1.226	1.396	1.513	4.357	1.596	2.968	0.787	0.787	X	0.257	0.705	1.024	1.073	3.137	6.154
1wjx	1.049	1.270	1.339	1.291	2.056	0.696	2.629	2.631	0.257	X	1.043	0.933	0.914	0.644	2.093
5chy	1.014	0.869	1.060	1.135	1.816	0.885	2.279	2.281	0.705	1.043	X	0.607	0.800	0.275	2.282
2id9	0.811	0.405	0.811	1.099	1.917	1.492	0.788	0.788	1.024	0.933	0.607	X	0.692	0.783	2.890
3i42	1.098	0.693	1.098	1.386	2.034	1.610	0.286	0.285	1.073	0.914	0.800	0.692	X	1.076	3.008
1d4o	0.628	0.988	1.011	3.599	3.122	3.102	0.980	0.980	3.137	0.644	0.275	0.783	1.076	X	4.406
2w0i	0.972	1.779	1.516	2.506	4.130	3.820	2.722	2.722	6.154	2.093	2.282	2.890	3.008	4.406	X

Tabelle 5.2: Distanzen der Proteingraphen, gemessen mit RGF

PDB-ID	1qpu	1qq3	1cgn	1he9	3gf9	1exs	1ngl	1qqs	3slo	1wjx	5chy	2id9	3i42	1d4o	2w0i
1qpu	X	4.405	5.056	2.271	10.75	3.567	2.309	13.66	14.10	2.128	1.933	2.165	1.877	8.791	4.422
1qq3	4.405	X	1.906	1.714	10.03	3.860	3.457	6.851	12.70	2.587	2.431	2.557	2.269	7.907	3.002
1cgn	5.056	1.906	X	0.985	9.469	1.712	1.979	7.333	12.20	1.954	1.120	0.249	0.037	6.501	0.980
1he9	2.271	1.714	0.985	X	5.881	3.456	1.386	4.069	5.432	1.466	1.291	1.099	1.386	9.950	2.506
3gf9	10.75	10.03	9.469	5.881	X	5.473	4.230	14.86	15.99	2.516	1.971	2.575	2.693	8.811	11.61
1exs	3.567	3.860	1.712	3.456	5.473	X	1.290	3.217	4.391	0.611	0.662	1.459	1.576	9.191	4.226
1ngl	2.309	3.457	1.979	1.386	4.230	1.290	X	3.637	2.620	2.697	2.644	0.788	0.286	4.373	2.722
1qqs	13.66	6.851	7.333	4.069	14.86	3.217	3.637	X	12.90	5.645	4.202	3.394	3.512	16.22	4.907
3slo	14.10	12.70	12.20	5.432	15.99	4.391	2.620	12.90	X	4.974	3.863	2.351	2.412	14.09	7.116
1wjx	2.128	2.587	1.954	1.466	2.516	0.611	2.697	5.645	4.974	X	0.360	0.961	0.965	2.225	2.042
5chy	1.933	2.431	1.120	1.291	1.971	0.662	2.644	4.202	3.863	0.360	X	0.797	0.914	1.728	2.093
2id9	2.165	2.557	0.249	1.099	2.575	1.459	0.788	3.394	2.351	0.961	0.797	X	0.692	1.921	2.890
3i42	1.877	2.269	0.037	1.386	2.693	1.576	0.286	3.512	2.412	0.965	0.914	0.692	X	2.039	3.008
1d4o	8.791	7.907	6.501	9.950	8.811	9.191	4.373	16.22	14.09	2.225	1.728	1.921	2.039	X	10.76
2w0i	4.422	3.002	0.980	2.506	11.61	4.226	2.722	4.907	7.116	2.042	2.093	2.890	3.008	10.76	X

Tabelle 5.3: Distanzen der Komplexgraphen, gemessen mit RGF

PDB-ID	1qpu	1qq3	1cgn	1he9	3gf9	1exs	1ngl	1qqs	3slo	1wjx	5chy	2id9	3i42	1d4o	2w0i
1qpu	X	1.0	1.0	0.621	0.666	0.276	0.25	0.25	0.224	0.25	0.538	0.714	0.5	0.363	0.333
1qq3	1.0	X	1.0	0.621	0.621	0.304	0.276	0.304	0.224	0.25	0.578	0.666	0.666	0.428	0.463
1cgn	1.0	1.0	X	0.621	0.714	0.333	0.224	0.224	0.224	0.276	0.5	0.666	0.578	0.333	0.428
1he9	0.621	0.621	0.621	X	0.935	0.463	0.428	0.176	0.276	0.304	0.621	0.875	0.764	0.333	0.463
3gf9	0.666	0.621	0.714	0.935	X	0.276	0.333	0.2	0.2	0.224	0.621	0.764	0.621	0.25	0.463
1exs	0.276	0.304	0.333	0.463	0.276	X	0.621	0.621	0.764	0.875	0.666	0.5	0.578	0.463	0.818
1ngl	0.25	0.276	0.224	0.428	0.333	0.621	X	0.276	0.428	0.666	0.621	0.538	0.578	0.463	0.666
1qqs	0.25	0.304	0.224	0.176	0.2	0.621	0.276	X	1.0	0.621	0.5	0.276	0.395	0.463	0.463
3slo	0.224	0.224	0.224	0.276	0.2	0.764	0.428	1.0	X	0.538	0.463	0.333	0.395	0.463	0.5
1wjx	0.25	0.25	0.276	0.304	0.224	0.875	0.666	0.621	0.538	X	0.538	0.363	0.5	0.395	0.621
5chy	0.538	0.578	0.5	0.621	0.621	0.666	0.621	0.5	0.463	0.538	X	0.818	0.764	0.428	1.0
2id9	0.714	0.666	0.666	0.875	0.764	0.5	0.538	0.276	0.333	0.363	0.818	X	0.818	0.304	0.621
3i42	0.5	0.666	0.578	0.764	0.621	0.578	0.578	0.395	0.395	0.5	0.764	0.818	X	0.333	0.714
1d4o	0.363	0.428	0.333	0.333	0.25	0.463	0.463	0.463	0.463	0.395	0.428	0.304	0.333	X	0.333
2w0i	0.333	0.463	0.428	0.463	0.463	0.818	0.666	0.463	0.5	0.621	1.0	0.621	0.714	0.333	X

Tabelle 5.4: Tanimoto-Koeffizienten der Aminosäuregraphen

PDB-ID	1qpu	1qq3	1cgn	1he9	3gf9	1exs	1ngl	1qqs	3slo	1wjx	5chy	2id9	3i42	1d4o	2w0i
1qpu	X	0.764	0.764	0.578	0.463	0.463	0.621	0.621	0.428	0.578	0.666	0.666	0.666	0.538	0.395
1qq3	0.764	X	0.935	0.764	0.463	0.621	0.714	0.714	0.538	0.714	0.764	0.875	0.818	0.666	0.5
1cgn	0.764	0.935	X	0.764	0.428	0.578	0.714	0.714	0.538	0.666	0.714	0.875	0.818	0.621	0.463
1he9	0.578	0.764	0.764	X	0.463	0.578	0.621	0.621	0.578	0.538	0.621	0.714	0.714	0.666	0.538
3gf9	0.463	0.463	0.428	0.463	X	0.428	0.463	0.463	0.463	0.463	0.538	0.428	0.463	0.5	0.333
1exs	0.463	0.621	0.578	0.578	0.428	X	0.5	0.5	0.714	0.5	0.578	0.666	0.578	0.714	0.621
1ngl	0.621	0.714	0.714	0.621	0.463	0.5	X	1.0	0.538	0.818	0.818	0.818	0.875	0.578	0.395
1qqs	0.621	0.714	0.714	0.621	0.463	0.5	1.0	X	0.538	0.818	0.818	0.818	0.875	0.578	0.395
3slo	0.428	0.538	0.538	0.578	0.463	0.714	0.538	0.538	X	0.538	0.5	0.578	0.621	0.578	0.463
1wjx	0.578	0.714	0.666	0.538	0.463	0.5	0.818	0.818	0.538	X	0.764	0.714	0.818	0.538	0.395
5chy	0.666	0.764	0.714	0.621	0.538	0.578	0.818	0.818	0.5	0.764	X	0.875	0.818	0.621	0.395
2id9	0.666	0.875	0.875	0.714	0.428	0.666	0.818	0.818	0.578	0.714	0.875	X	0.935	0.714	0.463
3i42	0.666	0.818	0.818	0.714	0.463	0.578	0.875	0.875	0.621	0.818	0.818	0.935	X	0.666	0.463
1d4o	0.538	0.666	0.621	0.666	0.5	0.714	0.578	0.578	0.578	0.538	0.621	0.714	0.666	X	0.5
2w0i	0.395	0.5	0.463	0.538	0.333	0.621	0.395	0.395	0.463	0.395	0.395	0.463	0.463	0.5	X

Tabelle 5.5: Tanimoto-Koeffizienten der Proteingraphen

PDB-ID	1qpu	1qq3	1cgn	1he9	3gf9	1exs	1ngl	1qqs	3slo	1wjx	5chy	2id9	3i42	1d4o	2w0i
1qpu	X	0.428	0.395	0.428	0.153	0.304	0.395	0.132	0.111	0.363	0.395	0.428	0.428	0.276	0.276
1qq3	0.428	X	0.666	0.621	0.2	0.395	0.5	0.2	0.090	0.5	0.5	0.538	0.538	0.333	0.428
1cgn	0.395	0.666	X	0.621	0.224	0.428	0.5	0.224	0.111	0.463	0.5	0.538	0.538	0.363	0.363
1he9	0.428	0.621	0.621	X	0.25	0.578	0.621	0.153	0.090	0.538	0.621	0.714	0.714	0.428	0.538
3gf9	0.153	0.2	0.224	0.25	X	0.276	0.224	0.153	0.153	0.276	0.304	0.25	0.224	0.276	0.224
1exs	0.304	0.395	0.428	0.578	0.276	X	0.5	0.2	0.153	0.578	0.578	0.666	0.578	0.5	0.621
1ngl	0.395	0.5	0.5	0.621	0.224	0.5	X	0.132	0.132	0.818	0.818	0.818	0.875	0.428	0.395
1qqs	0.132	0.2	0.224	0.153	0.153	0.2	0.132	X	0.224	0.132	0.132	0.132	0.132	0.132	0.2
3slo	0.111	0.090	0.111	0.090	0.153	0.153	0.132	0.224	X	0.111	0.090	0.090	0.111	0.111	0.153
1wjx	0.363	0.5	0.463	0.538	0.276	0.578	0.818	0.132	0.111	X	0.935	0.714	0.818	0.5	0.395
5chy	0.395	0.5	0.5	0.621	0.304	0.578	0.818	0.132	0.090	0.935	X	0.875	0.818	0.5	0.395
2id9	0.428	0.538	0.538	0.714	0.25	0.666	0.818	0.132	0.090	0.714	0.875	X	0.935	0.5	0.463
3i42	0.428	0.538	0.538	0.714	0.224	0.578	0.875	0.132	0.111	0.818	0.818	0.935	X	0.428	0.463
1d4o	0.276	0.333	0.363	0.428	0.276	0.5	0.428	0.132	0.111	0.5	0.5	0.5	0.428	X	0.333
2w0i	0.276	0.428	0.363	0.538	0.224	0.621	0.395	0.2	0.153	0.395	0.395	0.463	0.463	0.333	X

Tabelle 5.6: Tanimoto-Koeffizienten der Komplexgraphen

Literaturverzeichnis

[1] Hasegawa et al. Advances and pitfalls of protein structural alignment. *Current Opinion in Structural Biology*, 2009.

[2] May et al. Ptgl: a database for secondary structure-based protein topologies. *Nucleic Acids Research*, 2010.

[3] Przůlj et al. Modelling interactome: scale-free or geometric. *Bioinformatics*, 2004.

[4] Chris Sander Liisa Holm. Dali: a network tool for protein structure comparison. *Trends in Biomedical Sciences*, 1995.

[5] Tim Schäfer, Patrick May, and Ina Koch. Computation and Visualization of Protein Topology Graphs Including Ligand Information. In Sebastian Böcker, Franziska Hufsky, Kerstin Scheubert, Jana Schleicher, and Stefan 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 fuer Informatik.

[6] Adam Godzik Yuzhen Ye. Flexible structure alignment by chaining aligned fragment pairs allowing twists. *Bioinformatics*, 2003.