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

Ben Haladik

18. Januar 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 wurde die Strukturen von über 100000 biologischen Makromolekülen in der PDB. Der Vergleich dieser gigantischen Anzahl von Strukturdaten erfordert effiziente algorithmische Analysemethoden. Solche Analysen können tiefe Einblicke in die ferne evolutionäre Verwandschaft von Proteinen liefern und sie helfen bei der Bestimmung der Funktion eines Proteins. Des weiteren sind sie 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* [5]verwendet beispielsweise Distanzmatrizen, die die Abstände einzelner Residuen zueinander speichern, anstatt die Koordinaten jedes Atoms abzuspeichern. 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.

1.3 Ziele

TODO: Zitationen einfügen

Ziel der Arbeit war festzustellen, ob sich *Graphlets* eignen, um Ähnlichkeiten von Proteinstrukturen festzustellen. In den Arbeiten von *Hasegawa et al*, *Pržulj et al* und *Shervashidze* wurden *Graphlets* bereits erfolgreich angewandt, um die Ähnlichkeiten von Netzwerken zu bestimmen. Ihre analysen wurden hauptsächlich für sehr große netzwerke, wie PPI-Graphen und (siehe TODO, arbeit von Shervashidze) durchgeführt. Somit blieb die Frage offen, ob sich *Graphlets* auch zur analyse von Netzwerken eigenen, die deutlich kleiner sind und deutlich niedrigere Knotengrade haben. Weiterhin galt es, herauszufinden, welche Metriken sich am besten eignen, um die Unterschiede zwischen verschiedenen Topologien zu befiffern. Hiermit könnte schließlich eine Ähnlichkeitssuche in der PTGL implementiert werden.

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. Weiterhin wird das Programm`graphletAnalyser`vorgestellt, welches den *Graphlet*-Algorithmus implementiert. Es werden verschiedene Metriken vorgestellt und evaluiert, mit denen die erhaltenen *Graphlet*-Vektoren verglichen werden. Abschließend folgt eine kurze Beschreibung des FATCAT-SCOP-*Benchmarking*-Datensatzes und der für die Fallstudien ausgewählten Proteine. Die dabei erhaltenen Ergebnisse werden im gleichnamigen folgenden Kapitel

vorgestellt. Eine Diskussion der Ergebnisse und ein Ausblick auf weitere Anwendungen und Verbesserungsmöglichkeiten erfolgt im letzten Kapitel.

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 PLCC

PLCC ist das Programm, das die Graphen der PTGL erstellt. Hierfür benötigt es die PDB und DSSP Dateien des zu modellierenden Proteins. Der DSSP-Algorithmus weist den Residuen aus der PDB-Datei Sekundärstrukturen zu. Diese werden in den PTGL-Graphen als Knoten modelliert. Um die räumliche Nähe der Sekundärstrukturen zueinander festzustellen werden die Koordinaten aus der PDB-Datei genutzt. Wenn zwei Sekundärstrukturen räumlich benachbart sind, werden die entsprechenden Knoten durch eine Kante verbunden.

2.2 Die PTGL

Die Protein Topology Graph Library ist 2009 von *May et al.* [2] entwickelt worden. Sie stellt ein System zur Klassifizierung von 3D Proteinstrukturen zur Verfügung. Es gibt berei

2.3 Der *Graphlet*-Algorithmus

Motivation Die PTGL [4] 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 könen dann als Datenpunkte verglichen werden und man spart sich die Berechnungen, die man sonst für alle Paare von Graphen G, G' durchführen muss. Als Charakteristika eignen sich beispielsweise induzierte Teilgraphen

Beschreibung des Algorithmus *Graphlets* sind kleine induzierte Teilgraphen eines größeren ungerichteten Graphen. *N. Shervashidze* stellte diese Methode als Vergleichsschema für Graphen 2009 zum ersten Mal vor. (Literaturverweis einfügen). Folgendes Bild zeigt alle *Graphlets* der Größe 4:

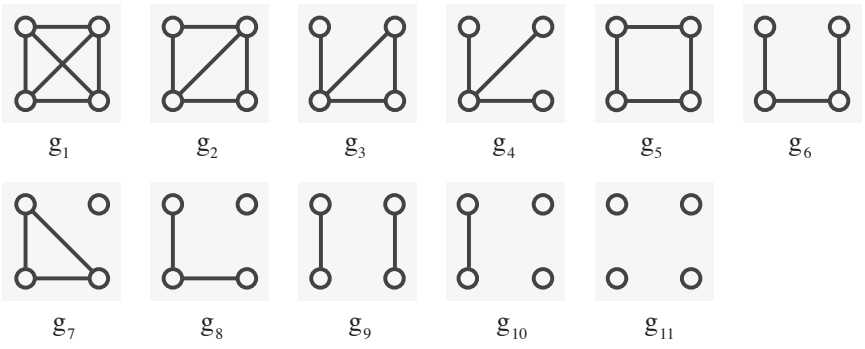

Abbildung 2.1: Graphlets der Größe 4 (*Shervashidze et al.*)

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?)

Markierte *Graphlets* werden ebenfalls durch den Algorithmus gezählt. In dieser Implementierung zählt der Algorithmus *Graphlets* mit Knotenmarkierungen, die in der Konfigurationsdatei vorgegeben werden können. Es werden alle kombinatorisch möglichen Markierungen von *Graphlets* bis zur Größe 3 betrachtet. Größere markierte *Graphlets* werden ignoriert, um die Laufzeit niedrig zu halten. Da die Markierungen von SSE-Graphen und AA-Graphen betrachtet werden, werden verschiedene Vektoren über verschiedenen Alphabeten verwendet.
Die Vektoren p_2, p_{3-Weg} und $p_{3-Kreis}$ beschreiben die verschiedenen Worte für den 2-Pfad, den 3-Weg und den 3-Kreis in Proteingraphen.

$$p_2 := (HH, HE,$$

(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)

2.4 `graphletAnalyser`

Der `graphletAnalyser` berechnet die oben beschreibenen *Graphlets* nach dem Algorithmus von *Shervashidze et al.*. ZUr Zeit unterstützt er die Berechnung für 3 verschiedene Graphen der PTGL: SSE-Graphen sind Graphen, die eine Polypeptidkette als Graph darstellen indem sie ihre Sekundärstrukturelemente als Knoten repräsentieren und diese mit Kanten verbinden, wenn sie räumlich benachbart sind.
Komplexgraphen sind Graphen für Proteinkomplexe; sie modellieren mehrere Polypeptidketten indem sie mehrere SSE-Graphen miteinander verknüpfen.
Aminosäuregraphen modellieren einzelne Aminosäuren als Knoten und verbinden diese miteinander, wenn sie räumlich benachbart sind. Dementsprechend ein Vielfaches der Knoten und Kanten eines SSE- oder Komplex-Graphen.
Zusätzlich zu den *Graphlets* bis zur Größe 5 berechnet das Programm markierte *Graphlets* mit bis zu 3 Knoten. Dabei werden zunächst alle Markierungen die kombinatorisch möglich sind berechnet und die entsprechenden *Graphlets* werden gezählt.
Das Programm wird über die Kommandozeile aufgerufen und erhält eine oder mehrere GML-Dateien als Argumente. Zusätzlich kann angegeben werden, ob die GML-Datei einen Proteingraphen, Komplexgraphen oder Aminosäuregraphen beschreibt. Alternativ kann auch in der Konfigurationsdatei angegeben werden, dass eine andere Art von Graph eingelesen werden soll. `graphletAnalyser` ist also in der Lage, jede GML-Datei, die einen ungerichteten Graphen enthält einzulesen und für sie *Graphlets* zu berechnen. Der Nutzer kann weiterhin ein Alphabet unterschiedlicher Knotenmarkierungen angeben, für das markierte *Graphlets* berechnet werden. Somit kann das Programm auch für graphenbezogene Fragestellungen verwendet werden, die sich nicht auf Proteine beziehen.
Aus der .gml-Datei wird ein Graph erstellt. Das Programm nutzt hierbei die *Boost-Graph-Library* zur internen Darstellung. Entsprechend der vom Nutzer ausgewählten Funktionen werden die *Graphlets* berechnet und als Datei ausgegeben. Zusätzlich können sie in einer lokalen Datenbank abgespeichert werden.

2.5 Scoring

2.5.1 Tanimoto-Koeffizient

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

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

Er berechnet also die relative Häufigkeit der gemeinsamen Elemente beider Mengen. Damit ist dieses Berechnungsschema auch für Vektoren geeignet.

2.5.2 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 $D(G, H)$. Diese Metrik bestimmt den Unterschied 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 hat den Vorteil, dass sie die normalisierten *Graphlet*-Vektoren verwendet. So ist ein guter Vergleich auch möglich, wenn einzelne *Graphlets* in den Netzwerken überrepräsentiert sind. Weiterhin wurde gezeigt, [3] dass diese Metrik auch bei verrauschten Daten noch sehr gut funktioniert.

2.5.3 *Root-Mean-Square-Deviation*

Die *Root-Mean-Square-Deviation* RMSD ist ein Standardmaß zur Abstandsmessung in Vektorräumen. Sie wird für zwei Vektoren x und y durch folgende Formel berechnet:

$$RMSD(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{2.5a}$$

2.5.4 Anpassung der Parameter

2.6 Datensätze

2.6.1 FATCAT-SCOP-*Benchmarking-Set*

Die FATCAT SCOP DB, beschrieben in: http://bioinformatics.oxfordjournals.org/content/23/2/e219.long könnte sich gut zum Vergleich eignen

Das FATCAT-SCOP-*Benchmarking-Set* ist für das Strukturalignment-Programm FATCAT [6] zusammengestellt worden, um seine Performance mit der von etablierten Strukturalignment-Programmen wie DALI und CE zu vergleichen. Der Datensatz besteht aus 2256 PDB-Einträgen, die bereits durch die SCOP-Datenbank klassifiziert worden sind. In dem Datensatz befinden sich Paare von PDB-Einträgen, die von allen drei Programmen als sehr ähnlich bewertet wurden.

2.6.2 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 Rrotein 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
1M4R	1.20.1250.10
1HGU	1.20.1250.10
1D9C	1.20.1250.10
1PV6	1.20.1250.20
1AWR	2.40.100.10
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

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 PRotein komplexe 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.

Kapitel 3

Ergebnisse

3.1 Fallstudie - vergleich mit Dali

Kapitel 4

Diskussion und Ausblick

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 Tabellenverzeichnis

5.2 Bildverzeichnis

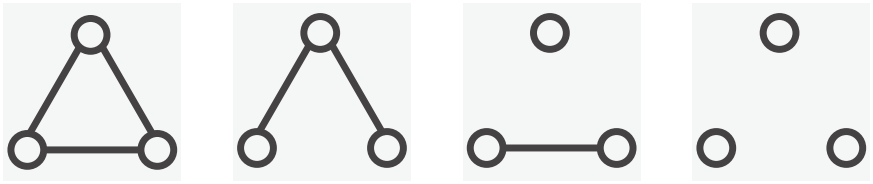

Abbildung 5.1: Graphlets der Größe 3 (*Shervashidze et al.*)

5.3 Literaturverzeichnis

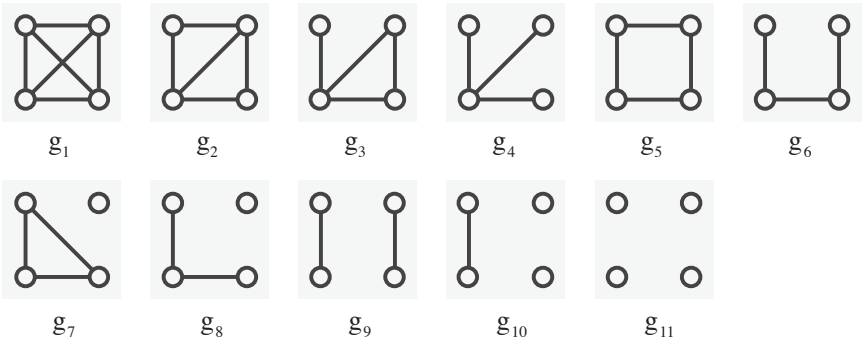

Abbildung 5.2: Graphlets der Größe 4 (*Shervashidze et al.*)

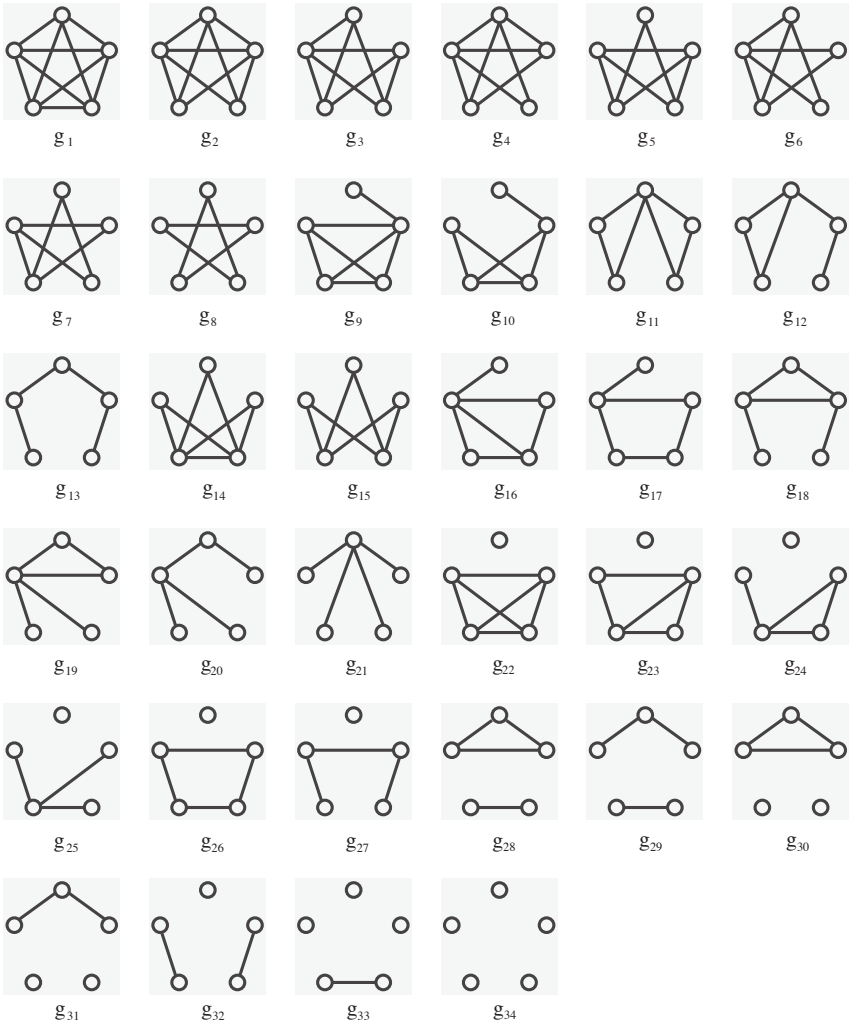

Abbildung 5.3: Graphlets der Größe 5 (*Shervashidze et al.*)

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*, 2009.

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

[4] Schafer et al. Computation and visualization of protein topology graphs including ligand information. *Bioinformatics*, 2015.

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

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