

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

Ben Haladik

1. Februar 2016

Inhaltsverzeichnis

1	Einleitung	2
1.1	Motivation	2
1.2	<i>State of the Art</i>	3
1.3	Ziele	4
1.4	Aufbau der Arbeit	4
2	Materialien und Methoden	6
2.1	PLCC	6
2.2	Die PTGL	6
2.3	Der <i>Graphlet</i> -Algorithmus	6
2.4	<code>graphletAnalyser</code>	9
2.5	Scoring	10
2.5.1	Tanimoto-Koeffizient	10
2.5.2	Relative <i>Graphlet</i> -Häufigkeiten-Distanz	10
2.5.3	<i>Root-Mean-Square-Deviation</i>	10
2.5.4	Anpassung der Parameter	11
2.6	Datensätze	11
2.6.1	FATCAT-SCOP- <i>Benchmarking-Set</i>	11
2.6.2	Fallstudien	11
3	Ergebnisse	15
3.1	Erster kleiner Test	15
3.2	Fallstudie - Vergleich mit CATH-Einordnung	15
3.2.1	AA- <i>Graphlets</i>	15
4	Diskussion und Ausblick	16
5	Anhang	17
5.1	Bildverzeichnis	17
5.2	Tabellenverzeichnis	18

Kapitel 1

Einleitung

1.1 Motivation

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

Die Ähnlichkeit 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 Verwandtschaft 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 liefern Informationen über mögliche 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 darstellen 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 Vergleich 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 entfernt 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*, *Prölj 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 eignen, 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 beiffern. 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 *graphletAnalyse* 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

vorge stellt. 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 Funktionalität wurde erweitert. Hierbei wurden Funktionen zur Analyse von Komplexgraphen, Aminosäuregraphen und den Sekundärstrukturgraphen implementiert. 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 bei

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 $f : (G, G') \rightarrow \mathbb{R}$, die für zwei Graphen G und G' deren Ähnlichkeit zueinander beziffert. Es gibt diverse Möglichkeiten dieses 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 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:

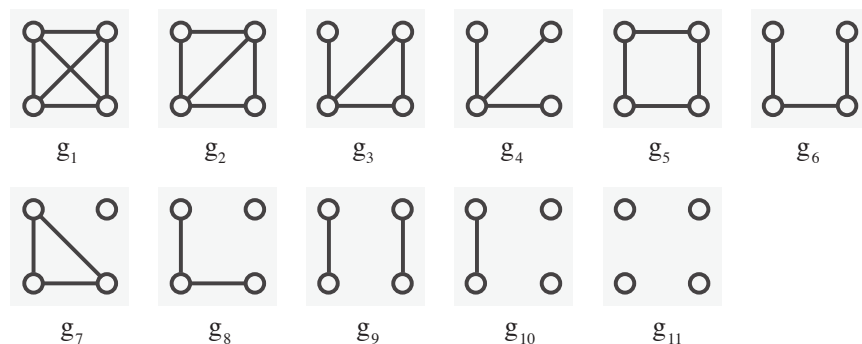


Abbildung 2.1: Graphlets der Größe 4 (*Shervashidze et al.*)

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 gehört. 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

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

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

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

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

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 beschriebenen *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:

$$D_{Tan}(A, B) := \frac{\sum_{x \in A \cap B} 1}{\sum_{x \in A \cup B} 1}$$

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 Zusammenhängen 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 $i \in 1, \dots, 29$ 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 definiert als:

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

$$\text{mit } F_i(G) := -\log\left(\frac{N_i(G)}{T(G)}\right) \quad (2.4b)$$

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:

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

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 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

Vergleich von Proteinen ähnlicher Größe

Da *Graphlets* in den Experimenten von Pruzlj et al und Shervashidze et al auf große Graphen angewandt wurden, bietet es sich an, sie auch auf große Proteinkomplexe anzuwenden und bei kleinen Proteinen den Aminosäuregraphen 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 großen 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 Klassifizierung von Biomolekülen entsprechend ihrer biologischen 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 Erster kleiner Test

Der erste kleine Test mit den Proteinen 1qf6, 1qf8 und 1qe0 war erfolgreich. Richtigerweise bewertet der Algorithmus mit der Relativen-*Graphlet*-Häufigkeiten-Distanz die Ähnlichkeit von 1qf8 und 1qe0 höher als die Ähnlichkeit zwischen den beiden und 1qf6. 1qf6 und 1qe0 sind beide Synthetasen, die Komplexe mit tRNA bilden, während 1qf8 eine Casein Kinase ist.

3.2 Fallstudie - Vergleich mit CATH-Einordnung

Die Ergebnisse sind für die KOMplexgraphen eher schlecht; bei Aminosäuregraphen funktioniert der Vergleich ok, allerdings hängt das von der Klasse ab. Die beste aktuelle Idee zur Verbesserung ist eine stärkere Gewichtung der gelabelten *Graphlets*. Diese könnte für die AA-Graphen jedoch dazu führen, dass die Scores am stärksten mit der Sequenzähnlichkeit korrelieren. Weiterhin könnte es sinnvoll sein, die Homologie der Proteine im Datensatz zu prüfen und zu schauen, ob sich Ideen für weitere Verbesserungen gewinnen lassen.

3.2.1 AA-*Graphlets*

Für die Proteine aus der Homologen Superfamilie der Cytochrome lieferten die Aminosäuregraphen die besten Ergebnisse. Nur hier stammten die ähnlichsten Proteine auch aus der gleichen Superfamilie. Als Metrik wurde die Relative-*Graphlet*-Häufigkeiten-Distanz verwendet.

PDB-ID	CATH-Code	Platz 1 - 4	Scores
1QPU	1.20.120.10	1QQ3, 2XL6, 1M4R, 2YL1	1.45 3.65 5.41 5.63
1QQ3	1.20.120.10	1QPU, 2XL6, 1M4R, 2YL1	1.45 3.61 5.08 5.65
2XL6	1.20.120.10	1QQ3, 1QPU, 2YL1, 2V5L	3.61 3.65 4.51 4.71
2YL1	1.20.120.10	2XL6, 1QPU, 1QQ3, 2V5L	3.61 3.65 4.51 4.71

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 Bildverzeichnis

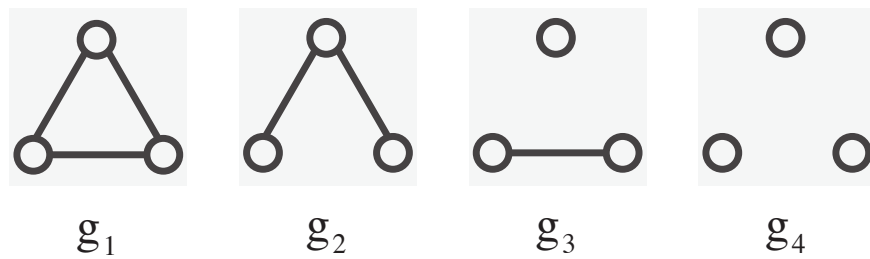


Abbildung 5.1: Graphlets der Größe 3 (*Shervashidze et al.*)

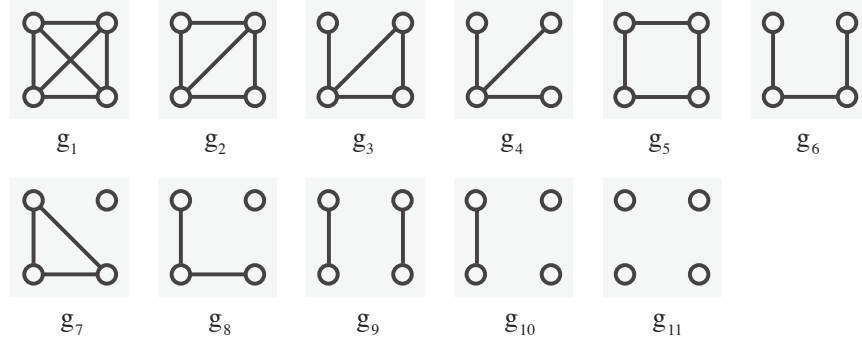


Abbildung 5.2: Graphlets der Größe 4 (*Shervashidze et al.*)

5.2 Tabellenverzeichnis

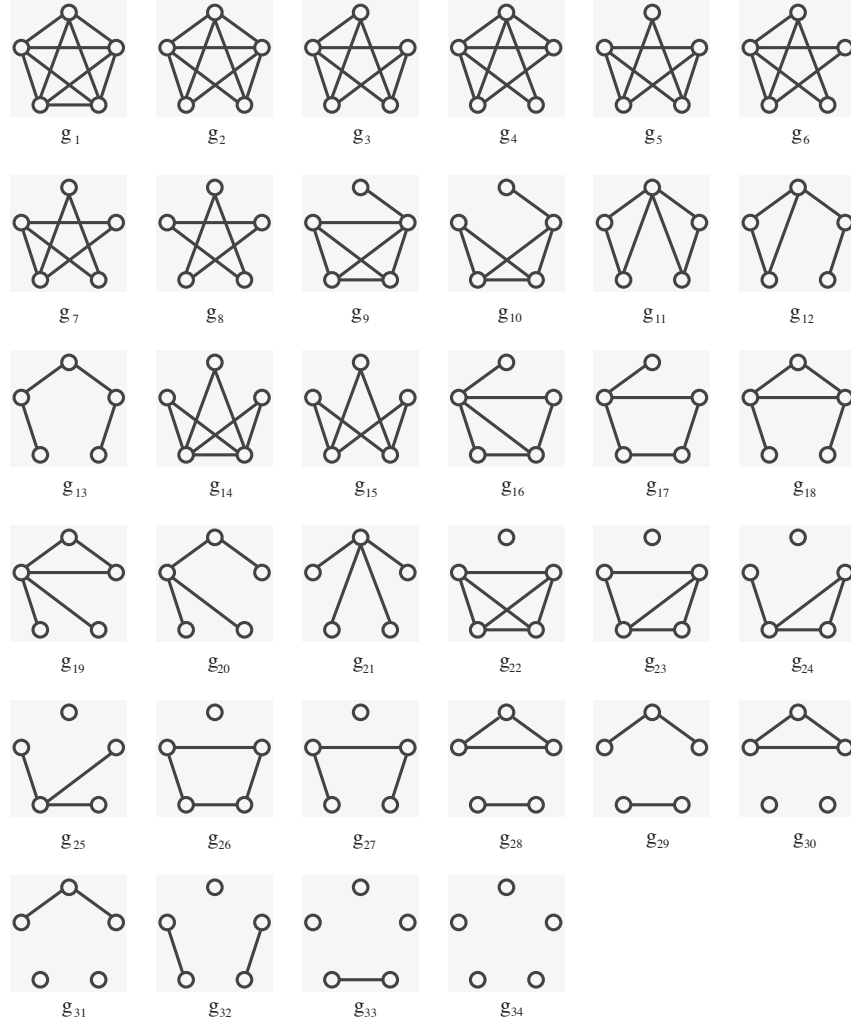


Abbildung 5.3: Graphlets der Größe 5 (*Shervashidze et al.*)

Proteine, die aus einer *Chain* bestehen, die genau eine Domäne enthält

PDB-ID	1vib	1qpu	1qq3	2xl6	2yl1	1rxq	2qe9	2rd9	1m4r	1hgu	1pv6	lawr	1d9c	1ngl	1n7v	7tim	1ney	1v3z	1al7	1ar0	2v5l
2utg	X	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	X	8.915	15.89	16.89	21.08	15.22	16.80	16.23	13.39	13.26	17.13	12.32	12.55	11.80	17.87	17.44	17.07	13.78	7.943	16.33	14.27
1qpu	X	8.626	15.54	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	X	8.150	15.89	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	X	9.759	16.89	3.655	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	X	13.62	21.08	5.632	5.651	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	X	6.408	15.22	6.064	5.106	7.737	X	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	X	8.128	16.80	7.554	5.523	7.197	X	2.024	4.006	6.167	5.959	8.708	4.649	8.068	10.83	4.470	5.180	6.065	6.728	4.542	3.043
2rd9	X	7.384	16.23	7.539	7.252	7.589	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	X	5.158	13.39	5.087	5.514	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	X	4.944	13.26	7.849	8.147	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	X	3.570	12.32	7.087	8.655	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
lawr	X	11.67	17.13	11.01	11.02	8.374	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	X	4.392	12.55	7.053	6.721	2.685	4.649	3.853	1.837	3.187	4.568	7.702	X	6.366	9.743	5.605	5.246	4.468	5.626	6.116	3.637
1ngl	X	7.387	11.80	9.864	9.918	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	X	13.02	17.87	12.54	12.56	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	X	9.171	17.44	8.261	7.787	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	X	8.737	17.07	8.829	8.354	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	X	8.025	13.78	8.822	8.892	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
1al7	X	2.490	7.943	9.192	9.531	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	X	9.329	16.33	9.692	9.216	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	X	6.253	14.27	6.462	6.019	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	1exs	1ngl	1qqs	3slo	1wjx	5chy	2id9	3i42	2w0i	1d4o	1qpu	1qq3	1cgn	1he9	3gf9
1exs	X	1.290	3.217	4.391	0.611	0.662	1.459	1.576	4.226	9.191	3.567	3.860	1.712	3.456	5.473
1ngl	1.290	X	3.637	2.620	2.697	2.644	0.788	0.286	2.722	4.373	2.309	3.457	1.979	1.386	4.230
1qqs	3.217	3.637	X	12.90	5.645	4.202	3.394	3.512	4.907	16.22	13.66	6.851	7.333	4.069	14.86
3slo	4.391	2.620	12.90	X	4.974	3.863	2.351	2.412	7.116	14.09	14.10	12.70	12.20	5.432	15.99
1wjx	0.611	2.697	5.645	4.974	X	0.360	0.961	0.965	2.042	2.225	2.128	2.587	1.954	1.466	2.516
5chy	0.662	2.644	4.202	3.863	0.360	X	0.797	0.914	2.093	1.728	1.933	2.431	1.120	1.291	1.971
2id9	1.459	0.788	3.394	2.351	0.961	0.797	X	0.692	2.890	1.921	2.165	2.557	0.249	1.099	2.575
3i42	1.576	0.286	3.512	2.412	0.965	0.914	0.692	X	3.008	2.039	1.877	2.269	0.037	1.386	2.693
2w0i	4.226	2.722	4.907	7.116	2.042	2.093	2.890	3.008	X	10.76	4.422	3.002	0.980	2.506	11.61
1d4o	9.191	4.373	16.22	14.09	2.225	1.728	1.921	2.039	10.76	X	8.791	7.907	6.501	9.950	8.811
1qpu	3.567	2.309	13.66	14.10	2.128	1.933	2.165	1.877	4.422	8.791	X	4.405	5.056	2.271	10.75
1qq3	3.860	3.457	6.851	12.70	2.587	2.431	2.557	2.269	3.002	7.907	4.405	X	1.906	1.714	10.03
1cgn	1.712	1.979	7.333	12.20	1.954	1.120	0.249	0.037	0.980	6.501	5.056	1.906	X	0.985	9.469
1he9	3.456	1.386	4.069	5.432	1.466	1.291	1.099	1.386	2.506	9.950	2.271	1.714	0.985	X	5.881
3gf9	5.473	4.230	14.86	15.99	2.516	1.971	2.575	2.693	11.61	8.811	10.75	10.03	9.469	5.881	X

Literaturverzeichnis

- [1] Hasegawa et al. Advances and pitfalls of protein structural alignment. *Current Opinion in Structural Biology*, 2009.
- [2] May et al. Ptg1: 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.