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

Ben Haladik

4. Januar 2016

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Übung: [1] Jetzt zitiere ich [2]

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 Anwendung dieser Methoden dient hauptsächlich zwei Zwecken. Der erste ist die schnelle Suche nach ähnlichen Strukturen in Datenbanken. Dies findet hauptsächlich im pharmakologischen Bereich statt. Der zweite Zweck ist der Erkenntnisgewinn über die ferne evolutionäre verwandschaft von Proteinen. Es wird davon ausgegangen, dass ähnliche Strukturen bei niedriger Sequenzidentität auf ferne Verwandschaft und ähnliche Funktion hindeuten. Wenn eine

1.3 Ziele

1.4 Aufbau der Arbeit

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

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 [3] 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 Be-

rechnungen. 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 Charakteristiken könen dann als Datenpunkte verglichen werden und man spart sich die Berechnungen, die man sonst für alle Paare von Graphen G, G' durchführen muss. Eine Algorithmus, mit dem sich solche Charakteristiken verechnen lassen wurde 2009 von *N. Shervashidze* vorgestellt: Der *Graphlet*-Algorithmus.

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:

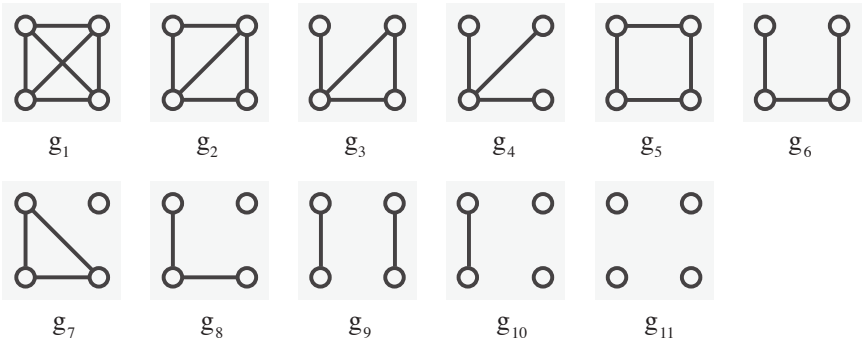

Abbildung 2.1: Graphlets der Größe 4 (*Shervashidze et al.*)

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?)

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

2.5.2 Relative *Graphlet*-Häufigkeiten-Distanz

2.5.3 *Root-Mean-Square-Deviation*

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 Programm FATCAT zusammengestellt worden, um seine Performance mit der von etablierten Strukturalignment-Programmen wie DALI und CE zu vergleichen. Der Datensatz setzt sich aus über 13000 PDB-Einträgen zusammen. Er ist nach Paaren mit großer strukturelller Ähnlichkeit sortiert. So eignen sich auch Teilmengen des Datensatzes, um *Graphlets* zu testen. Wenn der Ansatz funktioniert, dann sollten die Paare aus dem FATCAT-Datensatz auch in der PTGL die Paare mit dem höchsten Score sein.

2.6.2 Fallstudien

Kapitel 3

Ergebnisse

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

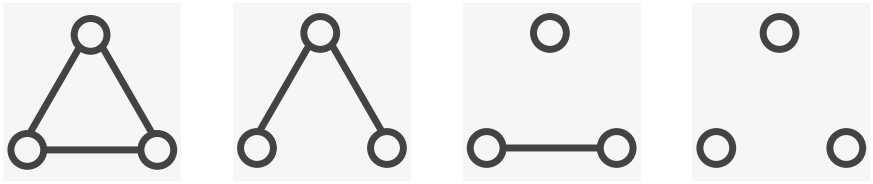

Abbildung 5.1: Graphlets der Größe 3 (*Shervashidze et al.*)

5.3 Literaturverzeichnis

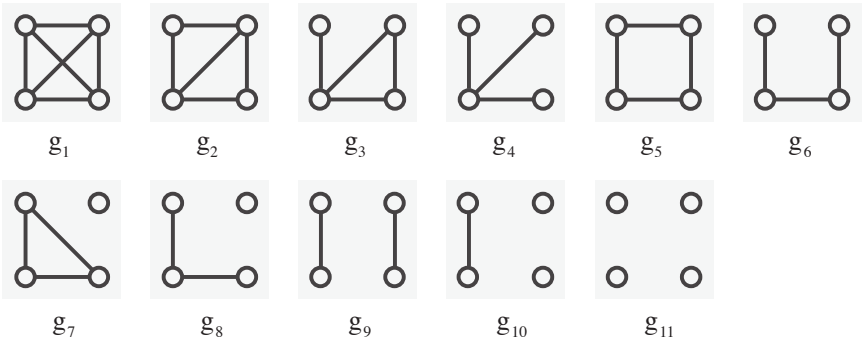

Abbildung 5.2: Graphlets der Größe 4 (*Shervashidze et al.*)

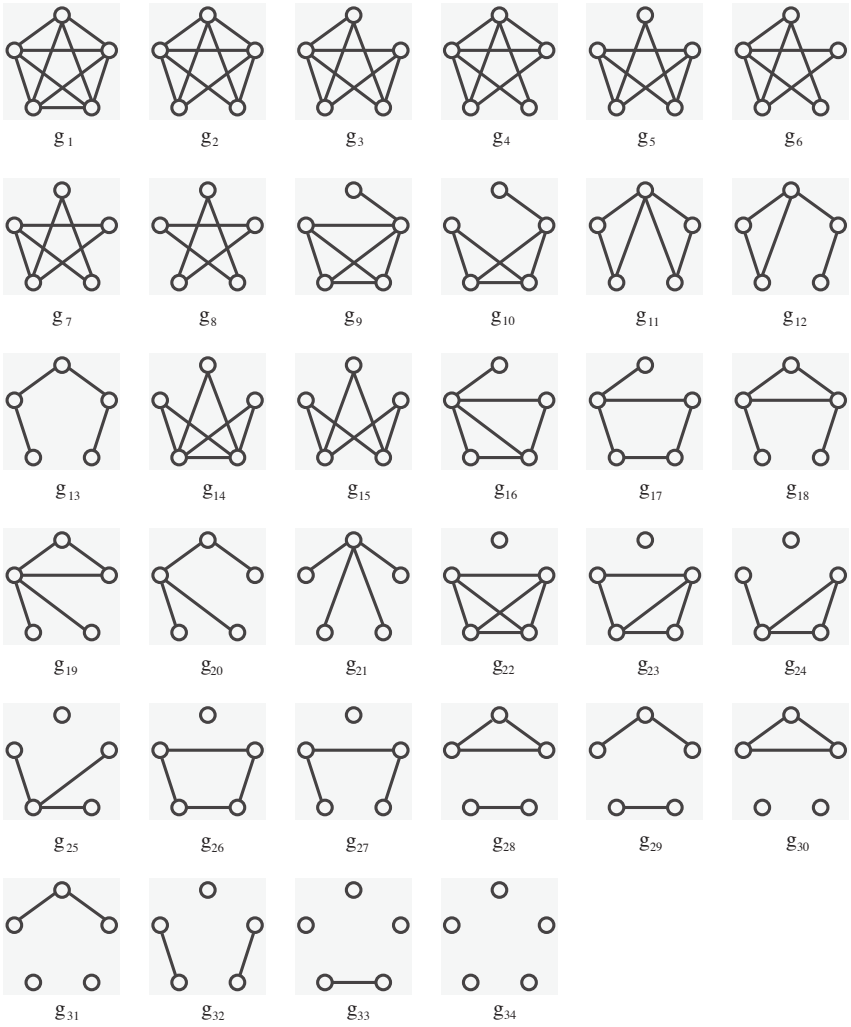

Abbildung 5.3: Graphlets der Größe 5 (*Shervashidze et al.*)

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