

**Bachelorarbeit**

**Labelpropagation - Analyse & Varianten**

Mike Pack

Themensteller: Univ. Prof. Dr. Schulz

Betreuer: Jonathan Rollin

Lehrgebiet Theoretische Informatik

Fachbereich Informatik

---

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>4</b>
<b>2 Theoretische Grundlagen</b>	<b>6</b>
2.1 Komplexe Netzwerke . . . . .	6
2.2 Community Struktur . . . . .	9
2.3 Community Detection Algorithmus . . . . .	10
2.3.1 Agglomerative Algorithmen . . . . .	11
2.4 Modularität . . . . .	12
<b>3 Methoden und Netzwerke</b>	<b>14</b>
3.1 Label Propagation . . . . .	14
3.1.1 Label Propagation Algorithmus . . . . .	14
3.1.2 Zeit Komplexität . . . . .	16
3.2 Auswertung des Label Propagation Algorithmus . . . . .	16
3.2.1 Reliabilität . . . . .	16
3.2.2 Qualität . . . . .	16
3.2.3 Konvergenz . . . . .	17
3.3 Variationen des Label Propagation Algorithmus . . . . .	17
3.3.1 Synchrone und Asynchrone Version . . . . .	19
3.3.2 Initialisierung der Label . . . . .	21
3.3.3 Reihenfolge der Knoten . . . . .	22
3.3.4 Nachbarschaft . . . . .	23
3.3.5 Update Regeln und Labelwahl . . . . .	24
3.3.6 Konvergenzkriterium . . . . .	26
3.3.7 Zusammenfassung . . . . .	26
3.4 Die Netzwerke . . . . .	27
3.4.1 Das Watts Strogatz Modell . . . . .	27
3.4.2 Die Modularität des Watts Strogatz Modell . . . . .	29
<b>4 Framework und Experimente</b>	<b>30</b>
4.1 Das Framework . . . . .	30
4.1.1 Aufbau des Frameworks . . . . .	31
4.2 Experimente . . . . .	32
4.2.1 Erheben der Grunddaten . . . . .	32
4.2.2 Anwendung der Varianten . . . . .	37
4.2.3 Detaillierte Experimente des LPA_S und Hop 0 - 1 Variante .	53

4.2.4	Auswertung der Varianten . . . . .	64
<b>5</b>	<b>Zusammenfassung und Ausblick</b>	<b>69</b>
5.1	Zusammenfassung . . . . .	69
5.2	Ausblick . . . . .	70

# 1 Einleitung

Die Beobachtung und Untersuchung komplexer Netzwerke hat in unterschiedlichen wissenschaftlichen Bereichen immer mehr an Bedeutung gewonnen. Insbesondere in den Bereichen, in denen es um die Analyse und Visualisierung komplexer Daten geht. Als Beispiel sei hier die Sozialwissenschaften [49, 32], die Biologie [41], die Netzwerkneurowissenschaften [42, 4], oder die Medizin [2] genannt.

Solche komplexen Netzwerke können unter anderem aus mehreren Millionen Knoten und Kanten bestehen. Innerhalb solcher Netzwerke existieren sogenannte Community Strukturen (siehe Kap 2.2). Der Begriff Community wird in unterschiedlichen Bereichen verwendet, sodass hier keine eindeutige Definition existiert.

Um nun solche Communities innerhalb solcher Netzwerke erkennen zu können, existieren sogenannte Community Detection Algorithmen (siehe Kap. 2.3). Einer dieser Algorithmen ist der Label Propagation Algorithmus (LPA) [40] (siehe Kap. 3.1.1). Der Algorithmus ist von dem Konzept der epidemischen Ausbreitung inspiriert. Das Erkennen der Communities innerhalb des Netzwerkes erfolgt dadurch, dass die Labels der Knoten durch das Netzwerk pro Iteration propagiert werden, bis es zur Konvergenz kommt. Die Laufzeit des Algorithmus (siehe Kap 3.1.2) beträgt im worst case  $O(n + m)$  pro Iteration, mit  $n$  für die Anzahl der Knoten und  $m$  für die Anzahl der Kanten. Die Komplexität des Algorithmus entspricht  $O(\mathcal{I}m)$ , wobei  $\mathcal{I}$  die Anzahl der Iterationen angibt. Für die Bewertung der gebildeten Communities wird die Modularität [33, 8] verwendet (siehe Kap. 2.4). Je höher dieser Wert ausfällt, desto optimaler verlief die Bildung der Communities mittels eines Community Detection Algorithmus.

Der Aufbau des Algorithmus selber erlaubt es, gewisse Bereiche des LPA zu verändern, da jeder Teilbereich des Algorithmus weitest gehend unabhängig voneinander ist. Sodass im Laufe der letzten Jahre einige Veränderungen an dem LPA durchgeführt wurden (siehe Kap. 3.3). Solche Änderungen betreffen unter anderem die Aktualisierungsregel (siehe Kap. 3.3.1), die Initialisierung der Label (siehe Kap. 3.3.2), die Reihenfolge in der die Knoten bearbeitet werden (siehe Kap. 3.3.3), die Wahl der Nachbarn (siehe Kap. 3.3.4), aber auch wie die Berechnung und Auswahl des neuen Labels erfolgt (siehe Kap. 3.3.5), sowie die Konvergenz (siehe Kap. 3.3.6).

Einige dieser Veränderungen, die an dem Algorithmus durchgeführt wurden, werden in den folgenden Experimenten (siehe Kap. 4.2) näher untersucht. Insbesondere wird hierbei die Konvergenz des LPA untersucht unter Einbezug der Modularität. Für die Durchführung dieser Experimente werden Netzwerke verwendet, die mittels dem Watts Strogatz Modell [34] nach Newman erstellt wurden (siehe Kap. 3.4).

Eine ähnliche Arbeitsweise wie die des LPA findet sich unter anderem auch bei Zellularautomaten [17] wieder. Hier erhalten die Knoten zunächst wie auch bei dem LPA ein entsprechendes Label zur Initialisierung. Es werden jedoch nicht  $n$  Label verteilt, sondern zwei. Jeder Knoten erhält ein blaues Label mit einer Wahrscheinlichkeit von  $p_b \leq \frac{1}{2}$  ( $p_b$  entspricht der Wahrscheinlichkeit, ob ein Knoten ein blaues Label erhält), andernfalls erhält er ein rotes Label. Die Knoten ermitteln die jeweiligen Label von ihren direkten Nachbarn wie auch der LPA und errechnen aus dieser Menge das maximal vorkommende Label. Bei einem Gleichstand behält ein Knoten anders als bei dem LPA das Label. Die Aktualisierung erfolgt auch hier synchron von Generation zu Generation, sodass auch hier nicht aktuelle Werte genutzt werden. Der Algorithmus terminiert, sobald ein sich wiederholender Zyklus der Generationen erreicht ist [17].

Eine weitere recht ähnliche Arbeitsweise wie die des LPA findet sich in der Bootstrap Perkolation [9, 23, 20]. Anders als bei dem LPA werden zufällige Knoten auf aktiv gesetzt, wären die übrigen Knoten inaktiv bleiben. Hierbei werden keine Labels verteilt. Inaktive Knoten betrachten nun ihre direkten Nachbarn, sodass es sich um einen lokalen Algorithmus handelt wie auch der LPA. Inaktive Knoten werden aktiv, wenn sie von mindestens  $r \geq 2$  aktiven Knoten umgeben sind. Diese Arbeitsweise lässt sich damit vergleichen, dass ein Knoten sein Label wechselt, wenn mehrere Nachbarn in seiner Umgebung dieses Label tragen. Dies wird iterativ für alle inaktiven Knoten wiederholt. Der Algorithmus terminiert, wenn alle Knoten aktiv sind, oder inaktive Knoten über weniger als  $r$  aktive Knoten verfügen.

## 2 Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen erläutert, die für das Verständnis der Experimente und Analysen benötigt werden.

Die Grundlagen von Community-Detection Algorithmen sind sogenannte Komplexe Netzwerke. Innerhalb solcher komplexen Netzwerke befinden sich gewisse Community Strukturen, die mithilfe eines solchen Algorithmus erkannt und identifiziert werden sollen.

Vor der eigentlichen Definition des Community Detection Algorithmus, ist es wichtig zu wissen, was ein Netzwerk ist und was eine Community ausmacht. Im ersten Abschnitt 2.1 wird zunächst eine formale Definition eines komplexen Netzwerkes beschrieben und wie die Darstellung solcher Netzwerke erfolgt.

Wenn man das Wort Community liest oder hört wird es mit einer Gruppe von Menschen in Verbindung gebracht, die als Beispiel ein gemeinsames Interesse teilen. In Verbindung mit der Graphen-Theorie besitzt dieses Wort eine etwas andere Bezeichnung und wird in dem Abschnitt 2.2 beschrieben.

Nach diesen beiden grundlegenden Definitionen wird im Abschnitt 2.3 beschrieben, was ein Community Detection Algorithmus ist.

Da die Struktur der Netzwerke aber auch die Struktur der darin enthaltenen Communities unbekannt ist, wird eine Messmethode benötigt, um die Qualität der Ergebnisse zu messen und zu bewerten. Eine gängige Methode um diese Messung durchzuführen, ist die Berechnung der Modularität, welche im Abschnitt 2.4 beschrieben wird.

### 2.1 Komplexe Netzwerke

Die Beobachtung und Untersuchung komplexer Netzwerke nimmt in unterschiedlichen wissenschaftlichen Bereichen immer mehr an Bedeutung zu. Insbesondere dort, wo es um die Analyse und Visualisierung komplexer Daten geht. Hier hat sich ein eigener wissenschaftlicher Bereich etabliert, die sogenannte Netzwerktheorie. Die Netzwerktheorie befasst sich mit der Studie und Analyse von Graphen, die Systeme darstellen, die unterschiedliche Objekte enthalten. Diese Objekte stehen in gewissen Beziehungen zueinander.

Die Netzwerktheorie wird in verschiedenen wissenschaftlichen Bereichen verwendet. Als Beispiel sei hier die Biologie, Ökologie, Informatik und Neurowissenschaft genannt. Ein Komplexes-Netzwerk ist ein System bestehend aus Knoten und Kanten. Wobei die Knoten unterschiedliche Objekte repräsentieren und die Kanten jeweils die Beziehungen zwischen diesen Objekten darstellen. Mathematisch kann ein

komplexes Netzwerk mithilfe eines Graphen beschrieben werden.

Die Formel  $G = (V, E)$  beschreibt den Graphen, wobei  $V$  eine endliche Menge von Knoten und  $E$  eine endliche Menge ungeordneter Knotenpaare, die sogenannten Kanten [7] darstellt.

Eine Kante  $\{u, v\}$  verbindet die Knoten  $u$  und  $v$  miteinander, und kann auch als  $uv$  geschrieben werden.  $G$  ist ein ungerichteter Graph, wenn die Knotenmenge einer Kante ungeordnet ist und gerichtet, wenn diese Menge geordnet ist.

Wenn es das Paar  $\{u, v\} \in E$  gibt, dann ist  $v$  ein Nachbar von  $u$ . Der Grad eines Knoten  $v$  ergibt sich aus der Anzahl der Kanten, die zum Knoten  $v$  führen und wird als  $\deg(v)$  geschrieben. Es ist also die Summe der Nachbarknoten des Knoten  $v$ . Verfügen alle Knoten des Graphen  $G$  über die gleiche Anzahl an Nachbarn, so ist  $G$  ein regulärer Graph wie die Abb. 1c zeigt. Verfügen die Kanten in  $G$  zusätzlich über ein Gewicht  $w$ , so ist  $G$  ein gewichteter Graph.

Ein Graph wird als einfach bezeichnet, wenn die Kanten nicht über Richtung und Gewicht, Mehrfachkanten, Hyperkanten oder Schleifen verfügt.

Zu den einfachen Graphen zählen unter anderem auch die sogenannten Bipartite Graphen sowie Bäume. Aus der Gruppe der Bäume werden hier nur die sogenannten Sterngraphen betrachtet.

Ein Graph  $G$  heißt Bipartite, wenn sich die Menge der Knoten in zwei disjunkte Teilmengen  $A$  und  $B$  aufteilen lässt. Die Knoten einer Teilmenge besitzen zu Knoten derselben Teilmenge keine Kanten. Dies bedeutet, die Knoten der Teilmenge  $A$  sind nur mit Knoten der Teilmenge  $B$  über Kanten verbunden. Für das Kantenpaar  $\{u, v\} \in E$  gilt entweder  $u \in A$  und  $v \in B$  oder  $u \in B$  und  $v \in A$ . Die Abb. 1e zeigt einen solchen Bipartiten Graphen.

Ein Graph  $G$  heißt Baum, wenn  $G$  ein zusammenhängender, kreisfreier Graph ist. Die Knoten mit Grad 1 heißen Blätter und die anderen Knoten sind innere Knoten.  $G$  ist ein Sterngraph  $S_n$  oder auch n-Stern genannt, wenn in ihm ein Knoten  $v$  existiert (die Wurzel), der mit allen anderen Knoten im Graphen verbunden ist wie die Abb. 1d zeigt.

Innerhalb eines Sterngraphen besitzt der Wurzelknoten den höchsten Knotengrad. Bei  $n$  Knoten ergibt sich ein Grad von  $n - 1$  für die Wurzel, während der Grad der anderen Knoten 1 ist. Der Sterngraph ist zudem ein vollständiger bipartiter Graph.

Um einen Graphen darstellen zu können existieren unterschiedliche Möglichkeiten. So kann ein Graph durch eine endliche Menge von Knoten und Kanten beschrieben werden, mithilfe einer Adjazenz- oder Inzidenzmatrix [7].

Sei  $G$  ein Graph mit einer Menge von Knoten  $V = \{v_1, v_2, \dots, v_n\}$  und einer

Menge von Kanten  $E$ . Die Adjazenzmatrix ist eine  $n \times n$  Matrix  $\mathcal{A}$  und ist wie folgt definiert:

$$\mathcal{A}_{ij} = \begin{cases} 1 & \text{wenn } \{v_i, v_j\} \in E, \\ 0 & \text{sonst} \end{cases} \quad (1)$$

In einem einfachen Graph sind die diagonalen Elemente der Matrix 0. In einem ungerichteten Graphen ist die Matrix symmetrisch. Um mit einer Adjazenzmatrix einen gewichteten Graphen zu beschreiben, werden die Einträge der Matrix mit dem jeweiligen Kantengewicht ersetzt.

Sei  $G$  ein Graph mit den Knoten  $V = \{v_1, v_2, \dots, v_n\}$  und einer Menge von Kanten  $E = \{e_1, e_2, \dots, e_m\}$ . Die Inzidenz Matrix ist eine  $n \times m$  Matrix  $\mathcal{I}$ , wobei jede Zeile einem Knoten entspricht und jede Spalte entspricht einer Kante. Die Inzidenz Matrix ist definiert als:

$$\mathcal{I}_{ij} = \begin{cases} 1 & \text{wenn } v_i \in e_j \\ 0 & \text{sonst} \end{cases} \quad (2)$$

Um einen Graphen zu visualisieren werden Kreise für die Knoten und Linien für die Kanten verwendet. Die nachfolgende Abbildung 1 zeigt unterschiedliche Graphen.

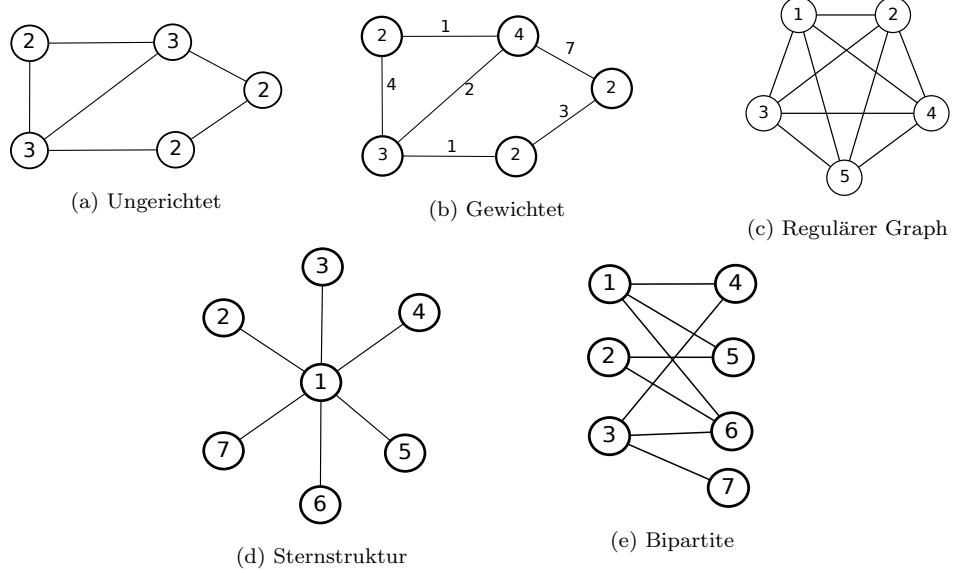


Abbildung 1: Die Abb. 1a zeigt einen ungerichteten Graphen mit dem Knotengrad eines jeden Knoten. Die Abb. 1b zeigt denselben Graphen, die Kanten tragen hier Gewichte. Ein regulärer Graph befindet sich in Abb. 1c. Die letzten beiden Graphen 1e und 1d stellen jeweils zwei Bipartite Graphen dar.

## 2.2 Community Struktur

Der Begriff Community Struktur wird in unterschiedlichen wissenschaftlichen Bereichen in unterschiedlicher Art und Weise verwendet, sodass dieser Begriff nicht eindeutig definiert ist. In der Ökologie und Biologie beschreibt eine solche Struktur beispielsweise die Artenvielfalt in einer gewissen Region oder einem Biotop.

In der Graphen Theorie besitzen komplexe Netzwerke eine Community Struktur, wenn die Knoten innerhalb eines solchen Netzwerkes zu Gruppen zusammen gefasst werden können. Die Knoten innerhalb einer solcher Community weisen eine dichtere Verbindung zu anderen Knoten der gleichen Community auf als zu den übrigen Knoten innerhalb des Netzwerkes [39].

Die Dichte eines Graphen entspricht der Anzahl der Kanten  $\mathcal{E}$  im Verhältnis zu den maximal möglichen Kanten [15]. Für eine Community  $\mathcal{C}$  entspricht die Dichte dem Verhältnis der Anzahl an möglichen Kanten, die ein Knoten haben könnte, wenn alle Knoten innerhalb dieser Community miteinander verbunden wären.

Die Dichte für einen ungerichteten Graphen ist wie folgt definieren:

$$\mathcal{D}_U(\mathcal{V}, \mathcal{E}) = \frac{m}{\frac{n(n-1)}{2}} = \frac{2m}{n(n-1)} \quad (3)$$

und für einen gerichteten Graphen:

$$\mathcal{D}_D(\mathcal{V}, \mathcal{E}) = \frac{m}{n(n-1)} \quad (4)$$

wobei  $m$  die Gesamtzahl der Kanten darstellt und  $n$  die Anzahl der Knoten.

Soziale Netzwerke sind ein Paradebeispiel für Graphen, die oftmals eine Community Struktur enthalten. Ein solches soziales Netzwerk besteht zumeist aus mehreren tausenden Mitgliedern, die dazu neigen, Gruppen mit anderen Personen zu bilden. Solch eine Gruppe kann z.B. ein gemeinsames Interesse umfassen, aus einem Freundeskreis bestehen oder aus Familienmitgliedern. Die Mitglieder eines solchen Netzwerkes können als Knoten aufgefasst werden und die Kanten stellen Verbindungen zwischen ihnen dar. Nun besteht zwischen Mitgliedern derselben Gruppe eine dichtere Verbindung als zu Mitgliedern aus anderen Gruppen. Diese Gruppierungen werden als Community aufgefasst.

Ein Beispiel für ein solches Netzwerk mit Community Strukturen stellt der bekannte und häufig genutzte Karate Club von Zachary dar [56]. Dieser Club teilte sich aufgrund eines Konfliktes zwischen dem Clubpräsidenten und Ausbilder in zwei Fraktionen. Das Netzwerk wird in Abb. 2 gezeigt. Der Knoten 0 stellt den Ausbilder und der Knoten 33 den Clubpräsidenten dar. Diese beiden Knoten können jeweils als Kern der Communities betrachtet werden.

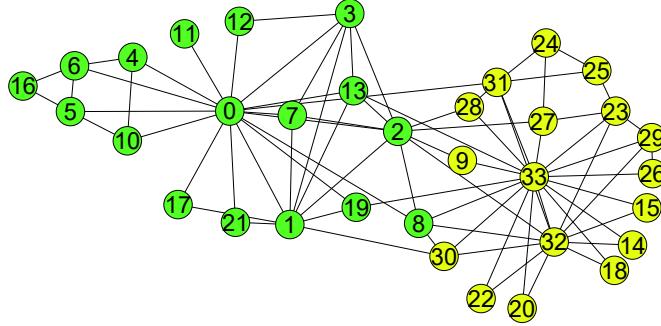


Abbildung 2: Das Netzwerk zu dem Karate Club. Die beiden Communities wurden farblich kenntlich gemacht.

Community Strukturen finden sich jedoch nicht nur in sozialen Netzwerken wieder. Sie sind auch präsent in Computer-Netzwerken, Netzwerkneurowissenschaften [42, 4], biologischen Netzwerken [41] oder auch in technologischen Netzwerken um ein paar zu nennen.

### 2.3 Community Detection Algorithmus

Die Menge an Knoten und Kanten innerhalb eines komplexen Netzwerkes reichen von einigen wenigen bis hin zu mehreren Millionen. In relativ kleinen Netzwerken mit einer sehr geringen Anzahl an Knoten lässt sich eine Community Struktur, wenn auch mit Mühe, manuell ermitteln. Je größer diese Netzwerke jedoch werden, desto schwieriger wird es, Communities zu identifizieren. Betrachtet man hier einmal das Internet, das mehrere Millionen Rechnern miteinander vernetzt, so ist das Ermitteln einzelner Communities ohne geeignete Hilfsmittel unmöglich. An diesem Punkt setzen die sogenannte Community Detection Algorithmen an. Solche Algorithmen finden Teilnetzwerke mit einer signifikant höheren Dichte zwischen Knoten, die zur selben Community gehören, als zu Knoten, die zu einer anderen Community gehören [18].

Die Auswahl an solchen Algorithmen ist relativ groß und auch die Arbeitsweise unterscheidet sich von Algorithmus zu Algorithmus. So existieren unter anderem Varianten, die als Eingabe lediglich den Graphen benötigen, wie etwa der Label Propagation [40], Girvan-Newman [18], oder der Greedy Modularity [13] Algorithmus. Andere Varianten wiederum benötigen neben den Graphen selber, zusätzlich die Informationen in wie viele Communities das Netzwerk aufgeteilt werden soll, wie etwa bei dem FluidC [36] oder Spectral Clustering [47].

Das Ziel aller Algorithmen ist jedoch die Einteilung der Netzwerke in geeignete Communities. Je nach Arbeitsweise des Algorithmus kann die Einteilung in Com-

munities pro Netzwerk unterschiedlich ausfallen. Aus diesem Grund lässt sich auch keine Aussage darüber treffen, welcher Algorithmus für welches Netzwerk geeignet ist. Eine wesentliche Rolle spielt hier die Struktur des Netzwerkes, aber auch wie der Algorithmus die Communities bildet.

Grob lassen sich die Community Detection Algorithmen in zwei Klassen, die Agglomerativen und Divisive [48, 31] aufteilen. Ausnahmen bilden hierbei Spectral Clustering sowie FluidC.

Die Divisiven Algorithmen sind sogenannte Top-Down Algorithmen. Jeder Knoten gehört hier zunächst der selben Community an. Mit jeder Iteration, wird diese Community in immer feiner Communities zerlegt. Dieses Verfahren wird solange durchgeführt, bis jeder Knoten eine eigene Community bildet.

### 2.3.1 Agglomerative Algorithmen

Die Agglomerativen Algorithmen sind sogenannte Bottom-Up Algorithmen und jeder Knoten stellt zunächst eine Community für sich dar. Nach jeder Iteration werden zwei Communities zu einer größeren Community zusammengefasst. Dies wiederholt sich, bis die Optimierung einer zuvor definierten Funktion erreicht ist, wie etwa das Maximieren der Modularität [8] (siehe Abschnitt 2.4). Zu den Agglomerativen Algorithmen zählen z.B. der Walktrap-Algorithmus [38], der auf der random walk [37] Methode basiert, der Greedy Modularity Algorithmus [13], sowie der Label Propagation Algorithmus [40] um nur ein paar zu nennen. Der Label Propagation Algorithmus wird im folgenden Kapitel 3 näher beschrieben und betrachtet. Zunächst folgt eine kurze Beschreibung des Greedy Modularity Algorithmus, da dieser zum Vergleich genutzt wird.

---

#### **Algorithm 2** Greedy Modularity Algorithmus

---

- 1: Jeder Knoten stellt eine Community für sich dar.
  - 2: Berechne die Modularität des gesamten Netzwerkes.
  - 3: Berechne die neue Modularität, wenn zwei benachbarte Communities miteinander vereint werden und speicher den Wert.
  - 4: Wiederhole Schritt 3 für jedes Community Paar.
  - 5: Vereine die Communities, die zu einer Steigerung der Modularität führen.
  - 6: Wiederhole die Schritte 2 - 5 solange, wie eine Steigerung der Modularität verzeichnet wird.
-

## 2.4 Modularität

Die Modularität  $\mathcal{Q}$  besitzt eine wichtige Bedeutung um die Qualität der gefundenen Communities mittels eines Community Detection Algorithmus zu messen [33]. Für den Wertebereich der Modularität gilt:  $-\frac{1}{2} \leq \mathcal{Q} \leq 1$  [8]. Liegt die Modularität im positiven Bereich, so lässt sich daraus schließen, dass eine Community Einteilung innerhalb des Netzwerkes existiert. Bei einem Wert von 0 besteht das Netzwerk lediglich aus einer großen Community [8] und ein negativer Wert deutet darauf hin, dass die Community Einteilung nicht erfolgreich ist.

Die Modularität  $\mathcal{Q}$  ergibt sich aus dem Anteil der Kanten der Knoten, die zu ein und derselben Community gehören minus dem erwarteten Anteil der Kanten, wenn diese zufällig verteilt werden. Mathematisch lässt sich die Modularität wie folgt schreiben [8]:

$$q(\mathcal{P}) = \sum_{c_j \in \mathcal{P}} \left[ \frac{E(C)}{m} - \left( \frac{\sum_{v_i \in C_j} \deg(v_i)}{2m} \right)^2 \right] \quad (5)$$

Wobei  $E(C)$  die Anzahl der Kanten innerhalb der Community beschreibt und  $\deg(v_i) = \sum_j A_{ij}$  beschreibt den Grad des Knoten  $v_i$ .  $m$  beschreibt die Menge aller Kanten im Netzwerk. Zur Maximierung der Modularität  $q(\mathcal{P})$  ist es wichtig, den ersten Term zu maximieren, indem die Knoten innerhalb der Community einen hohen Knotengrad erhalten. Der zweite Term wird minimiert, indem das Netzwerk in möglichst viele Communities mit einem geringen Knotengrad zerteilt wird. Die Formel (5) kann so umgeschrieben werden, dass die Berechnung direkt über die Knoten [10] anstelle der Community Struktur erfolgt.

$$\mathcal{Q} = \frac{1}{2m} \sum_i^n \sum_j^n \left( A_{ij} - \frac{\deg(v_i)\deg(v_j)}{2m} \right) \delta(c_i, c_j) \quad (6)$$

Mit  $\delta(c_i, c_j)$  als das Kronecker Delta, wobei  $c_i, c_j$  die Community Zugehörigkeit für einen Knoten  $i$  und  $j$  beschreibt,  $A_{ij}$  ist die Adjazenzmatrix und  $\frac{\deg(v_i)\deg(v_j)}{2m}$ , beschreibt den erwarteten Anteil an Kanten zwischen den Knoten  $v_i$  und  $v_j$ , wenn die Kanten zufällig verteilt werden, mit  $m$  als der Anzahl aller Kanten im Graphen. Das Kronecker Delta ergibt 1, wenn  $c_i = c_j$ , also wenn beide Knoten in derselben Community liegen, sonst 0.

Die Formel (6) lässt sich ohne das Kronecker Delta schreiben, indem man direkt über die Summe der Communities iteriert. Dies eignet sich eher für die Programmierung.

$$\mathcal{Q} = \frac{1}{2m} \sum_C \sum_{i \in C} \sum_{j \in C} \left( A_{ij} - \frac{\deg(v_i)\deg(v_j)}{2m} \right) \quad (7)$$

Die folgende Abb. 3 demonstriert, was sich hinter dem Term  $\frac{\deg(v_i)\deg(v_j)}{2m}$  verbirgt. Der Knoten  $i$  besitzt einen Grad von 4 und der Knoten  $j$  einen Knotengrad von 5. Jede Kante von  $i$  besitzt nun viermal fünf Möglichkeiten, mit dem Knoten  $j$  verbunden zu sein. Dies ist der Anteil an Kanten, wenn diese zufällig verteilt werden.

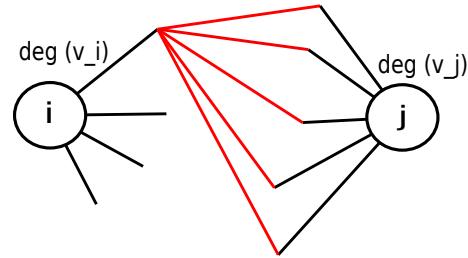


Abbildung 3: Der Anteil der Kanten, wenn diese zufällig verteilt werden (rot).

### 3 Methoden und Netzwerke

In diesem Kapitel folgt die Beschreibung des Label Propagation Algorithmus (LPA) [40]. Zu Beginn des Kapitels wird der Standard LPA in Abschnitt 3.1.1, sowie die Zeitkomplexität in Abschnitt 3.1.2 des Algorithmus beschrieben. In Abschnitt 3.3 folgt eine Beschreibung der LPA Varianten, die in den letzten Jahren mit dem Ziel entwickelt wurden, die Ergebnisse des Algorithmus zu verbessern.

Zum Abschluss des Kapitels, folgt im Abschnitt 3.4 eine Darstellung und Beschreibung der Netzwerke, die für den Test des LPA und der Varianten genutzt werden. Bei diesen Netzwerken handelt es sich um Watts Strogatz Modelle [34], die mit unterschiedlichen Werten erzeugt wurden.

#### 3.1 Label Propagation

In diesem Abschnitt des Kapitels folgt die Beschreibung des LPA. Im Anschluss daran folgt eine Aufstellung der LPA Varianten, sowie eine Beschreibung welche Bereiche des Algorithmus durch diese verändert werden.

##### 3.1.1 Label Propagation Algorithmus

Bei dem Label Propagation Algorithmus handelt es sich um einen agglomerativen Partitionsalgorithmus (siehe Abschnitt 2.3.1), bei dem jeder Knoten zunächst eine eigene Community bildet. Das Konzept des Algorithmus ist von einer Epidemischen Ausbreitung inspiriert, was sich auf die Ausbreitung der Labels bezieht. Die Arbeitsweise des Algorithmus ist einfach gehalten und lässt sich wie folgt beschreiben:

Jeder Knoten erhält zunächst ein initiales Label. Dabei darf jedes Label zu Beginn im Sinne der Agglomerativen Algorithmen (siehe Abschnitt 2.3.1) nur einmal vorkommen. Der Algorithmus beginnt mit  $n$  Communities, wobei  $n$  der Anzahl der Knoten im Graphen entspricht. Diese Labels werden durch das Netzwerk propagiert bis der Algorithmus terminiert. Die Ermittlung welches Label ein Knoten  $v$  nach einer Iteration annimmt, ist wie folgt geregelt:

Ein Knoten  $v$  sammelt zunächst die Labels seiner direkten Nachbarn und berechnet aus dieser Menge das Label, welches am häufigsten unter den Nachbarn vertreten ist GL. (8).

$$\mathcal{L}_{\mathcal{T}}(v) := \arg \max_{l \in \mathcal{L}} |N_{\mathcal{T}-1}^l(v)| \quad (8)$$

Wobei  $N_{\mathcal{T}-1}^l(v)$  die Menge der Nachbarn des Knoten  $v$  mit dem Label  $l$  zum Zeitpunkt  $\mathcal{T} - 1$  darstellt. Die Wahl der Labels beruht immer auf den Labels der vorhe-

rigen Runde. Es findet eine synchrone Arbeitsweise des Algorithmus statt. Stehen mehrere Labels zur Auswahl, die Gl. (8) erfüllen, wird zufällig eines gewählt. Dieses Verfahren wird für jeden Knoten innerhalb des Netzwerkes und während jeder Iteration wiederholt. Der Algorithmus terminiert, sobald es zu einer Stabilität der Labels innerhalb des Netzwerkes kommt. Eine Stabilität liegt vor, wenn folgende Bedingung erfüllt ist:

Entsprechen die Labels der Knoten zum Zeitpunkt  $\mathcal{T}$  den Labels zum Zeitpunkt  $\mathcal{T} - 1$  dann terminiere. Das bedeutet, zwischen der aktuellen und der vorherigen Iteration ist es zu keiner Veränderung der Labels gekommen. Die Community Einteilung ist somit stabil. Sobald der Algorithmus terminiert, besitzen mehrere Knoten ein identisches Label und gehören entsprechend der gleichen Community an.

Die Implementierung des LPA im Pseudocode kann Alg. 3 entnommen werden. Bevor der Graph eingelesen wird, kann eine Vorverarbeitung an dem Netzwerk erfolgen. Auf diese Weise können ‘Unreinheiten’ entfernt werden, die hinderlich für den Algorithmus sind. Als Beispiel sei hier das Entfernen von isolierten Knoten genannt. Solche Knoten haben auf die Berechnung der Communities keinen Einfluss, wirken sich jedoch auf die Laufzeit des Algorithmus aus. Der LPA würde diese Knoten mit jeder Iteration anwählen, um ein Label für sie zu ermitteln.

---

#### Algorithm 4 Die Synchrone Variante des Label Propagation Algorithmus

---

**Input:** Graph G (V, E)

**Output:** Communities  $\mathcal{C}$

```

1: Funktion Label.Propagation(G)
2:   for  $v \in V$  do
3:      $l_0 = \{\text{Unique label}\}$ 
4:   end for
5:   while keine Konvergenz do
6:      $\mathcal{T} \leftarrow 1$ 
7:     shuffle (V)
8:     for  $v \in V$  do  $\triangleq$  Iteriere über alle Knoten
9:        $c_v \leftarrow N(v)$   $\triangleq$  Ermittle Label der Nachbarn des Knoten v
10:       $\mathcal{L}_{\mathcal{T}}(v) \leftarrow \arg \max_{l \in \mathcal{L}} |N_{\mathcal{T}-1}^l(v)|$   $\triangleq$  Berechnung des neuen Labels
11:    end for
12:    if  $\mathcal{L}_{\mathcal{T}}$  equal  $\mathcal{L}_{(\mathcal{T}-1)}$  then  $\triangleq$  überprüfe auf Konvergenz
13:      break
14:    else
15:       $\mathcal{T} \leftarrow \mathcal{T} + 1$ 
16:      for  $v \in V$  do
17:         $\mathcal{L}_{\mathcal{T}+1}(v) \leftarrow \mathcal{L}_{\mathcal{T}}(v)$   $\triangleq$  Aktualisiere die Label
18:      return communities  $\mathcal{C}$ 

```

---

### 3.1.2 Zeit Komplexität

Die Laufzeit des Algorithmus ergibt sich wie folgt: Das Zuweisen eines Labels pro Knoten während der Initialisierungsphase benötigt eine Zeit von  $O(n)$ . Pro Iteration ergibt sich durch die Anzahl der Kanten  $m$  eine lineare Laufzeit von  $O(m)$ . Das hier die Kanten und nicht die Knoten betrachtet werden, hängt wie folgt zusammen. Für jeden Knoten  $v$  werden die Nachbarn  $\mathcal{N}$  aufsteigend nach ihrem Label gruppiert, was eine Komplexität von  $O(\mathcal{N}_v)$  entspricht. Der Knoten  $v$  wählt die größte Gruppe aus und übernimmt dessen Label, was im worst-case eine Komplexität von  $O(\mathcal{N}_v)$  entspricht. Dieses Verfahren wird für jeden Knoten wiederholt, sodass sich eine Komplexität von  $O(m)$  pro Iteration ergibt [40].

Daraus ergibt sich eine Komplexität von  $O(\mathcal{I}m)$ , wobei  $\mathcal{I}$  der Anzahl der Iterationen entspricht. Ist die Anzahl der Iterationen konstant über alle Graphen, so besitzt der LPA eine lineare Laufzeit. In allen anderen Fällen besitzt er diese nicht. Experimente haben gezeigt, dass bereits 95% der Knoten am Ende der fünften Iteration in der richtigen Community liegen [40]. Anhand dieser Experimente zeigt sich, dass die Anzahl der Iterationen bei nahezu allen Graphen konstant ist.

## 3.2 Auswertung des Label Propagation Algorithmus

Der LPA zählt mit seiner Laufzeit zu einem sehr effizienten und schnellen Algorithmus der Community Bestimmung. Neben dieser Schnelligkeit besitzt der Algorithmus auch einige Schwächen, die Spielraum für Erweiterungen schaffen. Diese Varianten werden in dem folgenden Abschnitt 3.3 beschrieben.

### 3.2.1 Reliabilität

Die Ergebnisse des LPA variieren [24, 44] von Anwendung zu Anwendung auf ein und demselben Graphen. Dadurch lassen sich Ergebnisse des Algorithmus nur schwer reproduzieren. Der Ursprung dieser Variation liegt unter anderem in der Zufälligkeit des LPA [43]. So wird eine zufällige Reihenfolge, in welcher die Knoten bearbeitet werden, nach jeder Iteration (in 3.3.3 näher untersucht) gewählt und es kommt zu einer zufälligen Auswahl der Labels bei einem Gleichstand [43]. So kann die Reihenfolge und Wahl der Labels mal optimal aber auch suboptimal erfolgen.

### 3.2.2 Qualität

Durch die Variation der Ergebnisse des Algorithmus, kommt es letztlich auch zu Schwankungen der Modularität, sowie der Bildung von Communities [24, 14]. Die-

ses Verhalten lässt sich auf die Zufälligkeit des Algorithmus zurückführen. So wird nach jeder Iteration eine sich zufällig ändernde Reihenfolge der Knoten gewählt. Bei Gleichstand wird aus mehreren Labels zufällig Eines ausgewählt [45, 6]. Dadurch entstehen Schwankungen der Communities, die wiederum zur Messung der Modularität genutzt werden. Zusätzlich wurde beobachtet, dass der Algorithmus zur Bildung von ‘*Super Communities*’ [24, 3, 6] neigt (siehe Abschnitt 3.3.1). Dabei besteht eine Community aus der großen Mehrheit aller Knoten im Netzwerk.

### 3.2.3 Konvergenz

Die Konvergenz des Algorithmus ist ein Bereich, der bisher nicht intensiver untersucht wurde. So kommt es vor, dass der LPA bei Eingabe eines Graphen konvergiert und beim nächsten Mal nicht. Dieses Verhalten hängt mit der zufälligen Auswahl eines Labels bei Gleichstand zusammen (siehe Abschnitt 3.3.5), sowie mit der Struktur des Netzwerkes selbst. Eine Divergenz lässt weiterhin auf eine Oszillation [40] schließen. Eine Oszillation liegt vor, wenn zwischen den Knoten die Labels ausgetauscht werden und sich dieser Tausch von Iteration zu Iteration wiederholt. Begünstigt wird eine Oszillation durch die synchrone Aktualisierung sowie durch die Struktur des Netzwerkes, wenn es über Teilgraphen verfügt, die eine Bipartite oder Sternen ähnliche Struktur aufweisen. Die Abb. 4 zeigt eine solche Oszillation.

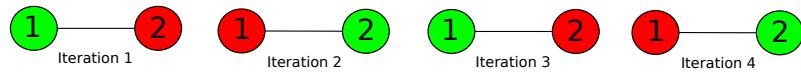


Abbildung 4: Ein einfaches Netzwerk bestehen aus zwei Knoten. Durch Anwendung der synchronen Variante des LPA kommt es zu einer Oszillation. Die Iterationen 1 und 2 werden unendlich oft wiederholt. Der Algorithmus terminiert nicht.

Eine Verbesserung der Konvergenz kann durch Nutzung der asynchronen Aktualisierung (siehe Abschnitt 3.3.1), durch Auflösen des Gleichstands (siehe Abschnitt 3.3.5) oder durch Änderungen der Knotenreihenfolge (siehe Abschnitt 3.3.3) erfolgen.

## 3.3 Variationen des Label Propagation Algorithmus

In diesem Abschnitt des Kapitels folgt eine Darstellung der verschiedenen LPA Varianten, die in den vergangenen Jahren entwickelt wurden. Das Ziel dieser Änderungen an dem LPA besteht darin, die Schwächen des Algorithmus aus Abschnitt 3.2 zu optimieren. Eine solche Optimierung liegt in der Verbesserung der Community Einteilung oder der Steigerung der Modularität.

Der LPA kann als ein Modul betrachtet werden, das aus kleineren Modulen zusammengesetzt ist. So besteht der Algorithmus

- aus der Festlegung, wann die Labels aktualisiert werden sollen (*Mode*) (siehe Abschnitt 3.3.1),
- aus der Initialisierung der Labels (Init) (siehe Abschnitt 3.3.2),
- aus der Reihenfolge, in der die Knoten bearbeitet werden (*Rhf.*) (siehe Abschnitt 3.3.3),
- aus der Bestimmung der Nachbarschaft des aktuellen Knotens  $v$  (*Nachb*) (siehe Abschnitt 3.3.4),
- aus der Berechnung des neuen Labels für den Knoten  $v$  (*Freq.*) (siehe Abschnitt 3.3.5),
- aus der Wahl, welches Label ausgewählt werden soll bei einem Gleichstand (*Sel.*) (siehe Abschnitt 3.3.5),
- wann der Algorithmus terminieren soll (*Konv.*) (siehe Abschnitt 3.3.6)

Jeder Teilabschnitt des Algorithmus kann verändert werden ohne den Charakter des LPA stark zu verändern. Änderungen innerhalb dieser Abschnitte können sowohl zu einer Steigerung als auch zu einer Verschlechterung des LPA führen. Die Abkürzungen in *kursiver* Schrift, hinter den jeweiligen Teilabschnitten des LPA beziehen sich auf die Bezeichnungen, die für Tabelle 3.3 verwendet werden. Die nachfolgende Tabelle 3.3 gibt einen Überblick über die Varianten des Algorithmus, die bislang entwickelt wurden. Hier sei angemerkt, dass nicht jede Variante des Algorithmus erfasst wurde.

Author	Jahr	Alg.	Basis	Mode	Init.	Rhf.	Nachb	Freq.	Sel.	Konv.
Raghavan et al. [40]	2007	LPA		Sync	Einzig.	zuf.	1	Knt.Präf.	Max	Stabil
Leung et al. [24]	2009	LPA - $\delta$	LPA	Async	Einzig.	zuf.	0 - 1	Knt.Präf.	Max	Stabil
Barber and Clark [3]	2009	LPAm	LPA	Async	Einzig.	zuf.	1	Ristrik.	Max	Stabil
	2009	LPar	LPA	Async	Einzig.	zuf.	1	Ristrik.	Max	Stable
Gregory [19]	2010	COPRA	LPA	Sync	Einzig.	NA	1	Knt.Präf.	Max	Stabil
Liu and Murata [29]	2010	LPAm+	LPA	Semi	Einzig.	zuf.	1	Ristrik.	Max	Stabil
Subelj and Bajec [45]	2011	DDALPA	LPA - $\delta$	Async	Einzig.	zuf.	0 - 1	Knt.Präf.	Max	Stabil
		ODALPA	LPA - $\delta$	Async	Einzig.	zuf.	0 - 1	Knt.Präf.	Max	Stable
		BDPA	DDALPA	Async	Kerne	zuf.	0 - 1	Knt.Präf.	Gruppe	Stabil
		ODALPA								
		DPA	BDPA	Async	Einzig.	zuf.	0 - 1	Knt.Präf.	Gruppe	Stabil
Xie et al [53]	2011	SLPA	COPRA	Async	Einzig.	zuf.	1	Zuf.	Max	Max Iter
Wu et al. [51]	2012	BMLPA	COPRA	Sync	Kerne	zuf.	1	Knt. Präf.	Max	Stabil
Zhao et al. [59]	2012	LPA-E	LPA	Sync	Einzig.	Entropie	1	Knt. Präf.	Max	Stabil
Zhank et al. [57]	2014	LPAp	LPA	Async	Einzig.	zuf.	1	Spez.	Perc.	Stabil
Xing et al. [55]	2014	NIBLPA	LPA	Async	Einzig.	Spez.	1	Label Präf.	Max	Stabil
Sum et al. [55]	2014	$\alpha$ NILP	LPA	Async	Einzig.	Spez.	Skali.	Knt Präf.	Max	Stabil
Hu et al. [21]	2016	roLPA	LPA	Async	Kerne	Abst.	1	Knt.Präf.	Max	Stabil
Chin and Ratnavelu [12]	2016	CLPA-GNR	LPA	Async	Einzig.	Fix	1	Nachb. Präf	Gruppe	mod
Liu et al. [27]	2016	CLBLPA	COPRA	Async	Kerne	zuf.	1	Zugeh.	Max	Stabil
Liu et al. [28]	2016	ELPA	LPA	Async	Kerne	Abst.	1	Kanten/Label	Max	Stabil
Zhang et al. [58]	2017	LPA_NI	LPA	Sync	Einzig.	Abst.	1	Label Einfl.	Max	Max Iter
Chen et al. [11]	2017	LPA-E	COPRA	Async	Einzig.	zuf.	2	Knt.Präf.	add. Thr.	Thr
Li et al. [26]	2017	LPA-S	LPA	Sync.	Einzig.	zuf.	1	ähnlich.	Max	Stabil
Sun et al. [46]	2017	LinkLPA	LAP	Async	Einzig.	zuf.	1	Verb.	Max	Stabil
Zhou et al. [60]	2018	SELP	SLP	Async	Spez.	Gruppe	1	Knt.Präf.	Max	Iter & Thr
Meng et al. [30]	2018	SSLP	SELP	Async	Spez.	Gruppe	1	Knt.Präf.	Max	Iter & Thr
Berahand and Boyer [5]	2018	LP-LPA	LPA	Sync	Einzig.	Abst.	1	Label Einfl.	Max	Stabil
Fiscarelli et al. [16]	2019	memLPA	LPA	Sync	Einzig.	NA	1	Knt. Präf	Max	Stabil
Li et al. [25]	2021	LPA-MNI	LPA	Async	Einzig.	Spez.	1	Spez.	Max	Stabil

Tabelle 1: Diese Tabelle gibt einen Überblick der verschiedenen Variationen des LPA, geordnet nach dem Jahr, in dem sie entwickelt wurden. Die Spalte Basis gibt an, auf welchen LPA der jeweilige Algorithmus *Alg.* beruht. Die Abkürzungen der Tabelle entsprechen den Begriffen aus Abschnitt 3.3

Nachfolgend folgt eine Erklärung der jeweiligen Spalten, sowie die Veränderungen, die an dem Algorithmus vorgenommen wurden.

### 3.3.1 Synchrone und Asynchrone Version

In dem Algorithmus 3 wurde die synchrone Variante des LPA beschrieben. Die Aktualisierung erfolgt bei Anwendung des herkömmlichen LPA in synchroner Weise. Die asynchrone Variante stellt eine Erweiterung des Algorithmus dar.

Der Unterschied zwischen beiden Versionen liegt in der Art, in der die Labels der Knoten aktualisiert werden. Unter Anwendung der synchronen Variante erfolgt eine Aktualisierung erst am Ende einer jeden Iteration. Dadurch ergeben sich zwei Besonderheiten, die zu beachten sind:

Erstens müssen die Labels der vorherigen Iteration  $\mathcal{T} - 1$  während der aktuellen Iteration  $\mathcal{T}$  bekannt sein, damit eine Berechnung des neuen Labels erfolgen kann. Zweitens müssen die Labels der aktuellen Iteration  $\mathcal{T}$  bis zum Ende der Iteration gespeichert werden. Unter Anwendung der synchronen Variante ist zu beachten, dass die Berechnungen der neuen Labels nicht auf aktuellen Werten beruht.

Anders ist es bei der asynchronen Version des LPA. Die Knoten innerhalb des Netzwerkes aktualisieren ihr Label direkt, nachdem das neue Label errechnet wurde. Hier sei angemerkt, dass die Knoten zur Berechnung des Labels, anders als bei der synchronen Variante immer aktuelle Werte verwenden.

Die Art, in der Knoten eine Aktualisierung ihres Label durchführen, beeinflusst die Konvergenz des Algorithmus. Unter Anwendung der synchronen Variante und je nach Struktur des Netzwerkes, kann es zu einer zyklischen Oszillation zwischen den Label kommen. Ein solches Verhalten des LPA kann begünstigt werden, wenn das Netzwerk über Teilgraphen verfügt, die eine Bipartite oder Sternen ähnliche Struktur enthalten [40]. Eine solche Oszillation führt zu einer Instabilität der Communities, und verhindert die Konvergenz des Algorithmus.

Das Problem der Oszillation lässt sich mit der asynchronen Version lösen [40]. Die Nutzung dieser Aktualisierung führt zu Nebeneffekten, die sich auf das Netzwerk auswirken.

Ein solcher Nebeneffekt ist die begünstigte Bildung von sogenannten '*Super Communities*' [24], sowie die Bildung von vielen kleineren Communities. Dieser Effekt hängt unter anderem damit zusammen, dass zu Beginn einer jeden Iteration eine zufällige Reihenfolge der Knoten festgelegt wird. So weisen die Knoten innerhalb der Community einen zu geringen Knotengrad auf, sodass sich das eigene Label im Netzwerk nicht behaupten kann [24]. Durch mehrere Experimente wurde gezeigt, dass die Bildung solcher '*Super Communities*' mithilfe der synchronen Version verlangsamt werden kann [24]. Die Abb. 5 zeigt die Entwicklung einer solchen '*Super Community*'.

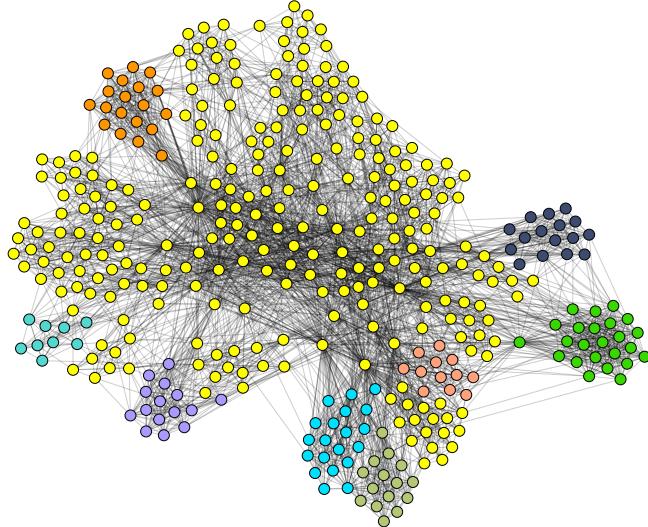


Abbildung 5: Dieser Graph zeigt die Städte NRW. Die Daten wurden im Rahmen eines Fachpraktikums der FernUniversität Hagen im Lehrgebiet Theoretische Informatik von mir erhoben. In diesem Graphen zeigt sich deutlich die Bildung einer ‘*Super Community*’ (gelb dargestellt) sowie die Bildung kleiner Communites.

Ein weiterer Nebeneffekt wirkt sich nicht auf das Netzwerk, sondern auf den Algorithmus aus. Bei Verwendung der asynchronen Variante, erfolgt die Aktualisierung der Knoten direkt, nachdem das neue Label errechnet wurde. Dies kann zu einer schnelleren Konvergenz beitragen. Diese Art der Aktualisierung verhindert eine Parallelisierung. Schließlich ist es für einen Knoten  $v$  nicht möglich sein neues Label zu berechnen, wenn die Label der Nachbarn sich ändern, sofern diese dazu kommen ihr Label zu aktualisieren.

Die Parallelisierung der synchronen Variante steigert die Geschwindigkeit des Algorithmus, führt aber zu keinerlei Beeinträchtigung der Ergebnisse.

### 3.3.2 Initialisierung der Label

Die erste Phase des LPA, besteht aus der initialen Labelverteilung. Bei dem herkömmlichen LPA, erhält jeder Knoten ein eindeutiges Label, welches im gesamten Netzwerk zu diesem Zeitpunkt nur einmal vorkommen darf. Diese Labels werden durch das Netzwerk propagiert, bis der Algorithmus terminiert. Die Anzahl der Labels nimmt mit jeder Iteration des LPA weiter ab. Ein Teil der Varianten aus der Tabelle 3.3 nutzen diese Art der initialen Verteilung.

Neben dieser recht schnellen Art der Verteilung existieren weitere Varianten, durch die die Knoten zu Beginn mit einem Label versehen werden können. Bei den sogenannten Semi-Supervised Propagation Algorithmen [60, 22] erhält nur ein Teil der Knoten ein Label, während die restlichen Knoten ohne Label verbleiben. Die

Vergabe der initialen Labels kann manuell oder zufällig erfolgen.

Eine weitere Art der Initialisierung stellt die Verwendung von Community Kernen dar. Das Berechnen der Kerne erfolgt in zwei Phasen, der Extraktions- und Expansionsphase.

Während der Extraktionsphase wird eine entsprechende Struktur innerhalb des Netzwerkes ermittelt. Entweder durch Bestimmung von Teilgraphen oder durch Ermittlung von Ursprungsknoten einer Community [51]. Eine solche Struktur zeichnet sich z.B. durch Knoten aus, die einen hohen Knotengrad besitzen [28]. Diese Knoten können als Zentrum der Communities betrachtet werden. Bei dem Netzwerk zum Karate Club aus Abb. 2 weisen die Knoten '0', '1', '2', '32' und '33' eine solche Struktur auf, da sie im Vergleich zu den übrigen Knoten des Netzwerkes über einen hohen Knotengrad verfügen.

Nachdem eine solche Struktur ermittelt und jeder Kern mit einem Label versehen wurde, folgt die Expansionsphase. Während dieser Phase erhalten die übrigen Knoten ein Label. Diese Phase kann auf zwei Arten umgesetzt werden. Jeder Knoten, ausgehend von den Kernen, erhält nun ein entsprechendes Label. Oder nur die Knoten, die sich in direkter Nähe der Kerne aufhalten [27]. Bei der letzten Möglichkeit erhalten die verbleibenden Knoten im Verlauf des Algorithmus ein Label, wie es bei den Semi-Supervised Algorithmen der Fall ist. Die Berechnung solcher Community Kerne wirkt sich jedoch auf die Laufzeit des Algorithmus aus, da zunächst solche Strukturen im Netzwerk ermittelt werden müssen.

In den Experimenten in Kapitel 4.2 wird die Vergabe der eindeutigen Labels pro Knoten beibehalten.

### 3.3.3 Reihenfolge der Knoten

Die zweite Phase des Algorithmus besteht darin, die Reihenfolge fest zulegen, in welcher die Knoten bearbeitet werden sollen. Änderungen an dieser Phase haben Auswirkungen auf den LPA. Sowohl auf die Ergebnisse wie auch auf die Stabilität des Algorithmus.

Bei dem Standard Algorithmus wird die Reihenfolge der Knoten von Iteration zu Iteration verändert. Dies hat zur Folge, dass Ergebnisse nicht reproduziert werden können und ihre Qualität vom Zufall abhängt. In Verbindung mit der asynchronen Aktualisierung wird die Bildung von '*Super Communities*' gefördert [24]. Um diese Zufälligkeit zu begrenzen legen bestimmte Varianten des LPA zu Beginn eine zufällige Reihenfolge fest. Diese bleibt während der gesamten Laufzeit des Algorithmus konstant.

Innerhalb der Regelungen in der ein Knoten sein Label aktualisiert, wurden weitere Methoden entwickelt. Ziel ist es, optimale Reihenfolgen der Knoten zu bestimmen, damit die Ergebnisse weniger vom Zufall bestimmt werden.

Zum Festlegung einer solchen Reihenfolge ist es möglich, jeden Knoten mit einem bestimmten Wert zu versehen. Der Wert lässt sich z.B. in Abhängigkeit davon errechnen, welchen Stellenwert der Knoten für das Netzwerk besitzt. Als Beispiele seien hier der NIBLPA [55] sowie der LPA\_NI [58] genannt. Nach Berechnung der Werte erfolgt eine Sortierung der Knoten, in auf- oder absteigender Reihenfolge. Auch der LPA-E [59] errechnet für jeden Knoten einen Wert, die sogenannte Label-Entropie.

Eine weitere Methode für das Festlegen einer Reihenfolge besteht darin, die Knoten nach speziellen Regeln zu gruppieren. Solche Gruppen können z.B. die Menge an Knoten mit und ohne Label umfassen, wie bei den Semi Supervised Algorithmen. Die Gruppen können aus aktiven und inaktiven Knoten bestehen [52], wobei nur die aktiven Knoten bearbeitet und aktualisiert werden. Wurden die Kerne der Communities errechnet, kann eine Einteilung in Kern- und Randknoten erfolgen. Der roLPA [21] unterteilt die Knoten neben Rand- und Kernknoten noch in sogenannte Brückenknoten. Solche Knoten besitzen Kanten in unterschiedlichen Communities.

### 3.3.4 Nachbarschaft

Damit ein Knoten  $v$  das neue Label berechnen kann, müssen zunächst die Nachbarn des Knotens betrachtet und die jeweiligen Label gesammelt werden.

Bei dem herkömmlichen LPA handelt es sich um einen lokalen Algorithmus, da ein Knoten  $v$  nur die Nachbarn betrachtet, die zu diesem adjazent sind. Diese Art der Betrachtung der nächstgelegenen Nachbarn wird auch als Hop 1 Distanz bezeichnet. Wird der Knoten  $v$  selber mit in diese Betrachtung einbezogen, so handelt es sich hierbei um eine Hop 0 - 1 Distanz. Sowohl die Hop 1 als auch die Hop 0 - 1 Variante wird von unterschiedlichen LPA Versionen verwendet, wie die Tabelle 3.3 zeigt. Bereits eine 0 - 1 Distanz führt zu einer Verhaltensänderung des LPA.

Die Entfernung zwischen dem Knoten  $v$  und den Nachbarn kann über die lokale Betrachtung hinaus erweitert werden. So nutzt der LPA-E [11] z.B. eine entsprechende Hop 2 Distanz. Hierbei werden die Nachbarn, sowie deren Nachbarn betrachtet. Der Knoten  $v$  kann auch bei dieser und jeder weiteren Hop Distanz mit beachtet werden, sodass sich immer eine 0 -  $n$  Distanz ergibt.

Das Ausdehnen der Distanz zwischen dem aktuellen Knoten und seinen Nachbarn wirkt sich auf die Laufzeit des Algorithmus aus. Da mit jeder weiteren Distanz die

Nachbarn der Nachbarn ermittelt werden müssen. Die Abb. 6 zeigt noch einmal die Hop Distanzen in einem Netzwerk.

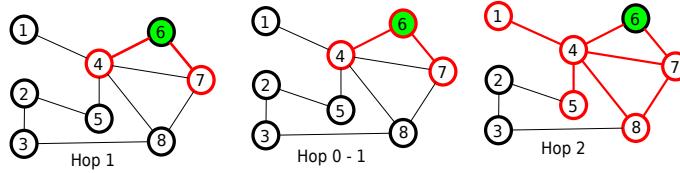


Abbildung 6: Darstellung der Hop Distanzen für ein Netzwerk. Bei Hop 1 werden nur die direkten Nachbarn betrachtet. Hop 0 - 1 schließt den Knoten  $v_i$  selbst mit ein. Bei der Hop 2 Distanz werden die Nachbarn des Knoten  $v_i$  sowie deren Nachbarn betrachtet.

### 3.3.5 Update Regeln und Labelwahl

Diese Phase des Algorithmus ist mitunter eine der wichtigsten. Innerhalb dieses Schrittes erfolgt die Berechnung sowie die Auswahl des neuen Labels.

Unter Anwendung des einfachen LPA ermittelt ein Knoten  $v$  zunächst seine direkten Nachbarn und sammelt die jeweiligen Labels der Knoten. Aus der folgenden Menge wird das Label gewählt, welches am häufigsten in der Menge vertreten ist GL. (8). Existieren nun mehrere Label, die diese Gleichung erfüllen, wird zufällig eines ausgewählt. Dies sorgt für zufällige Ergebnisse des Algorithmus und hat zudem Auswirkungen auf die Konvergenz, sowie die Stabilität der Communities. Es hat sich herausgestellt, dass die Art der Aktualisierung einen großen Teil der Instabilität des Algorithmus ausmacht. Durch ungünstige zufällige Auswahl wird die Bildung von ‘*Super Communities*’ gefördert [24]. Ausgehend von diesen Erkenntnissen wurde eine Reihe von Methoden entwickelt.

Zusammen mit dem Konvergenzkriterium, dass die Label sich ab einer Iteration  $\mathcal{T}$  nicht weiter ändern, ist es notwendig eine Lösung für das Problem des Gleichstandes zu ermitteln [3]. Mit der Lösung dieses Problems wird ein Teil der Zufälligkeit aus dem Algorithmus entfernt. Dies kann zur Stabilität beitragen. So nutzen einige Varianten des LPA folgenden Ansatz: Ist das Label des Knotens  $v$  in der Menge der Labels enthalten, so wechselt der Knoten  $v$  das Label nicht. Dieser Ansatz entfernt bereits einen Teil der Zufälligkeit und führt zu einer besseren Stabilität der Communities. Die Abb. 7 zeigt wie sich diese Änderung auf die Labelwahl und Konvergenz des Algorithmus auswirkt.

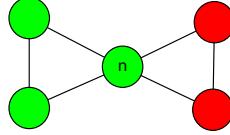


Abbildung 7: Der Knoten  $v$  besitzt vier Nachbarn. Hier würde der LPA nun zufällig eines auswählen, da beide Label zweimal vorhanden sind. Dies kann in einer Oszillation enden. Wird das Label des Knotens  $v$  selbst bei der Auswahl betrachtet, dann findet kein Wechsel statt.

Eine andere Methode zum Auflösen des Gleichstandes besteht darin, das Label aus der Menge zu wählen, dass eine Restriktion erfüllt. Eine solche Restriktion kann etwa die Steigerung der Modularität betreffen, wie bei dem LPAm [3] oder LPAm+ [29]. Wurden die Nachbarn eines Knotens  $v$  ermittelt, erfolgt eine Berechnung der Modularität. Es wird nun das Label gewählt, das zu einer Steigerung der Modularität führt. Der Kerngedanke bei dem LPAm besteht darin, Communities zu erzeugen, die in etwa über dieselbe Größe verfügen und eine hohe Modularität erreichen. Durch die Bildung gleich großer Communities erreicht der LPAm lediglich ein lokales Maximum der Modularität [29] und verharrt an diesem Punkt. Der LPAm+ versucht hingegen diesem lokalen Maximum zu entgehen, um so eine weitere Steigerung der Modularität zu erreichen.

Neben den oben genannten Möglichkeiten einen Gleichstand zu lösen, besteht ein weiteres genutztes Verfahren darin, die Gl. (8) wie folgt umzuschreiben:

$$\mathcal{L}'_i := \arg \max_{l \in \mathcal{L}} \sum_{u \in \Gamma(v, l)} f(u) \quad (9)$$

wobei  $f(u)$  eine Funktion ist, die jedem Knoten eine Präferenz zuordnet.  $\Gamma(v, l)$  sind die Nachbarn des Knoten  $u$  mit dem Label  $l$ . Die Funktion  $f(u)$  kann zum Beispiel als Knotengrad  $f(u) = \text{Deg}(u)$  [24] fungieren. Die Aktualisierungsregel wird unter anderem in dem Label Propagation with Hop Attenuation LPA  $\delta$  verwendet [24]. Bei dieser Variante des Algorithmus spielt der Knotengrad eine wesentliche Rolle, sowie ein Gewichtungsfaktor, der sogenannte Hop Score. Die Präferenz bei dieser Variante besteht darin, Knoten mit einem hohen oder niedrigen Knotengrad stärker zu gewichten.

Um die Bildung von '*Super Communities*' zu unterdrücken wurden weitere Varianten entwickelt. Bei dem LPAp [57] wird bei einem Gleichstand die Größe der Communities berechnet. In diesem Falle wird das Label gewählt, das zu der kleinsten Community gehört. So wird die Bildung der '*Super Communities*' unterdrückt.

### 3.3.6 Konvergenzkriterium

Der LPA terminiert, sobald die Label der Iteration  $\mathcal{T}$  den Label der Iteration  $\mathcal{T} - 1$  entsprechen. Dieses Konvergenzkriterium wird jedoch nicht immer erreicht. Insbesondere dann nicht, wenn es zu einer Oszillation zwischen den Labels innerhalb des Netzwerkes kommt.

Damit der Algorithmus terminiert, wurden weitere Methoden in Betracht gezogen. Eine Möglichkeit besteht darin, die Anzahl an Iterationen festzulegen. Sobald diese Anzahl erreicht ist, bricht der Algorithmus ab, unabhängig davon, ob die Bildung der Communities stabil ist. Ausgehend von der Aussage, dass bereits am Ende der fünften Iteration 95 % der Knoten richtig zugeordnet wurden [40], sollte die Anzahl der Iterationen nicht unter fünf liegen. Beide Möglichkeiten können auch kombiniert werden [5], sodass der LPA mit Sicherheit terminiert.

Neben diesen beiden Möglichkeiten ist der Einsatz einer Threshold zu beachten. Dabei handelt es sich um einen festgelegten Wert. So nutzt der SSLP [30] eine Kombination der maximalen Anzahl an Iterationen und legt zudem eine Threshold  $\delta$  fest. Sobald die Differenz zwischen den Iterationen  $\mathcal{T}$  und  $\mathcal{T} - 1$  unter diese Threshold fällt, terminiert der Algorithmus. Als Threshold kann die Differenz zwischen den Modularitäten genutzt werden. Solange eine Steigerung der Modularität verzeichnet wird, arbeitet der LPA weiter. Bei dem LabelRank [54] Algorithmus wird festgelegt, wie groß die Menge der zu aktualisierenden Knoten pro Iteration ist. Werden in einer Iteration weniger Knoten als angegeben aktualisiert, terminiert der LPA.

### 3.3.7 Zusammenfassung

Das Hauptaugenmerk bei der Auswertung der hier vorgestellten Varianten liegt vielmehr auf der Einteilung der Communities, die mit dem LPA erreicht werden und bewerten die Modularität des Algorithmus. Die Konvergenz des Algorithmus bleibt hier im Hintergrund. Ausnahmen bilden die Varianten, die den Algorithmus terminieren lassen. Dies kann durch eine Maximierung der Modularität, unterschreiten der Threshold oder erreichen einer festgelegten Iterationsgrenze erfolgen. Es bleibt jedoch unklar, ob der Algorithmus unter Anwendung der Varianten konvergiert und wenn ja, wie häufig dieses vorkommt. Aus diesem Grund wird in den Experimenten in Kapitel 4.2 die Standard Konvergenz des Algorithmus beibehalten, um zu ermitteln, wie sich einige Varianten auf die Konvergenz des LPA auswirken.

## 3.4 Die Netzwerke

In diesem Abschnitt folgt eine Beschreibung der Netzwerke, die im Kapitel 4.2 verwendet werden. Bei den hier genutzten Netzwerken handelt es sich um künstliche Netzwerke, die mittels dem Watts Strogatz Modell [34] nach Newman erzeugt wurden.

### 3.4.1 Das Watts Strogatz Modell

Bei dem Watts Strogatz Modell [50] handelt es sich um ein zufällig erzeugtes Netzwerk, das zur Erzeugung gewisse Parameter benötigt. Der Wert  $n$  für die Anzahl der Knoten, der Wert  $k$  für die Kanten, sowie den Wert  $p$  für die Wahrscheinlichkeit. Zur Erzeugung eines Netzwerkes werden zunächst die  $n$  Knoten des Graphen ringförmig angeordnet und mit den nächsten  $k$  Nachbarn verbunden. Handelt es sich bei  $k$  um einen ungeraden Wert, so werden  $k-1$  Nachbarn miteinander verbunden. Ein solches Netzwerk besitzt  $\frac{nk}{2}$  Kanten [50].

Der nächste Schritt besteht nun darin, die Kanten zufällig neu für  $p > 0$  anzugeordnen. Für  $p = 0$  wird jeder Knoten mit den  $k$  nächsten Nachbarn verbunden, sodass ein regulärer Graph entsteht.

Für  $p \neq 0$  wird aus dem regulären Graphen zunächst ein Knoten  $v_i$  gewählt, sowie eine Kante, die den Knoten  $v_i$  mit einem Nachbarn  $v_j$  verbindet. Dieses Verfahren erfolgt im Uhrzeigersinn. Mit der Wahrscheinlichkeit von  $p$  wird die Verbindung zwischen den Knoten  $v_j$  gelöst und zufällig mit einem anderen Knoten innerhalb des Netzwerkes verbunden. Für die zufällige Verbindung zwischen zwei Knoten existieren Regeln, sodass doppelte Kanten nicht erlaubt sind und die Verteilung gleichmäßig erfolgt. Kommt es zur Bildung einer doppelten Kante, so wird die ursprüngliche Verbindung zwischen dem Knoten  $v_i$  und  $v_j$  wiederhergestellt. Durch die gleichmäßige Verteilung soll zudem vermieden werden, dass der Knotengrad eines Knotens zu groß wird. Das Verfahren wird nun für jeden Knoten und seinen ersten Nachbarn innerhalb des Netzwerkes wiederholt, bis die Ausgangsposition erreicht ist. In der nächsten Iteration wird diese Prozedur wiederholt, diesmal für den zweiten Nachbarn des Knoten  $v_i$ . Auch hier wird jeder Knoten im Uhrzeigersinn betrachtet und die Kanten zu dem jeweils zweiten Nachbarn mit einer Wahrscheinlichkeit von  $p$  neu verbunden. Dieses Verfahren endet, sobald jeder Nachbar und somit auch jede Kante einmal betrachtet wurde und mit der Wahrscheinlichkeit von  $p$  neu verbunden wurde [50].

Für einen Wert von  $p = 1$  wird jeder Knoten innerhalb des Netzwerkes zufällig mit weiteren Knoten verbunden. Es entsteht ein komplett zufälliger Graph. Bei dem

hier genutzten Watts Strogatz Modell handelt es sich um das Modell von Newman [34]. So werden hier keine Kanten gelöscht, wie es bei dem ursprünglichen Modell [50] der Fall ist, sondern Kanten hinzugefügt.

Für  $0 < p < 1$  werden sogenannte "Small-World Networks" [50] erzeugt. Diese Graphen zeichnen sich durch die Verkürzung der Pfade zwischen zwei Knoten durch die zufällige Verteilung der Kanten aus. So repräsentieren sie in etwa die Gegebenheiten, die bei realen Netzwerken [50] zumeist vorliegen.

Die Abb. 8 zeigt, wie sich der Graph durch Veränderung des Wertes  $p$  von einem regulären Graphen in einen zufälligen Graphen wandelt.

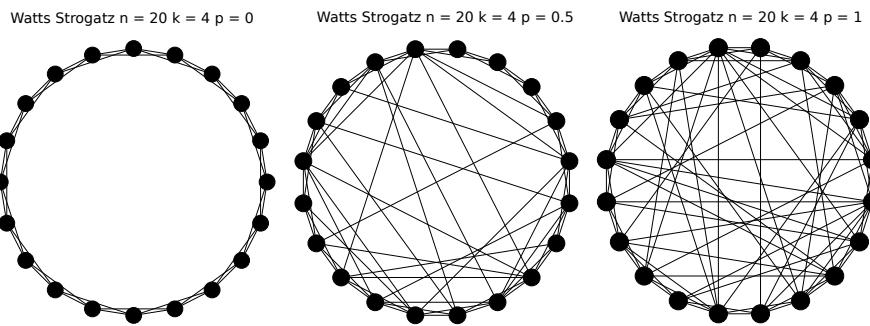


Abbildung 8: Das Watts Strogatz Modell mit unterschiedlichen  $p$  Wert. Die linke Abb. Zeigt einen regulären Graphen mit  $p = 0$ . Jeder Knoten besitzt den gleichen Anteil an Kanten. Die mittlere Abb. Zeigt einen zufälligen Graphen. Jeder Knoten erhält mit der Wahrscheinlichkeit  $p = 0.5$  zusätzliche Kanten. Die rechte Abb. Zeigt einen vollkommen zufälligen Graphen mit  $p = 1$ .

In der folgenden Tabelle sind die Watts Strogatz Graphen beschrieben, die für die Experimente genutzt werden.

Tabelle 2: Netzwerke

Netzwerk	Knoten ( $n$ )	Kanten	Kanten pro Knoten ( $m$ )	Wahrscheinlichkeit ( $p$ )
Watts Strogatz	200	400	5	0
Watts Strogatz	200	500	-	$\frac{1}{4}$
Watts Strogatz	200	539	-	$\frac{1}{3}$
Watts Strogatz	200	587	-	$\frac{1}{2}$
Watts Strogatz	200	668	-	$\frac{2}{3}$
Watts Strogatz	200	697	-	$\frac{3}{4}$
Watts Strogatz	200	800	-	1
<hr/>				
Watts Strogatz	200	1000	10	0
Watts Strogatz	200	1240	-	$\frac{1}{4}$
Watts Strogatz	200	1339	-	$\frac{1}{3}$
Watts Strogatz	200	1520	-	$\frac{1}{2}$
Watts Strogatz	200	1660	-	$\frac{2}{3}$
Watts Strogatz	200	1739	-	$\frac{3}{4}$
Watts Strogatz	200	2000	-	1

Tabelle 3: Ein Überblick über die Watts Strogatz Netzwerke

Was sich bei diesen Modellen ändert ist der Wert  $p$ . Dadurch wird die Zufälligkeit des Algorithmus beeinflusst und die Anzahl der Kanten wird erhöht.

### 3.4.2 Die Modularität des Watts Strogatz Modell

Für die Watts Strogatz Modelle existiert eine Besonderheit hinsichtlich der Modularität, die hier beschrieben wird. Mit dem Wachstum von  $p$  nimmt der Wert der Modularität entsprechend ab. Dies liegt an den zufälligen Graphen, die mit dem Wachstum von  $p$  erzeugt werden [35]. So nimmt die Modularität  $Q$  mit dem Wert von  $p$  weiter ab. Die maximale Modularität  $Q$  für die Watts Strogatz Netzwerke ergibt sich wie folgt:

$$Q = (1 - p) \left[ 1 - \sqrt{\frac{(\langle k \rangle + 2)}{N}} \right] [35] \quad (10)$$

mit  $N$  für die Anzahl der Knoten im Netzwerk mit einem durchschnittlichen Knotengrad von  $\langle k \rangle$ . Die Formel (10) zeigt: Läuft  $p$  gegen 1 so kann  $Q$  für ein Watts Strogatz Graphen nur gegen 0 laufen.

## 4 Framework und Experimente

In diesem Kapitel wird das Framework sowie die Experimente und die Analysen der Netzwerke vorgestellt. Zu Beginn folgt eine Beschreibung des in dieser Arbeit entwickelten Frameworks in Kapitel 4.1, mit welchem die Experimente im Abschnitt 4.2 durchgeführt wurden. Neben dem Basis LPA [40] lassen sich mit dem Programm weitere Varianten des Algorithmus testen. Durch die Wahl der Parameter synchrone- oder asynchrone Variante, Wahl des Label bei Gleichstand, Wahl der Hop-Distanzbestimmung, Wahl der initialen Labelverteilung und Festlegung des zu nutzenden Konvergenzkriteriums ergeben sich 480 Varianten des Algorithmus.

Der letzte Abschnitt 4.2 des Kapitels beschränkt sich auf die Experimente und Analyse der Ergebnisse, die mittels dem LPA erzeugt wurden. Bei der Experimentauswertung wird die Untersuchung der Konvergenz des Algorithmus unter zusätzlicher Berücksichtigung der Modularität untersucht. Es sollen Varianten gefunden werden, die konvergieren und an zweiter Stelle eine gute Modularität aufweisen.

### 4.1 Das Framework

In diesem Abschnitt des Kapitels folgt die Darstellung des Frameworks, das im Rahmen der Arbeit entwickelt<sup>1</sup> wurde. Mithilfe des Programms wird die Arbeitsweise des LPA visuell dargestellt. Durch diese visuelle Darstellung des Algorithmus lässt sich am ehesten erfassen, ob der LPA konvergiert, oder ob es zu einer Oszillation innerhalb des Netzwerkes kommt. Diese Beobachtungen können sodann in die Analyse der Daten einlaufen.

Neben dieser visuellen Darstellung des Graphen erfolgt eine Visualisierung der Modularität, die sich pro Iteration des LPA ergibt. Auf diese Weise lässt sich darstellen, wie die Wahl der Labels die Modularität beeinflusst. Zur Veranschaulichung wird in Abb. 9 das Framework für den Algorithmus präsentiert.

Die Anwendung ist optimal nutzbar bei Graphen mit einer Knotenmenge von maximal 100 bis 150 Knoten, sowie maximal 1000 bis 1100 Kanten. Bei komplexeren Graphen verringert sich die Übersichtlichkeit, sowie die Performance des Programms.

---

<sup>1</sup>Das Framework wurde mit Python 3 unter Nutzung von PyQt5 Ver. 5.15.4 entwickelt

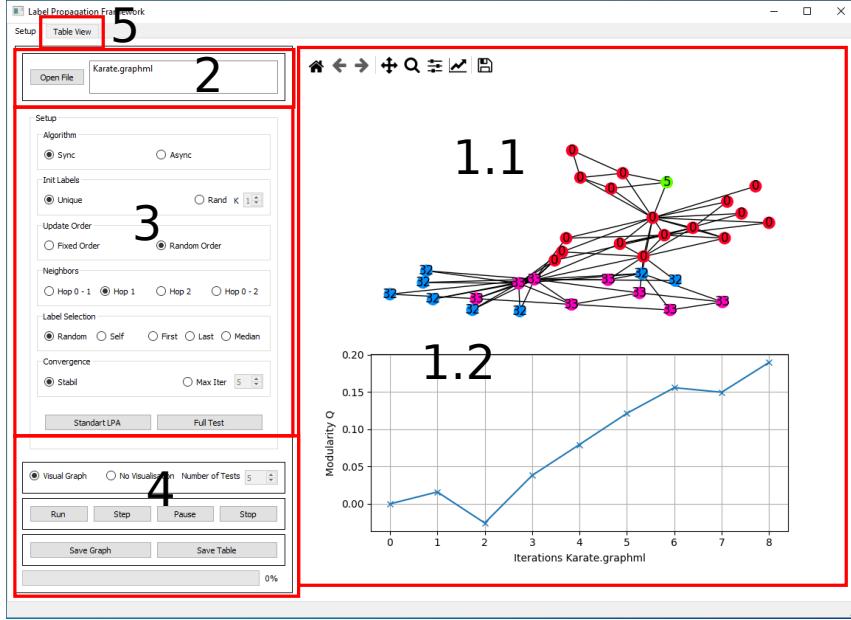


Abbildung 9: Das Framework zum Label Propagation Algorithmus, angewendet auf das Netzwerk des Karate Clubs. Die jeweiligen Nummern stellen die einzelnen Abschnitte des Programmes dar, die in Abschnitt 4.1.1 beschrieben werden.

#### 4.1.1 Aufbau des Frameworks

Das Framework aus Abb. 9 wurde in mehrere Teilabschnitte unterteilt, um die jeweiligen Bereiche entsprechend beschreiben zu können. In dem Bereich (1.1 - 1.2) aus Abb. 9 folgt die Visualisierung des Graphen, nachdem eine Datei über den Bereich (2) geladen wurde. Die Modularität im Bildbereich (1.2) wird nach jeder Iteration des LPA berechnet und aktualisiert. Dadurch lässt sich nachverfolgen, wie die Wahl der Labels die Modularität beeinflusst.

Die Einstellungen, mit welchen Parametereinstellung der Algorithmus ausgeführt werden soll, befinden sich im Abschnitt (3) der Abb. 9. Die Standardeinstellung des Programms entspricht dem Standard LPA [40].

Im Bereich (4) wird der Algorithmus gestartet. Hier ist es möglich den Algorithmus schrittweise laufen zu lassen, um so die Entwicklung der Community sichtbar zu machen. Neben diesen Funktionen kann der Algorithmus pausiert und später wieder aufgenommen oder abgebrochen werden. Der Algorithmus beendet hier die aktuelle Iteration. Die Ergebnisse eines jeden Testlaufes werden in einer Tabelle gespeichert, die über den Punkt (5) in Abb. 9 erreicht werden kann um sie zu betrachten oder zu speichern. Wird der Algorithmus vorzeitig abgebrochen, gehen die Ergebnisse nicht verloren, sondern werden in die Tabelle geladen.

Zusätzlich bietet das Programm die Möglichkeit, den Algorithmus mehrmals mit derselben Einstellung laufen zu lassen, ohne dass eine Visualisierung erfolgt. Wie

bereits in Abschnitt 3.2 beschrieben wurde kann es zu unterschiedlichen Ergebnisse pro Anwendung des LPA kommen. Zusätzlich können alle Varianten des Algorithmus an einem oder mehreren Netzwerken durchgeführt werden. Je nach Netzwerken und Anzahl der Testläufen kann ein solcher kompletter Test mehrere Stunden bis Tage in Anspruch nehmen <sup>2</sup>, da hier 480 mögliche Einstellungen des Algorithmus getestet werden.

Dieser Wert setzt sich wie folgt zusammen: Für die Initialisierung existieren drei Möglichkeiten, zwei Möglichkeiten für die Reihenfolge der Knoten, vier Möglichkeiten für die Distanz zwischen den Nachbarn, weitere fünf Möglichkeiten für die Wahl eines Labels, zwei Möglichkeiten für die Konvergenz sowie zwei Möglichkeiten für die Aktualisierung der Labels. Daraus ergibt sich eine totale Anzahl an Varianten von:

$$480 = 2^3 * 3 * 4 * 5 \quad (11)$$

Das Ergebnis eines Testlaufes wird in einer .csv Datei gespeichert, damit die Daten später weiter untersucht werden können. Neben diesem Framework existiert eine weitere Version des Programms für die Konsole. Dieses Programm dient dem schnellen Testen, welches für die Experimente verwendet wurde.

## 4.2 Experimente

In diesem Abschnitt werden die Experimente beschrieben, durchgeführt und ausgewertet. Für die Experimente wurden mit dem Watts Strogatz Modell (siehe Kapitel 3.4.1) eine Reihe von Netzwerken mit unterschiedlichen  $p$  und  $k$  Werten erzeugt.

Wie bereits erwähnt, liegt der Fokus dieser Experimente auf der Auswertung und Maximierung der Konvergenz unter zusätzlicher Berücksichtigung der Modularität.

### 4.2.1 Erheben der Grunddaten

In diesem Abschnitt des Kapitels folgen die ersten Experimente mit dem Standard LPA. An jedem Netzwerk wurden 200 Testläufe durchgeführt, um die Variation der Ergebnisse, benannt in Abschnitt 3.2.1, zu bestimmen. Von diesen Werten wurde für die Tabelle 4 der Eintrag gewählt, der die höchste Modularität am Ende des Algorithmus aufzeigt.

Die Tabelle 4 präsentiert die Ergebnisse, die nach Anwendung des LPA auf das Netzwerk gemessen wurde. Diese Tabelle enthält Werte für die Konvergenz "Konv. Rate", die sich aus allen 200 Testläufen errechnet. Im Verhältnis dazu, wie oft der

---

<sup>2</sup>Die Experimente wurde auf einem Windows 10 System mit einem AMD Ryzen 7 2700x CPU und 16 GB DDR4 RAM ausgeführt

Algorithmus terminiert, zeigt die Tabelle die Modularität ” $\mathcal{Q}$ ”, die beste gemessene Modularität ”*Best.  $\mathcal{Q}$* ”, sowie die durchschnittliche Modularität ” $\emptyset \mathcal{Q}$ ”, der Modularitätswert  $\mathcal{Q}$  sowie der Wert in ”*Best.  $\mathcal{Q}$* ” wurden mit dem Testlauf aus Spalte ”*Test #*” erreicht. Die durchschnittliche Modularität errechnet sich auf Grundlage von  $\mathcal{Q}$  über alle 200 Testläufe. Der Wert ”*Diff.  $\emptyset \mathcal{Q}$* ” beschreibt wie die Varianten des LPA sich auf die durchschnittliche Modularität auswirken, die mittels dem LPA erreicht wird. Die Spalte ” $\emptyset$  Iter.” zeigt an, wie viele Iterationen der LPA im Durchschnitt benötigt, bis er konvergiert oder zum Abbruch kommt.

Die Spalte ”*GM  $\mathcal{Q}$* ” der Tabelle stellt einen zweiten Algorithmus dar, den Greedy Modularity Algorithmus [33]. Damit soll ein Vergleich zwischen der Modularität beider Algorithmen hergestellt werden.

Watts Strogatz Modell n 200 k = 5								
Parameter	Test #	$\mathcal{Q}$	Best. $\mathcal{Q}$	$\emptyset \mathcal{Q}$	Diff. $\emptyset \mathcal{Q}$	Konv. Rate	$\emptyset$ Iter.	GM $\mathcal{Q}$
$p = 0$	1	0.7889	0.8196	0.7068	0	18 %	993.335	0.7197
$p = \frac{1}{4}$	99	0.6581	0.6594	0.6028	0	23.5 %	629.56	0.6444
$p = \frac{1}{3}$	133	0.6185	0.6214	0.4384	0	39%	558.355	0.5758
$p = \frac{1}{2}$	163	0.5503	0.5563	0.1119	0	82.5 %	173.415	0.5305
$p = \frac{2}{3}$	135	0.4392	0.4653	0.0114	0	98 %	36.655	0.4666
$p = \frac{3}{4}$	114	0.4038	0.4069	0.0111	0	98 %	38.32	0.4629
$p = 1$	148	0.2749	0.3179	0.0027	0	99.5 %	22.015	0.4078
Watts Strogatz Modell n 200 k = 10								
Parameter	Test #	$\mathcal{Q}$	Best. $\mathcal{Q}$	$\emptyset \mathcal{Q}$	Diff. $\emptyset \mathcal{Q}$	Konv. Rate	$\emptyset$ Iter.	GM $\mathcal{Q}$
$p = 0$	173	0.7401	0.7492	0.6788	0	16.5 %	843.375	0.5908
$p = \frac{1}{4}$	161	0.6065	0.6102	0.5696	0	41 %	233.855	0.5257
$p = \frac{1}{3}$	192	0.5608	0.5621	0.5273	0	43.5 %	141.115	0.4738
$p = \frac{1}{2}$	113	0.4869	0.4888	0.3147	0	62.5 %	150.06	0.4059
$p = \frac{2}{3}$	147	0.4483	0.4499	0.2367	0	68 %	96.28	0.3616
$p = \frac{3}{4}$	31	0.4378	0.4378	0.2568	0	77.5 %	43.045	0.3815
$p = 1$	99	0.3479	0.3524	0.0052	0	100 %	20.945	0.3034

Tabelle 4: Die Tabelle zeigt die Auswertung der Daten nach 200 Iterationen mit dem LPA ohne Modifikationen. Die Beschreibung der Spalten kann Abschnitt 4.2.1 entnommen werden.

Die Tabelle 4 präsentiert die Ergebnisse der Testläufe nach Anwendung des LPA. Bei den Watts Strogatz Modell zeigt sich, dass sich die Konvergenz und Modularität entgegengesetzt entwickeln. Bei zunehmender Konvergenz fällt die Modularität und andersrum. Dieses Verhalten zeigt sich auch in dem Greedy Modularity Algorithmus. Das sich die Modularität mit Zunahme von  $p$  verringert wird im Abschnitt 3.4.2 näher beschrieben. Für einen regulären Graphen  $p = 0$  wird die geringste Konvergenz verzeichnet. Die Modularität beider Algorithmen erreicht hier den höchsten Wert.

Bei einem vollständig zufälligen Graphen  $p = 1$  wird die höchste Konvergenz, jedoch die niedrigste Modularität erreicht.

Vermutlich sind die Konvergenz und die Modularität bei diesen Netzwerken abhängig voneinander. Es ist zu vermuten, dass der Wert  $p$  für diese Entwicklung verantwortlich ist. Ein kleines  $p$  führt bei dem Standard LPA zu einer geringen Konvergenz, mit steigendem  $p$  nimmt diese jedoch zu. Entsprechend entgegengesetzt verhält sich die Modularität. Mit steigendem  $p$  nimmt diese weiter ab [35]. So befindet sich die Konvergenz für  $0 \leq p \leq \frac{1}{3}$  unter 50 %. Dies bedeutet von den 200 Testläufen die durchgeführt wurden terminieren weniger als die Hälfte. Ab einem Wert von  $\frac{1}{2}$  für  $p$  steigt die Konvergenz.

Ein weiterer wichtiger Punkt betrifft die Anzahl der durchschnittlichen Iterationen, die der Algorithmus zur Konvergenz benötigt.

Die nachfolgende Abb. 10 zeigt anhand eines Curveplots, wie sich die Konvergenz (linke Grafik der Abb. 10) und durch einen Boxplot die Modularität (rechte Grafik der Abb. 10) für alle Testläufe entwickelt. Es zeigt sich für  $p \leq \frac{1}{3}$  konvergiert der LPA weniger oft, als mit einem größeren  $p$  Wert. Ab einem Wert von  $p \geq \frac{1}{2}$  liegt die Konvergenz im Schnitt viel höher. Pro Testlauf hier bei über 80 %. Bezogen auf die 200 Testläufe heißt es jeder 6te Testlauf würde nicht Konvergieren. Für  $p = 1$  zeigt sich, dass lediglich ein Lauf nicht zur Konvergenz geführt hat.

Aus dem Boxplot (rechte Grafik der Abb. 10) der Modularität lässt sich ablesen, dass der LPA bis zum Wert von  $p = \frac{1}{3}$  in der Lage ist, das Netzwerk geeignet in Communities zu unterteilen. Dies zeigt sich an den Boxen. Für einen regulären Graphen  $p = 0$  fällt die Modularität am höchsten aus. Mit dem Anstieg von  $p$  verringert sich dieser. Für einen Wert von  $p \geq \frac{1}{2}$  erzeugt der LPA vermehrt *null* Communities (siehe Abb. 12), was sich an dem Median (rote Linie) zeigt. Es kommt vereinzelt zur Bildung von Communities, die eine gute Modularität vorzeigen, was sich an den Ausreißern + (rechten Grafik der Abb. 10) zeigt. Jedoch überwiegt der Anteil an *null* Communities.

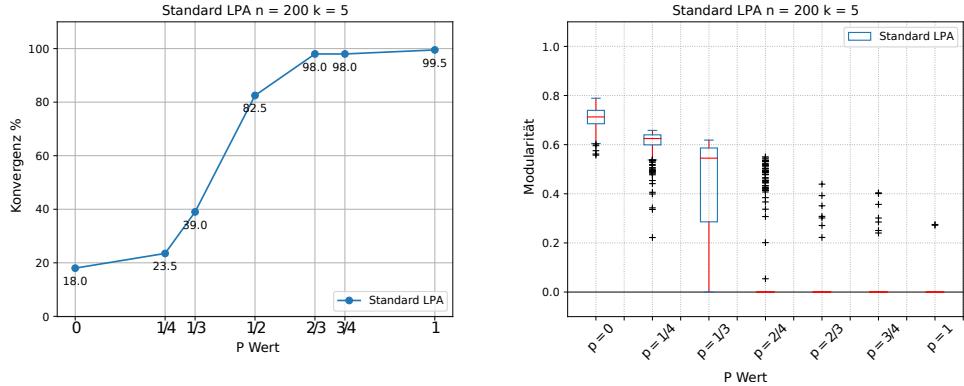


Abbildung 10: Die Konvergenz des LPA wird auf der y-Achse in % in der linken Grafik dargestellt. Der Wahrscheinlichkeitswert  $p$  wurde auf die x-Achse in beiden Grafiken aufgetragen. Die Konvergenz des LPA wird durch die Blauen Linie gezeigt. So steigert sich die Konvergenz des Algorithmus von 18% für  $p = 0$  bis zu 99.5% für  $p = 1$ . Die rechte Grafik zeigt die Entwicklung der Modularität anhand von einem Box-Plot. Auf der y-Achse wurde die Modularität aufgetragen. Für einen regulären Graphen  $p = 0$  wird die höchste Modularität gemessen. Dieser Wert nimmt mit dem Wachstum von  $p$  ab. Für einen vollkommen zufälligen Graphen wird eine Modularität von Null erreicht.

Ein etwas anderes Bild zeigt sich in Abb. 11 für  $k = 10$ . Die Konvergenz des Algorithmus (linke Grafik Abb. 11) macht nicht wie für  $k = 5$  einen extremen Anstieg zwischen  $p = \frac{1}{3}$  und  $p = \frac{1}{2}$ , sondern steigert sich mit wachsendem  $p$  in kleineren Schritten. Offensichtlich erreicht der LPA mit  $k = 5$  und steigendem  $p$  schneller eine hohe Konvergenz als mit  $k = 10$ .

Die Modularität (rechte Grafik Abb. 11) zeigt hier ein anderes Verhalten. Zwar kommt es auch hier zu einem Abfall der Werte, jedoch kommt es erst mit  $p = 1$  vermehrt zur Bildung einer *null* Community (siehe Abb. 12). Für  $\frac{1}{2} \leq p \leq \frac{3}{4}$  kommt es häufiger zu Schwankungen hinsichtlich der Community Bildung mittels des LPA, sodass die Modularität hier einen größeren Bereich beschreibt. Weiterhin liegt der Median hier weit über dem Wert 0. Für  $p = \frac{1}{3}$  zeigt sich zusätzlich, dass der LPA hier vermehrt Communities erkennt, die einander von dem Modularitätswert her sehr ähnlich sind. Das zeigt sich an der minimalen Schwankung und der recht schmalen Box.

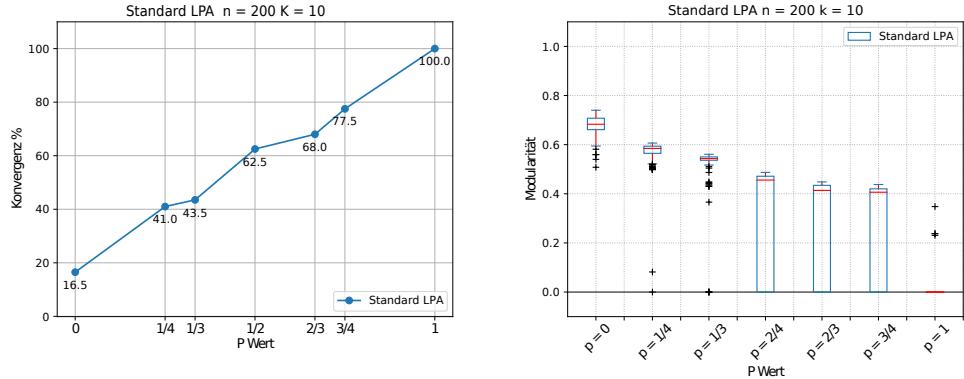


Abbildung 11: Die Konvergenz des LPA wird auf der y-Achse in % in der linken Grafik dargestellt. Der Wahrscheinlichkeitswert  $p$  wurde auf die x-Achse in beiden Grafiken aufgetragen. Anhand der Blauen Linie zeigt sich, dass der LPA bei diesem Netzwerk langsamer eine hohe Konvergenz erreicht im Vergleich zu dem Watts Strogatz Netzwerk mit  $k = 5$  aus Abb. 10. Die rechte Grafik zeigt die Entwicklung der Modularität anhand von einem Box-Plot. Auf der y-Achse wurde die Modularität aufgetragen. Für einen regulären Graphen  $p = 0$  wird die höchste Modularität gemessen. Dieser Wert verringert sich in kleineren Schritten mit dem Wachstum von  $p$  im Vergleich zu den Werten mit  $k = 5$  aus Abb. 10.

Die nachfolgende Abbildung 12 stellt die Entstehung einer *null* Community dar. Es zeigen sich noch farbliche Unterschiede der Labels, die Schritt für Schritt von dem roten Label dominiert werden. Zum Schluss des Algorithmus beschreibt das Netzwerk eine einzige Community, wodurch sich ein Modularitätswert von null ergibt.

Watt Strogatz Modell  $n = 200$   $k = 10$   $p = 1$

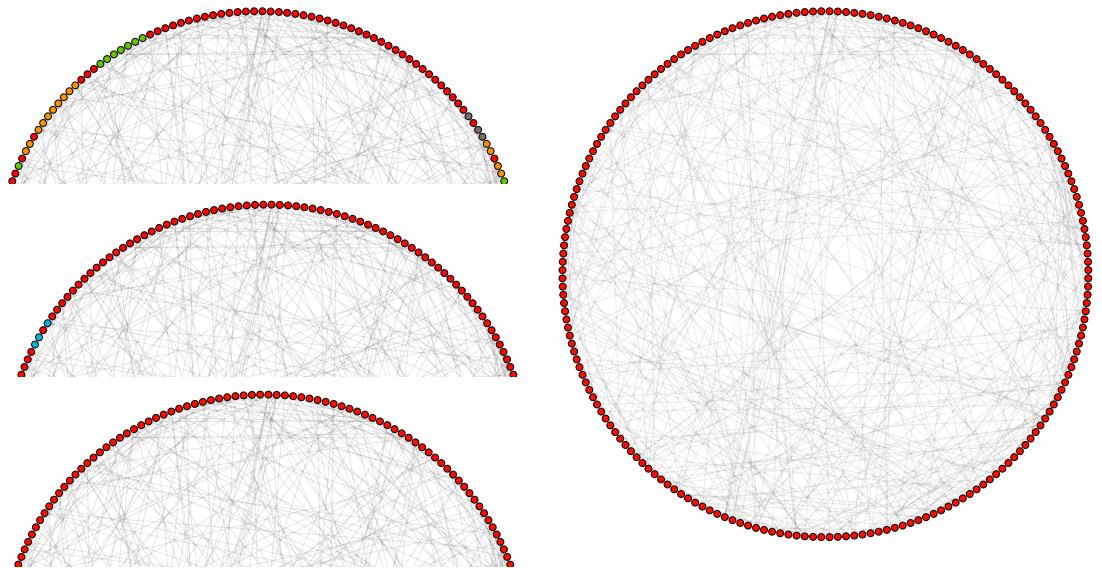


Abbildung 12: Entstehung einer *null* Community für ein Watts Strogatz Netzwerk mit  $n = 200$ ,  $k = 10$ ,  $p = 1$ . Die erste Grafik (oben links) zeigt, dass das Netzwerk zu Beginn noch aus mehreren Communities besteht, was sich an den unterschiedlich gefärbten Knoten zeigt. Die zweite Grafik (Mitte links) zeigt, wie das rote Label an Dominanz gewinnt, sodass in dem Ausschnitt nur noch zwei Farben rot und blau zu erkennen sind. Die dritte Grafik (unten links) wie auch die vierte Grafik (rechts) zeigen, wie das Netzwerk aus einer Community besteht.

Nachdem die Grunddaten ermittelt wurden, und eine erste Auswertung der Daten mittels dem LPA erfolgte, folgen nun die Anwendung einiger Varianten des LPA. Von den 480 Möglichkeiten hier, werden lediglich ein paar verwendet. Mit diesen Varianten soll ermittelt werden, ob eine Version existiert, die unabhängig von  $p$  dauerhaft eine hohe Konvergenz erreicht und zusätzlich die Modularität steigert.

#### 4.2.2 Anwendung der Varianten

In diesem Abschnitt folgen die Anwendungen der Varianten auf die Watts Strogatz Netzwerke. Mit diesen Varianten soll eine Steigerung der Konvergenz erreicht werden, bei gleichbleibender oder im besten Fall steigender Modularität, insbesondere mit dem Wachstum von  $p$ .

Im Folgenden wird analysiert, warum der Algorithmus nicht konvergiert.

Der Standard LPA ist so beschrieben, dass die Labels der Iteration  $\mathcal{T}$  den Labels der Iteration  $\mathcal{T} - 1$  entsprechen müssen, in diesem Falle terminiert der Algorithmus.

Kommt es nun zu der Situation, dass zwei Knoten pro Iteration jedes mal die Label tauschen, so kann der LPA nicht konvergieren. Es entsteht eine Oszillation.

Eine Oszillation liegt vor, wenn das Label der Iteration  $\mathcal{T}$  den Labels der Iteration  $\mathcal{T}-2$  entsprechen und Iteration  $\mathcal{T}+1$  entspricht der Iteration  $\mathcal{T}-1$ . Bei solch einem Verhalten wird der Algorithmus abgebrochen und es wird als nicht konvergierend gewertet.

Eine  $i$ -Oszillation liegt vor, wenn die Labels sich nach  $i$  Iterationen wiederholen. Dies bedeutet, während der Laufzeit des Algorithmus existierte schon einmal in einer früheren Phase eine solche Verteilung der Labels. Es kommt zu einer Schleifenbildung. Auch dieses Verhalten führt zu einem Abbruch des Algorithmus.

Ein Abbruch liegt vor, wenn der Algorithmus nach  $i$  Iterationen noch nicht terminiert. Die Anzahl der Iterationen wurde hier auf 1500 gesetzt. Es wurde aus dem Grund ein hoher Wert genommen, damit der Algorithmus genug Iterationen durchläuft um unter Umständen eine Oszillation/Schleifenbildung oder Konvergenz verzeichnen zu können. Im Schnitt benötigt der Algorithmus weit weniger Iterationen [40] als diese festgelegte Grenze.

Eine Oszillation kann durch zwei Wege entstehen. Die erste Möglichkeit ist, dass der Graph über eine Bipartite ähnliche Struktur, wie in Abschnitt 3.3.1 beschrieben, verfügt. Dies bedeutet die Knoten der einen Community besitzen mehr Verbindungen in eine andere Community und andersrum. Diese Knoten tauschen pro Iteration die Label.

Der zweite wesentliche Grund sind die Zufallsvariablen innerhalb des Algorithmus (siehe Abschnitt 3.2.1). Sowohl bei der Anordnung der Knoten wie auch bei der Auflösung des Gleichstandes. Durch eine ungünstig gewürfelte Reihenfolge kann es zu einer Oszillation kommen. Bei der Wahl mehrere Label kann zufällig das "falsche" Label gewählt werden, sodass es zu einer Oszillation führt.

Die nachfolgende Tabelle 5 zeigt die Parameter der Konfigurationen, die durch Oszillation, Schleifenbildung und Überschreitung der Iterationsgrenze wie oben erklärt, zum Abbruch des Algorithmus geführt haben.

Watts Strogatz Graphen n = 200							
Keine Konvergenz	k = 5 p = 0	k = 5 p = $\frac{1}{4}$	k = 5 p = $\frac{1}{3}$	k = 5 p = $\frac{1}{2}$	k = 5 p = $\frac{2}{3}$	k = 5 p = $\frac{3}{4}$	k = 5 p = 1
Oszillation	0	0	5	3	0	1	0
<i>i</i> Oszillation	133	127	81	28	4	3	0
Abbruch	31	25	36	4	0	0	1

Watts Strogatz Graphen n = 200							
Keine Konvergenz	k = 10 p = 0	k = 10 p = $\frac{1}{4}$	k = 10 p = $\frac{1}{3}$	k = 10 p = $\frac{1}{2}$	k = 10 p = $\frac{2}{3}$	k = 10 p = $\frac{3}{4}$	k = 10 p = 1
Oszillation	2	5	17	5	3	11	0
<i>i</i> Oszillation	149	109	95	64	57	34	0
Abbruch	16	4	1	6	4	0	0

Tabelle 5: Die Werte in der Tabelle geben an, wie oft der Algorithmus abgebrochen wurde. Die obere Tabelle stellt die Auswertung für  $k = 5$  mit unterschiedlichen  $p$  Wert dar. Die untere Tabelle zeigt die Auswertung der Daten für  $k = 10$ . Anhand der Werte hinter der jeweiligen Abbruchbedingung zeigt sie, wie viele der 200 Testläufe abgebrochen wurden. Eine der Hauptgründe warum der Algorithmus nicht terminiert liegt an der Schleifenbildung.

Anhand der Tabelle 5 lässt sich erkennen, wie oft der Algorithmus abgebrochen wurde. Für  $p \leq \frac{1}{2}$  kommt es häufiger zu solch einem Abbruch ausgelöst durch eine Schleifenbildung.

Wie in Abschnitt 3.3.1 beschrieben wurde in der Literatur [40] anhand von Experimenten gezeigt, dass sich eine Oszillation durch die asynchrone Version des Algorithmus unterdrücken lässt. So stellt sich eine Frage. Wirkt diese Variante auch dann, wenn es sich bei der Oszillation um Schleifen handelt. Hierzu folgt die Anwendung der asynchronen Variante des Algorithmus. Alle weiteren Parameter des LPA bleiben unverändert.

Watts Strogatz Modell n = 200 k = 5								
Parameter	Test #	$\mathcal{Q}$	Best. $\mathcal{Q}$	$\emptyset \mathcal{Q}$	Diff. $\emptyset \mathcal{Q}$	Konv. Rate	$\emptyset$ Iter.	GM $\mathcal{Q}$
$p = 0$	131	0.7836	0.813	0.6495	-0.0573	84.5 %	489.115	0.7197
$p = \frac{1}{4}$	129	0.6655	0.6678	0.6061	0.0033	57 %	333.455	0.6444
$p = \frac{1}{3}$	71	0.6222	0.6262	0.4498	0.0114	59.5%	358.4	0.5758
$p = \frac{1}{2}$	62	0.5381	0.5488	0.1113	-0.0006	90.5 %	85.27	0.5305
$p = \frac{2}{3}$	184	0.4789	0.4849	0.0237	0.0123	98.5 %	28.68	0.4666
$p = \frac{3}{4}$	127	0.4704	0.4738	0.0041	-0.007	99.5%	31.04	0.4629
$p = 1$	8	0.4259	0.4281	0.0041	-0.0014	98.5 %	18.44	0.4078
Watts Strogatz Modell n = 200 k = 10								
Parameter	Test #	$\mathcal{Q}$	Best. $\mathcal{Q}$	$\emptyset \mathcal{Q}$	Diff. $\emptyset \mathcal{Q}$	Konv. Rate	$\emptyset$ Iter.	GM $\mathcal{Q}$
$p = 0$	146	0.7216	0.7469	0.6145	-0.0643	81.5 %	392.145	0.5908
$p = \frac{1}{4}$	155	0.6084	0.6084	0.5658	-0.0038	58 %	89.635	0.5257
$p = \frac{1}{3}$	96	0.5651	0.5664	0.5109	-0.0164	70 %	74.745	0.4738
$p = \frac{1}{2}$	146	0.4855	0.4855	0.2508	-0.0639	90 %	36.36	0.4059
$p = \frac{2}{3}$	141	0.4467	0.4482	0.1412	-0.0955	88.5 %	25.26	0.3616
$p = \frac{3}{4}$	125	0.4325	0.4325	0.1319	-0.1249	97.5 %	16.735	0.3815
$p = 1$	9	0.3430	0.3443	0.0017	-0.0035	100 %	7.6	0.3324

Tabelle 6: Die Tabelle zeigt die Auswertung der Daten nach 200 Testläufe unter Anwendung der asynchronen Version des LPA. Die obere Tabelle stellt die Werte für  $k = 5$  und die untere Tabelle die Werte für  $k = 10$  dar. Es zeigt sich eine Steigerung der Konvergenz im Vergleich zu dem Standard LPA. Die Bezeichnungen der Spalten ist dieselbe wie in Tabelle 4 und kann dem Abschnitt 4.2.1 entnommen werden.

Anhand der Tabelle 6 zeigt sich, dass sich die Konvergenz (Tabelle 6, Spalte *Konv. Rate*) unter Anwendung der asynchronen Variante im Vergleich zur synchronen (siehe Tabelle 4) verbessert hat. Auch hinsichtlich der durchschnittlichen Anzahl an Iterationen sorgt die asynchrone Version des LPA für eine Verbesserung. Bei genauer Betrachtung der Werte zwischen  $\frac{1}{4} \leq p \leq \frac{1}{3}$  fällt jedoch auf, dass die Konvergenz hier niedriger ausfällt im Vergleich zu den übrigen Werten der Tabelle. Offenbar liegt zwischen diesem Intervall eine Besonderheit vor, sodass die Werte hier geringer ausfallen.

Die Modularität (Tabelle 6, Spalte  $\mathcal{Q}$ ) hat sich in gewissen Bereichen im Vergleich zu dem Standard LPA minimal verbessert, wie auch verschlechtert. Diese Verschlechterung der Modularität zeigt sich auch an den Werten der durchschnittlichen Modularität wie die Spalte *Diff.  $\emptyset \mathcal{Q}$*  zeigt. Beziiglich des Greedy Modularity Algorithmus (Spalte *GM  $\mathcal{Q}$* ) zeigt die asynchrone Variante die besseren Werte.

Die Grafik in Abb. 13 stellt die beiden Varianten des LPA für  $k = 5$  gegenüber. Die asynchrone Variante zeigt für  $p = \frac{1}{4}$ , sowie für  $p = \frac{1}{3}$  einen Einbruch der Konvergenz, wie die linke Grafik in Abb. 13 zeigt. Hier scheint diese Version des Algorithmus auf ein Problem zu stoßen, wie oben bereits erwähnt. Mit Steigerung des Wertes  $p$

verbessert sich auch die Konvergenz.

Ein Vergleich der Modularität (rechte Grafik Abb. 13) zeigt, dass sich die Werte hier mit minimalen Unterschieden beinahe identisch entwickeln. Lediglich für  $p = 0$  zeigt die asynchrone Variante, dass durchschnittlich mehr Testläufe eine Community Aufteilung so durchführen, dass sich die daraus ergebene Modularität unterhalb der synchronen Variante anordnet.

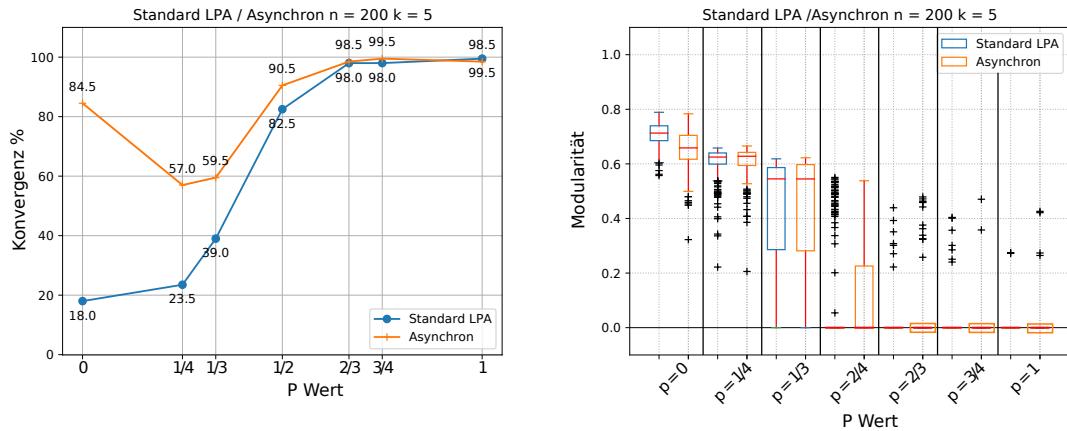


Abbildung 13: Die linke Grafik stellt die Konvergenz der Standard Variante (blaue Linie) und asynchronen Variante (orange Linie) des LPA gegenüber. Bereits für  $p = 0$  erreicht die asynchrone Variante eine hohe Konvergenz. Zwischen dem Intervall  $p = \frac{1}{4}$  und  $p = \frac{1}{3}$  wird der geringste Wert für die Konvergenz der asynchrone Variante gemessen. Die Modularität (rechte Grafik) kann durch Anwendung der asynchronen Variante nicht verbessert werden.

Für  $k = 10$  zeigt sich hinsichtlich der Konvergenz (linke Grafik Abb. 14) eine relativ ähnliche Entwicklung wie schon mit  $k = 5$  in Abb. 13. Auch hier zeigt sich, dass ein Tiefpunkt der Konvergenz für  $p = \frac{1}{4}$  erreicht wird. Mit dem Wachstum von  $p$  nimmt dieser Wert wieder zu.

Bezogen auf die Modularität (rechte Grafik Abb. 14) zeigt sich ein etwas anderes Verhalten ab  $p \geq \frac{2}{3}$  im Vergleich zum Standard LPA. Der Median liegt hier, anders als bei der synchronen Version, bei 0. Dies zeigt, dass der Algorithmus des öfteren das Netzwerk in nur eine Community unterteilt. Dies zeigt den Nachteil der asynchronen Version, da es hier vermehrt zur Bildung von "Super Communities" (siehe Abschnitt 3.2.2) kommt. Es spricht hier für eine Verschlechterung des LPA.

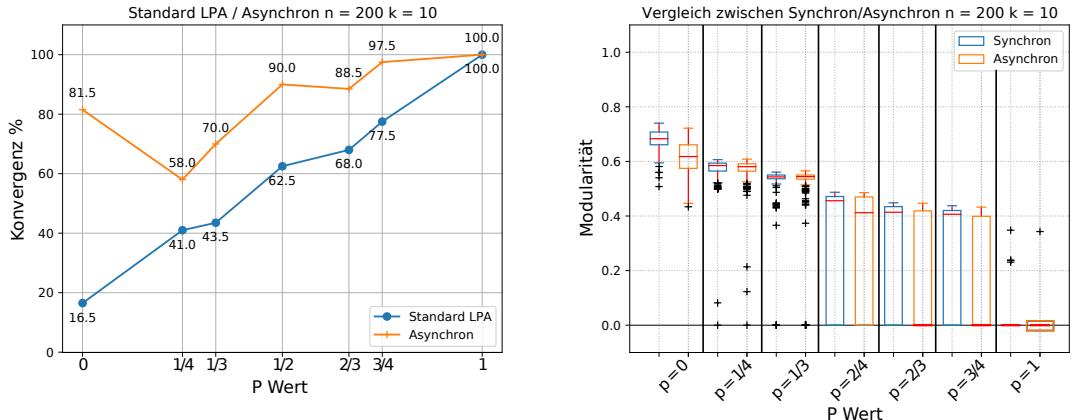


Abbildung 14: Die linke Grafik stellt die Konvergenz der Standard Variante (blaue Linie) und asynchronen Variante (orange Linie) des LPA gegenüber. Bereits für  $p = 0$  erreicht die asynchrone Variante wie schon in Abb. 13 eine hohe Konvergenz. Die Modularität (rechte Grafik) kann durch Anwendung der asynchronen Variante nicht verbessert werden. So liegt der Median der asynchronen Variante ab  $p = \frac{2}{3}$  bei Null.

Es zeigt sich, während mit der asynchronen Version eine hohe Konvergenz erreicht wird, verringert sich die Modularität mit dem Anstieg von  $p$ . Für  $k = 10$  Abb. 14 zeigt sich dies besonders deutlich ab einem Wert für  $p \geq \frac{2}{3}$ . Durch die asynchrone Version wird ab diesem Wert vermehrt eine *null* Community erzeugt, was sich an dem Median zeigt. Bei Anwendung der synchronen Variante werden seltener *null* Communities erzeugt. Für  $k = 5$  ist das Verhalten beider Varianten, bis auf kleine Abweichungen identisch.

Betrachtet man einzig die Konvergenz des Algorithmus, so wäre die asynchrone Variante des LPA zu bevorzugen. Anders sieht es aus, wenn die Modularität betrachtet wird. Hier würde die synchrone Version des LPA Sinn ergeben.

Aus der Tabelle 5 ist bekannt, dass es bei dem Watts Strogatz Modell zu einer Oszillation innerhalb des Netzwerkes kommt, unter Anwendung der synchronen Variante. Neben einer bipartiten Struktur kann auch die Wahl der Label zu einer Oszillation führen. Aus diesem Grund wird nun eine Variante gewählt, die diese Zufälligkeit des Algorithmus verringert soll. Dadurch soll eine Verbesserung der Konvergenz erreicht werden.

Die folgende Abbildung 15 zeigt einen Teil des Watts Strogatz Graphen mit  $n = 200$ ,  $k = 5$  und  $p = 0$ . Die zwei rot markierten Knoten besitzen je vier Nachbarn. Zwei der Nachbarn sind jeweils einer anderen Community zugeordnet. Hier kommt es zu einem Gleichstand der Label, was zu einer Oszillation führen kann. Dieses Problem lässt sich lösen, indem der aktuell betrachtete Knoten sein Label

behält, sofern es in der Menge der möglichen Label enthalten ist [3]. Dadurch lässt sich eine entsprechende Oszillation unterdrücken und führt zu einer Steigerung der Konvergenz [3].

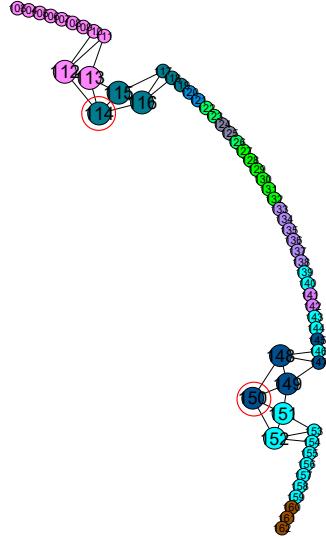


Abbildung 15: Jeder der Punkte stellt einen Knoten innerhalb des Netzwerkes dar. Zur Veranschaulichung sind die abweichenden Knoten vergrößert dargestellt. Beide rot markierten Knoten wechseln pro Iteration das Label. Da es hier pro Iteration zu einem Gleichstand der Label kommt. Es kommt zu einer Oszillation.

Der LPA wird so erweitert, wie bereits oben beschrieben (diese Variante wird im weiteren als LPA\_S bezeichnet). Die weiteren Parameter des Algorithmus entsprechen weiterhin dem Standard LPA. Die Tabelle 7 enthält die Daten nach Anwendung der Variante. Die Werte in Bezug auf die Konvergenz sind aussagekräftig.

Watts Strogatz Modell n = 200 k = 5								
Parameter	Test #	$\mathcal{Q}$	Best. $\mathcal{Q}$	$\emptyset \mathcal{Q}$	Diff. $\emptyset \mathcal{Q}$	Konv. Rate	$\emptyset$ Iter.	GM $\mathcal{Q}$
$p = 0$	3	0.6651	0.6651	0.6152	-0.1561	100 %	4.135	0.7197
$p = \frac{1}{4}$	43	0.5409	0.5409	0.4979	-0.1049	100 %	4.86	0.6444
$p = \frac{1}{3}$	26	0.5045	0.5045	0.4633	0.0249	100%	5	0.5758
$p = \frac{1}{2}$	145	0.4577	0.4577	0.4235	0.3116	100 %	5.12	0.5305
$p = \frac{2}{3}$	40	0.3936	0.3936	0.3683	0.3569	100 %	5.5	0.4666
$p = \frac{3}{4}$	177	0.3958	0.3958	0.3588	0.3477	100 %	5.5	0.4629
$p = 1$	179	0.3456	0.3456	0.3115	0.3071	100 %	5.645	0.4078
Watts Strogatz Modell n = 200 k = 10								
Parameter	Test #	$\mathcal{Q}$	Best. $\mathcal{Q}$	$\emptyset \mathcal{Q}$	Diff. $\emptyset \mathcal{Q}$	Konv. Rate	$\emptyset$ Iter.	GM $\mathcal{Q}$
$p = 0$	163	0.4715	0.4715	0.3787	-0.3001	100 %	2.98	0.5908
$p = \frac{1}{4}$	61	0.4143	0.4143	0.3513	-0.2183	100 %	3.06	0.5257
$p = \frac{1}{3}$	64	0.3886	0.3886	0.3348	-0.1925	100 %	3.25	0.4738
$10 p = \frac{1}{2}$	56	0.3495	0.3495	0.2995	-0.0152	100 %	3.445	0.4059
$p = \frac{2}{3}$	170	0.3203	0.3203	0.2621	0.0254	100 %	3.57	0.3616
$p = \frac{3}{4}$	8	0.3146	0.3146	0.2483	-0.0085	100 %	3.64	0.3815
$p = 1$	199	0.2498	0.2498	0.2004	-0.1952	100 %	3.84	0.3324

Tabelle 7: Auswertung der Daten, wenn ein Knoten  $v_i$  das Label hält, wenn diese in der Menge der Möglichen Label enthalten ist. Die obere Tabelle zeigt die Werte für  $k = 5$  und die untere Tabelle die Ergebnisse für  $k = 10$ . Die Konvergenz liegt für alle  $p$  Werte bei 100 %. Die Spalte  $Diff. \emptyset \mathcal{Q}$  zeigt, dass es zu einer Verringerung der Modularität kommt. Die Bezeichnungen der Spalten ist dieselbe wie in Tabelle 4 und kann dem Abschnitt 4.2.1 entnommen werden.

Der Tabelle 7 kann entnommen werden, dass mit dieser Version des LPA eine Konvergenz von 100% bei allen hier getesteten Watts Strogatz Netzwerken erreicht wird. Das Lösen des Gleichstandes scheint eine wichtige Rolle bei der Konvergenz zu spielen. Der Wert  $p$  hingegen besitzt weniger Einfluss auf die Konvergenz, wie zu Beginn zu vermuten war (siehe Abschnitt 4.2.1).

Ein weiterer wesentlicher Punkt betrifft die durchschnittliche Anzahl an Iteration (Tabelle 7, Spalte  $\emptyset$  Iter.). Bis der Algorithmus terminiert, werden im Schnitt weniger als 10 Iterationen benötigt, was anzeigt, dass der Algorithmus bezüglich der Anzahl an Iterationen effizienter ist.

Die Modularität zeigt im Vergleich zur synchronen und asynchronen Version eine Verschlechterung, wie die Spalte  $Diff. \emptyset \mathcal{Q}$  zeigt. Das Lösen des Gleichstandes, mit der hier getesteten Variante, führt weniger zu einer optimalen Bildung der Communities, was sich anhand der Modularität zeigt. Der Wert liegt hier weiterhin unter den Werten, die mittels des Greedy Modularity Algorithmus erreicht wurden (siehe Tabelle 7, Spalte GM  $\mathcal{Q}$ ).

Die folgende Abbildung 16 zeigt, dass mit der hier getesteten Variante des LPA immer eine Modularität erreicht wird, die über 0 liegt (linke Grafik Abb. 16). Dies

bedeutet, hinsichtlich der Community Bildung wurde eine Verbesserung erreicht. Es zeigt sich, dass die Werte der Modularität pro Testlauf nahe beieinander liegen. Daraus erschließt sich, dass pro Testlauf mit geringen Abweichungen identische Communities erzeugt werden. Für  $p < \frac{1}{2}$  und  $k = 5$  liegt die Modularität der LPA\_S Variante unterhalb der Werte, die mittels des Standard LPA erreicht wurden. Für  $p \geq \frac{1}{2}$  zeigt die LPA\_S Variante eine Verbesserung der Modularität. Für  $k = 10$  (linke Grafik Abb. 16) zeigt sich dieses erst für  $p = 1$ .

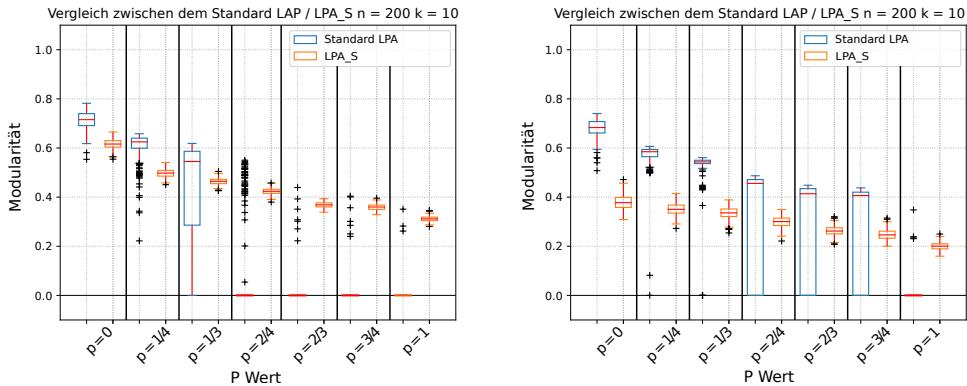


Abbildung 16: Die linke Grafik zeigt die Auswertung des Standard LPA (blaue Boxen) und der LPA\_S Variante (orange Boxen) mit  $k = 5$ . Die Modularität des Standard LPA liegt bis  $p = \frac{1}{3}$  oberhalb der Werte der Variante, so dass ein Knoten das Label hält. Diese Variante erreicht mit  $p = \frac{1}{2}$  eine Verbesserung der Modularität. Für  $k = 10$  (rechte Grafik) erreicht der LPA bis  $p = \frac{3}{4}$  die höchste Modularität. Für  $p = 1$  zeigt die zweite Variante das bessere Ergebnis.

Es hat sich gezeigt, wird das Problem des Gleichstandes gelöst, so wirkt sich dies positiv auf die Konvergenz des Algorithmus aus. Die Modularität zeigt zwar anders als bei dem Standard LPA geringere Werte auf, jedoch wird mit dieser Variante die Bildung von *null* Communities vermieden.

Im Folgenden soll eine weitere Variante getestet werden. In diesem Fall wird der Knoten  $v_i$  in die Betrachtung der Nachbarn mit einbezogen [24]. Es kann so verstanden werden, dass jeder Knoten über eine Schleife auf sich selbst verfügt. Es folgt die Anwendung der Hop 0 - 1 Distanz (diese Variante wird im folgenden als LPA Hop 0 - 1 bezeichnet). Die Darstellung einer Tabelle folgt hier nicht, sondern direkt die Auswertung der Daten in den Abbildungen 17, 18.

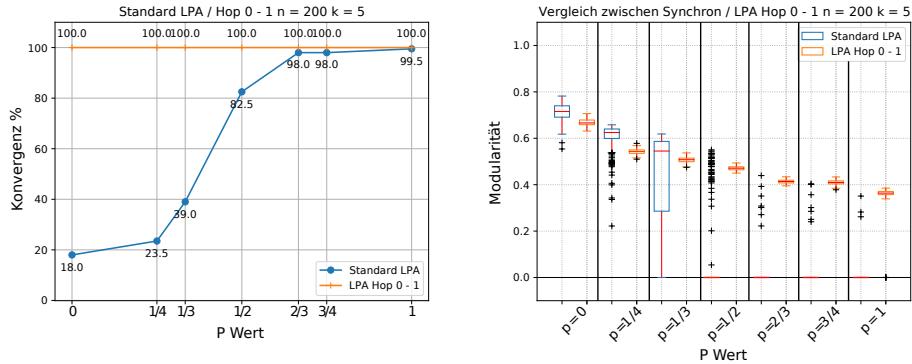


Abbildung 17: Die linke Grafik zeigt, wie sich die Konvergenz der Hop 0 - 1 Variante (orange Linie) im Vergleich zum Standard LPA (blaue Linie) für  $k = 5$  entwickelt. Die Hop 0 - 1 Variante erreicht eine Konvergenz von 100% für jeden Wert für  $p$ . Die rechte Grafik zeigt die Auswertung der Modularität für die Standard-Variante (blaue Box) sowie der Hop 0 - 1 Variante (orange Boxen). Bis  $p = \frac{1}{3}$  erreicht der Standard LPA die besseren Werte, während die Hop 0 - 1 Variante ab  $p = \frac{1}{2}$  eine Verbesserung zeigt. Dieses Verhalten konnte bereits in Abb. 16 beobachtet werden.

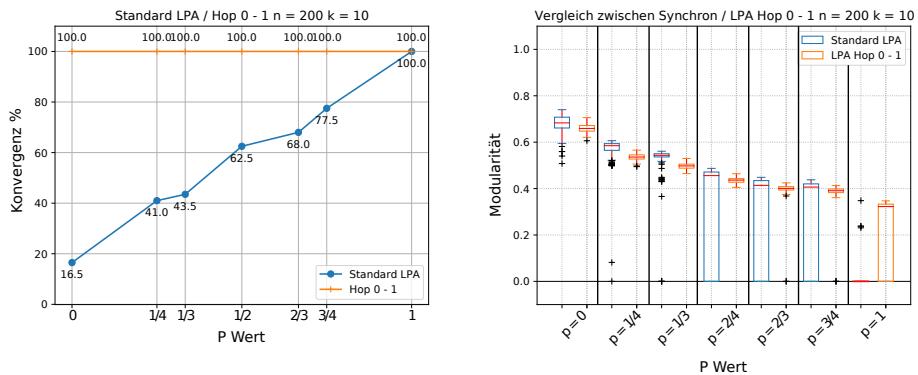


Abbildung 18: Für  $k = 10$  zeigt die Hop 0 - 1 Variante bezüglich der Konvergenz eine Identische Entwicklung wie für  $k = 5$  in Abb. 17. In der rechten Grafik zeigt sich, dass sich die Modularität der Hop 0 - 1 Variante bis  $p = \frac{3}{4}$  nur minimal von den Werten des LPA unterscheidet. Für  $p = 1$  erreicht die Hop 0 - 1 Variante, wie der Median zeigt eine Verbesserung der Modularität.

Sowohl für  $k = 5$  wie auch für  $k = 10$  zeigt sich eine Konvergenz von 100% mit der LPA Hop 0 - 1 Variante wie die linke Grafik in Abb. 17, 18 zeigt. Zwischen der LPA\_S und der LPA Hop 0 - 1 Variante zeigt sich diesbezüglich kein Unterschied.

Des Weiteren zeigt sich bei dieser Variante des Algorithmus eine Verbesserung der Modularität wie jeweils die rechte Grafik der Abb. 17, 18 zeigt. Für  $k = 5$  (rechte Grafik Abb. 17) liegt die Modularität für  $p \leq \frac{1}{3}$  unterhalb der Werte, die mit dem Standard LPA erreicht wurden. Für  $p \geq \frac{1}{2}$  zeigt die LPA Hop 0 - 1 Variante eine Verbesserung. Diese Verbesserung zeigte sich bereits durch die Anwendung der LPA\_S Variante in der linken Grafik der Abb. 16.

Wie die rechte Grafik der Abb. 18 zeigt, erreicht die LPA Hop 0 - 1 Variante für  $k = 10$  Modularitätswerte, die nahe an den Werten des Standard LPA liegen. Für  $p = \frac{2}{3}$  sowie  $p = \frac{3}{4}$  erreichen einige wenige Testläufe eine Modularität von 0, wie sich in der rechten Grafik Abb. 18 zeigt. Für  $p = 1$  zeigt die Hop 0 - 1 Variante eine größere Streuung auf, der Median liegt allerdings über Null. Bis  $p = 1$  zeigt sich, dass mit dieser Variante pro Testlauf Communities gebildet werden, die einander recht ähnlich sind. Dies zeigt sich durch die geringe Schwankung der Modularität. Dieses Verhalten konnte bereits bei  $k = 5$  in der Abb. 17 beobachtet werden.

Im Folgenden werden die LPA\_S und LPA Hop 0 - 1 Varianten bezüglich der Modularität verglichen. Die Abb. 19 stellt die Auswertung beider Varianten gegenüber.

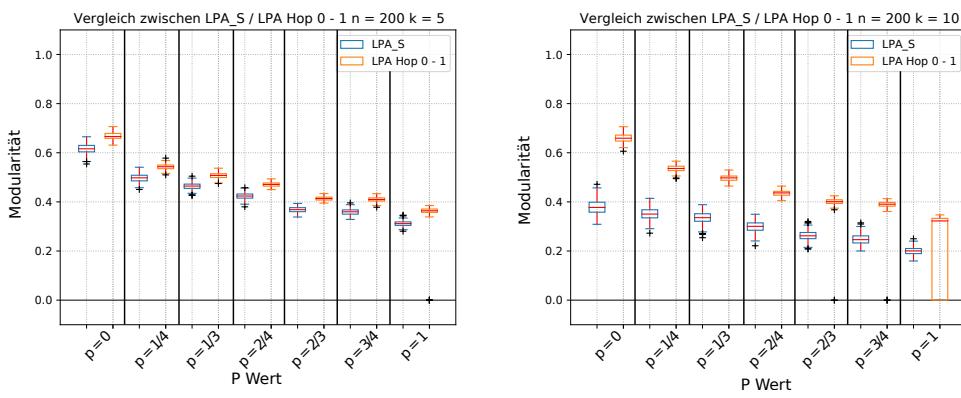


Abbildung 19: Die linke Grafik zeigt die Modularität für den LPA\_S (Blau) und der Hop 0 - 1 Variante (Orange) für  $k = 5$ . Die rechte Grafik zeigt die Auswertung für  $k = 10$ . In beiden Grafiken zeigt sich deutlich, dass die Hop 0 - 1 Variante höhere Ergebnisse erzielt. Weiterhin ist zu erkennen, dass beide Varianten schmale Boxen aufzeigen. Daraus lässt sich schließen, dass sich die gebildeten Communities pro Testlauf minimal unterscheiden. Ausnahmen bildet die Hop 0 - 1 Variante für  $p = 1$  und  $k = 10$  wie die rechte Grafik zeigt.

Es ist zu erkennen, dass die LPA Hop 0 - 1 Variante eine bessere Modularität im Vergleich zu dem LPA\_S erreicht. Deutlich wird dies für  $k = 10$  (rechte Grafik Abb. 19). Für einen vollkommen zufälligen Graphen  $p = 1$  weist die LPA Hop 0 - 1 Variante für  $k = 10$  eine deutliche Schwankung der Modularität auf. Ein solches Verhalten zeigt sich auch für  $k = 5$  mit  $p = 1$ . Bei der LPA Hop 0 - 1 Variante kommt es zu Ausreißern, die eine Modularität von Null erreichen. In beiden Fällen erweist sich die LPA\_S Variante, wenn auch mit der geringeren Modularität, als vorteilhafter.

Ausgehend von diesen Ergebnissen wäre die Hop 0 - 1 Variante der LPA\_S Variante zu bevorzugen. Neben einer Konvergenz von 100% erfolgt auch die Bildung der Communities optimaler, was sich an der Modularität zeigt. Eine Ausnahme bildet hier der Wert  $p = 1$ . Hier kann es deutlich zu Schwankungen der Community

Bildung kommen.

Eine mögliche Erklärung für dieses Verhalten wäre folgende: Wird der Knoten  $v_i$  bei der Betrachtung der Nachbarn mit einbezogen (Hop 0 - 1), so kann dies zu der Aufnahme des Labels des Knoten  $v_i$  in die Menge der maximalen Label führen. Besitzt der Knoten  $v_i$ ,  $m$  Nachbarn, wobei für  $m \geq j + l = m$  mit  $j < l$  und  $j + l = l$  gilt. Weiterhin gilt, die Menge  $j$  befindet sich in der Community  $\mathcal{C}_j$  und die Menge  $l$  in der Community  $\mathcal{C}_l$ . Der Knoten  $v_i$  gehört zu der Community  $\mathcal{C}_j$ . Wird der Knoten  $v_i$  nicht in die Betrachtung der Nachbarn einbezogen, so wird das Label der Community  $\mathcal{C}_l$  gewählt, da diese die größere Menge ausmacht. Wird der Knoten nun in die Berechnung einbezogen, so ergibt sich ein Gleichstand der Label. Vermutlich spielt genau dieser Gleichstand bei der Modularität eine wichtige Rolle. Zusätzlich ist es möglich, dass  $j = l$  gilt ohne den Knoten  $v_i$ . Durch Hinzunahme des Knoten  $v_i$  dominiert das Label der Community  $\mathcal{C}_j$ .

Dieses Verhalten ist bei der LPA\_S Variante nicht gegeben. Wenn  $j < l$  ist, so würde das Label des Knoten  $v_i$  wenn es zur Community  $\mathcal{C}_j$  gehört, nicht weiter berücksichtigt.

Es wurde vermutlich eine Variante gefunden, die zu einer hohen Konvergenz führt und für eine Verbesserung der Modularität bei dem Watts Strogatz Modell sorgt. In den nachfolgenden Experimenten werden weitere Varianten des LPA getestet. Dadurch sollen nach Möglichkeit weitere Varianten ermittelt werden, die Ergebnisse wie die des LPA\_S und der LPA Hop 0 - 1 Variante produzieren.

Der Beginn der nächsten Experimente erfolgt durch Festlegen einer zufälligen Reihenfolge der Knoten zu Beginn des Algorithmus. Die Anordnung der Knoten bleibt bis zum Ende des Algorithmus bestehen. Alle weiteren Parameter des LPA entsprechen dem Standard Algorithmus. Dadurch soll untersucht werden, ob die Reihenfolge Auswirkung auf die Community Bildung und den Algorithmus besitzt.

Der linken Grafik in Abb. 20 kann entnommen werden, dass sowohl der Standard LPA wie auch die LPA FO Variante ein ähnliches Konvergenzverhalten aufzeigen. Ein ebensolches Verhalten zeigt sich bei der Community Bildung. Dies kann der rechten Grafik der Abb. 20 entnommen werden. Für  $p < \frac{1}{2}$  zeigen der Standard LPA wie die LPA FO Variante mit leichten Abweichungen ähnliche Werte bezüglich der Modularität auf. Für  $p = \frac{1}{2}$  zeigt die LPA FO Variante, dass mehrere Testläufe eine Modularität über Null erreichen. Bei dem Standard LPA erreichen weniger Testläufe einen solchen Wert, was die Ausreißer zeigen. Ab  $p > \frac{1}{2}$  liegt der Median beider Algorithmen bei Null.

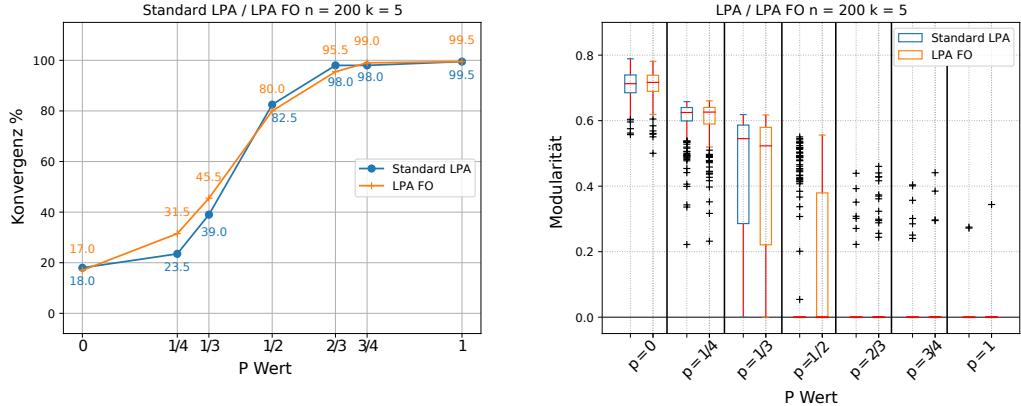


Abbildung 20: Die linke Grafik zeigt wie sich die Konvergenz des Standard LPA (blaue Linie) und der LPA FO Variante (orange Linie) für  $k = 5$  verhält . Erkenntlich wird, dass die Konvergenz der LPA FO Variante sich nur minimal von der des Standard LPA unterscheidet. Für  $p = \frac{1}{4}$  wie auch für  $p = \frac{1}{3}$  zeigt die LPA FO Variante die bessere Konvergenz. Die rechte Grafik zeigt die Modularität beider Varianten für  $k = 5$ . Bis  $p = \frac{1}{3}$  erreichen beide Varianten mit minimalen Abweichungen identische Werte. Für  $p = \frac{1}{2}$  weist die LPA FO Variante eine größere Streuung auf, sodass mehrere Testläufe einen Wert über Null erreichen. Mit dem Wachstum von  $p$  zeigen beide Varianten, dass sie zur Bildung von *null* Communities neigen.

Für  $k = 10$  zeigt sich eine ähnliche Entwicklung hinsichtlich der Konvergenz wie auf der Modularität, was die Abb. 21 zeigt.

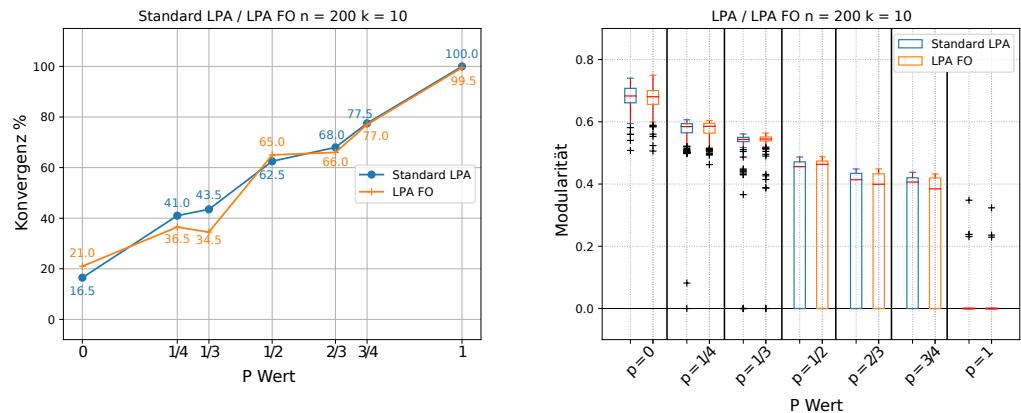


Abbildung 21: Die linke Grafik zeigt, wie sich die Konvergenz des Standard LPA (blaue Linie) und der LPA FO Variante (orange Linie) für  $k = 5$  verhält. Der Standard LPA erreicht durch eine sich änderte Reihenfolge für  $p = \frac{1}{4}$  und  $p = \frac{1}{3}$  die bessere Konvergenz. Mit dem Wachstum von  $p$  zeigen beide Algorithmen mit leichten Abweichungen eine ähnliche Konvergenz auf. Die rechte Grafik zeigt die Modularität beider Varianten für  $k = 10$ . Mit minimalen Abweichungen unterscheidet sich die Modularität dieser Varianten nur minimal voneinander, wie schon in Abb. 20 für  $k = 5$ .

Ein Vergleich beider Algorithmen lässt vermuten, dass die Reihenfolge der Knoten weniger eine Rolle spielt. Beide Varianten zeigen sowohl für  $k = 5$  wie auch für  $k = 10$ , wenn auch mit geringen Abweichungen, eine ähnliche Konvergenz wie auch

Modularität auf.

Bei der nächsten Variante die getestet wird, erfolgt eine Erweiterung der Hop-Distanz. Hier werden nun auch die Nachbarn der Nachbarn des Knoten  $v_i$  betrachtet. Dadurch soll untersucht werden, ob sich dies positiv auf die Konvergenz und Modularität auswirkt.

Der linken Grafik aus Abb. 22 kann entnommen werden, dass die LPA Hop 0 - 2 Variante im Vergleich zum Standard LPA bereits ab  $p = \frac{1}{4}$  eine hohe Konvergenz erreicht. Die rechte Grafik in Abb. 22 zeigt, wie sich die Modularität der LPA Hop 0 - 2 Variante verhält. Für einen regulären Graphen zeigt diese Variante eine ähnliche Modularität, wie der Standard LPA. Ab dem Wert  $p \geq \frac{1}{4}$  neigt die LPA Hop 0 - 2 Variante verstärkt dazu *null* Communities zu erzeugen.

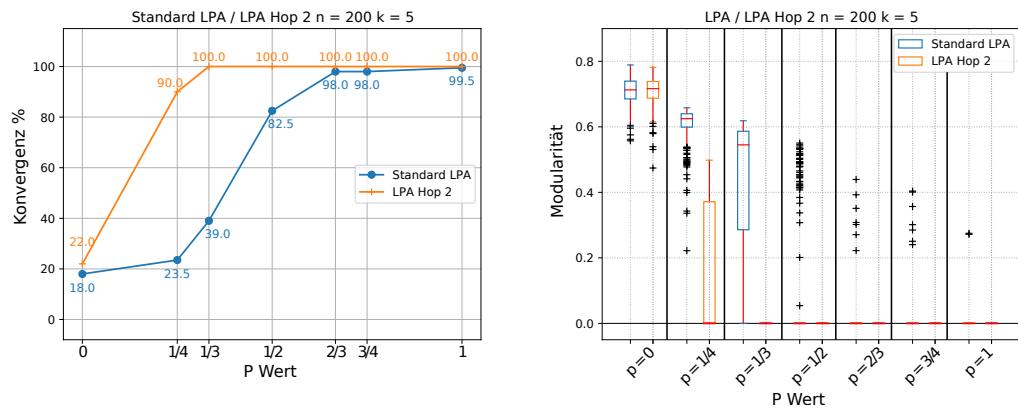


Abbildung 22: Die linke Grafik stellt die Konvergenz des Standard LPA (blaue Linie) und der Hop 0 - 2 Variante (orange Linie) für  $k = 5$  gegenüber. Bereits ab  $p = \frac{1}{4}$  zeigt die LPA Hop 0 - 2 Variante eine verbesserte Konvergenz an. Ab  $p \geq \frac{1}{3}$  erreicht diese Variante eine Konvergenz von 100%. Die rechte Grafik zeigt die Modularität für den Standart LPA (blaue Box) und LPA Hop 0 - 2 Variante (orange Box) für  $k = 5$ . Beide Varianten zeigen für einen regulären Graphen ( $p = 1$ ), dass ähnliche Communities gebildet werden. Dies zeigt sich mit leichten Abweichungen anhand der Modularität. Für  $p \geq \frac{1}{4}$  kommt es mit Anwendung der LPA Hop 0 - 2 Variante verstärkt zur Bildung von *null* Communities.

Die folgende Abb. 23 zeigt, wie sich die LPA Hop 0 - 2 Variante im Vergleich zu dem Standard LPA für  $k = 10$  verhält. Die linke Grafik in Abb. 23 zeigt, dass diese Variante bereits mit  $p = \frac{1}{4}$  eine Konvergenz von 100% erreicht. Die rechte Grafik in der Abb. 23 zeigt, dass die LPA Hop 0 - 2 Variante für alle Testläufe mit  $p \geq \frac{1}{4}$  eine Modularität von Null erreicht.

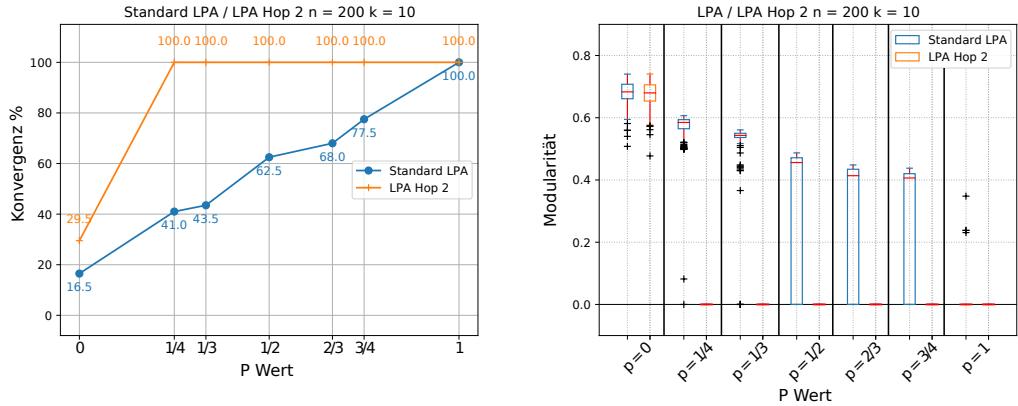


Abbildung 23: Die linke Grafik stellt die Konvergenz des Standard LPA (blaue Linie) und der Hop 0 - 2 Variante (orange Linie) für  $k = 10$  gegenüber. Bereits ab  $p = \frac{1}{4}$  zeigt die LPA Hop 0 - 2 Variante eine Konvergenz von 100%. Die rechte Grafik zeigt die Modularität für den Standart LPA (blaue Box) und LPA Hop 0 - 2 Variante (orange Box) für  $k = 10$ . Beide Varianten zeigen für einen regulären Graphen ( $p = 1$ ), dass ähnliche Communities gebildet werden. Dies zeigt sich mit leichten Abweichungen anhand der Modularität. Ab  $p \geq \frac{1}{4}$  zeigt die LPA Hop 0 - 2 für alle Testläufe eine Modularität von Null.

Das erweitern der Hop-Distanz führt schnell zu einer hohen Konvergenz von bis zu 100%, wie die linke Grafiken in den Abb. 22, 23 zeigen. Eine Verbesserung der Modularität konnte mit dieser Variante des LPA nicht erreicht werden.

Zum Abschluss dieser Experimente wird der LPA Hop 0 - 1 mit der LPA FO Variante kombiniert. Hierbei soll noch einmal untersucht werden, ob die Reihenfolge der Knoten eine Auswirkung auf den Algorithmus besitzt.

Die linke Grafik der Abb. 24 stellt die Konvergenz des Standard LPA, der LPA Hop 0 - 1 Variante und der LPA Hop 0 - 1 Fo Variante für  $k = 5$  gegenüber. Es zeigt sich, dass die LPA Hop 0 - 1 FO Variante eine Konvergenz von 100% erreicht. Dieses Verhalten zeigt auch die LPA Hop 0 - 1 Variante.

Die rechte Grafik in Abb. 24 zeigt, wie sich die Modularität der drei Varianten für  $k = 5$  verhält. Hier zeigt die LPA Hop 0 - 1 FO Variante, bis auf minimale Abweichungen, ähnliche Werte wie die LPA Hop 0 - 1 Variante.

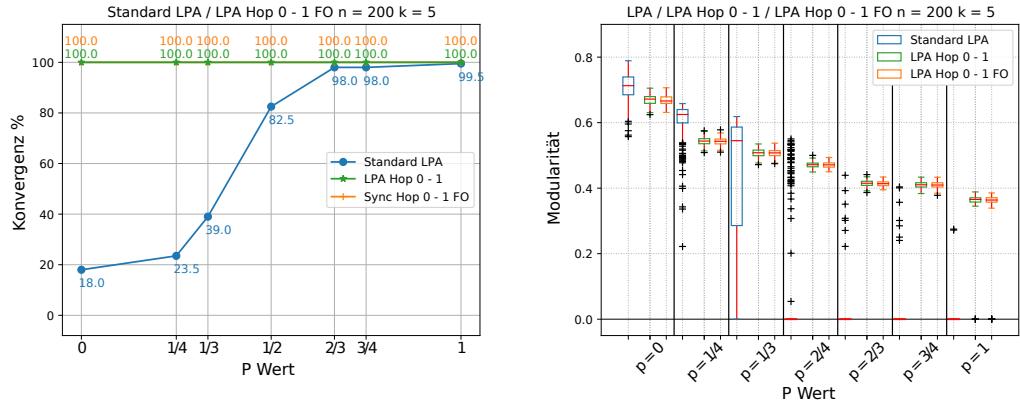


Abbildung 24: Die linke Grafik zeigt in % die Konvergenz des Standard LPA (blaue Linie), der Hop 0 - 1 Variante (grüne Linie) und der LPA Hop 0 - 1 FO Variante (orange Linie) für  $k = 5$ . Die LPA Hop 0 - 1 FO Variante zeigt, wie die LPA Hop 0 - 1 Variante, eine Konvergenz von 100%. Die rechte Grafik zeigt die Modularität der drei Varianten des LPA für  $k = 5$ . Bis auf minimale Abweichungen zeigen die LPA Hop 0 - 1 wie auch die LPA Hop 0 - 1 Fo Variante identische Ergebnisse.

Für  $k = 10$  zeigt sich zwischen der LPA Hop 0 - 1 FO Variante und der LPA Hop 0 - 1 Variante kein Unterschied bezüglich der Konvergenz, wie die linke Grafik der Abb. 25 zeigt. Ebenso zeigen beide Varianten des LPA mit geringen Unterschieden eine identische Community Bildung, was sich anhand der Modularität in der rechten Grafik der Abb. 25 zeigt.

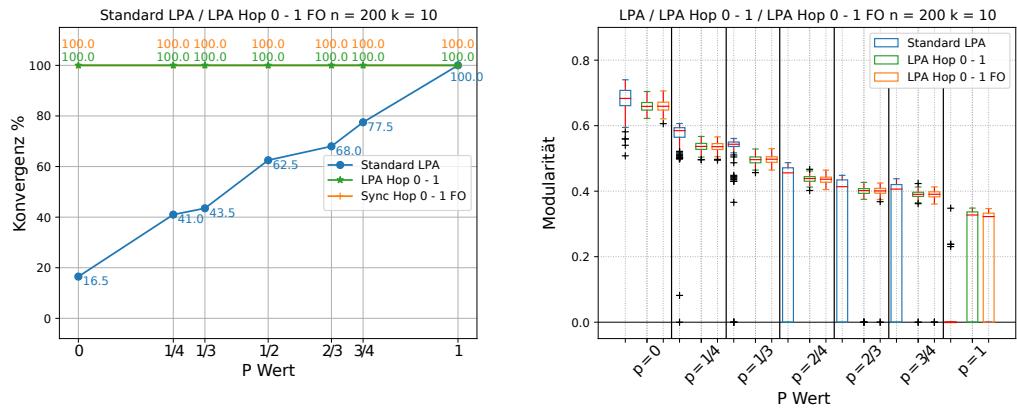


Abbildung 25: Die linke Grafik zeigt die Konvergenz in % des Standard LPA (blaue Linie), der Hop 0 - 1 Variante (grüne Linie) und der LPA Hop 0 - 1 FO Variante (orange Linie) für  $k = 10$ . Die LPA Hop 0 - 1 FO Variante zeigt, wie die LPA Hop 0 - 1 Variante, eine Konvergenz von 100%. Die rechte Grafik zeigt die Modularität der drei Varianten des LPA für  $k = 10$ . Bis auf minimale Abweichungen zeigen die LPA Hop 0 - 1, wie auch die LPA Hop 0 - 1 Fo Variante, identische Ergebnisse. Dies ist bereits in Abb. 24 zu sehen.

Nachdem nun mehrere Varianten des LPA bezüglich der Konvergenz und Modularität getestet wurden, lässt sich folgende Beobachtung festhalten. Der Wert  $p$

besitzt scheinbar weniger Einfluss auf die Konvergenz des Algorithmus, wie zunächst zu vermuten war. So zeigen einige Varianten bereits mit  $p = 0$  eine Konvergenz von 100%. Andere Varianten zeigen erst mit dem Wachstum von  $p$  eine hohe Konvergenz.

Bezogen auf die Modularität lässt sich festhalten, die getesteten Varianten zeigen, dass sie bei einem regulären Graphen, das Netzwerk in geeignete Communities unterteilen. Sodass eine hohe Modularität für  $p = 0$  erreicht wird [35]. Einige andere Varianten neigen mit dem Anstieg von  $p$  dazu, das Netzwerk in eine große Community zu unterteilen. Hierbei kommt es noch zu wenigen Ausreißern, die eine hohe Modularität aufzeigen. Die bislang besten Ergebnisse in Verbindung mit den Watts Strogatz Modell wurden mit der Hop 0 - 1 sowie der LPA\_S Variante erreicht.

#### 4.2.3 Detaillierte Experimente des LPA\_S und Hop 0 - 1 Variante

In diesen abschließenden Experimenten werden die Hop 0 - 1 sowie die LPA\_S Variante auf weitere Graphen angewendet. Beide Varianten zeigten in den bisherigen Experimenten die besten Ergebnisse. Die Anzahl der Kanten  $k$  bei dem Watts Strogatz Modell werden prozentual in Verbindung mit  $n$  erhöht. Dadurch soll ermittelt werden, ob die Menge der Kanten und Knoten die Konvergenz und Modularität beeinflusst. Des Weiteren wird die Modularität des Greedy Modularity (GM) Algorithmus mit der Modularität des jeweils besten Testlaufes der Varianten verglichen.

Die Grafiken in Abb. 26 zeigen die Auswertung der Konvergenz für  $n = 20$ . Der Standard LPA (blaue Linie) zeigt für  $k = 2$  die geringste Konvergenz. Für  $p = 0$ ,  $p = \frac{1}{4}$  und  $k = 2$  zeigt diese Variante eine Konvergenz von 0%. Der höchste Wert mit dem Standard LPA wird mit  $p = \frac{3}{4}$  erreicht für  $k = 2$ . Hier zeigt sich eine Konvergenz von 42%. Ab  $k = 4$  zeigt der Standard LPA eine Verbesserung der Konvergenz. Es zeigen sich Werte zwischen 53% und 100%.

Die LPA\_S Variante (grüne Linie) in der Abb. 26 zeigt für  $k = 2$  bis  $p = \frac{1}{4}$  eine Konvergenz von 0%, wie schon der Standard LPA. Für die Werte  $p = \frac{1}{3}$ ,  $p = \frac{3}{4}$  sowie für  $p = 1$  zeigt die LPA\_S Variante für  $k = 2$  die höchste Konvergenz. Für  $p = \frac{1}{2}$  sowie  $p = \frac{2}{3}$  zeigt sich eine Verschlechterung der Konvergenz wie schon mit dem Standard LPA. Für  $k \geq 4$  zeigt diese Variante des Algorithmus eine Konvergenz von 100%.

Die Hop 0 - 1 Variante (orange Linie) zeigt bereits für  $k = 2$  bis  $k = 16$  eine Konvergenz zwischen 92% und 100%, wie in der oberen linken Grafik der Abb. 26 zu sehen.

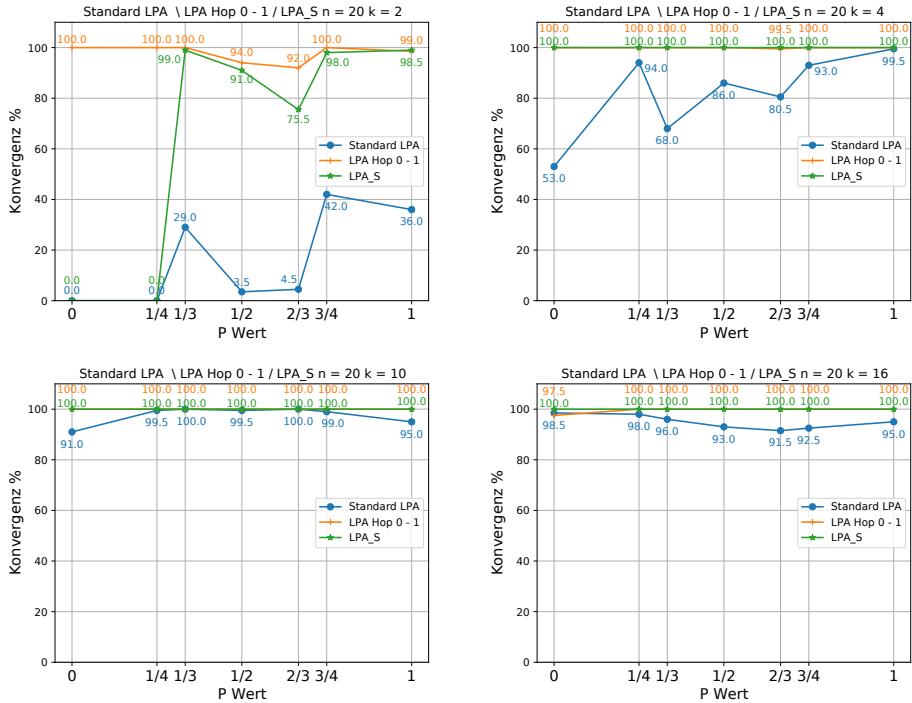


Abbildung 26: Wie die obere linke Grafik zeigt, erreicht der Standard LPA (blaue Linie) bis  $p = \frac{1}{4}$  eine Konvergenz von 0%. Im Schnitt erreicht der Standard LPA hier Werte zwischen 0% und 42%. Ein ähnliches Verhalten zeigt auch die LPA\_S Variante in der oberen linken Grafik. Bis  $p = \frac{1}{4}$  wird ein Wert von 0% erreicht. Je größer der Wert  $k$  wird, desto mehr verbessert sich die Konvergenz des Standard LPA. Die LPA\_Hop\_0 - 1 Variante (orange Linie) zeigt eine Konvergenz von 92% bis 100%.

Die Abbildung 27 zeigt wie sich die Modularität der LPA Varianten mit  $n = 20$  verhält. Für  $k = 2$  (linke obere Grafik) erkennt der Standard LPA (blaue Boxen) selten eine Community. In den meisten Fällen erreicht diese Variante eine negative Modularität. Daraus lässt sich schließen, dass der Standard LPA keine Communities für  $k = 2$  erkennt. Mit  $k = 4$  zeigt sich eine positive Modularität für  $p = 0$  sowie für  $p = \frac{1}{3}$  (obere rechte Grafik). Für alle weiteren Werte von  $p$  kommt es zu einzelnen Ausreißern, die eine Modularität größer Null zeigen. Für  $k \geq 10$  kommt es vermehrt zur Bildung von *null* Communities mit dem Standard LPA.

Anders verhält sich die LPA\_Hop\_0 - 1 Variante (orange Boxen), wie die Abb. 27 zeigt. Für  $k = 2$  erreicht diese Variante im Vergleich zu dem Standard LPA und der LPA\_S Variante die besseren Werte. Dieses Verhalten zeigt sich auch für  $k = 4$ , mit Ausnahme für den Wert  $p = 1$ . In der unteren linken Grafik der Abb. 27 zeigt sich, dass die LPA\_Hop\_0 - 1 Variante wie auch für  $p = 0$  eine positive Modularität erreicht.

Der LPA\_S (grüne Boxen) in der Abb. 27 erkennt ab  $p = \frac{1}{3}$  für  $k = 2$  innerhalb des Netzwerkes Communities. Für  $k = 4$  zeigt sich ein ähnliches Verhalten wie schon

mit der LPA Hop 0 - 1 Variante. Allerdings erreicht diese Version des LPA für  $p = 1$  bei  $k = 4$  konstantere Werte. In der unteren linken Grafik in der Abb. 27 zeigt sich, dass diese Variante des LPA bis  $p = \frac{1}{3}$  Communities in dem Netzwerk erkennt.

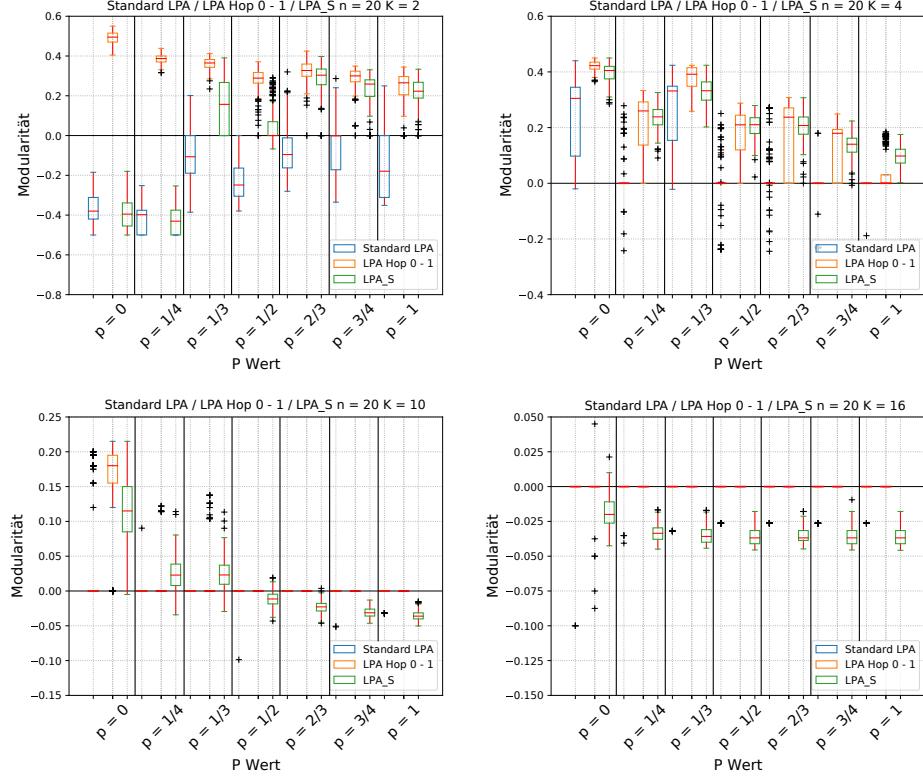


Abbildung 27: Die Abbildung zeigt, welche Modularität die unterschiedlichen LPA Varianten pro Testlauf erreicht haben. Der Standard LPA (blaue Boxen) zeigt für  $k = 2$  die geringste Modularität, wie die obere linke Grafik zeigt. Für  $k = 4$  kommt es zur Bildung von Communities mit der Standard LPA, sowohl für  $p = 0$  wie auch für  $p = \frac{1}{3}$  wie in der rechten oberen Grafik zu sehen. Mit dem Wachstum von  $p$  sowie von  $k$  nimmt der Wert der Modularität für den Standard LPA ab. Es kommt vermehrt zur Bildung von *null* Communities. Die LPA\_Hop\_0-1 Variante (orange Boxen) zeigt sowohl für  $k = 2$  wie auch für  $k = 4$  die höchsten Werte, wie die oberen beiden Grafiken erkennen lassen. Für  $k = 10$  erreicht diese Variante lediglich mit  $p = 0$  während mehreren Testläufen eine Modularität größer Null. Die LPA\_S Variante (grüne Boxen) zeigt für  $k = 2$  bis  $p = \frac{1}{4}$  mit leichten Abweichungen ähnliche Werte wie der Standard LPA. Für  $p \geq \frac{1}{3}$  kommt es zur Bildung von Communities, was sich in der linken oberen Grafik zeigt. Diese Variante erkennt für die meisten Testläufe, anders als der Standard LPA und die Hop 0 - 1 Variante, auch noch Communities für  $k = 10$  mit  $p \leq \frac{1}{3}$ .

Die folgende Grafik Abb. 28 zeigt einen Vergleich zwischen der Modularität des Greedy Modulariy Algorithmus mit der höchsten erreichten Modularität des Standard LPA, LPA\_S sowie dem LPA\_Hop\_0-1.

Sowohl die Hop 0 - 1 Variante (grüne Linie) wie auch der GM Algorithmus (orange Linie) zeigen für  $k = 2$  sowie  $k = 4$  eine mit minimalen Abweichungen

ähnlichen Modularität. Weiterhin zeigen sowohl der Standard LPA wie die LPA\_S Variante für  $k = 4$  eine ähnliche Entwicklung wie schon die LPA Hop 0 - 1 Variante. Lediglich für  $P = 1$  erreicht der Standard LPA einen Wert von Null. Für  $k = 10$  zeigt einzig der Gm Algorithmus bis  $p = \frac{2}{3}$  eine positive Modularität.

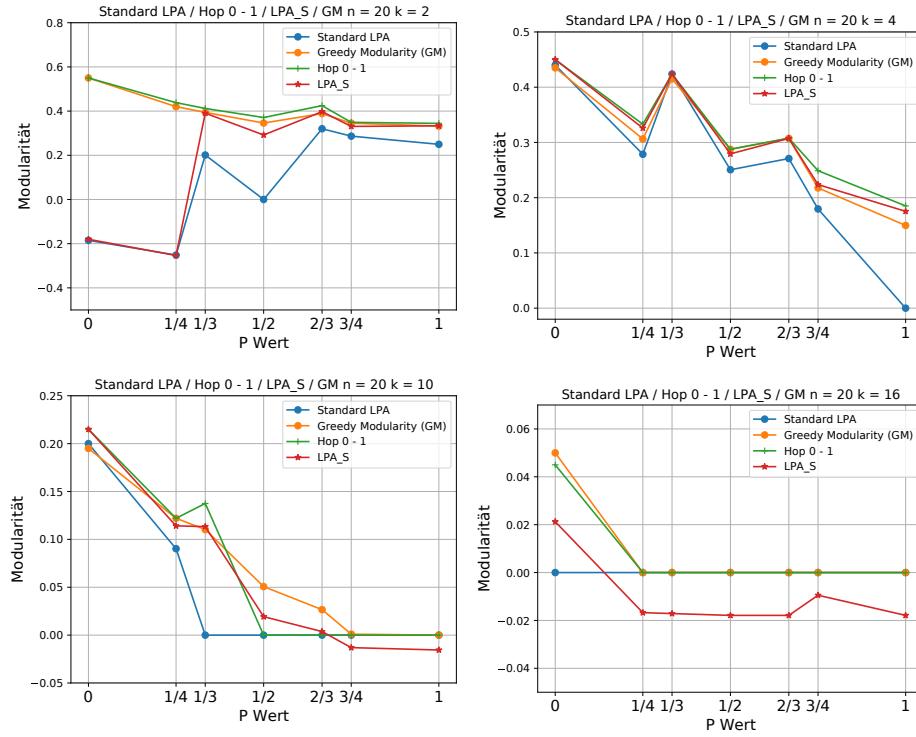


Abbildung 28: Vergleich der Modularität zwischen dem Standard LPA (blaue Linie), LPA Hop 0 - 1 (grüne Linie), LPA\_S Variante (rote Linie) sowie des GM-Algorithmus (orange Linie). Der Standard LPA wie auch die LPA\_S Variante zeigen für  $k = 2$  und  $p \leq \frac{1}{4}$  die geringsten Werte auf, wie die obere linke Grafik veranschaulicht. Die Hop 0 - 1 Variante wie auch der GM-Algorithmus zeigen mit minimalen Abweichungen einen ähnlichen Kurvenverlauf für  $k = 2$ . Dieses Verhalten lässt sich auch in der oberen rechten Grafik für alle Algorithmen erkennen. Für  $k = 10$  zeigt die LPA\_S Variante und der GM-Algorithmus das Communities bis  $p = \frac{2}{3}$  erkannt werden. Für  $k = 16$  zeigen bis auf den Standard LPA alle Algorithmen eine positive Modularität.

Für das nächste Experiment wird ein Netzwerk mit 100 Knoten verwendet. Die Abb. 29 zeigt wie sich die Konvergenz der Varianten entwickelt.

Im Vergleich zur Abb. 26 zeigt der Standard LPA (blaue linie) bei diesen Netzwerken einen steigenden Verlauf der Konvergenz für  $k = 10$  (obere linke Grafik). Mit dem Anstieg von  $k$  zeigt diese Variante eine verbesserte Konvergenz an. So wird mit  $p = \frac{1}{3}$  für  $k = 24$  (obere rechte Grafik) ein Wert von 98.5% erreicht. Für  $k \geq 50$  zeigen sich Werte zwischen 98.5% und 100% ab  $p = \frac{1}{4}$ .

Sowohl der LPA\_S (grüne linie) wie die Hop 0 - 1 Variante (orange Linie) zeigen für jedes  $k$  Werte von 99.5% bis 100% an, wie die Abb. 29 zeigt. Der LPA\_S hat sich

im Vergleich zur Abb. 26 verbessert.

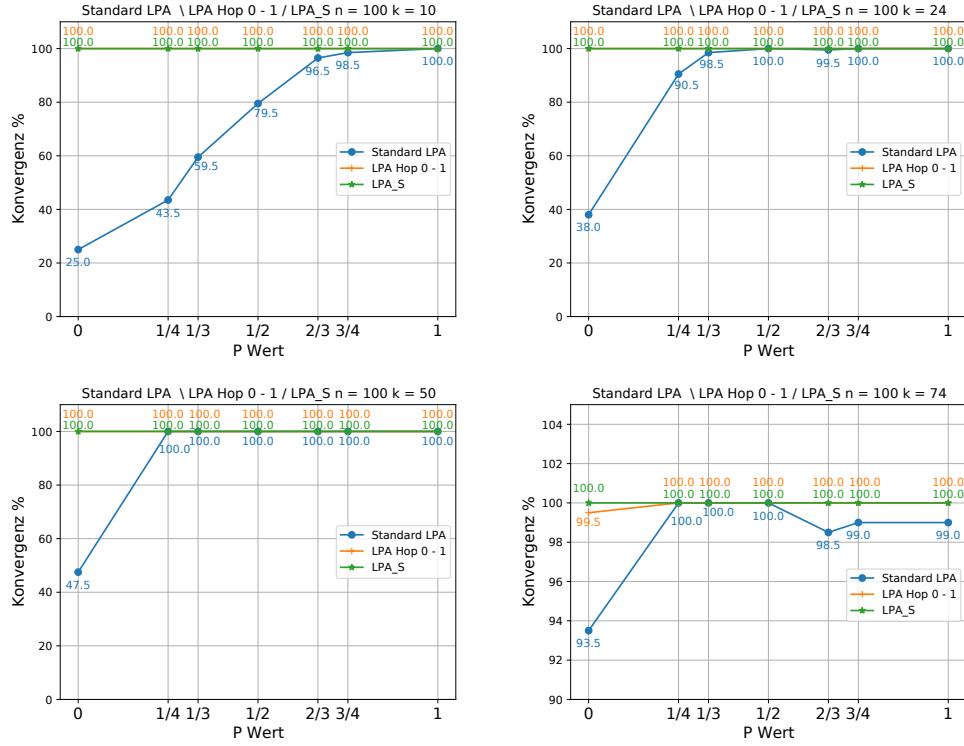


Abbildung 29: Wie die obere linke Grafik zeigt, steigert sich die Konvergenz des Standard LPA (blaue Linie) mit dem Wachstum von  $p$ . So erreicht diese Variante mit  $p = 1$  eine Konvergenz von 100%. Je größer der Wert  $k$  wird, desto schneller erreicht der Standard LPA einen Wert von 100%. Für  $k = 24$  (rechte obere Grafik) wird schon mit  $p = \frac{1}{4}$  eine Konvergenz von 90.5% erreicht. Für  $k = 50$  wird ein Wert von 100% bereits mit  $p = \frac{1}{4}$  erreicht. Für  $k = 74$  zeigt sich, dass der Standard LPA Werte zwischen 93.5% und 100% erreicht. Die LPA Hop 0 - 1 Variante (orange Linie) zeigt eine Konvergenz von 92% bis 100%. Die LPA\_S Variante (grüne Linie) zeigt hier für jedes  $k$  und  $p$  Werte von 100%.

Die Abb. 30 zeigt die Modularität der Varianten für ein Netzwerk mit 100 Knoten. Der Standard LPA (blaue Boxen) erreicht bis  $p = \frac{1}{2}$  für  $k = 10$  eine positive Modularität, wie die linke obere Grafik in Abb. 30 zeigt. Für  $k = 24$  und  $k = 50$  erreicht der Standard LPA noch mit einem regulären Graphen  $p = 0$  eine positive Modularität. Für die hier beschriebenen Werte kommt es zur Community Bildung mittels dem Standard LPA. Für alle weiteren Werte kommt es zur Bildung von *null* Communities mit dem Standard LPA. Dies wird besonders in der rechten unteren Grafik der Abb. 30 deutlich.

Die Hop 0 - 1 Variante (orange Boxen) in Abb. 30 erreicht für  $k = 10$  bis  $p = \frac{3}{4}$  eine positive Modularität (linke obere Grafik). Für  $k = 24$  zeigt diese Variante bis  $p = \frac{1}{3}$ , dass es zur Bildung von Communities kommt. Mit dem Wachstum von  $p$  erreichen nur noch wenige Testläufe einen Wert über Null. Für  $k = 50$  zeigt diese

Variante ein mit minimalen Abweichungen ähnliches Verhalten, wie der Standard LPA. Dies ist in der linken unteren Grafik der Abb. 27 zu sehen.

Ein anderes Verhalten zeigt die LAP\_S Version, wie die Grafiken in Abb. 30 zeigt. Für  $k = 10$  erreicht diese Variante des LPA die geringsten Modularitätswerte. Allerdings zeigt sich für  $p = 1$  (obere linke Grafik), dass mit dieser Variante für alle Testläufe eine positive Modularität erreicht wird. Für  $k = 24$  zeigt die LPA\_S Variante erneut die geringsten Werte auf. Jedoch kommt es bis  $p = 1$  zur Bildung von Communities mit jedem Testlauf, während die anderen beiden Varianten mit dem Wachstum von  $p$  einen Wert von Null erreichen. Dieses Verhalten zeigt sich auch für  $k = 50$ , wie die untere linke Grafik der Abb. 30 zeigt.

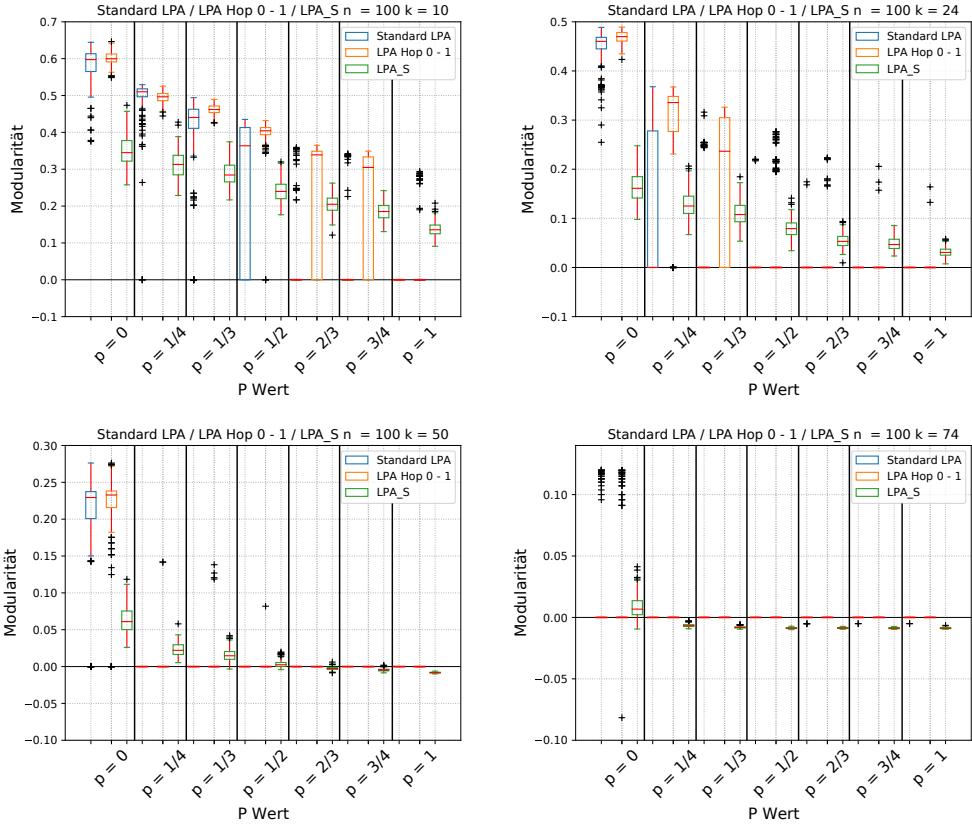


Abbildung 30: Die obere linke Grafik zeigt, dass der Standard LPA (blaue Boxen) bis  $p = \frac{1}{2}$  Communities in dem Netzwerk erkennt. Für  $p = \frac{1}{2}$  unterscheiden sich die gebildeten Communities mit dem Standard LPA stark voneinander. Das zeigt sich anhand der großen Streuung der Modularität. Für  $k = 24$  kommt es bereits mit  $p = \frac{1}{4}$  vermehrt zur Bildung von *null* Communities. Die LPA Hop 0 - 1 Variante (orange Boxen) bildet Communities bis  $p = \frac{3}{4}$  für  $k = 10$ , wie die obere linke Grafik zeigt. Mit dem Wachstum von  $k$  nimmt auch hier die Bildung der Communities pro  $p$  ab. Für  $k = 50$  erkennen der Standard LPA sowie die Hop 0 - 1 nur noch für  $p = 0$  Communities pro Testlauf, wie die linke untere Grafik zeigt. Die LPA\_S Variante zeigt auch hier wieder die geringsten Werte auf. Allerdings erkennt diese Variante für  $k = 10$  auch noch Communities für  $p = 1$  mit jedem Testlauf. Dies zeigt sich auch für  $k = 24$ . Für  $k = 50$  erreicht diese Variante des LPA eine positive Modularität bis  $p = \frac{1}{2}$ .

Die Grafiken in der Abb. 31 stellt die Modularität der drei LPA Varianten sowie die des Gm Algorithmus gegenüber.

Für  $k = 10$  zeigen alle Varianten wie auch der GM Algorithmus einen mit leichten Abweichungen ähnlichen Verlauf der Modularität. Lediglich der Standard LPA (blaue Linie) zeigt für  $p = 1$  und  $k = 10$  einen Wert von Null, wie die obere linke Grafik in Abb. 31 zeigt.

In der rechten oberen Grafik der Abb. 31 zeigt der Standard LPA bereits mit  $p = \frac{3}{4}$  einen Wert von Null. Für  $k = 50$  wie für  $k = 74$  zeigen die unteren Grafiken in Abb. 31, dass der Standard LPA bereits mit  $p = \frac{1}{4}$  Null Werte erreicht.

Die Hop 0 - 1 Variante (grüne Linie) zeigt erst für  $k = 50$  mit  $p = \frac{2}{3}$  Werte von Null an. Die LPA\_S Variante (rote Linie) zeigt erst mit  $p = \frac{3}{4}$  Werte von Null an, wie die untere linke Grafik in Abb. ?? veranschaulicht. Einzig der GM Algorithmus (orange Linie) erreicht hier selbst mit  $p = 1$  noch einen Wert über Null.

Für  $k = 74$  zeigen alle Varianten des LPA bereits mit  $p = \frac{1}{4}$  eine Modularität von Null. Lediglich dem GM Algorithmus gelingt eine positive Modularität bis  $p = \frac{1}{3}$

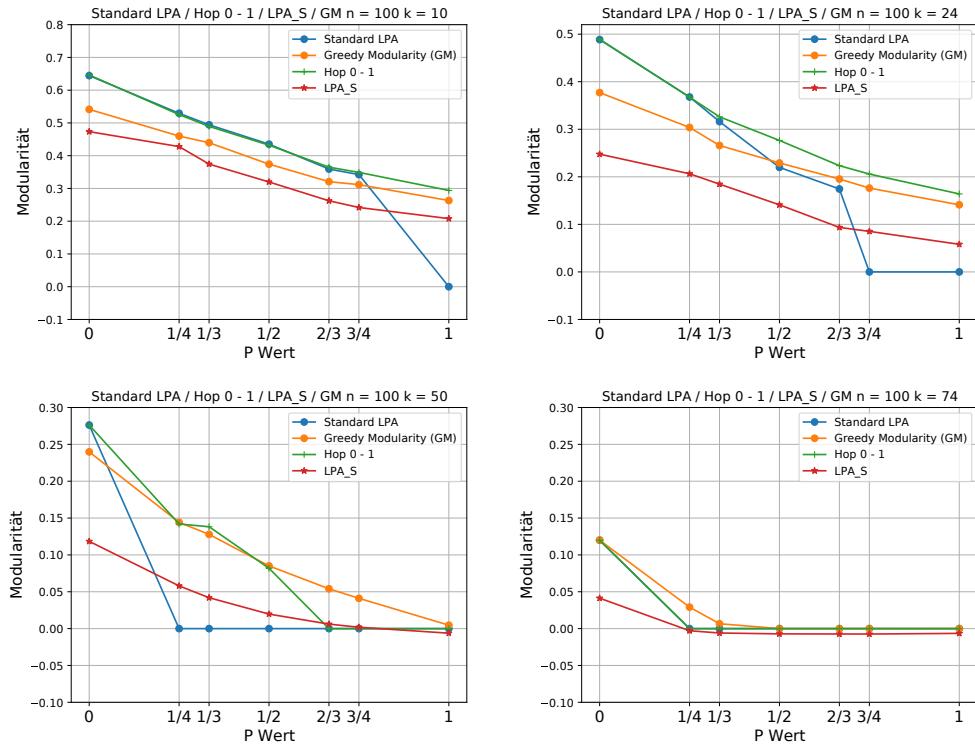


Abbildung 31: Vergleich der Modularität zwischen dem Standard LPA (blaue Linie), LPA Hop 0 - 1 (grüne Linie), LPA\_S (rote Linie) sowie des GM-Algorithmus (orange Linie). Für  $k = 10$  zeigen alle Algorithmen eine mit minimalen Abweichungen ähnlichen Verlauf der Kurve an, wie die obere linke Grafik verbildlicht. Lediglich der Standard LPA erreicht für  $p = 1$  einen Wert von Null. Für  $k = 24$  zeigt sich ein ähnliches Verhalten wie für  $k = 10$ . Der Standard LPA erreicht hier bereits mit  $p = \frac{3}{4}$  einen Wert von Null. Für  $k = 50$  zeigt sich, dass der GM-Algorithmus bis  $p = 1$  eine Modularität über Null erreicht, wie die untere linke Grafik zeigt.

Zum Abschluss der Experimente wird ein Netzwerk mit 500 Knoten verwendet. Der Standard LPA (blaue Linie) zeigt für  $k = 50$  in der oberen linken Grafik der Abb. 32 einen steilen Anstieg der Konvergenz. Für  $p = \frac{1}{4}$  zeigt der Standard LPA bereits einen Wert von 87.5%. In der oberen linken Grafik der Abb. 30 zeigt sich erst mit  $p = \frac{2}{3}$  ein Wert von 96.5%. Mit dem Wachstum von  $k$  erreicht diese Variante ab  $p \geq \frac{1}{4}$  Werte von 99% bis 100%. Sowohl die Hop 0 - 1 (orange Linie), sowie die LPA\_S Variante (grüne Linie) zeigen unabhängig von  $p$  und  $k$  eine Konvergenz von 100%, wie die Abb. 33 verdeutlicht.

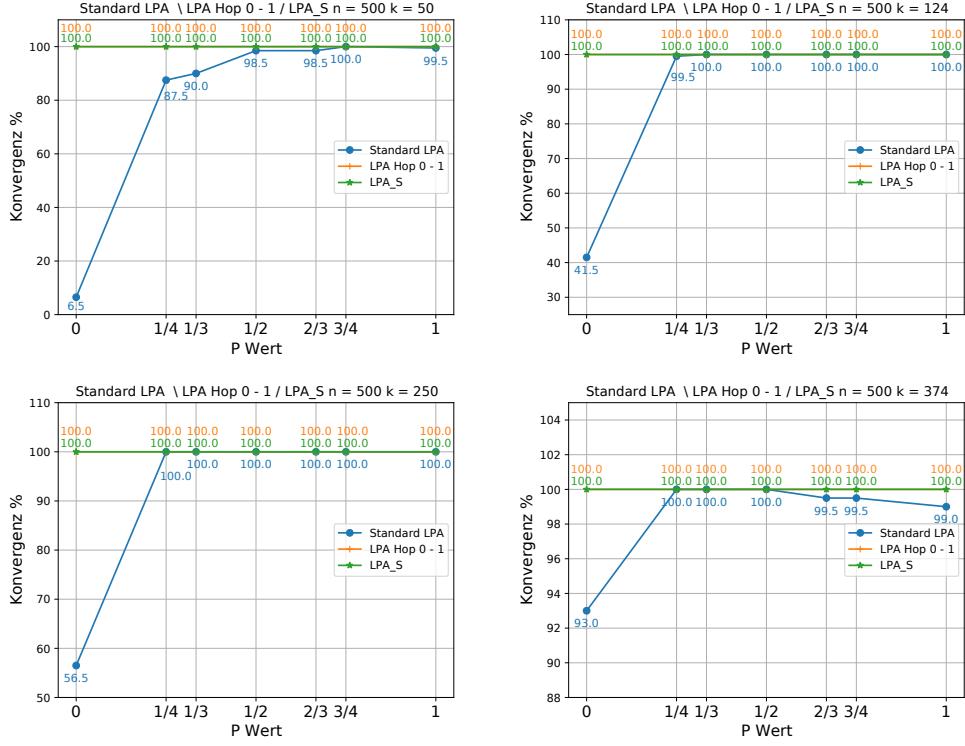


Abbildung 32: Wie die obere linke Grafik zeigt, steigert sich die Konvergenz des Standard LPA (blaue Linie) mit dem Wachstum von  $p$ . Bereits mit  $p = \frac{1}{4}$  wird ein Wert von 87.5% erreicht. Je größer der Wert  $k$  wird, desto früher erreicht der Standard LPA einen Werten zwischen 93% und 100%. Für einen regulären Graphen zeigt diese Variante für jedes  $k$  die geringste Konvergenz. Die LPA Hop 0 - 1 Variante (orange Linie) wie auch die LPA\_S Variante (grüne Linie) zeigen für jedes  $k$  und jedes  $p$  Werte von 100%.

Die folgende Abb. 33 zeigt, welche Modularitätswerte mit den Varianten des LPA erreicht wurden. Für  $k = 50$  zeigt der Standard LPA (blaue Boxen) für einen regulären Graphen die geringste Schwankung der Modularität. Für  $p = \frac{1}{4}$  und  $p = \frac{1}{3}$  zeigt der LPA die größte Schwankung in der Modularität. Dies deutet darauf hin, dass sich die Bildung der Communities pro Testlauf stark unterscheidet. Für  $k = 124$  und  $k = 250$  zeigt der Standard LPA einzlig für einen regulären Graphen eine Modularität über Null.

Die LPA Hop 0 - 1 Variante (orange Boxen) weist für  $k = 50$  in der oberen linken Grafik der Abb. 33 bis  $p = \frac{1}{3}$  eine hohe Modularität auf. Pro Testlauf werden mit leichten Abweichungen ähnliche Communities gebildet, was die geringe Schwankung der Modularität zeigt. Für  $p = \frac{1}{2}$  zeigt die Hop 0 - 1 Variante mit  $k = 50$  eine größere Schwankung auf. Für  $k = 124$  sowie  $k = 250$  zeigt die Hop 0 - 1 Variante mit minimalen Abweichungen ähnliche Werte wie der Standard LPA.

Die LPA\_S Variante (grüne Boxen) zeigt in der Abb. 33 wie in den vorherigen Experimenten die geringste Modularität. Pro Testlauf werden allerdings Commu-

nities gebildet, die sich minimal voneinander unterscheiden. Dies zeigt sich anhand der schmalen Boxen. Weiterhin zeigt diese Variante sowohl für  $k = 50$  wie auch für  $k = 124$  für jedes  $p$  eine Modularität über Null. Für  $k = 250$  erkennt die LPA\_S Variante bis  $p = \frac{1}{2}$  Communities in dem Netzwerk. Zusätzlich kommt es bei einem regulären Graphen für  $k = 374$ , wie die untere rechte Grafik in der Abb. 33 zeigt, zur deutlicheren Bildung von Communities.

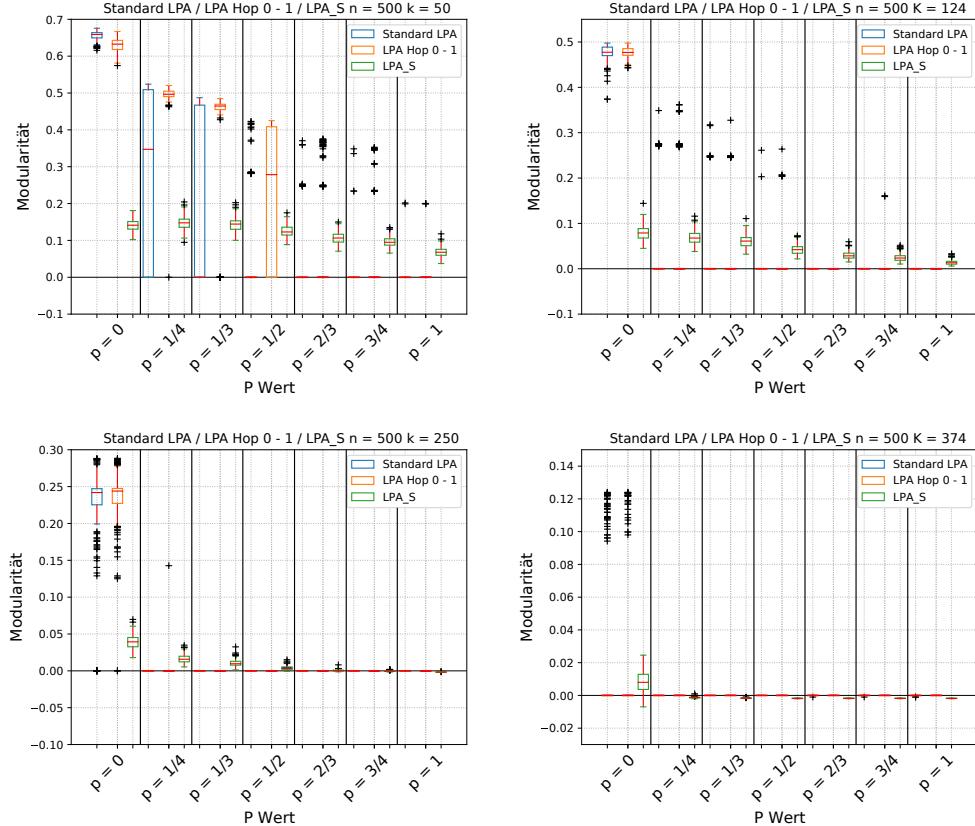


Abbildung 33: Die obere linke Grafik zeigt, dass der Standard LPA (blaue Boxen) bis  $p = \frac{1}{3}$  Communities in dem Netzwerk erkennt. Für  $p = \frac{1}{3}$  zeigt der Standard LPA vermehrt die Bildung von *null* Communities wie der Median in der linken oberen Grafik aufzeigt. Für  $k = 124$  sowie für  $k = 250$  zeigt der Standard LPA für  $p = 0$  mit jedem Testlauf eine Modularität über Null. Je größer der Wert  $p$  wird, desto vermehrt kommt es zur Bildung von *null* Communities. Die LPA Hop 0 - 1 Variante (orange Boxen) bildet Communities bis  $p = \frac{1}{2}$  für  $k = 50$ , wie die obere linke Grafik veranschaulicht. Je größer der Wert  $k$  wird, desto häufiger zeigt die LPA Hop 0 - 1 Variante ein ähnliches Verhalten wie der Standard LPA. Die LPA\_S Variante weist hier die geringsten Werte für jedes  $k$  auf. Allerdings erkennt diese Variante für  $k = 50$  auch noch Communities für  $p = 1$  mit jedem Testlauf. Dies zeigt sich auch für  $k = 124$ . Für  $k = 50$  erreicht diese Variante des LPA eine positive Modularität bis  $p = \frac{1}{2}$ .

Die Abb. 34 veranschaulicht die Gegenüberstellung der Modularität der LPA Varianten des jeweils besten Testlaufes und des GM-Algorithmus. Der Standard LPA (blaue Linie) wie die Hop 0 - 1 Variante (grüne Linie) weisen, in der oberen

linken Grafik der Abb. 34, mit minimalen Abweichungen einen ähnlichen Verlauf auf. Dieses Verhalten wiederholt sich auch in den weiteren Grafiken der Abb. 34. Für  $p = \frac{2}{3}$  bei  $k = 124$  (rechte obere Grafik) zeigt die Hop 0 - 1 Variante erneut einen Anstieg der Modularität und für  $p = 1$  wieder einen Wert von Null.

Die LPA\_S Variante (rote Linie) weist bis  $k = 250$  (linke untere Grafik der Abb. 34) und  $p = \frac{2}{3}$  eine Modularität über Null auf. Diese Variante erkennt auch dann noch Communities, wenn der Standard LPA bereits Werte von null erzeugt.

Der GM-Algorithmus erkennt, wie die LPA\_S Variante auch, für  $k = 250$  bis  $p = 1$  Communities in dem Netzwerk, wie die untere linke Grafik der Abb. 34 veranschaulicht. Für  $k = 374$  erreichen die Varianten des LPA lediglich für einen regulären Graphen Werte über Null, wie in der rechten unteren Grafik der Abb. 34 zu sehen. Der GM-Algorithmus erreicht positive Werte bis  $p = \frac{1}{3}$ .

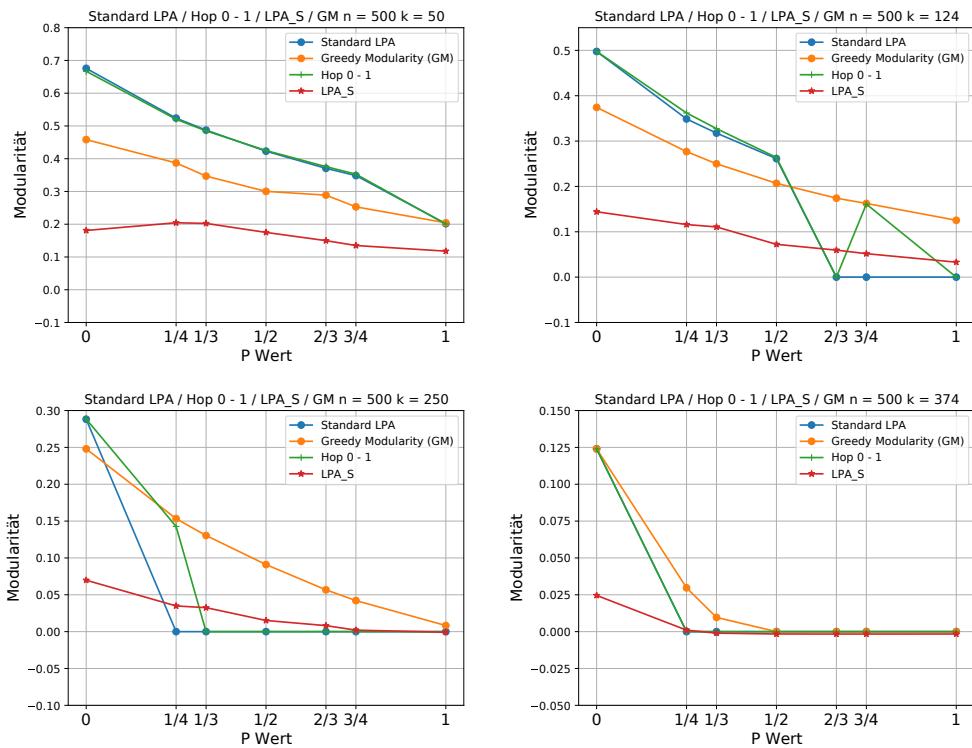


Abbildung 34: Vergleich der Modularität zwischen dem Standard LPA (blaue Linie), LPA Hop 0 - 1 (grüne Linie), LPA\_S (rote Linie) sowie des GM-Algorithmus (orange Linie). Für  $k = 50$  zeigen alle Algorithmen eine mit minimalen Abweichungen ähnlichen Verlauf der Kurve an, wie die obere linke Grafik aufzeigt. Für  $k = 124$  zeigt sich ein ähnliches Verhalten wie für  $k = 50$ . Der Standard LPA erreicht hier jedoch mit  $p = \frac{2}{3}$  einen Wert von Null, wie die obere rechte Grafik verbildlicht. Ein ähnliches Verhalten weist die Hop 0 - 1 Variante für  $k = 124$  auf. Mit  $p = \frac{3}{4}$  erreicht diese Variante erneut eine positive Modularität, bevor diese für  $p = 1$  auf Null fällt. Der GM-Algorithmus erkennt für  $k = 250$  bis  $p = 1$  Communities in dem Netzwerk, wie die untere linke Grafik zeigt. Für  $k = 374$  erkennen alle Varianten des LPA lediglich für einen regulären Graphen Communities. Der GM-Algorithmus erreicht hier bis  $p = \frac{1}{3}$  Werte über Null.

Nach Auswertung einiger unterschiedlich großer Netzwerke hat sich gezeigt, je größer der Wert  $k$ , desto geringer fällt die Modularität für alle Algorithmen aus. Dies lässt sich unter anderem damit begründen, dass dem Netzwerk weitere Kanten hinzugefügt werden und Knoten einer Community somit auch entsprechend mehr Kanten in anderen Communities aufweisen. Dadurch kommt es zu einem Abfall der Modularität. Es liegt hier also nicht an den Algorithmen, sondern an der Beschaffenheit des Netzwerkes. Zusätzlich ist auch bekannt, mit dem Anstieg von  $p$  fällt die Modularität [35].

#### 4.2.4 Auswertung der Varianten

In diesem letzten Abschnitt folgt eine Auswertung aller Varianten des LPA. Da die Konvergenz des Algorithmus untersucht wird fallen aus der Bewertung die Varianten heraus, die als Konvergenzkriterium eine maximale Anzahl an Iterationen nutzen. Somit ergeben sich für diese Experimente 240 Varianten des Algorithmus.

Die Auswertung der Ergebnisse erfolgt anhand von Tabellen, da die Darstellung als Schaubild bei dieser Menge unübersichtlich wird. Die Tab. 8 zeigt die Auswertung der 240 Varianten. Hierbei steht  $c$  für die Konvergenz des Algorithmus.

Jede Spalte grenzt ein Intervall für die Konvergenz  $c$  ab. Ein Wert von  $c \geq 90\%$  bedeutet  $100\% \geq c \geq 90\%$ . Anhand der Tab. 8 zeigt sich, dass von den 240 Varianten mehr als 70% eine Konvergenz von 100% erreichen. Die restlichen 30% zeigen eine Konvergenz von 90% bis 0% an. Daraus erschließt sich, dass die Konvergenz der Varianten weniger unzureichend zu sein scheint, wie zunächst zu vermuten war.

Auswertung der Konvergenz für $n = 200 k = 5$												
$p$	$c = 100\%$	$c \geq 90\%$	$c \geq 80\%$	$c \geq 70\%$	$c \geq 60\%$	$c \geq 50\%$	$c \geq 40\%$	$c \geq 30\%$	$c \geq 20\%$	$c \geq 10\%$	$c \geq 0\%$	
0	195	5	16	0	0	0	0	0	0	3	21	
$\frac{1}{4}$	176	25	12	3	5	3	2	1	6	5	2	
$\frac{1}{3}$	193	17	11	2	7	4	1	3	0	2	0	
$\frac{1}{2}$	202	17	16	3	0	2	0	0	0	0	0	
$\frac{2}{3}$	207	30	3	0	0	0	0	0	0	0	0	
$\frac{3}{4}$	206	31	2	1	0	0	0	0	0	0	0	
1	215	25	0	0	0	0	0	0	0	0	0	

Auswertung der Konvergenz für $n = 200 k = 10$												
$p$	$c = 100\%$	$c \geq 90\%$	$c \geq 80\%$	$c \geq 70\%$	$c \geq 60\%$	$c \geq 50\%$	$c \geq 40\%$	$c \geq 30\%$	$c \geq 20\%$	$c \geq 10\%$	$c \geq 0\%$	
0	203	1	0	4	8	0	0	0	2	14	8	
$\frac{1}{4}$	213	1	3	5	10	3	1	0	3	1	0	
$\frac{1}{3}$	213	4	9	7	1	2	0	0	3	1	0	
$\frac{1}{2}$	214	20	1	1	1	2	1	0	0	0	0	
$\frac{2}{3}$	215	19	1	1	1	1	2	0	0	0	0	
$\frac{3}{4}$	217	19	0	3	1	0	0	0	0	0	0	
1	228	12	0	0	0	0	0	0	0	0	0	

Tabelle 8: Die obere Tabelle zeigt die Auswertung der Konvergenz für  $k = 5$  und die untere Tabelle die Auswertung für  $k = 10$ . In beiden Tabellen zeigt sich, dass mehr als 70 der Varianten des LPA eine Konvergenz von 100% erreichen unabhängig von dem Wert  $p$ .

Ein wesentlicher Punkt wird in der Tabelle 8 jedoch nicht gezeigt, nämlich die Modularität. Eine hohe Konvergenz des Algorithmus ist weniger vorteilhaft, wenn die Einteilung in Communities nicht optimal erfolgt und so eine geringe oder keine Modularität erreicht wird. Die folgende Tabelle 9 veranschaulicht die Modularitätswerte der LPA Varianten. Hierbei wurden nur die Testläufe betrachtet, die mit allen Iterationen eine positive Modularität erreicht haben. Die Werte in den Tabellen 9, 10 lassen sich wie folgt lesen:

Der rechte Wert gibt an, wie viele Varianten eine Konvergenz von  $x\%$  erreicht haben. Der linke Wert gibt an, wie viele Varianten mit einer Konvergenz von  $x\%$  eine Modularität größer Null erreicht haben. Als Beispiel sei der Wert 90 : 195 aus Tabelle 9 mit einer Konvergenz von  $c = 100\%$  genommen. Es existieren 195 mögliche Varianten des LPA die eine Konvergenz von 100% erreichen, davon wiederum existieren 90 mögliche Varianten die eine Modularität größer Null erreichen.

Die Tabelle 9 weist auf, dass für  $k = 5$  mehrere Varianten existieren, die eine Konvergenz von 100% und eine positive Modularität zeigen. Für einen regulären Graphen  $p = 0$  existieren von 195 möglichen Varianten des LPA 90 Varianten, die eine Modularität über Null zeigen. Für  $p = \frac{1}{2}$  sind es 173 mögliche Varianten von 202. Je größer der Wert  $p$  wird, desto geringer die Anzahl an Varianten. Für  $p = 1$  existieren noch 15 Möglichkeiten von 215.

Alle weiteren Varianten des LPA führten entweder zu einer Modularität von 0 oder erkannten keine Communities, sodass hier ein negativer Wert erreicht wurde.

Modularität > 0 für n = 200 k = 5							
Konvergenz	$p = 0$	$p = \frac{1}{4}$	$p = \frac{1}{3}$	$p = \frac{1}{2}$	$p = \frac{2}{3}$	$p = \frac{3}{4}$	$p = 1$
$c = 100\%$	90 : 195	57 : 176	57 : 193	173 : 202	23 : 207	19 : 206	15 : 215
$c \geq 90\%$	0 : 5	2 : 25	1 : 17	17 : 17	0 : 30	2 : 31	0 : 25
$c \geq 80\%$	13 : 16	0 : 12	0 : 11	16 : 16	0 : 3	2 : 2	0 : 0
$c \geq 70\%$	0 : 0	0 : 3	0 : 2	3 : 3	0 : 0	1 : 1	0 : 0
$c \geq 60\%$	0 : 0	0 : 5	0 : 7	0 : 0	0 : 0	0 : 0	0 : 0
$c \geq 50\%$	0 : 0	0 : 3	0 : 4	2 : 2	0 : 0	0 : 0	0 : 0
$c \geq 40\%$	0 : 0	4 : 2	0 : 1	0 : 0	0 : 0	0 : 0	0 : 0
$c \geq 30\%$	0 : 0	0 : 1	0 : 3	0 : 0	0 : 0	0 : 0	0 : 0
$c \geq 20\%$	0 : 0	0 : 6	0 : 0	0 : 0	0 : 0	0 : 0	0 : 0
$c \geq 10\%$	3 : 3	0 : 5	2 : 2	0 : 0	0 : 0	0 : 0	0 : 0
$c \geq 0\%$	21 : 21	2 : 2	0 : 0	0 : 0	0 : 0	0 : 0	0 : 0

Tabelle 9: Die Tabelle zeigt, wie viele Varianten des LPA eine Modularität größer Null erreichten. So zeigt sich, dass für  $c = 100\%$  noch immer eine Menge möglicher Varianten existieren, die eine Modularität über Null erreichen. Für  $p = 0$  wie auch für  $p = \frac{1}{2}$  existieren die meisten Varianten. Ab  $p \geq \frac{2}{3}$  nimmt die Menge an Möglichkeiten ab.

Die Tabelle 10 zeigt für  $k = 10$  wie viele Varianten des LPA existieren, die

eine positive Modularität über alle Iterationen zeigten. Anders als wie für  $k = 5$  wie die Tabelle 9 zeigt liegen für einen regulären Graphen  $p = 0$  wie für  $p = \frac{1}{2}$  weniger Varianten des LPA vor, die einen positiven Wert zeigen. Für  $p = \frac{1}{3}$  wiederum existieren für  $k = 10$  180 mögliche Varianten, die eine positive Modularität zeigen. Für einen regulären Graphen  $p = 0$  existieren lediglich 70 von 203 möglichen Varianten des LPA, die einen Wert über Null erreichen. Für  $k = 5$  waren es noch 90 von 195 Möglichkeiten.

Modularität > 0 für n = 200 k = 10							
Konvergenz	$p = 0$	$p = \frac{1}{4}$	$p = \frac{1}{3}$	$p = \frac{1}{2}$	$p = \frac{2}{3}$	$p = \frac{3}{4}$	$p = 1$
$c = 100\%$	70 : 203	41 : 213	180 : 213	16 : 214	13 : 215	13 : 217	7 : 228
$c \geq 90\%$	0 : 1	1 : 1	4 : 4	0 : 20	0 : 19	0 : 19	0 : 12
$c \geq 80\%$	0 : 0	0 : 3	9 : 9	0 : 1	0 : 1	0 : 0	0 : 0
$c \geq 70\%$	1 : 4	0 : 5	7 : 7	0 : 1	0 : 1	0 : 3	0 : 0
$c \geq 60\%$	1 : 8	0 : 10	1 : 1	0 : 1	0 : 1	0 : 1	0 : 0
$c \geq 50\%$	0 : 0	0 : 3	2 : 2	0 : 2	0 : 1	0 : 0	0 : 0
$c \geq 40\%$	0 : 0	1 : 1	0 : 0	0 : 1	0 : 2	0 : 0	0 : 0
$c \geq 30\%$	0 : 0	0 : 0	0 : 0	0 : 0	0 : 0	0 : 0	0 : 0
$c \geq 20\%$	0 : 2	2 : 3	3 : 3	0 : 0	0 : 0	0 : 0	0 : 0
$c \geq 10\%$	11 : 14	1 : 1	1 : 1	0 : 0	0 : 0	0 : 0	0 : 0
$c \geq 0\%$	7 : 8	0 : 0	0 : 0	0 : 0	0 : 0	0 : 0	0 : 0

Tabelle 10: Die Tabelle zeigt, wie viele Varianten des LPA eine Modularität größer Null erreichen. So zeigt sich: Für  $c = 100\%$  existieren weniger mögliche Varianten, die eine Modularität über Null erreichen im Vergleich zu  $k = 5$ . Für  $p = \frac{1}{3}$  existieren hier die meisten Varianten. Ab  $p \geq \frac{2}{3}$  nimmt die Menge an Möglichkeiten rapide ab.

Die Abbildungen 35, 36 veranschaulicht die Varianten, die sowohl für  $k = 5$  wie für  $k = 10$  eine Konvergenz von 100% sowie einen Modularität größer Null für jedes  $p$  erreichen. Für die Modularitätswerte, wurde der beste Wert verwendet, der sich aus allen 200 Testläufen ergeben hat. Es erschließen sich 4 Varianten, die sowohl für  $k = 5$  wie auch für  $k = 10$  eine Konvergenz von 100% erreichen und eine positive Modularität für jedes  $p$ .

Bei diesen Varianten handelt es sich um die LPA\_S Variante des LPA. Die Bezeichnung *Hop 1* besagt, dass nur die direkten Nachbarn des Knoten  $v_i$  betrachtet werden. Weiterhin steht *UR* für *Unique Random*. Dies bedeutet, dass jedem Knoten ein eindeutiges Label zugewiesen wird. Zusätzlich ändert sich die Reihenfolge der Knoten pro Iteration. Die Bezeichnung *UF* steht für *Unique Fixed*. Hierbei wird vor Beginn des Algorithmus eine Reihenfolge der Knoten festgelegt. Diese Anordnung bleibt bis zum Terminieren des Algorithmus konstant.

Die letzten beiden Varianten *RR 2* sowie *RF 2* stehen für *Random Random* sowie *Random Fixed*. Dies bedeutet, dass die Knoten kein eindeutiges Label erhalten

sondern die Label zufällig verteilt werden, sodass ein Label mehrmals vorkommen kann.

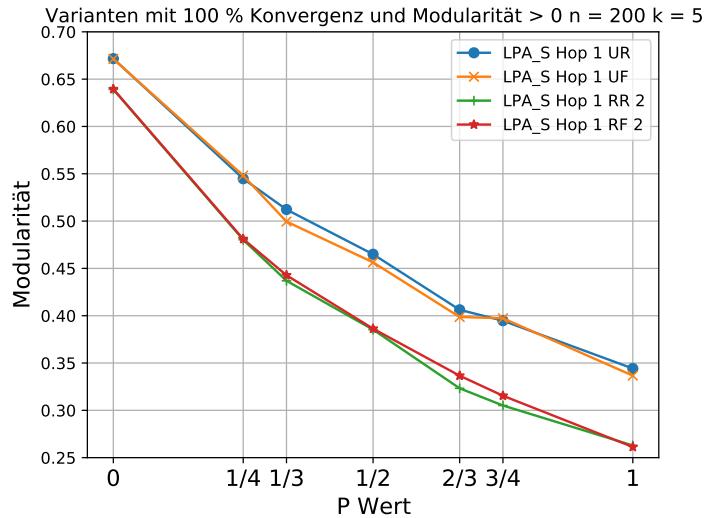


Abbildung 35: Darstellung der LPA Varianten, die für jedes  $p$  eine Modularität größer Null erreichen für  $k = 5$  mit einer Konvergenz von 100%. Die Hop 0 - 1 UR (blaue Linie) sowie die Hop 0 - 1 UF (orange Linie) erreichen die höchsten Werte und zeigen mit minimalen Abweichungen einen ähnlichen Kurvenverlauf. Einen ebenso ähnlichen Verlauf zeigen die Hop 1 RR 2 (grüne Linie) und die Hop 1 RF 2 (rote Linie) auf.

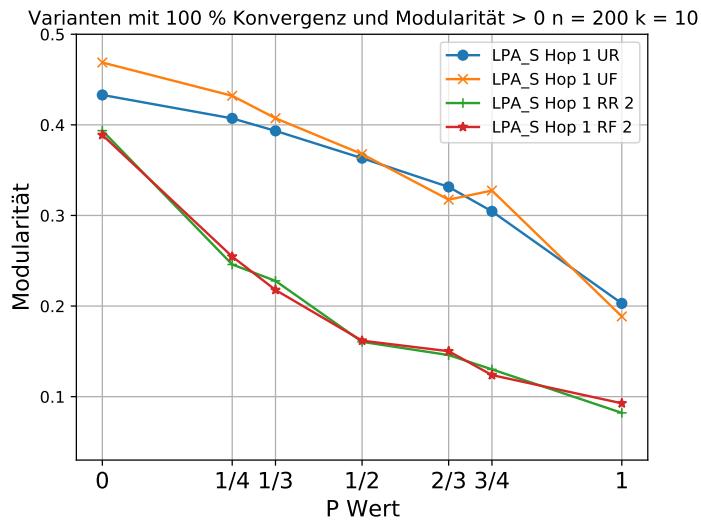


Abbildung 36: Darstellung der LPA Varianten, die für jedes  $p$  eine Modularität größer Null erreichen für  $k = 10$  mit einer Konvergenz von 100%. Die Hop 0 - 1 UR (blaue Linie) sowie die Hop 0 - 1 UF (orange Linie) erreichen die höchsten Werte und zeigen mit minimalen Abweichungen einen ähnlichen Kurvenverlauf. Einen ebenso ähnlichen Verlauf zeigen die Hop 1 RR 2 (grüne Linie) und die Hop 1 RF 2 (rote Linie) auf.

Von den möglichen 240 Varianten gelingt es vier Varianten des LPA, sowohl für

$k = 5$  wie auch für  $k = 10$  eine positive Modularität für jedes  $p$  zu erreichen.

Dies bedeutet jedoch nicht, dass die übrigen Varianten unzureichend sind. Das Augenmerk lag schlicht auf der Existenz von Varianten des LPA, die für beide Netzwerke eine Modularität größer Null erreichen und eine Konvergenz von 100% zeigen.

Die Tabellen 9 sowie 10 zeigen, dass weit mehr Varianten des LPA existieren, die die Netzwerke so in Communities unterteilt, dass eine Modularität größer Null erreicht werden kann. Welche Variante am Ende genutzt wird, hängt unter anderem auch von der Beschaffenheit des Netzwerkes ab. Sodass hier durchaus mehrere Varianten pro Netzwerk genutzt werden sollten, bevor eine Aussage über das Netzwerk, sowie die Community Aufteilung erfolgt.

## 5 Zusammenfassung und Ausblick

### 5.1 Zusammenfassung

Die hier durchgeführten Experimente untersuchten die Konvergenz des Label Propagation Algorithmus [40] unter Anwendung des Watts Strogatz Modells [50, 34]. Handelt es sich bei einem Netzwerk um einen regulären Graphen  $p = 0$ , so erreicht der Standard LPA eine geringe Konvergenz. Für einen vollkommen zufälligen Graphen  $p = 1$  erreichte der Standard LPA eine hohe Konvergenz. Durch vergrößern der Werte  $k$  und  $n$  bei dem Watts Strogatz Modell weist der LPA schon mit einem geringen Wert von  $p$  entsprechend hohe Konvergenzwerte auf. Verfügt das Netzwerk nur über wenige Knoten Abb. 26 und damit auch über wenig Kanten, so zeigt sich eine Schwankung der Konvergenz mit dem LPA.

Doch nicht alle Varianten des LPA weisen dieses Verhalten auf. Einige Varianten, wie der LPA Hop 0 - 1 oder der LPA\_S erreichen unabhängig von den Werten  $k$  und  $p$  eine hohe Konvergenz. Dies veranschaulicht Abb. 17, 26. Wieder andere Varianten wie der LPA FO zeigen ein ähnliches Verhalten wie der Standard LPA und weichen nur gering davon ab. Dies ist in der Abb. 20, 21 ersichtlich.

Ebenfalls war beim Standard LPA wie auch bei einigen anderen Varianten ein Abfall der Modularität zu beobachten, sobald die Konvergenz zunahm. So liegt die Vermutung nahe, dass mit dem Standard LPA lediglich ein Wert maximiert werden kann. Dieses Verhalten zeigte sich unter anderem auch bei der Hop 2 Abb. 22, 23 und LPA FO Variante Abb. 20, 21.

Die LPA Hop 0 - 1 und LPA\_S Variante zeigten für  $n = 200$  bezüglich der Konvergenz und Modularität zufrieden stellende Werte und wurden auf weitere Netzwerke angewendet. Die Hop 0 - 1 Variante verhielt sich mit dem Wachstum von  $k$  bei größeren Netzwerken mit minimalen Abweichungen ähnlich zu dem Standard LPA bezüglich der Modularität. Letztlich zeigten beide Varianten mit dem Wachstum von  $k$ , dass lediglich für einen regulären Graphen Communities erkannt werden.

Ein anderes Verhalten erschloss sich mit der LPA\_S Variante. Die Modularität fiel hier, im Vergleich zur LPA Hop 0 - 1 Variante, geringer aus. Allerdings unterteilte diese Variante mit dem Wachstum von  $k$  und  $p$  das Netzwerk noch immer in Communities, während der Standard LPA und die Hop 0 - 1 Variante mit diesen Werten bereits *null* Communities erzeugten. Dies zeichnete sich in den Abbildungen 30, 33 ab.

Zum Vergleich der Modularitätswerte der LPA Varianten wurde der Greedy Modularity Algorithmus [33] verwendet. Für ein kleines  $k$  zeigten die LPA Hop 0 - 1

und LPA\_S Varianten mit leichten Abweichungen einen ähnlichen Verlauf wie der GM-Algorithmus. Während die LPA Varianten ab einem gewissen Wert von  $k$  und  $p$  zur Bildung von Null Werten führten, erkannte der Gm-Algorithmus bis zu einem höheren  $p$  Wert noch Communities in dem Netzwerk.

Die Vermutung liegt nahe, dass das Watts Strogatz Modell eine große Community beschreibt, je größer der Wert  $k$  und  $p$  gewählt wird. Ein Modularitätswert von Null ist in diesem Falle optimal. Verfügt das Netzwerk nicht über Communities, sondern stellt bereits eine vollständige Community dar, so würden alle getesteten Algorithmen diese erkennen.

Im Abschnitt 4.2.4 wurde deutlich, dass bei den 480 bzw. 240 möglichen Varianten die Konvergenz des Algorithmus nicht die Problematik darstellte, sondern die Einteilung des Netzwerkes in geeignete Communities. Weiterhin zeigte sich, dass es sinnvoll ist, mehrere Varianten zu nutzen. So können bestmögliche Ergebnisse erzielt werden.

Beispielsweise existieren für  $n = 200$  und  $k = 5$  noch 173 von 202 mögliche Varianten für  $p = \frac{1}{2}$ , die eine Modularität größer Null erzeugen. Bei einem Netzwerk mit  $n = 200$  und  $k = 10$  waren es für  $p = \frac{1}{2}$  lediglich noch 16 von 214 mögliche Varianten, dafür jedoch für  $p = \frac{1}{3}$  180 Möglichkeiten von 213.

Der Standard LPA ist durch die Vielzahl an Varianten recht flexibel auf unterschiedliche Netzwerke einsetzbar. Erreicht eine Version des LPA ein unzureichendes Ergebnis, so ist es möglich, durch geschicktes Kombinieren einer oder mehrere Parameter den LPA für dieses Netzwerk anzupassen.

## 5.2 Ausblick

Durch die Vielzahl an Varianten, die genutzt werden können, wurden hier nur einige getestet. In dieser Arbeit wurde die Konvergenz des Algorithmus in Verbindung mit der Modularität untersucht. Wie die Tabelle in 4.2.4 zeigt, existieren einige Varianten, die zu einer hohen Konvergenz führen. Allerdings ist ein hoher Wert nicht gleichbedeutend mit einer guten Modularität. Sodass hier weitere Varianten untersucht werden sollten, um zu ermitteln, weshalb diese konvergieren jedoch eine geringe Modularität aufzeigen.

Zusätzlich wurde beim Watts Strogatz Modell deutlich, dass der Wertebereich der Modularität verringert wird, sofern der Parameter  $k > 10\%$  von  $n$  entspricht. Gleichzeitig erhöht sich die Konvergenz des Algorithmus. Ob bei einem Wert von  $k < 10\%$  der Knoten eine verbesserte Modularität vorliegt und sich dadurch die Konvergenz des Algorithmus verringert, muss jedoch noch erwiesen werden.

Die Wahl der Label nimmt Einfluss auf Konvergenz und Modularität. Bestimmte Varianten wählen ein Label nach einem festgelegten Kriterium aus. Erfüllen mehrere Label das Kriterium erfolgt die Auswahl nach dem Zufallsprinzip. Für eine hohe Konvergenz müsste dieser Gleichstand behoben werden. Ob die Zufälligkeit der Labelwahl bei einem Gleichstand wichtig für die Modularität ist, bleibt zu belegen. Da hier lediglich das Watts Strogatz Modell mit unterschiedlichen Parametern untersucht wurde, empfiehlt es sich, die Varianten des LPA auf weitere Graphmodelle wie reale Netzwerke oder unterschiedliche Barabasi Albert Modelle [1] anzuwenden. Dadurch könnten Varianten ermittelt werden, die Gemeinsamkeiten auf unterschiedliche Netzwerke aufzeigen.

## Literatur

- [1] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.
- [2] Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. Network medicine: a network-based approach to human disease. *Nature reviews genetics*, 12(1):56–68, 2011.
- [3] Michael J Barber and John W Clark. Detecting network communities by propagating labels under constraints. *Physical Review E*, 80(2):026129, 2009.
- [4] Danielle S Bassett and Olaf Sporns. Network neuroscience. *Nature neuroscience*, 20(3):353–364, 2017.
- [5] Kamal Berahmand and Asgarali Bouyer. Lp-lpa: A link influence-based label propagation algorithm for discovering community structures in networks. *International Journal of Modern Physics B*, 32(06):1850062, 2018.
- [6] Kamal Berahmand, Sogol Haghani, Mehrdad Rostami, and Yuefeng Li. A new attributed graph clustering by using label propagation in complex networks. *Journal of King Saud University-Computer and Information Sciences*, 2020.
- [7] Norman Biggs, Norman Linstead Biggs, and Biggs Norman. *Algebraic graph theory*. Number 67. Cambridge university press, 1993.
- [8] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE transactions on knowledge and data engineering*, 20(2):172–188, 2007.
- [9] J Chalupa, P L Leath, and G R Reich. Bootstrap percolation on a bethe lattice. 12(1):L31–L35, jan 1979.
- [10] Yun Kwan Chan and Dit-Yan Yeung. A convex formulation of modularity maximization for community detection. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
- [11] Naiyue Chen, Yun Liu, Haiqiang Chen, and Junjun Cheng. Detecting communities in social networks using label propagation with information entropy. *Physica A: Statistical Mechanics and its Applications*, 471:788–798, 2017.
- [12] Jia Hou Chin and Kuru Ratnavelu. Detecting community structure by using a constrained label propagation algorithm. *Plos one*, 11(5):e0155320, 2016.

- [13] Aaron Clauset, Mark EJ Newman, and Christopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.
- [14] Gennaro Cordasco and Luisa Gargano. Label propagation algorithm: a semi-synchronous approach. *International Journal of Social Network Mining*, 1(1):3–26, 2012.
- [15] Reinhard Diestel. Extremal graph theory. In *Graph Theory*, pages 173–207. Springer, 2017.
- [16] Antonio Maria Fiscarelli, Matthias R Brust, Grégoire Danoy, and Pascal Bouvry. Local memory boosts label propagation for community detection. *Applied Network Science*, 4(1):1–17, 2019.
- [17] Bernd Gärtner and Ahad N. Zehmakan. Color war: Cellular automata with majority-rule. In Frank Drewes, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications*, pages 393–404, Cham, 2017. Springer International Publishing.
- [18] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.
- [19] Steve Gregory. Finding overlapping communities in networks by label propagation. *New journal of Physics*, 12(10):103018, 2010.
- [20] Karen Gunderson, Sebastian Koch, and Michał Przykucki. The time of graph bootstrap percolation. *Random Structures & Algorithms*, 51(1):143–168, 2017.
- [21] Xuegang Hu, Wei He, Huizong Li, and Jianhan Pan. Role-based label propagation algorithm for community detection. *arXiv preprint arXiv:1601.06307*, 2016.
- [22] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5070–5079, 2019.
- [23] Svante Janson, Tomasz Łuczak, Tatyana Turova, and Thomas Vallier. Bootstrap percolation on the random graph  $g_{n,p}$ . *The Annals of Applied Probability*, 22(5), Oct 2012.

- [24] Ian XY Leung, Pan Hui, Pietro Lio, and Jon Crowcroft. Towards real-time community detection in large networks. *Physical Review E*, 79(6):066107, 2009.
- [25] Huan Li, Ruisheng Zhang, Zhili Zhao, and Xin Liu. Lpa-mni: An improved label propagation algorithm based on modularity and node importance for community detection. *Entropy*, 23(5):497, 2021.
- [26] Wei Li, Ce Huang, Miao Wang, and Xi Chen. Stepping community detection algorithm based on label propagation and similarity. *Physica A: Statistical Mechanics and its Applications*, 472:145–155, 2017.
- [27] Shichao Liu, Fuxi Zhu, Huajun Liu, and Zhiqiang Du. A core leader based label propagation algorithm for community detection. *China Communications*, 13(12):97–106, 2016.
- [28] Wei Liu, Xingpeng Jiang, Matteo Pellegrini, and Xiaofan Wang. Discovering communities in complex networks by edge label propagation. *Scientific reports*, 6(1):1–10, 2016.
- [29] Xin Liu and Tsuyoshi Murata. Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Physica A: Statistical Mechanics and its Applications*, 389(7):1493–1500, 2010.
- [30] Jintao Meng, Dongmei Fu, and Tao Yang. Semi-supervised soft label propagation based on mass function for community detection. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 1163–1170. IEEE, 2018.
- [31] Glenn W Milligan and Martha C Cooper. Methodology review: Clustering methods. *Applied psychological measurement*, 11(4):329–354, 1987.
- [32] Alan Mislove, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, 2007.
- [33] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.
- [34] Mark EJ Newman and Duncan J Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, 1999.

- [35] Raj Kumar Pan and Sitabhra Sinha. Modularity produces small-world networks with dynamical time-scale separation. *EPL (Europhysics Letters)*, 85(6):68006, 2009.
- [36] Ferran Parés, Dario Garcia Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguadé, Jesús Labarta, Ulises Cortés, and Toyotaro Suzumura. Fluid communities: A competitive, scalable and diverse community detection algorithm. In *International Conference on Complex Networks and their Applications*, pages 229–240. Springer, 2017.
- [37] KARL PEARSON. The Problem of the Random Walk. *Nature*, 72(1865):294–294, 1905.
- [38] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293. Springer, 2005.
- [39] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663, 2004.
- [40] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.
- [41] Benno Schwikowski, Peter Uetz, and Stanley Fields. A network of protein–protein interactions in yeast. *Nature biotechnology*, 18(12):1257–1261, 2000.
- [42] Olaf Sporns and Richard F Betzel. Modular brain networks. *Annual review of psychology*, 67:613–640, 2016.
- [43] Lovro Šubelj. Label propagation for clustering. *Advances in network clustering and blockmodeling*, pages 121–150, 2019.
- [44] Lovro Šubelj and Marko Bajec. Robust network community detection using balanced propagation. *The European Physical Journal B*, 81(3):353–362, 2011.
- [45] Lovro Šubelj and Marko Bajec. Unfolding communities in large complex networks: Combining defensive and offensive label propagation for core extraction. *Physical Review E*, 83(3):036103, 2011.

- [46] Heli Sun, Jiao Liu, Jianbin Huang, Guangtao Wang, Xiaolin Jia, and Qinbao Song. Linklpa: A link-based label propagation algorithm for overlapping community detection in networks. *Computational Intelligence*, 33(2):308–331, 2017.
- [47] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.
- [48] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.
- [49] Stanley Wasserman, Katherine Faust, et al. Social network analysis: Methods and applications. 1994.
- [50] Duncan J Watts and Steven H Strogatz. Collective dynamics of “small-world” networks. *nature*, 393(6684):440–442, 1998.
- [51] Zhi-Hao Wu, You-Fang Lin, Steve Gregory, Huai-Yu Wan, and Sheng-Feng Tian. Balanced multi-label propagation for overlapping community detection in social networks. *Journal of Computer Science and Technology*, 27(3):468–479, 2012.
- [52] Jierui Xie and Boleslaw K Szymanski. Community detection using a neighborhood strength driven label propagation algorithm. In *2011 IEEE Network Science Workshop*, pages 188–195. IEEE, 2011.
- [53] Jierui Xie and Boleslaw K Szymanski. Towards linear time overlapping community detection in social networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 25–36. Springer, 2012.
- [54] Jierui Xie and Boleslaw K Szymanski. Labelrank: A stabilized label propagation algorithm for community detection in networks. In *2013 IEEE 2nd Network Science Workshop (NSW)*, pages 138–143. IEEE, 2013.
- [55] Yan Xing, Fanrong Meng, Yong Zhou, Mu Zhu, Mengyu Shi, and Guibin Sun. A node influence based label propagation algorithm for community detection in networks. *The Scientific World Journal*, 2014, 2014.
- [56] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

- [57] Aiping Zhang, Guang Ren, Yejin Lin, Baozhu Jia, Hui Cao, Jundong Zhang, and Shubin Zhang. Detecting community structures in networks by label propagation with prediction of percolation transition. *The Scientific World Journal*, 2014, 2014.
- [58] Xian-Kun Zhang, Jing Ren, Chen Song, Jia Jia, and Qian Zhang. Label propagation algorithm for community detection based on node importance and label influence. *Physics Letters A*, 381(33):2691–2698, 2017.
- [59] Yuxin Zhao, Shenghong Li, and Xiuzhen Chen. Community detection using label propagation in entropic order. In *2012 IEEE 12th International Conference on Computer and Information Technology*, pages 18–24. IEEE, 2012.
- [60] Kuang Zhou, Arnaud Martin, Quan Pan, and Zhunga Liu. Selp: Semi-supervised evidential label propagation algorithm for graph data clustering. *International Journal of Approximate Reasoning*, 92:139–154, 2018.