Kontakt : <xuan--2004.07.14--graphentheorie-bioinformatik@studium.baldauf.org>

Datum: 16.04.2004

Graphentheorie in der Bioinformatik

Organisatorisches

- Termine
  - 21.04.2004 (2 Mal)
  - 28.04.2004
  - 05.05.2004 (2 Mal)
  - 19.05.2004 (durch Vertretung)
  - 26.05.2004
  - 09.06.2004 (durch Vertretung)
  - 16.06.2004 (2 Mal)
  - 30.06.2004 (2 Mal)
- Prüfungsform
  - „so unbürokratisch, wie gesetzlich möglich“
  - wahrscheinlich:
    - reale Prüfungsform mündlich
      - „Da setzen wir uns zu einem Kaffee hin und unterhalten uns über Graphentheorie. Dann sollten Sie zeigen, dass Sie sich erinnern können an die Vorlesung, …“
    - offizielle Prüfungsform mündlich oder schriftlich

Themen

„Graphentheorie ist doch eins der Themen, die in der Bioinformatik eine relativ große Rolle spielen. Nicht nur dass sie für Algorithmen und Datenstrukturen die Grundlage ist, sondern auch weil Graphen in einem gewissen Sinne eine natürliche Art und Weise sind, biologische Strukturen zu beschreiben. Wir werden uns mit graphentheoretischen Beschreibungen auseinandersetzen, zum einen Molekülstrukturen (grobkörnige Beschreibung von Molekülstrukturen, typischerweise ungerichtete Graphen), zum anderen die Beschreibung von Metabolischen Netzwerken, Signalnetzwerken, funktionalen Zusammenhängen in der Zelle.“

„Wir werden uns mehr auf die Themen konzentrieren, die im Mainstream der Literatur eher vernachlässigt werden.“

- struktur von Kreisen in Graphen
- Begriffe von Zentralität in einem Graphen (Welcher Knoten ist zentraler als ein anderer Knoten?)
- eher klassische Geschichten
  - Einbettbarkeit von Graphen in die Ebene
  - ohne Überschneidung von Kanten
  - planare Graphen
- Teilgraphen
- Färbungen von Graphen (und ihre Beziehung zu Sequenzen)

„Die Vorlesung wird eher einzelne Teilthemen streifen, die untereinander eher wenig Verbunden sind."

Wenn es Themengebiete gibt, die Sie besonders interessieren, dann sagen Sie das bitte rechtzeitig, da dies in das Vorlesungsprogramm bis zu einem gewissen Grad einfließen kann.

Definition: ungerichteter Graph

„So'n Graph ist ein Ding, das besteht aus Knoten und es besteht aus Kanten, und an der Stelle gibt es schon die ersten Unterschiede in der Literatur, wie man einen Graphen aufzufassen hätte. Einiges davon ist mathematisches Detail, was in der Praxis eigentlich nicht so relevant ist. Hier können wir sehen, wie man sich das Leben erschweren kann."
Sei

- $V \in Mengen$
- $|V| = N$
- $|V| \in \mathbb{N}_0^+$ (Menge der Kanten ist abzählbar und endlich)
- $E = \{(x, y) | (x \in V) \wedge (y \in V) \wedge (x \neq y)\}$

Ein **ungerichteter Graph** ist eine Liste $G = (V, E)$.

Bemerkung

- E steht für „edges" („Kanten"),
- V steht für „vertices" („Knoten")

Bemerkung

Diese Definition von ungerichteten Graphen lässt wegen $\forall((x, y) \in E) : (x \neq y)$ keine Schlaufen (Kanten von einem Knoten zu dem selben Knoten) zu.

Definition: gerichteter Graph

Sei

- $V \in Mengen$
- $|V| = N$
- $|V| \in \mathbb{N}_0^+$ (Menge der Kanten ist abzählbar und endlich)
- $E = \{(x, y) | (x \in V) \wedge (y \in V)\}$

Es gilt $E \subseteq (V \times V)$
Ein **gerichteter Graph** ist eine Liste $G = (V, E)$.

Definition: orientierter Graph

Ein gerichteter Graph ohne Schlaufen $G = (V, E)$ heißt **orientierter Graph**, wenn

- $\forall((x, y) \in E) : ((y, x) \notin E)$

also wenn es für jedes Knotenpaar maximal eine Kante, also

- entweder in die eine Richtung
- oder in die andere Richtung
- oder garnicht

existiert.

Definition: Multimenge

Eine Multimenge ist eine Menge, in der ihre Elemente mehrmals vorkommen dürfen und jedes Vorkommen eines Elements unterschiedliche Identität hat.

Definition: Multigraph

Sei
- $V \in Mengen$
- $|V| = N$
- $|V| \in \mathbb{N}_0^+$ (Menge der Kanten ist abzählbar und endlich)
- $i: E \to V$ eine Funktion
- $t: E \to V$ eine Funktion
- $E \subseteq (V \times V).createSuperMultiset()$

Eine Liste $\Gamma = (V, E, i, t)$ heißt **Multigraph**.

Bemerkung

Die Menge E ist eine Multimenge. Beispielsweise ist $E = \{(1,2),(1,2),(1,2)\}$.

Bemerkung

Für eine Kante e beschreibt $i(e)$ den Anfangsknoten der Kante e.
Für eine Kante e beschreibt $t(e)$ den Endknoten der Kante e.

Definition: Knotenanzahl

Sei $G = (V, E)$ ein Graph. Die Zahl $|V| = n = |G.getKnoten()| = G.getKnoten().length()$ heißt **Knotenanzahl** dieses Graphen.

Definition: Kantenanzahl

Sei $G = (V, E)$ ein Graph. Die Zahl $|E| = m = |G.getKanten()| = G.getKanten().length()$ heißt **Kantenanzahl** dieses Graphen.

Definition: Inzidenzmatrix

Sei $G = (V, E)$ ein ungerichteter Graph.

Eine Matrix H mit $\forall(x \in V): \forall(e \in E): \left(H(x,e) = \begin{cases} 1 & wenn\ x \in e \\ 0 & sonst \end{cases} \right)$ heißt **Inzidenzmatrix** dieses Graphen.

Definition: Inzidenzmatrix

Sei $G = (V, E)$ ein gerichteter Graph.

Eine Matrix H mit $\forall(x \in V): \forall(e \in E): \left(H(x,e) = \begin{cases} +1 & wenn\ \exists(z \in V): (e = (x,z)) \\ -1 & wenn\ \exists(z \in V): (e = (z,x)) \\ 0 & sonst \end{cases} \right)$ heißt **Inzidenzmatrix** dieses Graphen.

Definition: Adjazenzmatrix.

Sei $G = (V, E)$ ein ungerichteter Graph.

Eine Matrix A mit $\forall(x\in V):\forall(y\in V):\left(A(x,y)=\begin{cases}1 & wenn\ (x,y)\in E\\ 0 & sonst\end{cases}\right)$ heißt **Adjazenzmatrix** in dieses Graphen.

Definition: Adjazenzmatrix.

Sei $G=(V,E)$ ein gerichteter Graph.

Eine Matrix A mit $\forall(x\in V):\forall(y\in V):\left(A(x,y)=\begin{cases}1 & wenn\ (x,y)\in E\\ 0 & sonst\end{cases}\right)$ heißt **Adjazenzmatrix** in dieses Graphen.

Definition: Grad

Sei $G=(V,E)$ ein ungerichteter Graph.
Es sei die Funktion $d:V\to\mathbb{N}$ mit $\forall(v\in V):\left(d(v)=\left|\{e|(e\in E)\wedge(v\in e)\}\right|\right)$ definiert.
Für einen Knoten $v\in V$ heißt $d(v)$ **Grad** dieses Knotens.

Definition: eingehende Kanten

Sei $G=(V,E)$ ein gerichteter Graph.
$\forall(v\in V):\left(v.getEingehendeKanten()=\left\{e\middle|(e\in E)\wedge\exists(y\in V):((y,v)=e)\right\}\right)$

Definition: ausgehende Kanten

Sei $G=(V,E)$ ein gerichteter Graph.
$\forall(v\in V):\left(v.getAusgehendeKanten()=\left\{e\middle|(e\in E)\wedge\exists(y\in V):((v,y)=e)\right\}\right)$

Definition: indegree

Sei $G=(V,E)$ ein ungerichteter Graph.
Es sei die Funktion $indegree:V\to\mathbb{N}$ mit

- $\forall(v\in V):\left(indegree(v)=\left|v.getEingehendeKanten()\right|\right)$

definiert.

Definition: outdegree

Sei $G=(V,E)$ ein ungerichteter Graph.
Es sei die Funktion $outdegree:V\to\mathbb{N}$ mit:

- $\forall(v\in V):\left(outdegree(v)=\left|v.getAusgehendeKanten()\right|\right)$

definiert.

Definition: Grad

Sei $G=(V,E)$ ein gerichteter Graph.
Es sei die Funktion $d:V\to\mathbb{N}$ mit $\forall(v\in V):(d(v)=indegree(v)+outdegree(v))$ definiert.
Für einen Knoten $v\in V$ heißt $d(v)$ **Grad** dieses Knotens.

Definition: Admittanz-Matrix

Sei

- $G=(V,E)$ ein ungerichteter Graph und

- $H = G.getInzidenzmatrix()$

Was ist dann $(H \cdot H^T)$?

$$\forall (x \in V): \forall (y \in V): ($$
$$(H \cdot H^T)(x, y) = \sum_{\forall (e \in E)} (H(x, e) \cdot H^T(e, y))$$
$$= \sum_{\forall (e \in E)} (H(x, e) \cdot H(y, e))$$
$$= \begin{cases} 0 & wenn\ (\{x, y\} \notin E) \wedge (x \neq y) \\ 1 & wenn\ (\{x, y\} \in E) \wedge (x \neq y) \\ d(x) & wenn\ x = y \end{cases}$$
$$)$$

Damit ist $(H \cdot H^T) = A + D$ mit $\forall (x \in V): \forall (y \in V): (D(x, y) = d(x) \cdot \delta(x, y))$ (wobei $\forall (x): \forall (y): \left(\delta(x, y) = \begin{cases} 1 & wenn\ x = y \\ 0 & sonst \end{cases}\right)$ die Kronecker-Funktion ist).

Die Matrix $H \cdot H^T$ heißt **Admittanz-Matrix** des Graphen G.

Definition: Laplace-Matrix

Sei

- $G = (V, E)$ ein gerichteter Graph und
- $H = G.getInzidenzmatrix()$

Was ist dann $(H \cdot H^T)$?

$$\forall (x \in V): \forall (y \in V): ($$
$$(H \cdot H^T)(x, y) = \sum_{\forall (e \in E)} (H(x, e) \cdot H^T(e, y))$$
$$= \sum_{\forall (e \in E)} (H(x, e) \cdot H(y, e))$$
$$= \begin{cases} 0 & wenn\ ((x, y) \notin E) \wedge (x \neq y) \\ -1 & wenn\ (((x, y) \in E) \vee ((y, x) \in E)) \wedge (x \neq y) \\ d(x) & wenn\ x = y \end{cases}$$
$$)$$

Damit ist $(H \cdot H^T) = A - D$ mit $\forall (x \in V): \forall (y \in V): (D(x, y) = d(x) \cdot \delta(x, y))$

Die Matrix $H \cdot H^T$ heißt **Laplace-Matrix** des Graphen G.

Bemerkung

Die Laplace-Matrix macht für gerichtete Graphen keinen Sinn.

Definition: Gradmatrix

Sei

- $G = (V, E)$ ein Graph.

Dann ist die Matrix $D: V \times V \to \mathbb{N}_0^+$ diesen Graphen so definiert

$$\forall (x \in V): \forall (y \in V): (D(x, y) = d(x) \cdot \delta(x, y))$$

Definition: Laplace-Matrix

Sei

- $G=(V,E)$ ein ungerichteter Graph.
- $G_2=G.verrichte(orient)$ (Erzeuge den selben Graphen, wobei jede Kante genau eine Richtung bekommt)
- $H_2=G_2.getInzidenzmatrix()$
- $L=H_2\cdot H_2^T$

Die Matrix L heißt Laplace-Matrix des Graphens G.

Satz

Sei
- $G=(V,E)$ ein ungerichteter Graph.
- $L=G.getLaplaceMatrix()$
- $A=G.getAdjazenzMatrix()$
- $D=G.getGradMatrix()$

Dann gilt:
- $L=D-A$

Definition: Weg

Sei $G=(V,E)$ ein Graph und
eine Liste $w=(e_1,e_2,e_3,\ldots,e_n)$ mit
- $\forall(i):(e_i\in G.getDirectedEdges())$ und
- mit $knoten=w.getVertexList()$:
  $\forall(i\in([0,|knoten|[\cap\mathbb{N})):((x_i,x_{i+1})\in G.getDirectedEdges())$

heißt **Weg** in diesem Graphen.

Definition: getStartVertexList

Sei $G=(V,E)$ ein Graph w ein Weg in diesem Graph.
Dann ist die Liste $w.getStartVertexList():\mathbb{N}\to V$ definiert als
$\forall(k\in([0,|w|[\cap\mathbb{N})):(w.getStartVertexList()[k]=w[k].getStartVertex())$.

Definition: getVertexList

Sei $G=(V,E)$ ein Graph w ein Weg in diesem Graph.
Dann ist die Liste $w.getVertexList():\mathbb{N}\to V$ definiert als
$w.getVertexList()=w.getStartVertexList()\cdot(w.getLastElement().getEndVertex())$

Definition: getStartVertex

Sei $G=(V,E)$ ein Graph w ein Weg in diesem Graph.
Dann ist $w.getStartVertex()=w.getVertexList().getFirstElement()$.

Definition: getEndVertex

Sei $G=(V,E)$ ein Graph und w ein Weg in diesem Graph.
Dann ist $w.getEndVertex()=w.getVertexList().getLastElement()$.

Definition: geschlossen

Sei w ein Weg. Wenn $w.getEndVertex()=w.getStartVertex()$, dann heißt w geschlossen.

Definition: offen

Sei w ein Weg. Wenn $\neg w.isClosed()$, dann heißt w offen.

Definition: Weg ohne mehrfache Kanten

Sei w ein Weg. Enthält w nicht die selbe Kante mehrfach, dann heißt w **Weg ohne mehrfache Kanten**.

Definition: Kreis

Sei w ein Weg ohne mehrfache Kanten. Ist w geschlossen, dann heißt w **Kreis**.

Definition: Pfad

Sei w ein Weg ohne mehrfache Kanten. Ist w offen, dann heißt w **Pfad**.

Definition: Weg ohne mehrfache Knoten

Sei w ein Weg. Enthält $w.getStartKnotenListe()$ nicht den selben Knoten mehrfach, dann heißt w **Weg ohne mehrfache Knoten**.

Definition: einfacher Kreis

Sei w ein Weg ohne mehrfache Knoten. Ist w geschlossen, dann heißt w **einfacher Kreis**.

Definition: einfacher Pfad

Sei w ein Weg ohne mehrfache Knoten. Ist w offen, dann heißt w **einfacher Pfad**.

[...]

$$|W_{x,z}| = |W_{x,y}| + |W_{y,z}|$$

Definition: Länge

Sei $G=(V,E)$ ein kantengewichteter Graph.
Für eine Kante $e \in E$ sei $l(e)=e.getLength()$ die **Länge** (das Gewicht) dieser Kante.

Definition: Länge eines Weges

Sei $G=(V,E)$ ein kantengewichteter Graph und w ein Weg in diesem Graphen.
Die **Länge** des Weges ist $|w| = \sum_{\forall (e \in w)} (e.getLength()) = w.getLength()$.

Definition: istZwischen

Sei $G=(V,E)$ ein kantengewichteter Graph und w ein Weg in diesem Graphen.
$\forall (x \in V): \forall (y \in V): (w.istZwischen(x,y) \Leftrightarrow$
$((w.getVertexList().getFirstElement()=x) \wedge (w.getVertexList().getLastElement()=y))$

Definition: Abstand

Sei

- $G=(V,E)$ ein kantengewichteter Graph

- $x \in V$
- $y \in V$

Die Zahl $d(x,y) = \{|w| w.istZwischen(x,y)\}.getMinimalElement() = G.getDistance(x,y)$ heißt Abstand zwischen dem Knoten x und y.

Definition: zusammenhängend

Sei $G=(V,E)$ ein kantengewichteter Graph.
Der Graph G heißt **zusammenhängend** („**connected**“) genau dann, wenn
$\forall(x \in V): \forall(y \in V): \exists(w \in G.getWege()): (w.istZwischen(x,y))$

Satz

Sei $G=(V,E)$ ein kantengewichteter Graph.
$G.istZusammenhängend() \Leftrightarrow \forall(x \in V): \forall(y \in V): (d(x,y) < +\infty)$

Satz

Sei
- $G=(V,E)$ ein ungerichteter Graph.
- $L = G.getLaplaceMatrix()$
- $eigenwertemenge = L.getEigenwerte()$
- $n = |eigenwertemenge|$

Dann gilt:
n ist die Anzahl der Zusammenhangskomponenten (zusammenhängenden Subgraphen) des Graphen G.

Bemerkung

Es ist möglicherweise überraschend, dass geometrische Eigenschaften wie Anzahl zusammenhängender Subgraphen rein algebraisch ermittelt werden können.

[...irgendwas über den Laplace-Operator, warum er als $\Delta = A - D$ und nicht $\Delta = A + D$ definiert ist ...]
Betrachten wir ein ebenes Gitter als Spezialfall eines Graphen. Jeder Knoten x bekommt eine Höhe (ein Gewicht) $f(x)$ zugeordnet. Dann gilt:

$$\begin{aligned}\Delta f(x) &= „\frac{d^2 \cdot f}{d \cdot x^2}(x)“ + „\frac{d^2 \cdot f}{d \cdot y^2}(x)“ \\ &= f(x_1) + f(x_2) - 2 \cdot f(x) + f(y_1) + f(y_2) - 2 \cdot f(x) \\ &= \sum_{\forall(|z|z,x| \in E)} (f(z) - f(x)) \\ &= \sum_{\forall(z)} (A(z,x) \cdot f(z) - d(x) \cdot f(x)) \\ &= (A - D) \cdot f(x)\end{aligned}$$

Also ist $\Delta = A - D$.

Definition: Durchmesser

Sei

- $G=(V,E)$ ein ungerichteter Graph.

Die Zahl $diam(G)=\{d(x,y)\,|\,(x\in V)\wedge(y\in V)\}.\,getMaximum()=G.getDiameter()$ heißt **Durchmesser** dieses Graphen.

Definition: Exzentrizität

Sei
- $G=(V,E)$ ein ungerichteter, kantengewichteter Graph,
- $G.istZusammenhängend()$,
- $x\in V$ ein Knoten.

Die Zahl $e(x)=\{d(x,y)\,|\,y\in V\}.\,getMaximum()=x.getExcentricity()$ heißt **Exzentrizität** des Knotens x.

Bemerkung

Die Exzentrizität eines Knotes ist ein Maß dafür, wie „zentral“ der Knoten innerhalb des Graphens ist.
Je kleiner die Exzentrizität des Knotens ist, desto zentraler ist der Knoten.
Je größer die Exzentrizität des Knotens ist, desto exzentrischer („weiter draußen“) ist der Knoten.

Definition: minimale Exzentrizität

Sei
- $G=(V,E)$ ein ungerichteter, kantengewichteter Graph,
- $G.istZusammenhängend()$.

Die Zahl $G.getMinimaleExzentrizität()=\{x.getExcentricity()\,|\,x\in V\}.\,getMinimum()$ bezeichnet die **minimale Exzentrizität** des Graphen, also den minimalen maximalen Abstand, den ein Knoten zum „Rand“ des Graphen haben kann.

Definition: Zentrum

Sei
- $G=(V,E)$ ein ungerichteter Graph,
- $G.istZusammenhängend()$.

Die Menge $C(G)=\{x\in V\,|\,e(x)=G.getMinimaleExzentrizität()\}=G.getCentralVertices()$ heißt **Zentrum**.

Resouce-Placement-Probleme

Thema

Wir nehmen an, dass wir einen ungerichteten, nicht kantenbewerteten Graphen G haben. Uns interessiert:

1. Gegeben sei Straßennetzwerk. Wohin soll ein Krankenhaus platziert werden, sodass die (maximalen) Anfahrtswege (-Zeiten) für die Rettung minimal sind.
   - Wir suchen also $\rho(G)=\min\limits_{\forall(x\in X)}\Big(\underbrace{\max\limits_{\forall(y\in Y)}(G.getShortestPathFromTo(x,y).\,getLength())}_{e(x)}\Big)$,

     wobei
     - X die Menge der Krankenhäuse und

- Y die Menge der potentiellen Patienten ist.

und damit

$$C(G)=\left\{x\left|\rho(G)=\underbrace{\max_{\forall(y\in Y)}\left(G.getShortestPathFromTo(x,y).getLength()\right)}_{e(x)}\right.\right\}$$
$$=G.getCentre()$$

2. Gegeben sei Straßennetzwerk. Wohin soll ein Einkaufszentrum platziert werden, sodass die (durchschnittlichen) Anfahrtsweg für die Kunden minimal sind. (Die Menge der Kunden maximal sind.) („Das ganze ist eine amerikanisierte Variante, wo solche Dinge wie Ensembleschutz keine Berücksichtigung finden.“)
3. Gegeben sei Straßennetzwerk. Wohin soll ein Einkaufszentrum platziert werden, sodass die (durchschnittlichen) Anfahrtsweg für die Kunden minimal sind (die Menge der Kunden maximal sind), wobei wir wissen, dass ein Konkurrent kurz nach unserer Entscheidung ebenfalls ein Einkaufszentrum platzieren will. Ziel ist, den Konkurrenten möglichst zu behindern, um im Verhältnis die meisten Kunden erreichen zu können.

Definition: Status

Sei
- $G=(V,E)$ ein ungerichteter, kantengewichteter[?] Graph,
- $G.istZusammenhängend()$,
- $x\in V$ ein Knoten.

Dann heißt $x.getStatus()=\sum_{\forall(y\in G.getVertices())}\left(d(x,y)\right)$ **Status** des Knotens x.

Definition: Status

Sei
- $G=(V,E)$ ein ungerichteter, kantengewichteter[?] Graph,
- $G.istZusammenhängend()$.

Dann heißt $\sigma(G)=G.getStatus()=\left\{\frac{1}{|V|}\cdot x.getStatus()\left|x\in V\right.\right\}.getMinimum()$ **Status** des Graphen G.

Definition: Median

- $G=(V,E)$ ein ungerichteter, kantengewichteter[?] Graph,
- $G.istZusammenhängend()$.

Dann heißt $M(G)=G.getMedian()=\{x.getStatus()=G.getStatus()\}$ **Median** dieses Graphen.

Beispiel: Einkaufszentrumplatzierung mit Konkurrenz

Sei
- $G=(V,E)$ ein ungerichteter, kantengewichteter[?] Graph,
- $G.istZusammenhängend()$,
- x der Knoten für mein Einkaufszentrum,
- y der Knoten für das Einkaufszentrum meines Konkurrenten

Dann bezeichne
- $V_{x,y}=\left\{w\left|(w\in V)\wedge(G.getDistance(x,w)<G.getDistance(y,w))\right.\right\}$ die Kunden, die mir näher sind als dem Konkurrenten und

- $V_{y,x}=\{w|(w\in V)\wedge(G.getDistance(x,w)>G.getDistance(y,w))\}$ die Kunden, die meinem Konkurrenten näher sind als mir.

Ziel ist dann $\min_{\forall(x\in V)}\left(\max_{\forall(y\in(V\setminus\{x\}))}(|V_{y,x}|-|V_{x,y}|)\right)$.

Wir gehen hier aber implizit davon aus, dass der Konkurrent rational entscheidet.

Der Aufwand, um $\min_{\forall(x\in V)}\left(\max_{\forall(y\in(V\setminus\{x\}))}(|V_{y,x}|-|V_{x,y}|)\right)$ auszurechnen, ist aber kubisch. Wir können aber beweisen, dass gilt:

- $\forall(x\in V):\forall(y\in V):(x.getStatus()+|V_{x,y}|=y.getStatus()+|V_{y,x}|)$ und damit gilt
- $\forall(x\in V):\forall(y\in V):(x.getStatus()-y.getStatus()=|V_{y,x}|-|V_{x,y}|)$

Also ist

$$\begin{aligned}\min_{\forall(x\in V)}\left(\max_{\forall(y\in(V\setminus\{x\}))}(|V_{y,x}|-|V_{x,y}|)\right)&=\min_{\forall(x\in V)}\left(\max_{\forall(y\in(V\setminus\{x\}))}(x.getStatus()-y.getStatus())\right)\\&=\min_{\forall(x\in V)}\left(x.getStatus()+\max_{\forall(y\in(V\setminus\{x\}))}(-y.getStatus())\right)\\&=\min_{\forall(x\in V)}\left(x.getStatus()-\min_{\forall(y\in(V\setminus\{x\}))}(y.getStatus())\right)\end{aligned}$$

Definition: Zentroid-Wert

Also benenn'ich die zu minierende Funktion $f:V\to\mathbb{Q}$ mit

- $\forall(x\in V):\left(f(x)=x.getCentroidValue()=x.getStatus()-\min_{\forall(y\in(V\setminus\{x\}))}(y.getStatus())\right)$

Diese Funktion liefert mir einen **Zentroid-Wert** für jeden Knoten des Graphen.

Definition: minimaler Zentroid-Wert

Dann heiße $\varphi(G)=G.getMinimalCentroidValue()=min\forall(x\in V)(x.getCentroidValue())$ **minimaler Zentroid-Wert** des Graphen G.

Definition: Zentroid

Dann heißt die Knotenmenge
$Z(G)=G.getCentroid()=\{x|(x\in V)\wedge(f(x)=G.getMinimalCentroidValue())\}$
Zentroid des Graphen G.

Wir haben hier also 3 Begriffe von Zentralität, nämlich

- Zentrum und
- Median und
- Zentroid.

Es gibt Beweise, die zeigen, dass man Graphen konstruieren kann, dass diese 3 Knotenmengen disjunkt sind. Man kann sogar Graphen konstruieren, die zeigen, dass die paarweisen Abstände dieser Mengen eine bestimmte Mindestgrenze nicht unterscheiden dürfen.

Literaturverweis

Slater, Nework 34: 224..249

Bemerkung

- Der Median eines Graphen sollte eigentlich Zentroid heißen, weil er sowas wie einen Schwerpunkt darstellt.
- Das Zentrum eines Graphen sollte eigentlich Median heißen, da links und rechts etwa gleich viel Knoten sind.

Definition: liegt zwischen

Sei
- $G=(V,E)$ ein ungerichteter, kantengewichteter[?] Graph,
- $G.istZusammenhängend()$,
- $x \in V$,
- $y \in V$,
- $m \in V$

Betrachten wir die Menge aller kürzesten Pfade von x nach y im Graphen G.
Der Knoten m **liegt zwischen** x und y ($m.liegtZwischen(x,y)$) genau dann, wenn
- $G.getDistance(x,m)+G.getDistance(m,y)=G.getDistance(x,y)$

Definition: betweenness centrality

Sei
- $G=(V,E)$ ein ungerichteter, kantengewichteter[?] Graph,
- $G.istZusammenhängend()$,
- $m \in V$.

Dann heißt $b(m)=m.getBetweennessCentrality()=\left|\{(x,y)\,|\,m.liegtZwischen(x,y)\}\right|$
betweenness centrality von m.

Bemerkung

Die betweenness centrality bezeichnet also ungefähr die Eignung von Knoten als Verkehrsknotenpunkt (also Knoten, an denen man sowieso vorbei kommt, wenn man sich schon im Graphen bewegt) (z.B. für die Tankstellenplatzierung, da diese gerne ohne Umweg (d.h. Wegverlängerung, nur um die Tankstelle zu erreichen) angefahren werden wollen).

Definition: Teilgraph

Sei
- $G=(V,E)$ ein ungerichteter, nicht kantengewichteter Graph,
- $W \subseteq V$
- $F \subseteq E$
- $H=(W,F)$
- $H.istGraph()$

Dann heißt H **Teilgraph** von G.

Definition: induzierter Teilgraph

Sei
- $G=(V,E)$ ein ungerichteter, nicht kantengewichteter Graph,
- $H=(W,F)$,
- $H.istTeilgraphVon(G)$,
- $\forall(x \in W):\forall(y \in W):((\{x,y\} \in F) \Leftrightarrow (\{x,y\} \in E))$

Dann heißt H **induzierter Teilgraph** von G.

Bemerkung

Bei induzierten Teilgraphen reicht die Angabe der Knotenmenge W einfach aus, die „dafür noch möglichen“ Kanten werden einfach „mitgeerbt“.

Schreibweise

$G.getInduzierterTeilgraphFür(W)=G[W]$

Definition: konvexer Teilgraph

Sei
- $G=(V,E)$ ein ungerichteter, nicht kantengewichteter Graph,
- $H=(W,F)$,
- $H.istInduzierterTeilgraphVon(G)$
- $\forall(x\in W):\forall(y\in W):(G.getKürzesteWegeFür(x,y)\subseteq H.getWege())$

Dann heißt H **konvexer Teilgraph** von G.

„Der Begriff der Konvexität macht nur Sinn in Bezug auf eine übergeordnete Struktur (Supergraph, etc).“
„Wenn ich einen Hünergott habe, und sage, er ist konvex, dann bezieht sich das automatisch auf $\mathbb{R}^3$“.

Definition: Zweig

Sei
- $G=(V,E)$ ein ungerichteter, nicht kantengewichteter Graph,
- $x\in V$,
- $H=(W,F)$ sei ein konvexer Teilgraph von G, wobei
  - $x\notin W$ und
  - H ein maximaler (bezüglich der Zahl der Knoten) (bezüglich aller Teilgraphen von G, zu dessen Knotenmengen x nicht gehört) konvexer Teilgraph von G.

Dann heißt H **Zweig** von G in x.

Definition: Zweiggewicht, branch weight

- $G=(V,E)$ ein ungerichteter, nicht kantengewichteter Graph,
- $x\in V$,
- $H=(W,F)$ sei ein Zweig von G in x.

Dann heißt die Zahl $h(x)=|H.getVertices()|$ **Zweiggewicht** von G am Knoten x („**branch weight** at x“).

Bemerkung

Man denke an die Bäume. Da ist das Zweiggewicht ein Maß für die Assymetrie.

Bemerkung

Man kann einen Zentralitätsbegriff definieren, der eine Knotenmenge beschreibt im Graphen beschreibt, die minimales Zweiggewicht nach allen Seiten haben.

[...hier sind ziemlich viele Lücken :-(...]

Beispiel: Königsberger Brückenproblem

Das älteste Problem der Graphentheorie ist wahrscheinlich das Königsberger Brückenproblem. Es gibt dort mehrere Brücken und Inseln. Die Frage war wahrscheinlich schon im 18. Jahrhundert, wie man einen Rundgang durch Königsberg macht, sodass man jede Brücke (Kante) nur genau einmal passiert.

Definition: eulerscher Weg

Gegeben sei
- ein Graph $G=(V,E)$,
- ein Weg W.

Der Weg W heißt **eulerscher Weg** genau dann, wenn er jede Kante aus E genau einmal passiert.

Beispiel: Eulersches Problem

Suche nach einem eulerschen Weg für einen gegebenen Graphen.

Definition: eulerscher Graph

Gegeben sei
- ein Graph $G=(V,E)$.

Der Graph heißt **eulerscher Graph** genau dann, wenn
- $\exists(W \in G.getWege()):(W.istEulerscherWeg())$

Lemma

Sei
- $G=(V,E)$ ein ungerichteter Graph,
- $G.istEulerscherGraph()$.

Dann gilt:
- $G.istZusammenhängend()$.

Lemma

Sei
- $G=(V,E)$ ein ungerichteter Graph,
- $G.istEulerscherGraph()$.

Dann gilt:
- $\forall(x \in V):\left(G.getGradFür(x) \in \left(\mathbb{N}_0^+ \cdot 2\right)\right)$

Beweis

Sei
- $W \in G.getEulerscheWege()$,
- $x \in V$

Dann sei

- $W_T \in G.getWege()$ mit
  - $W_T.istTeilweg(W)$ und
  - $W_T.getStartVertex() = x$ und
  - $W_T.getEndVertex() = x$.

Dann definieren wir den Graphen
$G_2 = (V, E \setminus W_T.getEdges())$. Wir wissen dann, dass
$G_2.getGradFür(x) = G.getGradFür(x) - 2$.
Der reduzierte Graph enthält wieder einen eulerschen Weg.

Dieses Verfahren wiederhole man so lange, wie der Grad des Knoten x im jeweiligen Graphen größer 0 ist.

- Ist er genau 0, so ist klar, dass der Grad des Knoten x im ursprünglichen Graphen G eine gerade Zahl ist.
- Ist der Grad einmal genau 1 in einem Graphen der Reduktionskette, so gibt es keinen eulerschen Weg für diesen Graphen und somit auch für alle Graphen davor, somit auch für die Graphen G. Das heißt, dass die Voraussetzung, $G.istEulerschergraph()$ falsch ist. Dies ist ein Widerspruch.

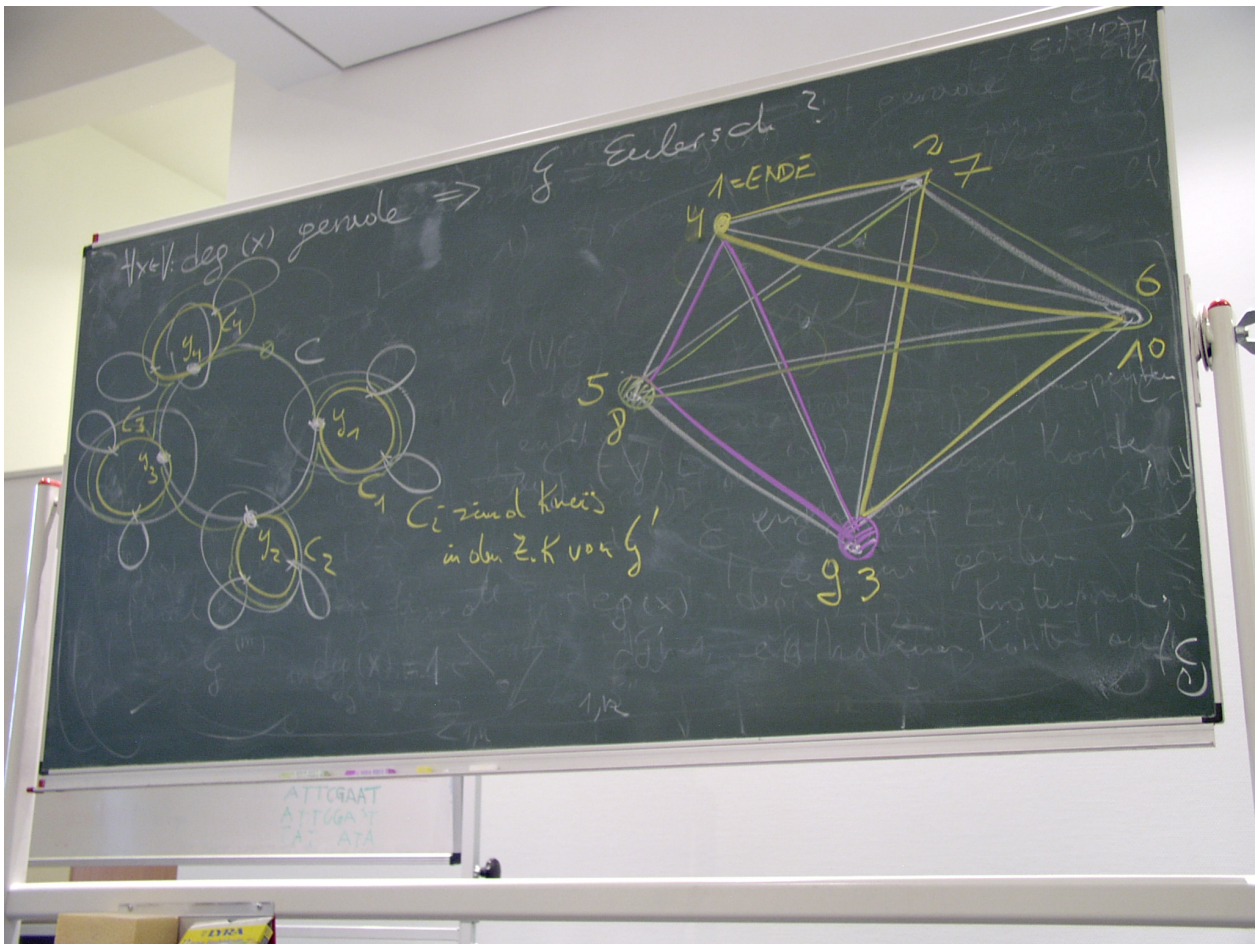
Frage

Sei

- $G = (V, E)$ ein ungerichteter Graph,
- $G.istZusammenhängend()$,
- $\forall (x \in V): \left(G.getGradFür(x) \in \left(\mathbb{N}_0^+ \cdot 2\right)\right)$.

Gilt dann?

- $G.istEulerscherGraph()$.



Betrachtung

1. Für jeden Knoten $x \in V$:
   1. finden wir einen geschlossenen Weg C.
   2. bilden wir einen Graphen $G_C = (V, E \setminus C.getEdges())$.
   3. Für jede Zusammenhangskomponente $G_{c,i} \in G_C.getZusammenhangskomponenten()$:
      1. $Y = G_{C,i}.getVertices() \cap C.getVertices()$
      2. Wir wissen, dass $|Y| > 0$, da die einzelne Zusammenhangskomponente nur deswegen „allein geworden ist“, weil Kanten an ihrem Knoten entfernt wurden.
      3. Für diese Zusammenhangskomponente finden wir einen eulerschen Weg (Rekursion)
   4. Wir verbinden den Weg C mit den Eulerschen Wegen der einzelnen Zusammenhangskomponenten geeignet zu einem Weg E. Dies ist ein Eulerscher Weg.

Schlussfolgerung:
Ganz offensichtlich lassen sich eulersche Wege finden für jeden solchen Graphen.

Bemerkung

Wenn der maximale Grad des Graphen k ist, dann brauchen wir nur eine Rekursionstiefe von $\frac{1}{2} \cdot k$.

Definition: Teilgraph-charakterisierender-Vektor

Sei

- $G = (V, E)$ ein Graph,

- $T \subseteq E$

Dann

- existiert eine Funktion $X(T): E \to \{0,1\}$ mit

$$\forall (e \in E): \left(X(T)(e) = \begin{cases} 1 & e \in T \\ 0 & e \notin T \end{cases} \right)$$

Diese Funktion heißt **charakterisierender Vektor** für die Teilmenge T der Kantenmenge von G.

Frage

Sei

- $G = (V, E)$ ein Graph,
- $C \in G.getWege()$,
- $C.istKreis()$,
- H ein Graph mit
  - $H.istTeilgraphVon(G)$
  - $H.getEdges() = C.getEdges()$

Was kann dann über X_H ausgesagt werden?

Es gilt dann:

- für jeden Knoten $x \in V$:
  - $$\sum_{\forall (e \in E)} \left(H_{x,e} \cdot X_{C.getEdges()}(e) \right) = \begin{cases} 2 & x \in C.getVertices() \\ 0 & sonst \end{cases}$$

Diese 2 da oben ist ein wenig unschön. Dann rechne ich lieber im Körper GF2.

Definition: GF2

Sei

- $\square \oplus \square : \{0,1\} \times \{0,1\} \to \{0,1\}$ mit $\forall (x_0 \in \{0,1\}): \forall (x_1 \in \{0,1\}): \left((x_0 \oplus x_1) = mod(x_0 + x_1, 2) \right)$,
- $\square \odot \square : \{0,1\} \times \{0,1\} \to \{0,1\}$ mit $\forall (x_0 \in \{0,1\}): \forall (x_1 \in \{0,1\}): \left((x_0 \odot x_1) = mod(x_0 \cdot x_1, 2) \right)$.

Dann heißt der Körper $GF2 = (\{0,1\}, \square \oplus \square, \square \odot \square)$ **GF2**.

Bemerkung

Im folgenden betrachten wir die Teilgraph charakterisierenden Vektoren als zugehörig zum Vektorraum über dem Körper GF2.

[...]

Sei,

- $A \subseteq E$,
- $B \subseteq E$,
- $$\forall (e \in E): \left(X_A(e) = \begin{cases} 1 & wenn\ e \in A \\ 0 & sonst \end{cases} \right)$$
- $$\forall (e \in E): \left(X_B(e) = \begin{cases} 1 & wenn\ e \in B \\ 0 & sonst \end{cases} \right)$$

- Was ist dann $\forall (e \in E): \left(X_D(e) := X_A(e) \oplus X_B(e) \right)$?
  - offensichtlich ist [...]

- Also ist $D=(A\cup B)\setminus(A\cap B)$ (also die symmetrische Differenz von A und B).
- Damit ist die Operation $\square\oplus\square$ die Operation XOR.

Definition: XOR_{set}

Nennen wir $\forall(A):\forall(B):(XOR_{set}(A,B)=(A\cup B)\setminus(A\cap B))$
[...]

Satz

Sei
- C_0 eine Menge,
- C_1 eine Menge,
- $(C_0\cap C_1)=\emptyset$.

Dann gilt:
- $XORset(C_0,C_1)=C_0\cup C_1$

Satz

- $G=(V,E)$ ein Graph,
- $C\in G.getWege()$,
- $C.istKreis()$,
- H ein Graph mit
  - $H.istTeilgraphVon(G)$
  - $H.getEdges()=C.getEdges()$
- $x\in V$

Dann gilt:
- $\sum_{\forall(e\in E)}(H_{x,e}\cdot X_H(e)=H.getDegreeFor(x))$

Satz

Daraus folgt:
- $G=(V,E)$ ein Graph,
- $C\in G.getWege()$,
- $C.istKreis()$,
- H ein Graph mit
  - $H.istTeilgraphVon(G)$
  - $H.getEdges()=C.getEdges()$
- $x\in V$

Dann gilt:
- $\left(\sum_{\forall(e\in E)}(H_{x,e}\oplus X_H(e)=0)\Leftrightarrow(H.getDegreeFor(x)\in(\mathbb{N}_0^+\cdot 2))\right)$

Definition: Kreisraum

[...]
Offensichtlich spannen alle möglichen Kreise mit ihren Teilgraph charakterisierenden Vektoren einen Vektorraum auf. Dieser Vektorraum heißt **Kreisraum**

$C(G)=G.getCycleSpace()$. Der Kreisraum ist damit ein Vektorraum über dem Körper GF2 (bezüglich eines Graphen).
[...]

Definition: Basis

Sei
- $G=(V,E)$ ein Graph,
- $C(G)=G.getCycleSpace()$,
- $M\subseteq G.getCycleSpace().getMengeAllerVektoren()$,
- Die mit den aufspannenden Vektoren ist jeder Vektor über Linearkombination erreichbar
  $\forall(U\in G.getCycleSpace().getMengeAllerVektoren()):\exists(\lambda_0):\exists(\lambda_1):\ldots:\exists(\lambda_{|M|-1}):$
  $\left(\bigodot_{k=0}^{|M|-1}(\lambda_k\odot M_i)=U\right)$
- M ist linear unabhängig:
  $\forall(\lambda_0):\forall(\lambda_1):\ldots:\forall(\lambda_{|M|-1}):\left(\bigodot_{i=0}^{|M|-1}(\lambda_i\cdot M_i)=\vec{0}\right)\Rightarrow(\lambda_0=\lambda_1=\ldots=\lambda_{|M|-1}=0)$

Satz

Sei
- $G=(V,E)$ ein Graph,
- $G.istBaum()$.
- $x\in V$
- $y\in(V\setminus\{x\})$

Dann gilt:
- es gibt genau einen Weg W mit
  - $W.getStartVertex()=x$ und
  - $W.getStartVertex()=y$

Definition: Fundamental-Kreis

Sei
- $G=(V,E)$ ein Graph,
- $T\in G.getSpannbäume()$.

Dann gibt es für jedes Paar von Knoten (x,y) dieses Spannbaums genau einen Weg W zwischen diesen beiden Knoten. Ist $|W.getEdges()|>1$, so erzeugt das Einfügen der Kante (x,y) in den Baum T einen Kreis $C=W.concat((x,y))$.
Dieser Kreis heißt **Fundamental-Kreis** für (x,y) bezüglich des Spannbaums T und wird bezeichnet als $cyc(T,e)=T.getCycleFor(e)$.

Definition: Basis

Sei
- $G=(V,E)$ ein Graph,
- $T\in G.getSpannbäume()$,
- $B_T=\{T.getCycleFor(e)\,|\,e\in(E\setminus T.getEdges())\}$.

Dann heißt B_T Basis des Kreisraums von G bezüglich des Spannbaums T.

Satz

Sei
- $G=(V,E)$ ein Graph,
- $T \in G.getSpannbäume()$.

Die Menge $B_T=\{T.getCycleFor(e) \mid e \in (E \setminus T.getEdges())\}$ ist eine Basis für den Kreisraum von G.

Beweis

- lineare Unabhängigkeit:
  Sei
  - $B_T=\{C_0, C_1, \ldots, C_{|B|-1}\}$
    Dann gilt:
  - Wenn wir die Gleichung $\lambda_0 \cdot X(C_0)+\lambda_1 \cdot X(C_1)+\ldots+\lambda_{|B|-1} \cdot X(C_{|B|-1})=0$ lösen wollen, dann klappt das nur, wenn $\lambda_0=\lambda_1=\ldots=\lambda_{|B|-1}=0$.
  - Das liegt daran, dass jeder dieser Basisvektoren $\forall(k):(X(C_k))$ genau dem Basisvektor des Spannbaums $X(T.getEdges())$ entspricht bis auf eine Kante e_k. Diese Kante e_k ist aber für jeden Kreis eine andere, wie aus der obigen Konstruktion ersichtlich. Somit sind alle $\forall(k):(X(C_k))$ voneinander linear unabhängig bezüglich der Kantenmenge $G.getEdges() \setminus T.getEdges()$.
- Erzeugenden.System
  - Betrachten wir $C \in G.getCycleSpace()$:
    - Wir können C wie folgt partitionieren: $C=(C \cap T.getEdges()) \cup (C \setminus T.getEdges())$.
    - Sei $K=C \setminus T.getEdges()$.
    - Sei $\forall(e \in (E \setminus T.getEdges())):(C_e=T.getCycleFor(e))$
      - Damit ist $\forall(e \in K):(C_e \in B_T)$
    - [...]
    - $K \subseteq C'=\bigoplus_{\forall(e \in K)} (C_e)$ [?]
    - $C'=K \cup (C \cap T)$ [?] und damit
    - $C'=K \cup (C \cap T)=C$ „und wir sind fertig“ [???]

Also:

Jeder Kreis C kann dargestellt werden als $C=\bigoplus_{\forall(e \in (C \setminus T))} (T.getCycleFor(e))$

Frage

Was ist $G.getCycleSpace().getDimension()$?
Wir werden später sehen, dass $G.getCycleSpace().getDimension()=$

$$+C.getEdges().length()$$
$$-C.getVertices().length()$$
$$+G.getZusammenhangskomponenten().length()$$

Theorem

Sei
- $G=(V,E)$ ein Graph,
- $G.istZusammenhängend()$.

- $T \in G.getSpannbäume()$,
- $B_T = G.getBaseForSpanningTree(T)$.

Dann gilt:

- $|B_T| = |E| - |V| + 1 = G.getCycleSpace().getDimension()$

Definition: cyclomatische Zahl

Sei

- $G = (V, E)$ ein Graph,
- $G.istZusammenhängend()$

Dann heißt die Zahl $G.getCyclomatischeZahl() = |E| - |V| + 1$ **cyclomatische Zahl** dieses Graphen.

[...]

[...]

Planare Graphen

[Dieses Kapitel ist wahrscheinlich eher wenig Prüfungsrelevant, da es nicht von Peter Stadler persönlich gehalten wurde.]

Definition: Weg in der Ebene

Sei
- $p:[0,1]\to\mathbb{R}^2$ eine Funktion,
- $p.istInjektiv()$,
- $p.istStetig()$.

Diese Funkion p heißt **Weg in der Ebene**.

Bemerkung

Aus der Stetigkeit folgt, dass der Weg zusammenhängend ist.
Aus der Injektivität der Funktion folgt die Kreuzungsfreiheit.

Definition: inneres

Sei
- $p:[0,1]\to\mathbb{R}^2$ ein Weg in der Ebene,

Dann ist $I: p\to Potenzmenge(\mathbb{R}^2)$ mit
- $\forall(p:[0,1]\to\mathbb{R}):(I(p)=p.getInterior()=\{p(x)\,|\,x\in]0,1[\})$

das **Innere** des Weges p.

Definition: planare Einbettung

Sei
- $G=(V,E)$ ein Graph,
- $coordinateOf: V\to\mathbb{R}^2$ eine Funktion,
- $wegVon: E\to\{p_e\,|\,(e\in E)\wedge(p_e:[0,1]\to\mathbb{R})\}$ eine Funktion,
  - $\forall(\{v,w\}\in E):\left(\left(p_{\{v,w\}}(0)=v\right)\wedge\left(p_{\{v,w\}}(1)=w\right)\right)\wedge\left(\left(p_{\{v,w\}}(0)=w\right)\wedge\left(p_{\{v,w\}}(1)=v\right)\right)$,
  - $\forall(e_0\in E):\forall\left(e_1\in\left(E\setminus\{e_0\}\right)\right):\left(p_{e_0}.getInterior()\cap p_{e_1}.getInterior()=\emptyset\right)$,
  - $\forall(e\in E):\left(wegVon(e).getInterior()\cap coordinateOf.getBild()=\emptyset\right)$

Dann heißt $(coordinateOf, wegVon)$ **planare Einbettung** von G.

Fragen

- Welche Graphen sind Planar?
- [...]

Satz von Euler

Sei
- $G=(V,E)$ ein Graph,
- $(coordinateOf, wegVon)$ eine Einbettung von G,
- $G.istZusammenhängend()$,

- k die Anzahl der Flächen der planaren Einbettung $(coordinateOf\ ,wegVon)$ des zusammenhängenden Graphen G.

Dann gilt:

- $|V|-|E|+k=2$

Beweis

Der Beweis geht über die vollständige Induktion über $|V|$.

Definition: Clique, vollständiger Graph

Sei

- $G=(V\ ,E)$ ein ungerichteter Graph,
- $E=\left\{\{v_0\ ,v_1\}\middle|(v_0\in V)\wedge\left(v_1\in\left(V\setminus\{v_0\}\right)\right)\right\}$ (Jeder Knoten ist mit jedem anderen Knoten verbunden.)

Dann heißt der Graph G **Clique** oder $|V|$-ter **vollständiger Graph**.

Beispiel

[...Foto...]

Definition: vollständig bipartiter Graph

Sei

- $G=(V\ ,E)$ ein ungerichteter Graph,
- $(\{V_0\ ,V_1\}).istPartitionierungVon(V)$
- $\forall(v_0\in V_0):\forall(v_1\in V_1):(\{v_0\ ,v_1\}\in E)$ (Jeder Knoten aus der einen Partition V_0 ist mit jedem Knoten aus der anderen Partition V_1 verbunden.)

Dann heißt G vollständig bipartiter Graph $K_{|V_0|,|V_1|}$

Beispiel

[...Foto...]

Definition: Weg

Sei

- $G=(V\ ,E)$ ein ungerichteter Graph,
- $l\in\mathbb{N}$,
- $p:([0,l[\cap\mathbb{N})\to V$,
- $p.istInjektiv()$,
- $\forall(i\in(]0,l[\cap\mathbb{N})):(\{p(i-1)\ ,p(i)\}\in E)$.

Dann heißt p Weg in diesem Graph G.

Definition: Inneres

Sei

- $G=(V\ ,E)$ ein ungerichteter Graph,
- $P:([0,l[\cap\mathbb{N})\to V$ ein Weg in diesem Graph G.

Dann heißt $I(p)=P.getInterior()=\left\{P(i)\middle|i\in(]0,l-1[\cap\mathbb{N})\right\}$ das **Innere** dieses Weges.

Definition: Unterteilung

Sei
- $G=(V,E)$ ein ungerichteter Graph,
- $G_2=(V_2,E_2)$,
- $\forall(e\in E):\exists(P_e\in G_2.\,getWege()):$
  - $\forall(e_2\in(E\setminus\{e\})):((P_e.\,getInterior()\cap P_{e_2}.\,getInterior())=\emptyset)$
  - $\bigcup_{\forall(e\in E)}(P_e.\,getUrbild())=V_2$

Dann heißt G_2 eine Unterteilung von G.

Definition: Teilgraph

- $G=(V,E)$ ein ungerichteter Graph,
- $G_2=(V_2,E_2)$,
- $V_2\subseteq V$,
- $E_2=E\cap\{\{v_0,v_1\}\mid(v_0\in V_2)\wedge(v_1\in V_2)\}$.

Dann heißt G_2 **Teilgraph** über V von G.

Satz: Satz von Kuratowski

Sei
- $G=(V,E)$ ein ungerichteter Graph,

G ist planar genau dann, wenn
- G hat keinen Teilgraph, der Unterteilung des K_5 ist, und
- G hat keinen Teilgraph, der Unterteilung des $K_{3,3}$ ist.

Beweis

[...Geduld...]

Definition:

Sei
- $G=(V,E)$ ein ungerichteter Graph,
- $e\in E$,
- $\{v,w\}=e$

Dann sei
- $G/e=G.kontrahiere(e)=(V_2,E_2)$ definiert durch:
  - $V_2=(V\setminus\{v,w\})\cup\{v_2\}$ ($v_2\notin V$) (v_2 ist ein neuer Knoten) und
  - $E_2=\left(E\cap\{\{v_0,v_1\}\mid(v_0\in V_2)\wedge(v_1\in V_2)\}\right)\cup\{\{v_2,w_2\}\mid(\{v,w_2\}\in E)\vee(\{w,w_2\}\in E)\}$

Beispiel

[...Foto...]

Definition: k-zusammenhängend

Sei
- $G=(V,E)$ ein ungerichteter Graph,
- $k\in\mathbb{N}$,

- $\forall (V_2 \subseteq V): \left(\left(|V_2| < k \right) \Rightarrow \left(G.getTeilgraphÜber(V_2).istZusammenhängend() \right) \right)$

Dann heißt der Graph G k-**zusammenhängend**.

Bemerkung

Von nun an sei G immer 3-zusammenhängend.

Lemma 1

Sei
- $G = (V, E)$ ein ungerichteter Graph,
- $e \in E$,
- G hat keinen Teilgraph, der Unterteilung des K_5 ist, und
- G hat keinen Teilgraph, der Unterteilung des $K_{3,3}$ ist.

Dann gilt für $G_2 = G.kontrahiere(e)$:
- G_2 hat keinen Teilgraph, der Unterteilung des K_5 ist, und
- G_2 hat keinen Teilgraph, der Unterteilung des $K_{3,3}$ ist.

Beweis

Siehe „Topological Graph Theory“ (urn:isbn:0-486-41741-7)

Lemma

Sei
- $G = (V, E)$ ein ungerichteter Graph,
- $G.ist3Zusammenhängend()$.

Dann liegt v_2 im Graphen $G.getTeilgraphÜber(V \setminus \{v_2\})$ innerhalb eines Zyklus.

Satz: Satz von Kuratowski, Abwandlung 1

Sei
- $G = (V, E)$ ein ungerichteter Graph,
- G ist 3-zusammenhängend,
- G hat keinen Teilgraph, der Unterteilung des K_5 ist, und
- G hat keinen Teilgraph, der Unterteilung des $K_{3,3}$ ist.

Dann gilt:
- $G.istPlanar()$.

Beweis

Wir betrachten den Graphen $G = (V, E)$.

Induktions-Verankerung

Der Satz gilt für alle Graphen mit maximal 4 Knoten.

Induktions-Hypothese

Der Satz gilt für alle Graphen mit weniger als $|V|$ Knoten.

Induktions-Schritt

- Gegeben sei

    - $G=(V,E)$ mit $|V|$ Knoten, sodass
    - G hat keinen Teilgraph, der Unterteilung des K_5 ist, und
    - G hat keinen Teilgraph, der Unterteilung des $K_{3,3}$ ist.
- Zu zeigen ist:
    - $G.istPlanar()$
- Wir wissen:
    - Für jedes $e \in E$ gilt:
        - $G.kontrahiere(e)$ hat keinen Teilgraph, der Unterteilung des K_5 ist, und
        - $G.kontrahiere(e)$ hat keinen Teilgraph, der Unterteilung des $K_{3,3}$ ist.
    - [...Foto...]
    - Für den Knoten v_2 gibt es einen Zyklus $C=(c_1, c_2, \ldots)$ mit $\forall(i):(c_i \in V)$.
- (1) Es gibt maximal 2 Knoten im Zyklus C, die jeweils sowohl mit v als auch mit w verbunden sind.
- (2) Die Knoten vom Typ u (also alle die, die mit u, aber nicht mit v verbunden sind) kommen an einem Stück auf C (also auf einer Teilliste von C) vor.
  Die Knoten vom Typ v (also alle die, die mit v, aber nicht mit u verbunden sind) kommen an einem Stück auf C (also auf einer Teilliste von C) vor.
  (Diese einzelnen Teillisten enthalten sogar keinen gemeinsamen Knoten)
    - Angenommen, diese Aussage gilt nicht.
        - [...dann?...]
- Wegen (1) und (2) funktioniert die Kontraktion wie abgebildet.

Bemerkung

K_5 ist zwar nicht auf der Ebene, aber auf dem Möbius-Band einbettbar.

Ergänzungen zum Satz von Kuratowski

- Wir hatten mehrere Formen von „Enthalten-Sein“ definiert
  - topologisches Enthalten-Sein
    - Durch Einfügen zusätzlicher Knoten entlang von Kanten
    - [Grafik]
  - Enthalten-Sein über Kontraktion
    - Durch Kontraktion

Satz: Erweiterter Satz von Kuratowski

Sei
- G ein ungerichteter Graph.

Dann sind folgende Aussagen äquivalent:
1. $G.istPlanar()$
2. $\neg K_5.istTopologischEnthaltenIn(G) \wedge \neg K_{3,3}.istTopologischEnthaltenIn(G)$
3. $\neg K_5.istKontraktivEnthaltenIn(G) \wedge \neg K_{3,3}.istKontraktivEnthaltenIn(G)$

Lineare Algebra auf Graphen in einer Nussschale

Sei
- $G=(V,E)$ ein ungerichteter Graph.

Dann erhalten wir
- einen GF2-Vektorraum.
  - Das ist der Vektorraum über dem Körper $(\{0,1\}, \square\oplus\square, \square\wedge\square)$, wobei
    - $\square\oplus\square$ die Boole'sche XOR-Operation,
    - $\square\wedge\square$ die Boole'sche AND-Operation ist.
- Knotenraum:
  - $V(G) := \{f \mid f: V \to GF2\}$
  - $(f \in V(G)) \Leftrightarrow (f \text{ entspricht einer Teilmenge von } V)$
  - $f+g$ entspricht der symmetrischen Differenz von f und g.
- Kantenraum
  - analog zum Knotenraum
  - $E(G) := \{f \mid f: E \to GF2\}$
  - Die Standard-Basis von $E(G)$ ist: $\{\{e_1\}, \{e_2\}, \ldots, \{e_n\}\}$, wobei $E = \{e_1, e_2, \ldots, e_n\}$
  - Sei
    - $f_0 = \left((v_0)_1, (v_0)_2, \ldots, (v_0)_m\right),\ \forall(i): \left((v_0)_i \in \{0,1\}^m\right)$
    - $f_1 = \left((v_1)_1, (v_1)_2, \ldots, (v_1)_m\right),\ \forall(i): \left((v_1)_i \in \{0,1\}^m\right)$

    Dann können wir das innere Produkt (Skalarprodukt) definieren als:
    - $\langle f_0, f_1 \rangle = \bigoplus_{\forall(i)} \left((v_0)_i \wedge (v_1)_i\right)$
    - $(f_0 \perp f_1) \Leftrightarrow \left(\langle f_0, f_1 \rangle = 0\right)$

- Sei
  - F ein Unter-Vektor-Raum von $E(G)$.
- Dann sei
  - $F^{\perp} := \{f \mid (f \subseteq E) \wedge \forall (g \in F.getMenge()) : (\langle f, g \rangle = 0)\}$ (Orthogonales Komplement)
    - Alle Vektoren, die Senkrecht zur Vektormenge von F stehen.
  - Unter-Räume von $E(G)$:
    - Zyklen-Raum $C(G)$
    - Cut-Raum $C^*(G)$

Definition: cut

Sei
- $G = (V, E)$ ein Graph,
- $E_0 \subseteq E$,
- $G_0 = (V, E \setminus E_0)$.

Wenn $\neg G_0.istZusammenhängend()$, dann heißt E_0 **cut** (**Schnitt**) von G.

Sei G ein zusammenhängender Graph.

Satz

$C(G)$ wird erzeugt von den durch G induzierten Zyklen

Satz

$C^*(G)$ wird erzeugt durch $\{E(v) \mid v \in V\}$, wobei $E(v) := \{\{x, y\} \mid (\{x, y\} \in E) \wedge (x = v)\}$

Satz

1. $C(G) = (C^*(G))^{\perp}$
2. $C^*(G) = (C(G))^{\perp}$

Satz

1. $dim(C^*(G)) = |V| - 1$
2. $dim(C(G)) = |E| - |V| + 1$

Definition: einfach

Sei
- $G = (V, E)$ ein Graph,
- $F \subseteq E(G)$,
- $\forall (e \in E) : (|\{f \mid (f \in F) \wedge (e \in f)\}| \leq 2)$.

Dann heißt die Menge F **einfach**. ($F.istEinfachBezüglich(G)$)

Definition: Zusammenhangszahl

Sei
- $G = (V, E)$ ein Graph,

Dann heißt die Anzahl der Knoten, die entfernt werden müssen, damit der Graph in mindestens 2 Zusammenhangskomponenten zerfällt Zusammenhangszahl des Graphen G.

$K(G) = G.getZusammenhangszahl()$

Beispiel

Satz: Satz von MacLane

Sei

- $G = (V, E)$ ein Graph,

$G.istPlanar() \Leftrightarrow G.getCycleSpace().getMenge().istEinfachBezüglich(G)$

Beweis

- Sei $G.getZusammenhangszahl(G) \leqslant 1$
  - Wir beweisen mittels vollständiger Induktion über die Anzahl der Knoten.
  - Induktionsverankerung:
    - Sei $|V| = 2$
    - Dann gilt:
      - der Satz ganz offensichtlich.
  - Induktionsschritt:
    - Voraussetzung:
      - Der Satz gilt für jeden Graphen mit weniger als $|V|$ Knoten.
    - Dann gilt:
      - Wenn
        - $G = G_1.vereinigeMit(G_2) \quad G_1.schneideMit(G_2).getKnoten().getLength() \leqslant 1$,
      - dann:
        - Behauptung:
          - $G.istPlanar() \Leftrightarrow (G_1.istPlanar() \wedge G_2.istPlanar())$
        - Beweis:
          - $G.istPlanar() \Leftarrow (G_1.istPlanar() \wedge G_2.istPlanar())$ klar, da planare Graphen nur über einen Knoten miteinander verbunden wieder einen planaren Graphen bilden.
          - $G.istPlanar() \Rightarrow (G_1.istPlanar() \wedge G_2.istPlanar())$ nicht ganz so klar, steht im Buch von Diestel.
        - Behauptung:
          - $G.getCycleSpace()$ hat eine einfache Zyklen-Basis genau dann, wenn wenn
            - $G_1.getCycleSpace()$ hat eine einfache Zyklen-Basis und
            - $G_2.getCydeSpace()$ hat eine einfache Zyklen-Basis.
        - Beweis:
          - Rückrichtung $\Leftarrow$:
            - Es gilt:
              - $G.getCycleSpace() = G_1.getCycleSpace() + G_2.getCycleSpace()$
              - $(G_1.getCycleSpace() \cap G_2.getCycleSpace()) = \emptyset$
              - B_1 sei einfache Basis für $G_1.getCycleSpace()$
              - B_2 sei einfache Basis für $G_2.getCycleSpace()$
              - $B = B_1 \cup B_2$ ist dann Basis für $G.getCycleSpace()$
          - Hinrichtung $\Rightarrow$:

- Sei B eine einfache Basis für $G.getCycleSpace()$.
- Jede $b \in B$ enthält entweder ausschließlich Kanten von G_1 oder ausschließlich Kanten von G_2.
- B_1 ist einfache Basis für $G_1.getCycleSpace()$,
- B_2 ist einfache Basis für $G_2.getCycleSpace()$.
- $B = B_1 \cup B_2$
- [Grafik]

- Sei $G.getZusammenhangszahl(G) > 1$.
  - $G.istPlanar() \Rightarrow G.getCycleSpace().getMenge().istEinfachBezüglich(G)$:
    - Sei
      - $G.istPlanar()$
    - Wir wählen eine planare Einbettung von G.
    - Lemma:
      - In planarem eingebettetem Graphen G mit $G.getZusammenhangszahl(G) > 1$ wird jede Fläche von einem Zyklus begrenzt.
    - B sei die Menge aller Kantemnegen, die eine Fläche in der Einbettung begrenzen.
    - Behauptung:
      - B ist einfache Zyklenbasis
    - Beweis:
      - B ist einfach, weil jede Kante an maximal 2 Flächen angrenzt.
      - [...Erklärung an der Tafel...]
    - Also erzeugt B $G.getCycleSpace()$.
  - $G.istPlanar() \Leftarrow G.getCycleSpace().getMenge().istEinfachBezüglich(G)$:
    - Sei
      - $B = \{C_1, C_2, \ldots, C_k\}$ eine einfache Basis für $G.getCycleSpace()$.
    - Wähle $e \in E$, $G_2 = (V, E \setminus \{e\})$.
    - Behauptung:
      - $G_2.getCydeSpace()$ hat eine einfache Basis.
    - Beweis:
      - Fall: e liegt in genau einem C_i.
        - Ohne Beschränkung der Allgemeinheit liegt $e \in C_1$.
        - Also $\{C_2, C_3, \ldots, C_k\}$ ist Basis für $G_2.getCydeSpace()$.
      - Fall: e liegt in C_1 und C_2 (ohne Beschränkung der Allgemeinheit)
        - $B_2 = \{C_1 + C_2, C_3, C_3, \ldots, C_j\}$ ist Zyklus-Basis für $G_2.getCydeSpace()$
        - [Grafik]
    - Es verbleibt dann noch der Fall zu zeigen, dass weder K_5 noch $K_{3,3}$ eine einfache Zyklen-Basis hat.
      - Angenommen, K_5 hat eine einfache Zyklen-Basis. Dann gilt:
        - $K_5.getCycleSpace().getDimension() = 6$,
        - $B = \{C_1, C_2, C_3, C_4, C_5, C_6\}$.
        - Sei $C_0 = C_1 + C_2 + C_3 + C_4 + C_5 + C_6$.
        - Wir wissen, dass in jedem C_i mindestens 3 Kanten sein müssen, da es schließlich Zyklen sind. Also $\forall (i \in \{1,2,3,4,5,6\}) : (|C_i| \geq 3)$.
        - Also:

$$
\begin{aligned}
18 &\leqslant \sum_{\forall (i \in [1,2,3,4,5,6])} : \left(\left|C_i\right|\right) \\
&\leqslant 2 \cdot K_5 . \, getEdges() . \, getLength() - \left|C_0\right| \\
&\leqslant 2 \cdot 10 - \left|C_0\right| \\
&\leqslant 20 - \left|C_0\right| \\
\left|C_0\right| &\leqslant 20 - 18 \\
\left|C_0\right| &\leqslant 2
\end{aligned}
$$

- Aber $\left|C_0\right| \geqslant 3$, da C_0 ein Zyklus ist.

Zufallsgraphen

Variante

- Man wähle 2 Parameter
  - Anzahl der Knoten N
  - Anzahl der Kanten m
- Dann erzeuge man einfach die Knoten und verbinde wahllos so lange Knoten, bis genau m kanten existieren.

Variante: Binomial-Modell

- Man wähle 2 Parameter
  - Anzahl der Knoten N
  - eine Kanten-Existenz-Wahrscheinlichkeit $p \in [0,1]$
- Dann betrachte man jede mögliche Kante und erwürfele mit Wahrscheinlichkeit p, ob diese Kante existieren soll.
- Der Erwartungswert ist $m_{erwartet} = p \cdot \frac{1}{2} \cdot N \cdot (N-1)$.

Eigenschaften

- Teilgraphen:
  - Betrachten wir einen Teilgraphen G des Zufallsgraphen
    - mit l Knoten
    - mit k Kanten
  - Sei $n_G(l,k)$ die Anzahl solcher Teilgraphen. Dann gilt für den Erwartungswert dieser Anzahl:
  - $$E(n_G(l,k)) \approx \frac{N!}{(N-l)!} \cdot p^k \cdot (1-p)^{\frac{l \cdot (l-1)}{2} - k}$$
    $$\approx N^l \cdot p^k$$

Zusammenhang Zufallsgraphen-Parameter mit Cluster-Bildung

Ein Cluster ist eine Zusammenhangskomponente im Zufallsgraphen.
Sei

- $P(s,m)$ die Wahrscheinlichkeit, dass von einem zufällig gewählten Knoten ausgehend dieser Knoten sich in einem Cluster mit gerade s Knoten befindet, wobei m die Anzahl der Kanten des Zufallsgraphen ist.

$$P(s,0) = \begin{cases} 1 & wenn\ s=1 \\ 0 & sonst \end{cases}$$

$$P(s,m+1) - P(s,m) = -2 \cdot \frac{s}{N} \cdot P(s,m) + \frac{s}{N} \cdot \sum_{r=1}^{s-1} \left(P(r) \cdot P(s-r) \right)$$

$$\mu_n(m) = \sum_{s=1}^{\infty} \left(s^n \cdot P(s,m) \right)$$

$$\mu_0(m) = \sum_{s=1}^{\infty} \left(s^0 \cdot P(s,m) \right) = 1$$

$$\mu_1(m)=\sum_{s=1}^{\infty}\left(s^1\cdot P(s,m)\right)=E(s)$$

$$\mu_2(m)=\sum_{s=1}^{\infty}\left(s^2\cdot P(s,m)\right)=V(s)$$ [wurde zumindest mündlich so angegeben]

$$\mu_n(m+1)-\mu_n(m)=\sum_{s=1}^{\infty}\left(s^n\cdot\left(P(s,m+1)-P(s,m)\right)\right)$$

$$=\sum_{s=1}^{\infty}\left(s^n\cdot\left(-2\cdot\frac{s}{N}\cdot P(s,m)+\frac{s}{N}\cdot\sum_{r=1}^{s-1}\left(P(r,m)\cdot P(s-r,m)\right)\right)\right)$$

$$=-\frac{2}{N}\cdot\mu_{n+1}(m)+\frac{1}{N}\cdot\sum_{s=1}^{\infty}\left(\sum_{r=1}^{s-1}\left(s^{n+1}\cdot P(r,m)\cdot P(s-r,m)\right)\right)$$

$$=-\frac{2}{N}\cdot\mu_{n+1}(m)+\frac{1}{N}\cdot\sum_{k=1}^{n+1}\left(\binom{n+1}{k}\cdot\mu_k(m)\cdot\mu_{u+1-k}(m)\right)$$

Für $n=1$ gilt: $\mu_1(m+1)-\mu_1(m)=\frac{2}{N}\cdot\left(\mu_1(m)\right)^2$.

[...]

Definition: Länge einer Kreisbasis

Sei B eine Kreis-Basis eines Graphen.
Dann ist $l(B)=B.getLänge()=\sum_{\forall(C\in B)}(|C|)$.

$u(B):=max(\{|C\|C\in B\})$

Definition: minimale Kreisbasis

B heißt **minimale Kreisbasis**, wenn $l(B)$ minimal ist.

Definition: Matroid

„Der Begriff des Matroiden wurde geschaffen, um über lineare Unabhängigkeit reden zu können, ohne über Vektoren sprechen zu müssen."

Sei
- X eine endliche Menge
- $J\subseteq Potenzmenge(X)$.

J heißt Matroid genau dann, wenn:
1. $\emptyset\in J$
2. $\forall(I\subseteq X):((I\in J)\Leftrightarrow I.heißtLinearUnabhängig())$
3. $\forall(I_0\subseteq X):\forall(I_1\subseteq I_0):((I_0\in J)\Rightarrow(I_1\in J))$
4. $(I_p\in J)\wedge(I_{p+1})$, wobei $(|I_p|=p)\wedge(|I_{p+1}|=p+1)$
5. $\exists(x\in I_{p+1}):((I_p\cup\{x\})\in J)$

Die Basis von (X,J), ist eine maximale unabhängige Menge.

Satz

Alle Basen haben die selbe Anzahl von Elementen (Rang eines Matroiden).

Beweis

Angenommen, B_0 und B_1 seien Basen, wobei $|B_0|<|B_1|$. Dann $\exists(A\subseteq B_1):(|A|=|B|+1)$.
Aus dem Axiom 5 folgt: $\exists(x\in A):((B\cup\{x\})\in J)$. Also ist B nicht maximal. Also ist B keine Basis.

Gegeben sei (X,J). Wir definieren eine Gewichtsfunktion $w:X\to\mathbb{R}_0^+$ (die soll die Länge der Kreise liefern).
Gesucht ist eine Basis mit $\omega(B):=\sum_{\forall(x\in B)}(w(x))$ minimal.
- Gegeben sei eine Basis B_0 und ein Element x.
- $\exists(y\in B):\left(\left(\underbrace{(B_0\setminus\{y\})\cup\{x\}}_{=B_1}\right).istLinearUnabhängig()\right)$

- $\omega(B_1)=\omega(B_0)+w(x)-w(y)$
- Daraus folgt:
  - Ist M eine minimale Basis, dann $\exists(x_{min,0}\in M):\left(w(x_{min,0})=\underset{\forall(y\in X)}{min}(w(y))\right)$. Da $x_{min,0}$ vom Gewicht her minimal ist, wollen wir, dass dieses Element Element unserer Basis ist. Wir entfernen also ein Element aus B_0 und fügen x hinzu, und damit erhalten wir B_1. Wir wissen, dass $\omega(B_1)\leqslant\omega(B_0)$.
  - Jetzt wollen wir wieder ein Element aus B_1 substituieren, sodass B_2 entsteht, aber am liebsten $\omega(B_2)<\omega(B_1)$. Dazu suchen wir uns das kleinste Element (bezüglich der Gewichtsfunktion), was als Kandidat gelten kann, welches ein Element aus B_1 ersetzen kann. Das zu ersetzende Element aus B_1 darf aber nicht $x_{min,0}$ sein. Wir wählen also das kleinste $x_{min,1}$, wofür gilt : $\{x_{min,0},x_{min,1}\}\in J$.
- Daraus folgt ein **Greedy-Algorithmus**:
  - Eingegeben sei:
    - Gewichtsfunktion $w:X\to\mathbb{R}_0^+$
    - X aufsteigend entsprechen w sortiert,
  - $B=\emptyset$
  - $\forall(x\in X):$
    - $if\left(\left(B\cup\{x\}\right)\in J\right):$ /* Erstbestes passendes x wird genommen -> gierig */
      - $B=B\cup\{x\}$

Satz

Der Kreis C ist in einer minimalen Kreisbasis M enthalten, wenn $\neg\exists(C_1,C_2,\ldots,C_l):\left(\forall(i):\left(|C_i|<|C|\right)\wedge\left(C=\bigoplus_{i=1}^{l}(C_i)\right)\right)$, also wenn es keine kleineren Kreise gibt, die zusammen XOR-Verknüpft den Kreis C ergeben.

Beweis

Offensichtlich kann ich die Menge $\{C_1,C_2,\ldots,C_l\}$ so wählen, dass sie selbst linear unabhängig ist. Wenn sie's nicht sein würde, dann kann man aus der Menge einfach ein paar Kreise entfernen, bis sie's ist.
Sei $C\in M$, M eine minimale Kreisbasis und $\exists(C_1,C_2,\ldots,C_l):\left(\forall(i):\left(|C_i|<|C|\right)\wedge\left(C=\bigoplus_{i=1}^{l}(C_i)\right)\right)$. Dann lässt sich ja C ganz offensichtlich als Linearkombination von $C_1,C_2,\ldots,C_l$ darstellen, aber $\forall(i):\left(|C|>|C_i|\right)$ (also C ist größer als alle C_i). Das bedeutet, dass M nicht minimal ist, da $M\setminus\{C\}$ minimaler wäre. Also ist dies ein Widerspruch.

Sei
- G ein ungerichteter Graph.
- $e=(u,v)$ eine Kante
- x ein Knoten.

- $Q_{u,x}$ ein nicht kürzester Weg von x nach u
- $Q_{v,x}$ ein nicht kürzester Weg von x nach v
- $\hat{P}_{u,x}$ ein kürzester Weg von x nach u
- $\hat{P}_{v,x}$ ein kürzester Weg von x nach v

$$C=\left(Q_{u,x}\oplus\{e\}\oplus\hat{P}_{v,x}\right)\oplus\left(\hat{P}_{u,x}\oplus\{e\}\oplus Q_{v,x}\right)\oplus\left(\hat{P}_{u,x}\oplus\{e\}\oplus\hat{P}_{v,x}\right)=\left(Q_{u,x}\oplus\{e\}\oplus Q_{v,x}\right)$$

Blöderweise sind die kürzesten Wege nicht eindeutig. Wir können diese aber vereindeutlichen, indem jeder Weg auf seine Länge einfach eine eindeutige ganz kleine Zahl hinzuaddiert bekommt.

[...verpasst...]

Algorithmus für minimale Kreisbasen nach Horton

(1987)
„Unglücklicherweise funktioniert er nicht."

- $L=\emptyset$
- $\forall(e\in G.getEdges()):$
  - $\forall(x\in V):$
    - $C_{e,x}=\hat{P}_{u,x}\oplus\hat{P}_{x,v}\oplus\{e\}$
    - $L=L\cup\{C_{e,x}\}$
- $sort(L,w)$

Definition: Vereinigung der minimalen Kreisbasen

Sei

- G ein ungerichteter Graph.
- $Basen$ die Menge aller Kreisbasen dieses Graphen.

Dann heißt $R=\bigcup_{\forall(B\in Basen)}(B)$ **Vereinigung der minimalen Kreisbasen** oder **Menge der relevanten Kreise**.

Es gilt: $\forall(C\in G.getKreise()):\left((C\in R)\Leftrightarrow\neg\exists(C_1,C_2,\ldots,C_l):\left(C=\bigoplus_{i=1}^{l}(C_i)\right)\right)$

Für manche Graphen steigt $|R|$ exponentiell mit der Anzahl der Knoten.

Satz

Sei

- $C_e\in R$

Dann gilt:

-

Beweis

Angenommen M ist

- $M_1=\{C|(C\in M)\wedge(e\in C)\}$

- $M_2 = \{C \mid (C \in M) \wedge (e \notin C)\}$

$M' = \left(M_1 \setminus \underbrace{\{C\}}_{\in M}\right) \cup C_e \quad |M'| \leqslant |M|$

$S(e)$ sei die Menge der kürzesten Kreise durch die Kante e

$S_{super} = \bigcup_{\forall(e)} (S(e)) \subseteq \mathbb{R}$

Anwendung: Smallest set of shortest rings

In der Chemie lassen sich Strukturformeln schlecht effizient suchen. Um dies zu verbessern kann man eine Strukturformel nach der Anzahl der Kreise der Länge k in einer minimalen Kreisbasis charakterisieren.

Anwendung: metabolische Netzwerke

Für die Inzidenz-Matrix $H_{x,e}$ gilt: $\sum_{\forall(e)} (H_{x,e} \cdot C_e) = 0$

Chemische Reaktionen haben im Allgemeinen diese Form:

- $n_1 \cdot X_1 + n_2 \cdot X_2 + n_3 \cdot X_3 + \ldots \rightarrow m_1 \cdot Y_1 + m_2 \cdot Y_2 + m_3 \cdot Y_3$
- $2 \cdot H + 1 \cdot O \rightarrow 1 \cdot H_2O$
- $2 \cdot H_2 + O_2 \rightarrow 2 H_2O$
- $\rho : 2 \cdot H_2O - 2 \cdot H_2 - O_2$ (hierbei ist die „ 2 “ vor „ H_2O “ der so genannte „stöchiometrische Koeffizient“)
- $\dot{c}_x = \sum_{\forall(\rho)} \left(I_\rho(\vec{c}) \cdot S_{x,\rho}\right)$
- $\dot{c} = S \cdot J$, wobei J ein Flussvektor ist.
- In einem stationären Metabolismus gilt: $0 = S \cdot J$.

Elementare Flussmoden

1. $S \cdot J = \emptyset$
2. $\forall(\rho) : (J_\rho \geqslant 0)$
3. $\neg\exists(Flussmode\ J_1) : \exists(Flussmode\ J_2) : \exists(\lambda_1 > 0) : \exists(\lambda_2 > 0) : (J = \lambda_1 \cdot J_1 + \lambda_2 \cdot J_2)$

   Im Fall von Graphen: Elementare, gerichtete Kreise.

Flussmoden, die elementare Flussmoden sind, entsprechen im wesentlichen den „metabolic pathways“.

Produkte von Graphen

Definition: mittlerer Knotengrad

Sei

- $G = (V, E)$ ein Graph.

Dann heißt $G.getAverageVertexDegree() = \overline{d} = \frac{2 \cdot |E|}{|V|}$ **mittlerer Knotengrad** des Graphen G

.

Definition: kartesisches Produkt

Sei
- $G_0=(V_0, E_0)$ ein Graph,
- $G_1=(V_1, E_1)$ ein Graph.

Dann sei:
- $V=V_0\times V_1$
- $E=\left\{((x_0, x_1),(y_0, y_1))\middle|((x_0=y_0)\wedge((x_1, y_1)\in E_1))\vee((x_1=y_1)\wedge((x_0, y_0)\in E_0))\right\}$

Eigenschaften

- $|V|=|V_0|\cdot|V_1|$
- $|E|=|E_0|\cdot|V_1|+|E_1|\cdot|V_0|$
- mittlerer Knotengrad:
  - $\overline{d}_0=\frac{2\cdot|E_0|}{|V_0|}$
  - $\overline{d}_1=\frac{2\cdot|E_1|}{|V_1|}$

$$\begin{aligned}\overline{d}&=\frac{2\cdot|E|}{|V|}\\&=\frac{2\cdot(|E_0|\cdot|V_1|+|E_1|\cdot|V_0|)}{|V_0|\cdot|V_1|}\\&=2\cdot\frac{|E_0|\cdot|V_1|+|E_1|\cdot|V_0|}{|V_0|\cdot|V_1|}\end{aligned}$$

- $$\begin{aligned}&=2\cdot\left(\frac{|E_0|\cdot|V_1|}{|V_0|\cdot|V_1|}+\frac{|E_1|\cdot|V_0|}{|V_0|\cdot|V_1|}\right)\\&=2\cdot\left(\frac{|E_0|}{|V_0|}+\frac{|E_1|}{|V_1|}\right)\\&=\frac{2\cdot|E_0|}{|V_0|}+\frac{2\cdot|E_1|}{|V_1|}\\&=\overline{d}_0+\overline{d}_1\end{aligned}$$

Satz

Für die Abstände im Graphen gilt: $\forall(x\in V_0):\forall(y\in V_1):(d((x, y))=d(x)+d(y))$.
Behauptung: $d((x_1, y_1),(x_2, y_2))=d(x_1, x_2)+d(x_1, y_2)$

gehalten von Peter Stadler

Sequenzdesign, Graphen und alles, was damit zu tun hat

RNA-Sekundärstruktur

Was ist eine RNA-Sekundärstruktur aus Graphen-Sicht?
1. G ist ein aufspannender Pfad (Hamiltonscher Kreis, wenn wir Anfang und Ende uns als verbunden denken)
2. G ist außerplanar
3. $\forall (x \in G.getKnoten()) : (deg(x) \leqslant 1)$: der Graph ist sub-kubisch
4. G ist dreiecksfrei

Beispiel

[Foto]

Definition: außerplanar

Ein Graph heißt **außerplanar** genau dann, dass es eine plannare Einbettung des Graphen gibt, sodass alle Knoten außen sind und alle Kanten zwischen den äußeren Knoten am Rand oder innen (aber nicht außen) verlaufen.

Satz

Ein Graph ist genau dann außerplanar, wenn er keinen Teilgraph hat, der ein „geplätteter Tetraeder“ ist.

Definition: dreiecksfrei

Ein Graph heißt dreiecksfrei genau dann, wenn er keinen Teilgraph hat, welcher ein Dreieck ist.

Definition: 2-zusammenhängend

Ein Graph heißt 2-zusammenhängend genau dann, wenn es für jedes Paar von Knoten einen elementaren Kreis (also einen, der sich nicht in kleinere Kreise aufteilen lässt) gibt, der diese beiden Knoten enthält.

Satz

Sei
- G
- $G.istAußerplanar()$ und
- $G.ist2zusammenhängend()$

Dann gilt:
- Die minimale Kreisbasis des Graphen ist eindeutig und gleich der „planaren“ Basis.

Beispiel

[Grafik]

Definition: Färbung

Sei
- $G=(V,E)$ ein Graph.
- M eine endliche Menge.
- $c: V \to M$

Dann heißt die Abbildung c **Färbung**.

Definition: Farbe

Sei zudem $a \in A$. Dann heißt a Farbe der Färbung c des Graphen G.

Beispiel

Eine RNA-Sekundärstruktur, wobei jeder Knoten eine Basen A,U,G,C ist.
$M=(A,U,G,C)$
[Grafik]
„Wir nehmen den linken Graph G_0 und reißen ihm einfach das Backbone aus. Dann erhalten wir den rechten Graphen G_1."
Für den Graphen G_1 gilt: $\forall(x \in G_1.getKnoten(): (G_1.getDegreeFor(x) \leq 1)$.
Wenn wir nun die einzelnen Knoten numerieren, und alle Basenpaare in die Menge B packt, dann bildet $(V, \square < \square, B)$ eine Art Ordnung.

$$((\{i,j\} \in B) \wedge (\{k,l\} \in B)) \Rightarrow ((i<j<k<l) \vee (i<k<l<j) \vee ...Rollenvertauschung\ von\ i\ mit\ j,\ k\ mit\ l...)$$

[...viel erzählt, wenig verstanden...]

Zwei Basenpaare sind genau dann nicht vergleichbar (die „Dinger" (Bögen der Basenpaare im Beispiel) überkreuzen sich), wenn $(B, \square <_O \square)$ keine Halbordnung bildet.

Definition: co-Vergleichbarkeitsgraph

Sei
- G ein Graph.
- B die Menge der Basenpaare im B
- $\square <_O \square$ eine Halbordnung über diesen Basenpaaren.
- $(B, \square <_O \square)$ die entsprechende algebraische Struktur,

Dann ist $I(G)=G.getCoVergleichbarkeitsGraph()=(B,U)$, wobei $(\{\{i,j\},\{k,l\}\} \in U) \Leftrightarrow (\{i,j\}\ und\ \{k,l\}\ nicht\ vergleichbar\ in\ \square <_O \square)$

Satz

Der co-Vergleichbarkeitsgraph von G ist genau dann bipartit, wenn G planar „wie unten" (also als Summe von außerplanaren Graphen) zwei zeichenbar ist.

Satz

Sei G ein Graph. Der Graph G ist bipartit genau dann, wenn alle seine Kreise gerade Länge haben.

Bemerkung

Warum? Naja, man geht einfach die Kreise entlang und markiert den ersten Knoten rot, den zweiten Knoten grün, den dritten Knoten rot, usw...

Definition: Buchdicke

Wenn ein Graph nicht vollständig außerplanar ist, braucht man vielleicht mehrere Ebenen, in den man ihn einbetten kann, wobei alle Ebenen sich in einer Gerade schneiden. Diese Gerade enthält einen Teilgraphen, der alle Kanten enthält. Die einzelnen Ebenen sind dann eine planare Einbettung eines Teilgraphen, und zwar eine außerplanare. Diese vielen Ebenen kann man sich als Buch vorstellen. Die **Buchdicke** eines Graphen ist dann die Anzahl der Seiten, die ein solches Buch mindestens haben muss, damit in jeder Ebene der jeweilige Teilgraph außerplanar ist.

Definition: erlaubte Färbung

Sei

- $(V, ❑<❑, B)$ gegeben.

Dann ist für ein Basenpaar (x, y) nur folgende Färbung erlaubt:

- $(c(x), c(y)) \in \{(G, C), (C, G), (A, U), (U, A), (G, U), (U, G)\}$

Theorem

(von Christian Reidys, St.)
Für alle Paare $(V, ❑<❑, B')$, $(V, ❑<❑, B'')$ existiert eine gemeinsame erlaubte Färbung.

Beweis

Betrachten wir $(V, ❑<❑, B' \cup B'')$ und den dazugehörigen Graphen $G = (V, B' \cup B'')$. Da gilt: $\forall (x \in V) : (G.getDegreeFor(x) \leqslant 2)$.

- Der Graph ist dann eine disjunkte Vereinigung von
  - isolierten Knoten $G.getDegreeFor(x) = 0$
    - diese lassen sich beliebig färben
  - Pfaden
    - diese lassen sich legal färben
  - Kreisen
    - Die Kanten entlang der Kreise alternieren in der Hinsicht, dass sie aus unterschiedlichen ursprünglichen Graphen stammen.
    - Weil die Kanten alternieren, sind die Kreise immer gerader Länge.

gehalten von Axel Mosig

Sei $w = w_1 \cdot w_2 \cdot \ldots \cdot w_l$ eine gemeinsame Teilsequenz (Teilfolge) von sowohl x als auch y. Eine Folge w ist **Teilsequenz** von x dann, wenn man Buchstaben beliebig aus x so entfernen kann, dass w entsteht. Eine Elementare Teilsequenz von sowohl x als auch y (also eine Teilsequenz der Länge 1), kann durch ein Paar (i, j) dargestellt werden, wobei i der Index des Zeichens in x und j der Index des Zeichens in y ist. Diese einzelnen Paare bilden nun eine partielle Ordnung [Grafik] bezüglich ihrer Reihenfolge in x bzw. y.

Eine allgemeine Teilsequenz kann also durch eine Folge von Paaren $((i_1, j_1), (i_2, j_2), \ldots, (i_l, j_l))$ dargestellt werden, wobei $(i_1, j_1) \prec (i_2, j_2) \prec \ldots \prec (i_l, j_l)$ gilt.
Eine Aufsteigende Folge in einer partiellen Ordnung nennt man **Kette**.
Mengen von paarweise nicht vergleichbaren Elementen nennt man **Antiketten**.

Satz

Die maximale Länge p einer Kette $C \subseteq M$ ist gleich der maimalen Kardinalität t einer Menge von paarweise disjunkten Antiketten $A_1, A_2, \ldots, A_t$ welche M partitionieren, das heißt: $M = A_1 \dot\cup A_2 \dot\cup \ldots \dot\cup A_t$.

Beweis

- Zu jedem $c \in M$ definieren wir $rank(c)$ als die Länge der längsten Kette, die in c endet.
- Sei
  - $c \in M$,
  - $c' \in M$
  - $c' \neq c$,
  - $rank(c) = rank(c')$.
- Dann gilt:
  - c und c' sind miteinander nicht vergleichbar.
- Wir definieren nun $\forall \left(i \in \left([1, p] \cap \mathbb{N} \right) \right) : \left(A_i^{min} := \left\{ c \middle| (c \in M) \wedge (rank(c) = i) \right\} \right)$.
- Dann gilt: $M = A_1^{min} \dot\cup A_2^{min} \dot\cup \ldots \dot\cup A_p^{min}$.
- Bleibt zu zeigen, dass $p \geq t$:
  - Zu jedem $a \in A_i^{min}$ mit $i > 1$ existiert ein $a' \in A_{i-1}^{min}$ mit $a' \prec a$. Man konstruiere also aus $A_1^{min}, A_2^{min}, \ldots, A_t^{min}$ eine Kette der Länge t.

Satz von Dilworth

Es sei M partielle geordnete endliche Menge M. Die minimale Anzahl disjunkter Ketten, die M partitiionieren, ist gleich der Kardinalität der größtem Antikette.

Beweis

R. Stanley: „Enumerative Combinatoric“

Berechnung von maximalen Ketten und Antiketten

Gegeben sei $(M, \square \preccurlyeq \square)$.

Man konstruiere den gerichteten Vergleichbarkeitsgraphen $G_{(M,\square\preccurlyeq\square)}=(V,E)$ mit:

- $V=M$,
- $E=\{(v,w)\,|\,v\prec w\}$

Ketten in M entsprechen Pfaden in $G_{(M,\square\preccurlyeq\square)}$.

Das heißt: Berechnen einer Kette maximale Länge entspricht dem Berechnen eines längsten Weges in $G_{(M,\square\preccurlyeq\square)}$. Dies geht gut mit dynamischer Programmierung:

- Für den dynamischen-Programmierungs-Algorithmus benötigen wir 2 Definitionen:
  - $\forall(W\subseteq V):\left(N(W):=\{v\,|\,(v\in V)\wedge\exists(w\in W):((w,v)\in E)\}\right)$
  - $\forall(v\in V):\left(P(v):=\{w\,|\,(w\in V)\wedge((w,v)\in E)\}\right)$
- Algorithmus $MaxChain(M,\square\preccurlyeq\square)$
  - Berechne $G_{(M,\square\preccurlyeq\square)}$.
  - $W:=$*Menge aller minimalen Elemente*
  - *while* $(W\neq M)$:
    - Wähle $v\in(N(W)\cap(V\setminus W))$
    - $c(v):=max(\{c(w)\,|\,w\in P(v)\})+1$
    - $W:=W\cup\{v\}$
- Laufzeit:
  - $O(|V|\cdot\delta+|E|)$

Bemerkung

Der längste Pfad, den wir berechnet haben, liefert

- eine maximale Clique im Vergleichbarkeitsgraphen.
- eine Färbung im Vergleichbarkeitsgraph.

Daraus folgt:

- Im Vergleichbarkeitsgraphen ist die Färbbarkeitszahl gleich der Cliquen-Zahl.

Dies führt auf perfekte Graphen.

Siehe: M.C.Golumbic: „Algorithmic Graph Theory and Perfect Graphs“

Datum: 14.07.2004 (11:15 Uhr)

[nicht anwesend gewesen, möglicherweise aber Uta Schulze, Madlen Hartmann, Marcel Kretzschmann, Armin Vollmer]

Datum: 14.07.2004 (15:15 Uhr)

[nur Vorträge von anderen, keine Mitschriften]

Graphentheorie

Einordnung der Vorlesung

- Graphentheorie gehört zu den diskreten Theorien
- Hier soll Wert gelegt werden auf strukturelle Eigenschaften von Graphen, weniger auf algorithmische Aspekte der Graphentheorie. „Die Vorlesung soll also keine abartige Algorithmen-Vorlesung sein."

Anwendungen

- metabolische Netzwerke
- chemische Reaktion

Definition: gerichteter Graph

Sei
- V eine Menge von Knoten
- $E \subseteq (V \times V)$ eine Menge von Kanten über V

Dann heißt:
- $G = (V, E)$ **gerichteter Graph**.

Definition: einfacher gerichteter Graph

Sei
- $G = (V, E)$ ein gerichteter Graph,
- $\forall (x \in V) : (\neg((x, x) \in E))$ (Es gibt keine Schlingen von einem Knoten in sich selbst).

Dann heißt:
- G **einfacher gerichteter Graph**.

Definition: Kante

Sei
- $G = (V, E)$ ein Graph
- $e \in E$

Dann heißt:
- e **Kante**.

Schreibweise

Sei
- $G = (V, E)$ ein gerichteter Graph
- $e \subseteq E$
- $e = (x, y)$ eine gerichtete Kante (also eine Kante eines gerichteten Graphen).

Dann bedeutet die Schreibweise $x \to y$, dass eine Kante von Knoten x nach Knoten y (implizit in der Kantenmenge E des Graphen G) existiert.

Definition: Knoten-Labelfunktion

Sei
- $G=(V,E)$ ein Graph
- $Labels$ eine Menge (von Labels)
- $\lambda_V: V \to Labels$

Dann heißt
- λ_V **Knoten-Labelfunktion**.

Definition: Kanten-Labelfunktion

Sei
- $G=(V,E)$ ein Graph
- $Labels$ eine Menge (von Labels)
- $\lambda_E: E \to Labels$

Dann heißt
- λ_E **Kanten-Labelfunktion**.

Definition: Graphen-Unrichten

Sei
- $G_0=(V_0,E_0)$ ein gerichteter Graph,
- $G_1=(V_0,E_1)$ ein ungerichtere Graph,
- $\forall(x\in V_0):\forall(y\in V_0):\left(((x,y)\in E_1)\Leftrightarrow\left(((x,y)\in E_0)\vee((y,x)\in E_0)\right)\right)$

Dann gilt:
- G_1 ist das Ergebnis der Methode „**Graphen-Unrichten**“ angewendet auf G_0.

Definition: Multigraph

Sei
- V eine Menge (von Knoten),
- E eine Menge (von Kanten),
- $\tau_{head}: E \to V$ eine Funktion, die zu jeder Kante ihren Start-Knoten zuordnet,
- $\tau_{tail}: E \to V$ eine Funktion, die zu jeder Kante ihren End-Knoten zuordnet.

Dann heißt:
- $G=(V,E,\tau_{head},\tau_{tail})$ **Multigraph**.

Definition: Multigraph zur Graph

Sei
- $G=(V,E,\tau_{head},\tau_{tail})$ ein Multigraph

Einen Multigraphen kann man in einen normalen Graphen übersetzen, indem man jede Kante $e\in E$ mit $e.getHead()=x$ und $e.getTail()=z$ eines Multigraphen ersetzt durch zwei Kanten e_0, e_1 mit $e_0.getHead()=x$ und $e_0.getTail()=y_e$ und $e_1.getHead()=y_e$ und $e_1.getTail()=z$, wobei y_e ein neuer Knoten (in der Mitte von e) ist, der nur für die ehemalige Kante e existiert.

Definition: Hypergraph

Ein Hypergraph ist ein Graph, bei dem der Kopf einer Hyperkante eine Knotenmenge (statt

genau ein Knoten) ist und der Schwanz einer Hyperkante eine Knotenmenge (statt genau ein Knoten) ist.

Beispiel

Eine chemische Reaktion $A+2B \rightarrow 2E+F+G$ lässt sich durch eine Hyperkante $h=(\{A,B\},\{E,F,G\})$ darstellen.
Dies müssen wir noch verkomplizieren durch Gewichte, die Gewichtsfunktionen $w_{head}: V \times Hyperkante \rightarrow Gewichtswerte$, $w_{tail}: V \times Hyperkante \rightarrow Gewichtswerte$ liefern:

- $w_{head}(A,h)=1$
- $w_{head}(B,h)=2$
- $w_{head}(E,h)=0$
- $w_{head}(F,h)=0$
- $w_{head}(G,h)=0$
- $w_{tail}(A,h)=0$
- $w_{tail}(B,h)=0$
- $w_{tail}(E,h)=2$
- $w_{tail}(F,h)=1$
- $w_{tail}(G,h)=1$

Eine natürliche Erweiterung der Graphentheorie auf mehr-als-zwei-stellige Relationen (etwa $E \subseteq V \times V \times V$) ist Peter F. Stadler nicht bekannt. „Da hätte ich mehr darüber stolpern müssen, wenn es da etwas gäbe, weil ich mich mit Relationensystemen beschäftige."

wesentliche Konzepte

Definition: Pfad

Sei

- $G=(V,E)$ ein Graph.
- $p=(x_0,e_1,x_1,e_2,x_2,\ldots,x_{n-1},e_n,x_n)$ mit
  - $\forall(i):(x_i \in V)$ sowie $\forall(i):(e_i \in E)$
- $\forall(i \in (\mathbb{N} \cap]0,n[)):\forall(j \in (\mathbb{N} \cap]0,n[)):((x_i=x_j) \Rightarrow (i=j))$ (Jeder innerer Knoten kommt im Pfad nur einmal vor, aber Kreise (Anfangs- und Endknoten sind gleich) sind zulässig.)

Dann heißt:

- p **Pfad** (von x_0 nach x_n bezüglich G).

Bemerkung

Für jeden Pfad gilt, dass er ein Weg ist.

Definition: Weg

Sei

- $G=(V,E)$ ein Graph.

- $p=(x_0, e_1, x_1, e_2, x_2, \ldots, x_{n-1}, e_n, x_n)$ mit
  - $\forall(i):(x_i \in V)$ sowie $\forall(i):(e_i \in E)$
- $\forall(e_i):\forall(e_j):((e_i=e_j)\Rightarrow(i=j))$ (Jede Kante kommt im Weg nur einmal vor.)

Dann heißt:

- p **Weg** (von x_0 nach x_n bezüglich G).

Bemerkung

Für jeden Weg gilt, dass er ein Trail ist.

Definition: Trail

Sei

- $G=(V, E)$ ein Graph.
- $p=(x_0, e_1, x_1, e_2, x_2, \ldots, x_{n-1}, e_n, x_n)$ mit
  - $\forall(i):(x_i \in V)$ sowie $\forall(i):(e_i \in E)$

Dann heißt:

- p **Trail** (von x_0 nach x_n bezüglich G).

Bemerkung

Für einen Trail gibt es keine Einschränkung, dass Knoten oder Kanten nicht doppelt vorkommen dürfen.

Schreibweise

Einen Trail $p=(x_0, e_1, x_1, e_2, x_2, \ldots, x_{n-1}, e_n, x_n)$ (sowie seine Spezialfälle Wege und Pfade) kann man auch als **Kantenzug** $p_e=(e_1, e_2, \ldots, e_n)$ (kürzer) schreiben, denn die Kanten implizieren die Knoten zwischen den Kanten.

Definition: Teilgraph

Sei

- $G_0=(V_0, E_0)$ ein Graph,
- $G_1=(V_1, E_1)$ ein Graph.

Dann gilt:

$$(G_1.istTeilgraphVon(G_0))\Leftrightarrow\left((V_1\subseteq V_0)\wedge(E_1\subseteq E_0)\wedge\underbrace{(V_1\subseteq(E_1\times E_1))}_{(V_1,E_1)\ ist\ Graph}\right)$$

Schreibweise

$$(G_1.istTeilgraphVon(G_0))\Leftrightarrow((V_1, E_1)\subseteq_G(V_0, E_0))$$

Definition: induzierter Teilgraph

Sei

- $G_0=(V_0, E_0)$ ein Graph,
- $G_1=(V_1, E_1)$ ein Graph.
- $W\subseteq V_0$ eine Knotenmenge
- $\forall(x\in W):\forall(y\in W):((x, y)\in E_0)\Rightarrow((x, y)\in E_1)$: Jede Kante aus G_0, deren

Endknoten zu W gehören, gehört auch zu G_1

Dann gilt:

- G_1 heißt der durch W auf G_0 **induzierte Teilgraph**.

Definition: Kreis

Sei

- $G=(V,E)$ ein Graph,
- $p=(x_0,e_1,x_1,e_2,x_2,\ldots,x_{n-1},e_n,x_n)$ ein Pfad bezüglich G,
- $x_0=x_n$.

Dann gilt:

- p heißt Kreis bezüglich G.

Definition: verallgemeinerter Kreis

Sei

- $G_0=(V_0,E_0)$ ein Graph,
- $G_1=(V_1,E_1)$ ein Teilgraph von G_0.
- G_1 lässt sich zerlegen in Teilgraphen, die nur aus genau einem Kreis bestehen, wobei die Vereinigung aller Teilgraphen wieder G_1 ergibt.

Dann gilt:

- G_1 heißt **verallgemeinerter Kreis**.

Definition: spannender Wald

Sei

- $G_1=(V_1,E_1)$ ein Graph,
- $\neg\exists(p\in G_1.getPfade()):(p.istKreis())$

Dann gilt:

- G_1 heißt **spannender Wald**.

Definition: spannender Wald

Sei

- $G_0=(V_0,E_0)$ ein Graph,
- $G_1=(V_1,E_1)$ ein Teilgraph von G_0.
- $V_1=V_0$,
- $G_1.istSpannenderWald()$.

Dann gilt:

- G_1 heißt **spannender Wald** von G_0.

Definition: zusammenhängend

Sei

- $G=(V,E)$ ein Graph.

Dann gilt:

- $G.istZusammenhängend()\Leftrightarrow$
  $\forall(x\in V):\forall(y\in V):\exists(p\in G.getPfade()):$
  $((p.getStartVertex()=x)\wedge(p.getEndVertex()=y))$

Definition: spannender Baum

Sei
- $G_1=(V_1, E_1)$ ein spannender Wald,
- $G_1.istZusammenhängend()$.

Dann gilt:
- G_1 heißt **spannender Baum**.

Definition: spannender Baum

Sei
- $G_0=(V_0, E_0)$ ein Graph,
- $G_1=(V_1, E_1)$ ein spannender Wald von G_0

Dann gilt:
- G_1 heißt **spannender Baum** von G_0.

Satz

$\forall(G_0 \in Graphen): \exists(G_1 \in G_0.getTeilgraphen()): (G_1.istSpannenderBaumVon(G_0))$

Für jeden Graph gibt es mindestens einen spannenden Baum.

Definition: kreisfrei

Sei
- $G=(V, E)$ ein Graph,

Dann gilt:
- $G.istKreisfrei() \Leftrightarrow \forall(p \in G.getPfade()): (\neg p.istKreis())$

Definition: Baum

Sei
- $G=(V, E)$ ein Graph,

Dann gilt:
- $G.istBaum() \Leftrightarrow G.istKreisfrei()$.

Satz

$\forall(G_0 \in Graphen): \exists(G_1 \in G_0.getTeilgraphen()): (G_1.istSpannenderBaumVon(G_0))$

Für jeden Graph, der ein Baum ist, gibt es genau einen spannenden Baum. Dieser spannende Baum ist der betrachtete Graph selbst.

Wir beschäftigen uns heute damit, wie wir Graphen als Matrizen darstellen können.

Definition: Adjazenz-Matrix

Sei
- $G=(V,E)$ ein ungerichteter Graph
- $A: V\times V\to\{0,1\}$ mit $\forall(x\in V):\forall(y\in V):\left(A_{x,y}=\begin{cases}1 & \{x,y\}\in E\\ 0 & sonst\end{cases}\right)$

Dann gilt:
- A heißt **Adjazenz-Matrix** von G.

Definition: Inzidenz-Matrix

Sei
- $G=(V,E)$ ein ungerichteter Graph
- $H: V\times E\to\{0,1\}$ mit $\forall(x\in V):\forall(e\in E):\left(H_{x,e}=\begin{cases}1 & x\in e\\ 0 & sonst\end{cases}\right)$

Dann gilt:
- H heißt **Inzidenz-Matrix** von G.

Bemerkung

Man kann eine Inzidenz-Matrix auch umgekehrt als $H: E\times V\to\{0,1\}$ definieren. Dann lassen sich aber charakteristische Vektoren nicht als Spaltenvektoren anmultiplizieren, sondern nur als Zeilenvektoren, und „das führt bei mir zu Kopfweh“.

Definition: charakteristischer Vektor

Sei
- $G=(V,E)$ ein ungerichteter Graph,
- $F\subseteq E$
- $X^F: E\to\{0,1\}$ mit $\forall(e\in E):\left(X^F_e=\begin{cases}1 & e\in F\\ 0 & sonst\end{cases}\right)$

Dann gilt:
- X^F heißt charakteristischer Vektor für F bezüglich G.

Definition: Transposition

Sei
- $H: V\times E\to\{0,1\}$ eine Inzidenz-Matrix.
- $H^+: E\times V\to\{0,1\}$ mit $\forall(x\in V):\forall(e\in E):\left(H^+{}_{e,x}=H_{x,e}\right)$

Dann gilt:
- H^+ ist die transponierte Matrix von H.

Definition: Multiplikation

Sei

- $A: X \times Y \to W$
- $B: Y \times Z \to W$
- $C: X \times Z \to W$ mit $\forall (x \in X): \forall (z \in Z): \left(C_{x,z} = \sum_{\forall (y \in Y)} \left(A_{x,y} \cdot B_{y,z} \right) \right)$

Dann gilt:

- $C = A \cdot B$

Bemerkung: Multiplikation von Inzidenz-Matrizen mit der eigenen transponierten

Sei

- $H: V \times E \to \{0,1\}$ eine Inzidenz-Matrix.
- $H^+: E \times V \to \{0,1\}$ die transponierte Maatrix von H.

Dann gilt:

- $\forall (x \in V): \forall (y \in V): \left(\left(H \cdot H^+ \right)_{x,y} = \sum_{\forall (e \in E)} \left(H_{x,e} \cdot H^+{}_{e,y} \right) = \sum_{\forall (e \in E)} \left(H_{x,e} \cdot H_{y,e} \right) \right)$
- $\left(H \cdot H^+ \right)_{x,y} = ¿$
  $\forall (x \in V): \forall (y \in V): ¿$
  ¿
  ¿

$$\begin{cases} x = y & \sum_{\forall (e \in E)} \left(\begin{cases} 1 & x \in e \\ 0 & sonst \end{cases} = deg(x) \right) \\ x \neq y & \left[\sum_{\forall (x \in E)} \left(\begin{cases} 1 & wenn\,(x,y) = e \\ 0 & sonst \end{cases} \right) \right] \end{cases} ¿¿¿$$

$$\begin{cases} 1 & (x,y) \in E \\ 0 & sonst \end{cases} = A_{x,y}$$

Definition: Inzidenz-Matrix

Sei

- $G = (V, E)$ ein gerichteter Graph ohne Schlingen.
- $H: V \times E \to \{-1,0,1\}$ mit $\forall (x \in V): \forall (e \in E): \left(\vec{H}_{x,e} = \begin{cases} +1 & x = e.getHead() \\ -1 & x = e.getTail() \\ 0 & sonst \end{cases} \right)$

Definition:

Sei

- $G = (V, E)$ ein gerichteter Graph
- $A^{ungerichtet}{}_{x,y} = Anzahl\ der\ Kanten\ in\ G\ zwischen\ x\ und\ y\ ohne\ Berücksichtigung\ der\ Richtung$

Satz

$$A^{ungerichtet}_{x,y} = A^{ungerichtet}_{y,x}$$

Satz: Multiplikation von Inzidenz-Matrizen mit der eigenen transponierten

$$\left(\vec{H}\cdot\vec{H}^{+}\right)_{x,y}=\sum_{\forall(e\in E)}\left(\vec{H}_{x,e}\cdot\vec{H}^{+}{}_{e,y}\right)=\sum_{\forall(e\in E)}\left(\vec{H}_{x,e}\cdot\vec{H}_{y,e}\right)$$

$$\left(\vec{H}\cdot\vec{H}^{+}\right)_{x,y}=\begin{cases}x=y & \underbrace{\sum_{\forall(e\in E)}\left(\begin{cases}1 & x\in e\\ 0 & sonst\end{cases}\right)}_{:=deg(x)=degin(x)+degout(x)}\\ x\neq y & \underbrace{\begin{cases}-1 & wenn\ \left(e=(x,y)\right)\vee\left(e=(y,x)\right)\\ 0 & sonst\end{cases}}_{=-A^{ungerichtet}{}_{x,y}+\delta_{x,y}\cdot deg(x)}\end{cases}$$

$$\left(\vec{H}\cdot\vec{H}^{+}\right)_{x,y}=\begin{cases}x=y & \sum_{\forall(e\in E)}\left(\left(\vec{H}_{x,e}\right)^{2}\right)\\ x\neq y & [...nie\ an\ die\ Tafel\ geschrieben...]\end{cases}$$

$$\vec{H}\cdot\vec{H}^{+}=-A+diag(deg)$$

mit $diag:\{v:X\to Y\}\to\{m:X\times X\to Y\}$ mit $diag(deg)=\begin{pmatrix}deg(x_0) & 0 & \dots & 0\\ 0 & deg(x_1) & & \vdots\\ \vdots & & \ddots & \vdots\\ 0 & 0 & \dots & deg(x_{n-1})\end{pmatrix}$

Definition: Laplace-Matrix eines Graphen

$$\vec{H}\cdot\vec{H}^{+}=diag(deg)-A=:L$$

Eigenschaft: Symmetrie

$$L_{x,y}=L_{y,x}$$

Satz

$$\forall(y\in V):\left(\sum_{\forall(x\in V)}\left(L_{x,y}\right)=-\underbrace{\sum_{\forall(x\in V)}\left(A_{x,y}\right)}_{=deg(y)}+\underbrace{\sum_{\forall(x\in V)}\left(\delta_{x,y}\cdot deg(x)\right)}_{=deg(y)}=0\right)$$

weil

$$\sum_{\forall(x\in V)}\left(A_{x,y}\right)=\sum_{\forall(x\in V)}\left(\begin{cases}1 & wenn\ \{x,y\}\in E\\ 0 & sonst\end{cases}\right)=\#\,Kanten,\ die\ y\ enthalten\ =deg(y)$$

Satz

Aus obigem Satz und der Symmetrie folgt:

$$\forall(x\in V):\left(\sum_{\forall(y\in V)}\left(L_{x,y}\right)=0\right)$$

Satz

Aus $\forall(x\in V):\left(\sum_{\forall(y\in V)}\left(L_{x,y}\right)=0\right)$ sowie $\forall(y\in V):\left(\sum_{\forall(x\in V)}\left(L_{x,y}\right)=0\right)$ folgt:

$L \cdot \underline{1} = \underline{0}$ mit $\underline{1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$ sowie $\underline{0} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

Sätze

Da $\underline{0} = 0 \cdot 1$ gilt, gilt auch: $L \cdot 1 = 0 \cdot 1$. Damit gilt: $0 \in L.getEigenwerte()$ (denn für jeden Eigenwert λ einer Matrix A gilt: $A \cdot x = \lambda \cdot x$). Damit gilt: $det(L)$. Damit gilt: $\neg L.istInvertierbar()$. Damit gilt: Lineare Gleichungssysteme der Form $L \cdot A = B$ haben keine eindeutige Lösung.

Satz

Für jede Matrix A und jeden Spaltenvektor x gilt: $(A \cdot x)^+ = x^+ \cdot A^+$.

Satz

Da für Eigenwerte gilt: $A \cdot \underline{x} = \lambda \cdot \underline{x}$, gilt auch:
$\underline{x}^+ \cdot A \cdot \underline{x} = \underline{x}^+ \cdot \lambda \cdot \underline{x} = \lambda \cdot \underline{x}^+ \cdot \underline{x} = \lambda \cdot \|\underline{x}\|^2$, wobei $\|\underline{x}\|$ die Norm des Vektors $\underline{x}$ ist.
Damit gilt: $\lambda = \frac{\underline{x}^+ \cdot A \cdot \underline{x}}{|\underline{x}|^2}$.
Gilt $\|\underline{x}\| = 1$ (ist also $\underline{x}$ normiert), so gilt sogar $\lambda = \underline{x}^+ \cdot A \cdot \underline{x}$.

Satz

Angenommen, dass $\underline{x}$ normierter Eigenvektor von L ist. Dann gilt:
$\lambda = \underline{x}^+ \cdot L \cdot \underline{x} = \underline{x}^+ \cdot (H \cdot H^+) \cdot \underline{x} = \underbrace{(\underline{x}^+ \cdot H)}_{=\underline{h}} \cdot \underbrace{(H^+ \cdot \underline{x})}_{=\underline{k}} = \left\| (H^+ \cdot \underline{x}) \right\|^2 \geq 0$. Es gilt: $\underline{h}^+ = k$
Die Eigenwerte von dieser Laplace-Matrix L sind alle nichtnegativ (also echt positiv, wenn man den Eigenwert 0 ausnimmt).

Was nützt uns der Laplace-Operator? Über die Definition $\vec{H} \cdot \vec{H}^+ = diag(deg) - A =: L$ haben wir in der Hauptdiagonale den Grad der einzelnen Knoten, außerhalb der Hauptdiagonale haben wir die Adjazenzmatrix. Da die Adjazenzmatrix eindeutig für einen Graphen ist, ist auch der Laplace-Operator eindeutig. Wir hoffen, über algebraische Eigenschaften des Laplace-Operators auf geometrische Eigenschaften des Graphen schließen zu können.

Wir zeigen nun den Zusammenhang zwischen Laplace-Operator und Zusammenhangseigenschaften des repräsentierten Graphen.

Wieviele Eigenvektoren gibt es zum Eigenwert 0 bei der Laplace-Matrix eines Graphen?

Definition: Kern

Sei
- L eine Matrix

Dann gilt:

- $ker(L) = L.getKern() = \{\underline{x} | L \cdot \underline{x} = 0\}$.

Es stellt sich die Frage: Wie groß ist $dim(ker(L))$? Wir wissen $1 \in L.getKern()$. Damit enthält $L.getKern()$ mindestens einen nichttrivialen Vektor (also einen anderen Vektor als $\underline{0}$). Damit gilt $L.getKern().getDimension() \geq 1$.

Wir werden zeigen, dass ein Graph G genau dann zusamenhängend ist, wenn $G.getLaplaceMatrix().getKern().getDimension() = 1$

Definition: Zusammenhangskomponente

Zusammenhangskomponenten sind maximale zusammenhängende Teilgraphen. Für die Teilgraphen-Knotenmengen C_1 , C_2 , … , C_k gilt:

- $C_i \subseteq V$
- $\bigcup_{\forall(i)} (C_i) = V$
- $\forall(i) : \forall(j) : \left(\left((C_i \cap C_j) \neq \emptyset \right) \Rightarrow (i = j) \right)$

Satz: Zusammenhang zwischen Laplace-Matrizen von Mehrkomponenten-Graphen

$L = concat\left(\begin{pmatrix} L_1 & 0 & 0 \\ 0 & L_2 & 0 \\ 0 & 0 & L_3 \end{pmatrix} \right)$, wobei L_i jeweils die Laplace-Matrix der kleineren Zusammenhangskomponente i ist und 0 hier für mit Nullen gefüllten Matritzen steht.

Allgemein gilt: $L = concat\left(\begin{pmatrix} L_1 & 0 & 0 & 0 \\ 0 & L_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & L_n \end{pmatrix} \right)$.

$$L \cdot 1_j = concat\left(\begin{pmatrix} L_1 & & & & \\ & \ddots & & & \\ & & L_j & & \\ & & & \ddots & \\ & & & & L_k \end{pmatrix} \right) \cdot concat\left(\begin{pmatrix} 0 \\ \vdots \\ 1_j \\ \vdots \\ 0 \end{pmatrix} \right) = concat\left(\begin{pmatrix} \underline{0} \\ \vdots \\ L_j \cdot 1_j \\ \vdots \\ \underline{0} \end{pmatrix} \right) = \underline{0}$$

Literatur

- http://cip.uni-trier.de/lange/semna/spektralanalyse-ausarbeitung.pdf
- http://citeseer.ifi.unizh.ch/mohar97some.html
- Bojan Mohar: „...“

[...20 Minuten zu spät...]
[...heute nicht gehalten von Peter F. Stadler...]

Chemische Reaktionsnetzwerke

Eine chemische Reaktion besteht aus mehreren Produkten, mehreren Edukten, die auch noch gewichtet sind.

Beispiele

- $\underbrace{2\,Ag + 1\,S}_{Edukte} \rightarrow \underbrace{1\,Ag_2S}_{Produkte}$
- $4\,Fe + 3\,O_2 \rightarrow 2\,Fe_2O_3$
- $\underbrace{C_2H_5OH}_{Alkohol} + O_2 \rightarrow \underbrace{HC_2H_3O_2}_{Essig} + H_2O$

Eine verallgemeinerte chemische Reaktion lässt sich so darstellen:

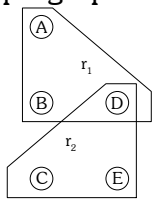
$$a\,A + cC \overset{vi}{\rightarrow} eE + hH$$

Jede chemische Reaktion mit m Edukten und n Produkten lässt sich durch eine stöchiometrische Matrix der Größe m times n darstellen.

Wenn die chemischen Stoffe Knotenpunkte eines Graphen darstellen sollen, dann sind normale Kanten nicht die geeignete Darstellung für chemische Reaktionen. Viel geeigneter sind Hyperkanten.

Beispiel

Wir haben zwei chemische Reaktionen r_1, r_2, die gleichzeitig ablaufen können. Dies wollen wir in einem chemischen Reaktionsnetzwerk darstellen.

- in standard-Notation
  - $r_1: \; A + B \overset{k_1}{\rightarrow} D$
  - $r_2: \; C \overset{k_2}{\rightarrow} D + E$
- als Hypergraph:
  - 

- als bipartiter Graph
  - [...Foto...]
  - Jeden Hypergraphen kann man als bipartiten Graphen darstellen
- als stöchiometrische Matrix:

Stoff	r_1	r_2
A	-1	0

Stoff	r_1	r_2
B	-1	0
C	0	-1
D	1	1
E	0	1

- also $S = \begin{pmatrix} -1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$

$$\vec{J} = \begin{pmatrix} k_1 \cdot A.getConcentration() \cdot B.getConcentration() \\ k_2 \cdot C.getConcentration() \end{pmatrix}$$

Wir berechnen nun die einzelnen Konzentrationsänderungsgeschwindigkeiten nach der Formel $\frac{\overrightarrow{d\, x.getConcentration()}}{dt} = S \cdot \vec{J}$:

- $\frac{d\, A.getConcentration()}{dt} = -k_1 \cdot A.getConcentration() \cdot B.getConcentration()$
- $\frac{d\, B.getConcentration()}{dt} = -k_1 \cdot A.getConcentration() \cdot B.getConcentration()$
- $\frac{d\, C.getConcentration()}{dt} = -k_2 \cdot C.getConcentration()$
- $\frac{d\, D.getConcentration()}{dt} = +k_1 \cdot A.getConcentration() \cdot B.getConcentration() + k_2 \cdot C.getConcentration()$
- $\frac{d\, C.getConcentration()}{dt} = +k_2 \cdot C.getConcentration()$

Wenn man biochemische Netzwerke unterschiedlicher Organismen nimmt, dann kann man ein bestimmtes Teilnetzwerk (z.B. den Citrat-Zyklus) nehmen und schauen, wie dieses in verschiedenen Organismen „implementiert“ ist. Die Teilnetzwerke der unterschiedlichen Organismen kann man Vergleichen (z.B. indem man die symmetrische Differenz (XOR) nimmt) und je nach der relativen Größe der symmetrischen Differenz kann man Abstände zwischen Organismen berechnen. Daraus kann man wieder Verwandschaftsinformationen und -entfernungen berechnen.

Eigenschaften von stöchiometrischen Matrizen

Balancierung von Atom-Anzahlen

Beispiel

Betrachtenn wir diese Reaktion:
$Glucose + Adenosintriphosphat \rightarrow Glucose6phosphat + Adenosindiphosphat$

Wir wollen wissen, ob eine solche Reaktion überhaupt Sinn haben kann.

Eine Tabelle, die darstellt, welcher Stoff aus jeweils wieviel Atomen besteht. (Die konkreten Zahlen sind hier aber nicht garantiert.)

Atom	***Glucose***	***ATP***	***G6P***	***ADP***
C	6	10	6	10
H	12	13	11	13
O	6	13	9	10
P	0	3	1	2
N	0	5	0	5

also $D = \begin{pmatrix} 6 & 10 & 6 & 10 \\ 12 & 13 & 11 & 13 \\ 6 & 13 & 9 & 10 \\ 0 & 3 & 1 & 2 \\ 0 & 5 & 0 & 5 \end{pmatrix}$

Diese Matrix D können wir mit einer stöchiometrischen Matrix S für die Reaktion multiplizieren:

Stoff	***Gewicht***
Glucose	-1
ATP	-1
G6P	1
ADP	1

also $S = \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}$

damit gilt: $D \cdot S = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$. Dieser Vektor muss aber ein Null-Vektor sein, damit diese Reaktion Sinn macht. Hier macht die Reaktion keinen Sinn. Offensichtlich ist der Wasserstoff-Anteil

nicht ausgeglichen. Also müssen wir Wasserstoff hier berücksichtigen:
$Glucose + Adenosintriphosphat \rightarrow Glucose6phosphat + Adenosindiphosphat + H$

Balancierung von Ladungs-Anzahlen

Beispiel

Betrachten wir die chemische Reaktion $2O_2^- + 2H^+ \rightarrow H_2O_2 + O_2$
Hier ist die Tabelle E wie folgt aufgebaut:

	O_2	H	H_2O_2	O_2
e^+	-1	1	0	0

Hier ist die stöchiometrische Matrix S wie folgt aufgebaut:

Stoff	Gewicht
O_2	-1
H	-1
H_2O_2	1
O_2	1

$$E = \begin{pmatrix} -1 & 1 & 0 & 0 \end{pmatrix}$$
$$S = \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}$$
$$E \cdot S = \begin{pmatrix} 0 \end{pmatrix}$$
Die Ladung ist hier also ausgeglichen.

lineare Abbildung zwischen Räumen durch stöchiometrische Matrix

Die Gleichung $\frac{\overrightarrow{d\,x.getConcentration()}}{dt} = S \cdot \vec{J}$ verbindet den Reaktionsraum mit dem Konzentrationsänderungsgeschwindigkeitsraum. Wir können annehmen, dass sich ein Organismus (zum Beispiel eine Zelle) im Gleichgewicht findet. das heißt, dass die sich die Konzentrationen der einzelnen Stoffe nicht ändert, also es gilt: $\frac{\overrightarrow{d\,x.getConcentration()}}{dt} = S \cdot \vec{J} = \vec{0}$. Diese Gleichung können wir nach $\vec{J}$ auflösen. Die Löungsmenge aller passenden Vektoren $\vec{J}$ ergibt gerade die Menge aller möglichen Zustände (steady states), bei denen sich in der Zelle keine Konzentration mehr ändert. Dies ist der Null-Raum der linearen Gleichung $\frac{\overrightarrow{d\,x.getConcentration()}}{dt} = S \cdot \vec{J}$. Ein solcher Vektor $\vec{J}$ repräsentiert gerade eine Reaktions-Fluss-Konfiguration, also eine Zuordnung von chemischen Reaktionen zu ihrer jeweiligen Intensität bzw. Reaktionsgeschwindigkeit (z.B. in mol/s).

Setzt man einen Organismus unter Stress (stellt man zum Beispiel die Nahrungszufuhr ab), so ändern sich die Reaktionsgeschwindigkeiten und schließlich die Konzentrationen der einzelnen Stoffe. Diese kann man unter Umständen (nachdem sich der Organismus an den Stress angepasst hat, er sich also in einem Steady-State befindet) messen und aufgrund der

Messungen die Menge der möglichen Steady-State-Punkte stark einschränken und letztendlich einen neuen Steady-State-Punkt herausfinden. Aus diesem Punkt wiederum lässt sich wiederum die Reaktions-Fluss-Konfiguration ermitteln. Dazu hilft uns die stöchiometrische Matrix.

Betrachten wir einen Graphen G, und davon dessen Laplace-Matrix $L = G.getLaplaceMatrix()$. Dann gilt:
$L \cdot \underline{1} = \underline{0}$. Das gilt übrigens genau dann, wenn $L \cdot \underline{1} = 0 \cdot \underline{1}$. Daraus folgt: $L.getKern().getDimension() \geq 1$. Genaue genommen gilt sogar $L.getKern().getDimension() \geq |G.getZusammenhangskomponent()|$.

Nun wollen wir noch zeigen $L.getKern().getDimension() = |G.getZusammenhangskomponent()|$

Satz: Theorem von Perron Frobenius

Sei

- A eine irreduzible n times n-Matrix.
- $a_{i,j} \geq 0$ und der gerichtete Graph $\vec{\Gamma}(A)$
  - $V = (1,2,\ldots,n) \quad \forall (i \neq j) : ((i,j) \in E \Leftrightarrow a_{i,j} > 0)$
- stark zusammenhängend.

Dann gibt es einen Eigenvektor $\underline{x}$ mit $x_k > 0$ und zugehörigem Eigenwert λ_1, sodass λ_1 einfach reell ist und $\lambda_1 = \max_{Eigenwerte} := \rho(A.getSpektralradius())$. Es gilt auch $\forall (\lambda \neq \lambda_1) : (|\lambda| < |\lambda_1|)$.

Noch zu zeigen:
Aus dem Perron-Forbenius-Theorem folgt:

- Wenn G zusammenhängend ist, dann ist 0 [Null] einfacher Eigenwert von $G.getLaplaceMatrix()$

Das letzte Mal haben wir bewiesen: für $L = D - A$: Alle Eigenwerte sind größer gleich null und es gibt einen Eigenwert, der genau 0 ist. 0 ist kleinster Eigenwert von L mit Eigenvektor $\underline{1}$.
Wenn wir nun $(-L) = A - D$ betrachten, kehren sich die Eigenwerte um. Hier ist dann 0 der größte Eigenwert (immer noch mit Eigenvektor $\underline{1}$), alle anderen Eigenwerte sind negativ.

Sei $a > \rho(L)$ ($\rho(L)$ ist der Spektralradius von L, also der Wert des größten Eigenvektors).
Dann sei $\tilde{A} = A - D + a \cdot I$ wobei I die Einheitsmatrix ist.

$B \cdot \underline{x} = \lambda \cdot x$.
$(B + a \cdot I) \cdot \underline{x} = B \cdot \underline{x} + a \cdot I \cdot \underline{x} = \lambda \cdot \underline{x} + a \cdot \underline{x} = (\lambda + a) \cdot \underline{x}$
Damit ist a größter Eigenwert mit Eigenvektor $\underline{1}$ und es gilt: alle Einträge von $\tilde{A}$ sind größer gleich 0.
Dann sagt uns das Perron-Frobenius-Theorem, dass a ein einfacher Eigenwert ist von $\tilde{A}$. Damit ist 0 ein einfacher Eigenwert von $(-L) = A - D$. Damit ist 0 ein einfacher Eigenwert von $L = D - A$.

Beispiel

$$A = \begin{pmatrix} 1.3 & 0 & 1.7 \\ 0 & 1.7 & 1.92 \\ 0.16 & 0 & 0 \end{pmatrix}$$

Der zugehörige Graph sieht dann so aus:
[...Foto...]

Satz
Sei

- G ein Graph

Dann gilt:

- $$\begin{aligned} |G.getZusammenhangskomponenten()| &= L.getLaplaceMatrix().getKern().getDimensionen() \\ &= |G.getLaplaceMatrix().getEigenwerte().getVielfachheitFürEigenwert(0)| \end{aligned}$$

$$\widetilde{A}_{i,j} = \widetilde{A}_{j,i}$$
$$\vec{\Gamma}(\widetilde{A}).istStarkZusammenhängend() \Leftrightarrow G.istZusammenhängend()$$

Ausgehend von G

- erhalten wir $L = D - A$
- erhalten wir A
- erhalten wir $\widetilde{A} = A - \underbrace{(D - a \cdot I)}_{\text{diagonal}}$

$\vec{\Gamma}(\widetilde{A}) =$
$\forall (i \neq j)$:

- $\left((i,j) \in E(\vec{\Gamma})\right) \Leftrightarrow (A_{i,j} = i)$
- $\left((i,j) \in G\right) \Leftrightarrow (A_{i,j} = 1)$
- $\left((i,j) \in G\right) \Leftrightarrow (\widetilde{A}_{i,j} = 1)$

Wenn ein Graph G nicht zusammenhängend ist, kann man seine Laplace-Matrix L so schreiben:

$$L = concat\left(\begin{pmatrix} L_1 & 0 & 0 & 0 \\ 0 & L_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & L_n \end{pmatrix}\right)$$

Eine interessante Anwendung der Berechnung von Zusammenhangskomponenten über Eigenwerte von Laplace-Matrizen von Graphen ist das Clustering. Denn es kann sein, dass die Laplace-Matrix nicht überall den Wert 0 hat, sondern dort, wo sie den Wert 0 eigentlich haben könnte, einen sehr kleinen, aber von 0 verschiedenen Wert hat. Das heißt, dass der Graph schon zusammenhängend ist, aber es gibt Cluster, die untereinander intensiv zusammenhängend sind und gegeneinander nur sehr schwach zusammenhängend sind. (Wenn man sich in dem Graphen also „bewegt“, ist es unwahrscheinlich, in ein anderes Cluster zu kommen, aber nicht unmöglich.) Was wir beobachten, ist, dass schwacher

Zusammenhang sich in bestimmten Eigenvektoren darstellt, die folgendes Format haben:

$$\begin{pmatrix} 0+\epsilon_0 \\ 0+\epsilon_1 \\ \vdots \\ 0+\epsilon_{i-1} \\ 1+\epsilon_i \\ 1+\epsilon_{i+1} \\ \vdots \\ 1+\epsilon_{j-1} \\ 0+\epsilon_j \\ 0+\epsilon_{j+1} \\ \vdots \\ 0+\epsilon_n \end{pmatrix}$$

Der Eigenvektor hat also „fast“ einen Teil, indem seine Einträge „fast 1“ sind, sonst sind die Einträge „fast 0“. Je nach der Sortierung der Knoten sind aber die Einträge des Eigenvektors ebenfalls nicht sortiert, sodass die „fast 1“-Einträge nicht in einem Block sind. Durch geeignetes Sortieren der Knoten kann man aber auch die „fast 1“-Einträge so sortieren, dass sie einen Block ergeben.

Wir können weiter beobachten:

- $\lambda_1 = 0$ (λ_1 ist der kleinste Eigenwert von G)
- λ_2 ist der zweit-kleinste Eigenwert von G
- $0 = \lambda_1 \leqslant \lambda_2 \leqslant \lambda_3 \leqslant \ldots \leqslant \lambda_n$
- $G.istZusammenhängend() \Leftrightarrow (\lambda_2 > 0)$

Wir beobachten aber weiter: wenn G fast nicht zusammenhängend ist, dann ist λ_2 sehr klein.

Definition: k-Knoten-zusammenhängend

Ein Graph G ist k-**Knoten-zusammenhängend** genau dann, wenn $G[V \setminus \{x_1, \ldots, x_{k-1}\}]$ immer zusammenhängend ist und es Knoten gibt $y_1, y_2, \ldots, y_K$ gibt, sodass $G[V \setminus \{y_1, \ldots, y_k\}]$ nicht zusammenhängend ist.

Bemerkung

k ist die kleinste Zahl von Knoten, die ich aus dem Graphen herausreißen muss, damit der Graph nicht mehr zusammenhängend ist

Schreibweise

$\alpha(G) = k$, wobei G k-Knoten-zusammenhängend ist.

Beispiele

[...Foto...]

Das gleiche können wir auch für Kanten (k-Kanten-zusammenhängend) definieren.

Schreibweise

$\epsilon(G)=k$, wobei G k-Knoten-zusammenhängend ist.

Satz

Ein Graph ist zusammenhängend genau dann, wenn es ein k größer 0 gibt, sodass er k-Knoten-zusammenhängend ist.

$$\lambda_2 \leqslant \alpha(G)$$

$$2 \cdot \epsilon(G) \cdot \left(1 - \cos\left(\frac{\pi}{|V|}\right)\right) \leqslant \lambda_2$$

Definition: algebraische Konnektivität

Der zweite Eigenwert λ_2 heißt auch **algebraische Konnektivität**, weil er ja viel über die Konnektivität es Graphen aussagt.

Man kann noch viel mehr Zusammenhänge finden zwischen strukturellen Eigenschaften von Graphen und algebraischen Eigenschaften von Graphen.

Literatur

- Bojan Mohar: <Artikel über die „Laplacian“ [also den Laplace-Operator]> aus Lubljana
- Dragos M Cvetkovic, Micahel Doob, Horst Sachs: „Spectra of Graphs“ „Theory and Applications“

Wir können auch etwas sagen über den Eigenvektor zu λ_2.

$$W^+ = \{u \mid (u \in V) \wedge (x(u) \geqslant 0)\}$$
$$W^- = \{u \mid (u \in V) \wedge (x(u) \leqslant 0)\}$$

Dann können wir den Graphen G in zwei Teilgraphen $G[W^+]$, $G[W^-]$ zerlegen.

Definition: Fiedler-Vektor

Der Eigenvektor x zum zweiten Eigenwert λ_2 heißt Fiedler-Vektor.

Diese Zerteilung in Teilgraphen wirkt so, dass der Graph gerade dort zerschnitten wird, so besonders geringer Zusammenhang besteht. Der Graph wird also in genau 2 Teilcluster partitioniert. Diese Partitionierung kann man beispielsweise nutzen, um bei Parallelrechnern die Aufgabenmengen so zu partitionieren, dass die Kommunikation zwischen einzelnen Knoten des Parallelrechners (entsprichend die Knoten des Graphen) möglichst gering ist.

Wenn x Eigenvektor von $G.getLaplaceMatrix()$ zum Eigenwert λ_k ist, dann gilt: $\left|G\left[W^+\right].getZusammenhangskomponenten()\right|+\left|G\left[W^-\right].getZusammenhangskomponenten()\right|\leqslant k$

Für eine schwingende Saite gilt: Die Anzahl der Nullstellen (der Stellen, an der eine schwingende Saite nicht schwingt) einer Sinus-Schwingung entspricht gerade der Nummer des Obertons bzw. die Nummer ihres Eigenwertes.

Im Wesentlichen wenden wir dieses Problem nur auf Graphen an.

„Can you hear the shape of a drum?“: Anhand des Klangspektrums kann man unter bestimmten Bedingungen tatsächlich die Form einer Trommel rekonstruieren.

„Can you hear the shape of a graph?“ Nein. Es gibt unterschiedliche Graphen, die gleiche Eigenwerte liefern.

[...25 Minuten zu spät...]

Satz

In einem zusammenhängenden Graphen, wobei jeder Knoten eine gerade Anzahl von Kanten hat, kann ich an einem beliebigen Knoten loslaufen, alle bereits begangenen Kanten vermeiden und ich komme garantiert wieder beim Startknoten an (und bin einen Kreis gelaufen).

Satz

Einen so generierten Kreis kann ich vom Graphen G_0 abziehen und ich erhalte einen Graphen G_1, wobei der Grad aller Knoten von G_1 wiederum gerade ist. Dies kann ich immer wiederholen. Wenn der Graph G_0 endlich war, dann stoße ich mit diesem Verfahren bis zum Graphen G_k, der kantenlos ist.

Beweis

[...]

Satz

Aus der Liste der so generierten Kreise lässt sich ein eulerscher Weg konstruieren, und zwar mit folgendem Verfahren: Man starte mit Kreis Q_0 und laufe Q_0 entlang, bis man auf einen Kreis Q_i mit höherer Nummer stößt. Dann unterbreche man das Langlaufen entlang von Q_0, sondern laufe Q_i entlang. Falls man wieder auf einen Kreis Q_j mit höherer Nummer stößt, dann laufe man diesen rekursiv entlang, und so weiter. Ist man auf einen Kreis Q_j vollständig entlanggelaufen, so setze man den Kreislauf am jeweils vorherigen Kreis fort.

Der so abgelaufene Weg ist ein eulerscher Weg durch den Graphen.

Satz

Jeder solcher eulerscher Graph ist eine kantendisjunkte Vereinigung einfacher Kreise.

Sei

- K ein zusammenhängender verallgemeinerter Kreis, für den wir einen eulerschen Weg kennen.

Wir können diesen Kreis reduzieren in einen elementaren Kreis, wobei alle Knoten den Knotengrad 2 haben (wo es also keine Verzweigungen gibt). Dies geht so: Man laufe entlang von K. Sobald man auf eine Verzweigung (=Knoten mit Knotengrad größer 2) stößt und man eigentlich einen Umweg laufen würde (weil man ja später selbst wieder auf den Knoten stößt), geht man den Umweg nicht, sondern lässt den Umweg einfach weg. Dann stößt man wieder auf den Startknoten und ist einen Weg langgelaufen, bei dem kein Knoten mehrfach durchgangen wird.

Satz

Jeder Graph, der kein eulerscher Graph ist, ist keine kantendisjunkte Vereinigung einfacher Kreise.

Wann sind zwei Graphen isomorph?

Zum Beispiel in Datenbanken über chemische Formeln möchte man einen chemischen Stoff anhand der Strukturformel wiederfinden, aber nicht nur anhand genau des selben Graphen, sondern auch anhand von („gleichberechtigten“) isomorphen Graphen (wo die Knoten vielleicht anders nummeriert sind).

Definition: isomorph

Sei

- $G_0=(V_0, E_0)$
- $G_1=(V_1, E_1)$

Dann gilt:

- G_0 heißt isomorph zu G_1 genau dann, wenn eine Funktion $\phi: V_0 \to V_1$ existiert, für die gilt:
  1. $\phi.isInvertierbar()$ und
  2. $\forall(x \in V_0): \forall(y \in V_0): \left(\left(\{\phi(x), \phi(y)\} \in E_1\right) \Leftrightarrow \left(\{x, y\} \in E_0\right)\right)$ (also wenn man den einen Graphen durch Knotenumnummerieren in den anderen Graphen überführen kann)

Suche anhand isomorpher Graphen

Man kann aus der großen Menge möglicher Graphen diejenigen herausfiltern, die gleiche strukturelle Eigenschaften haben wie der Graph, mit dem angefragt wird. Solche strukturellen Eigenschaften können zum Beispiel sein:

- Knotenanzahl
- Kantenanzahl
- Spektrum des Graphen
- [Kreisstruktur?]
- [chromatische Zahl?]

Definition: Subgraphenisomorphie

Ein verwandtes Problem ist **Subgraphenisomorphie** (wobei Graphisomorphie nur ein Teilproblem davon ist). Beispielsweise möchte ich ein chemisches Molekül bauen aus Molekülen, für die ich bereits weiß, wie ich sie baue.

Graphen aus Kreisen aufbauen

Wir wollen nun betrachten, wie wir Graphen aus Kreisen bauen können, obwohl diese Graphen nicht unbedingt eulersche Graphen sind.

Beispiel

$C_3 = C_1 \oplus C_2$ (wobei $\square \oplus \square$ die symmetrische Differenz zwischen zwei Kreisen ist).

Kontext

Im folgenden betrachten wir immer einen Graphen G mit n Knoten und m Kanten. Ein Kreis C wird einfach durch einen Kantenvektor $\{0,1\}^m$ dargestellt, wobei $(C[i]=1) \Leftrightarrow (e \in C)$, also wobei C für die Kante i gleich 1 genau ist, wenn die Kante i zum Kreis C gehört.

Im Wesentlichen ist das ein Vektor aus einem Vektorraum, der auf dem $Körper\left(\{0,1\}, \underbrace{\square \oplus \square}_{\text{XOR bzw. Addition modulo 2}}, \underbrace{\square \odot \square}_{\text{AND bzw. Multiplikation}}\right)$, wobei hier $\square \oplus \square : \{0,1\} \times \{0,1\} \to \{0,1\}$ sowie $\square \odot \square : \{0,1\} \times \{0,1\} \to \{0,1\}$.

Diese XOR-Operation und diese AND-Operation können wir aber auf Kantenvektoren entsprechend erweitern: $\square \oplus \square : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}^m$ sowie $\square \odot \square : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}^m$

Wenn wir zwei Kreise C_0, C_1 miteinander XORn, entsteht wieder ein Kantenvektor $C_2 = C_0 \oplus C_1$. Dieser Kantenvektor kann

- wieder ein einfacher Kreis sein, oder
- zwei disjunkte Kreise repräsentieren, oder auch
- den Null-Kantenvektor (gewissermaßen einen Null-Graphen) repräsentieren, wenn $C_0 = C_1$, denn da löschen sich alle Kanten gerade heraus durch die XOR-Operation

Offensichtlich wollen unsere Kreis-Kantenvektoren gleich eulersche Graphen repräsentieren, also lassen wir dies.

Satz

Sei

- C_0 ein eulerscher Graph als Kantenvektor
- C_1 ein eulerscher Graph als Kantenvektor

Dann gilt:

- $C_0 \oplus C_1$ ist ein eulerscher Graph.

Beweis

- $|C|$ sei die Anzahl der gesetzten Kanten in C für jeden Kantenvektor C
- $\underbrace{|C_0|}_{\text{gerade}} + \underbrace{|C_0|}_{\text{gerade}} - \underbrace{2 \cdot |C_0 \cap C_1|}_{\text{gerade}} = \underbrace{|C_0 \oplus C_1|}_{\text{erst recht gerade}}$

Satz

Die Menge der eulerschen Teilgraphen von G bilden einen Vektorraum bezüglich der Operation $\square \oplus \square$ (symmetrische Differenz der Kantenmengen) über GF(2).

Satz

Sei

- $G=(V,E)$ ein Graph
- $C_0 \in G.getEulerscheTeilgraphen()$
- $C_1 \in G.getEulerscheTeilgraphen()$
- $C_2 = C_0 \oplus C_1$

Dann gilt:

- $C_2 \in G.getEulerscheTeilgraphen()$

Definition: Graphen-XOR

Sei

- $G=(V,E)$ ein Graph
- $C_0 \in G.getEulerscheTeilgraphen()$
- $C_1 \in G.getEulerscheTeilgraphen()$
- Für jeden Teilgraphen C ist $X(C)$ wie folgt definiert: $\forall (e \in E): \left(X(C)(e) = \begin{cases} 1 & \text{wenn } e \in C \\ 0 & \text{sonst} \end{cases} \right)$

Satz

Es gilt:

- $\forall (e \in E): \left(\left(X(C_0)(e) \; XOR \; X(C_1)(e) \right) = X(C_0 \oplus C_1)(e) \right)$

Offensichtlich sind C_0 und $X(C)$ miteinander isomorph.

Eigenschaften

Es gilt:

1. $C_0 \oplus C_1 = C_1 \oplus C_0$
2. $(C_0 \oplus C_1) \oplus C_2 = C_0 \oplus (C_1 \oplus C_2)$
3. $(C \oplus C) = \emptyset$
4. $(-C)$ ist das Inverse bezüglich $\square \oplus \square$, und zwar $(-C) = C$, weil $C \oplus (-C) = \emptyset$
5. $C \oplus \emptyset = C$, $\emptyset \oplus C = C$

Definition: Vektorraum der eulerschen Teilgraphen

Sei

- $EV = (G.getEulerscheTeilgraphen(), \square \oplus \square)$

Es gilt:

- $EV.istKommutativeGruppe()$
- EV ist ein Vektorraum über dem Grundkörper $(\{0,1\}, \square \oplus \square, \square \odot \square)$

Definition: Dimension

Die Dimension eines Vektorraums ist die maximale Anzahl von miteinander linear unabhängigen Vektoren, also die größtmögliche Länge einer Menge von Vektoren, die alle miteinander linear unabhängig sind.

Die Dimension eines Vektorraums ist auch die minimale Anzahl der Elemente eines Erzeugendenensystems im Vektorraum.

Definition: Erzeugendensystem

ES heißt **Erzeugendensystem** von einem Vektorraum V, wenn $\forall(z \in V): \exists\left(\underbrace{\lambda_i}_{\forall(i \in ES)} \in V.getGrundkörper()\right): \left(z = \sum_{\forall(i \in ES)} (\lambda_i \cdot i)\right)$

Definition: linear abhängig

Sei
- V ein Vektorraum
- $M \subseteq V.getElemente()$

Dann gilt:

$M.istLinearAbhängigBezüglich(V) \Leftrightarrow$
- $\exists\left(\underbrace{\lambda_i}_{\forall(i \in M)} \in V.getGrundkörper()\right): \left(\exists(\lambda_i): (\lambda_i \neq 0) \wedge \left(\sum_{\forall(i \in M)} (\lambda_i \cdot i) = \vec{0}\right)\right)$

Das heißt also: Wenn ich Koeffizienten λ_i finden kann (wobei mindestens einer dieser Koeffizienten nicht 0 sein darf), sodass die Summe der mit diesen Koeffizienten gewichteten Elemente der Menge M gleich dem 0-Vektor ist, dann heißt M **linear abhängig**.

Definition: Basis eines Vektorraums

Sei
- V ein Vektorraum
- $M \subseteq V.getElemente()$

Dann gilt:
- M ist eine **Basis** des Vektorraums V, wenn M
  1. ein Erzeugendensystem für V ist und
  2. eine linear unabhängige Menge von V ist.

Bemerkung

Wenn M eine Basis von V ist, dann gilt:
1. M ist ein minimales Erzeugendensystem für V
2. M ist eine maximale noch linear unabhängige Menge von V

Definition: linear unabhängig

Sei
- V ein Vektorraum
- $M \subseteq V.getElemente()$

Dann gilt:

- $M.istLinearUnbhängigBezüglich(V) \Leftrightarrow \left(\left(\sum_{\forall (i \in M)} (\lambda_i \cdot i) = \vec{0} \right) \Rightarrow \forall (i \in ES) : (\lambda_i = 0) \right)$ [?]

Sei B eine Basis von EV.
$z = \bigoplus_{\forall (i \in B)} (\lambda_i \cdot i)$ mit $\lambda_i \in \{0,1\}$

Jedes Element z von EV können wir durch genau einen Vektor $z \cong \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{|B|} \end{pmatrix}$ darstellen. Jeder solcher entspricht einem Element z von EV.

Wieviele eulersche Teilgraphen gibt es für einen Graphen G (bezüglich einer Basis B)? Die Anzahl der eulerschen Teilgraphen ist gleich der Anzahl der Elemente von EV, das ist gleich $2^{|B|}$. Da $|B|$ immer gleich ist für den selben Graphen G, unabhängig von den konkreten Elementen in B, ist auch die Anzahl der eulerschen Teilgraphen von G unabhängig von der konkreten Wahl von B.

Sei T ein Spannbaum von $G = (V, E)$. Für jede Kante $e \in (E \setminus T)$ (mit $e = \{x, y\}$ gibt es genau einen Weg $P_{x,y}$ [...abgewischt...]

$\Rightarrow cyc(T, e) := P_{x,y} \cup \{e\}$

[...abgewischt...]

Satz

$\forall (T \in G.getSpannbäume()) : (|T| = |G.getKnotenMenge()| - 1)$
Ein Spannbaum T eines Graphen G hat genau so viele Kanten, wie G Knoten minus 1 hat.

Satz

Sei
- $B_K = \{cyc(e, T) | e \in (E \setminus T)\}$ (das K von B_K steht für Kirchhoff)

Dann gilt:
- $B_K.istLinearUnabhängigBezüglich(EV)$

Behauptung

$\{cyc(e, T) | e \in (E \setminus T)\}.istLinearUnabhängigBezüglich(EV)$

Beweis

[...Foto...]

$$(\lambda_{m-n+1} \cdot 1 = 0) \Rightarrow (\lambda_{m-n+1} = 0)$$
$$\vdots$$
$$(\lambda_2 \cdot 1 = 0) = (\lambda_2 = 0)$$
$$(\lambda_1 \cdot 1 = 0) = (\lambda_1 = 0)$$

Wir beweisen noch (später irgendwann vielleicht), dass B_K auch ein Erzeugendensystem ist.

Dann wissen wir, dass B_K eine Basis ist.

Der Beweis wird ungefähr so gehen:
- G ist zusammenhängend,
- K ist eine Menge von Kanten, sodass $\neg(V, (E \setminus K)).istZusammenhängend()$

[...]

Definition: cut

Sei
- $G = (V, E)$ ein Graph,
- $x \in V$,
- $K_x = \{e \mid (e \in E) \wedge (x \in e)\}$ (K_x ist die Menge aller Kanten aus G, die inzident mit x sind)

Dann heißt
- K **cut** von x bezüglich G.

Satz

Sei
- $G = (V, E)$ ein Graph,
- $G.istSchlaufenfrei()$

Dann gilt:
- $\bigoplus_{\forall (x \in V)} \left(\underbrace{G.getCutFor(x)}_{=K_x} \right) = \emptyset$

Begründung

Jeder Kante kommt bei dieser XOR-Operation genau zweimal vor, die XOR-Operation führt gerade zur Auslöschung beider Kanten miteinander. Jede Kante kommt deswegen genau zweimal voraussetzung

Satz

Sei
- $G = (V, E)$ ein Graph,
- $G.istSchlaufenfrei()$
- $G.istZusammenhängend()$

Dann gilt:

- $\forall (M \subseteq V): \left(\left((M \neq V) \wedge (M \neq \emptyset) \right) \Rightarrow \left(\bigoplus_{\forall (x \in M)} \left(\underbrace{G.getCutFor(x)}_{=K_x} \right) \neq \emptyset \right) \right)$
- $\forall (M = V): \left(\bigoplus_{\forall (x \in M)} \left(\underbrace{G.getCutFor(x)}_{=K_x} \right) = \emptyset \right)$
- $\forall (M = \emptyset): \left(\bigoplus_{\forall (x \in M)} \left(\underbrace{G.getCutFor(x)}_{=K_x} \right) = \emptyset \right)$

Es gibt also nur eine einzige nichtleere Knotenmenge, wo die XOR-Summe der entsprechenden Cuts die leere Menge ist, und das ist die vollständige Knotenmenge.

Zusammenfassung

Wir können aus einem Graphen eine Eulersche-Graphen-Basis erstellen, indem wir einen spannenden Baum nehmen, dort noch eine Kante hinzufügen und dann haben wir einen eulerschen Kreis.

Graphen-Umformung

[...Foto...]

Satz

Ebenen-Einbettung und Torus-Einbettung ist nicht äquivalent.

Beispiel

Den bipartiten Graphen mit je 3 Knoten pro Partition ($K_{3,3}$) kann man leicht im Torus einbetten, aber nicht in der Ebene.

Beispiel

Den vollständigen Graphen auf 5 Knoten (K_5) kann man leicht im Torus einbetten, aber nicht in der Ebene.

Satz

Für jeden Graphen G gibt es eine Fläche (aber nicht notwendigerweise eine Ebene), in die er eingebettet werden kann.

Satz

Die Einbettung in die Ebene und die Einbettung in die Kugel-Oberfläche sind äquivalent.

Begründung

Einen in die Ebene eingebetteten Graphen kann ich ausschneiden, zusammenfalten und den Rand zum Südpol erklären.

Begründung

Einen in die Kugeloberfläche eingebetteten Graphen kann so in die Ebene überführt werden, indem ein Punkt der Kugeloberfläche, der von keiner Kante berührt wird, aufgeblasen wird

zum Rand der Ebene.

Das nächste Mal

- Zusammenhang zwischen Planarität und Kreisbasen
- Minimale Kreisbasen
- aber: nächste Woche ist keine Vorlesung „Christi Himmelfahrt“, Männertag,
- aber: 1. Juni 2006: Vorlesungsvertretung
- aber: danach Pfingsten
- aber: danach noch einmal Vorlesungsvertretung
- danach: 20. Juni 2006: Prof. Stadler ist zurück.

Definition: planer Graph, plane graph

Sei

1. $\Gamma = (V, E)$
2. $V \subseteq \mathbb{R}^2$
3. $\forall (e \in E): \left((e: [0,1] \to \mathbb{R}^2) \wedge e.istStetig() \wedge e.istInjektiv() \right)$ (Stetigkeit bedeutet „eine Linie vom Startpunkt zum Endpunkt) (Injektivität bedeutet „keine Schlaufen in der Linie“)
4. $\forall (e \in E): \forall (x \in [0,1]): \left((e(x) \in V) \Leftrightarrow ((x=0) \vee (x=1)) \right)$ (Knoten sind nur Endpunkte von Kanten)
5. $\forall (e \in E): (e(0) \neq e(1))$ (Wir verbieten Schlaufen, auch wenn dies schon aus der Injektivität folgt)
6. $\forall (e_0 \in E): \forall (e_1 \in E): \forall (x \in]0,1[): \forall (y \in]0,1[): (e_0(x) \neq e_1(y))$ (Kanten dürfen sich nicht schneiden)

Dann gilt:

- Γ heißt **planer Graph** oder (englisch) **plane graph**

Definition: planarer Graph

Sei

- $G = (V, E)$ ein Graph

Dann gilt:

- $G.istPlanar() \Leftrightarrow \exists (\Gamma \in PlaneGraphs): \exists (f):$
  $\left((f: G.getKantenmenge() \to \Gamma.getKantenmenge()) \wedge f.istBijektiv() \right)$

Definition: Kontraktion

[...] (Eine Kontraktion nimmt eine Kante und verschmilzt diese Kante samt ihrer Endknoten zu einem neuen Knoten, der diese Kante und diese beiden Endknoten ersetzt.)

Definition: Minor

[...] (Ein Minor eines Graphen G_0 ist ein Graph G_1, der aus dem Graphen G_0 durch wiederholte Kontraktion oder Deletion (von Kanten oder Knoten) hervorgegangen ist.)

Definition: n-Zusammenhängend

Sei

- $G = (V, E)$ ein Graph
- $n \in \mathbb{N}$
- $\forall (F \subseteq E): \left((|F| = n) \Rightarrow G.ohneKanten(F).istZusammenhängend() \right)$

Dann gilt:

- $G.istNZusammenhängend(n)$

Lemma (Struktur 3-zusammenhängender Graphen)

Sei

- $G=(V,E)$ ein Graph
- $|G|>4$
- $G.ist3Zusammenhängend()$

Dann gilt:

- $\exists(e\in E):(G.ohneKante(e).ist3Zusammenhängend())$

Beweis

Nehmen wir an, dass die Behauptung nicht gilt.

- Für jedes $e\in E$:
  - Betrachten wir $G_1=G.ohneKante(e)$, $e=\{x,y\}$
  - Der kontrahierte Vertex heiße $v_{x,y}$.
  - Nun finden wir z, sodass $S=\{v_{x,y},z\}$ G_1 trennt.
  - Wähle e und z so, sodass [...abgewischt...]

Grafik

[...abgewischt...]

Satz von Kuratowski

Sei

- $G=(V,E)$ ein Graph

Dann gilt:

- $G.istPlanar()\Leftrightarrow\neg(G.hatAlsMinor(K_5)\vee G.hatAlsMinor(K_{3,3}))$

Bemerkung

$\forall(G_0\in Graphen):(G_0.istPlanar()\Rightarrow\forall(G_1\in G_0.getMinoren()):(G_1.istPlanar()))$

Beweis

Wir beweisen $G.istPlanar()\Leftarrow\neg(G.hatAlsMinor(K_5)\vee G.hatAlsMinor(K_{3,3}))$ mit Induktion über die Anzahl der Knoten ($|V|$) von 3-zusammenhängenden Graphen.

- Induktionsanfang
  - $|V|=4$.
  - Hier haben wir den K_4 (alle anderen Graphen mit einer Knotenanzahl gleich 4 sind nicht 3-Zusammenhängend.) Der K_4 ist tatsächlich planar.
- Induktionsschritt:
  - Annahme:
    - Wir nehmen an, dass alle 3-zusammenhängende Graphen $G_1=(V_1,E_1)$ mit $|V_1|\leqslant|V_0|$, die weder K_5 noch $K_{3,3}$ als Minor haben, planar sind.
  - Beweis
    - Nach obigem Lemma wählen wir eine Kante $e\in E_0$ so, dass $G.ohneKante(e)$ 3-zusammenhängend ist. Sei $e=\{x,y\}$.
    - Wir betrachten $G.ohneKnoten(\{y\})$

- [...unverständlich...]

Angenommen, der Graph G ist nicht 3-zusammenhängend, aber wenigstens 2-zusammenhängend. Dann kann man den Graphen G in zwei Graphen G_1, G_2 zerlegen, wobei G_1 und G_2 miteinander überlappen in zwei Knoten x, y (und wir nehmen an, dass G_1 und G_2 3-zusammenhängend sind). Wenn nun G_1 und G_2 selbst planar sind, dann kann ich beide Graphen an den Knoten x, y zusammenstecken.

Somit gilt der Satz von Kuratowski auch für Graphen, die nicht 3-zusammenhängend sind.

Satz von McLane

Sei

- $G=(V,E)$ ein Graph

Dann gilt:

- $$G.istPlanar()\Leftrightarrow \exists(Kreisbasis\in G.getCycleSpace()):\forall(e\in E):\left(\left|\{basiskreis|(basiskreis\in Kreisbasis)\wedge(basiskreis.enthält(e))\}\right|\leqslant 2\right)$$

Definition: einfach

Sei

- $G=(V,E)$ ein ungerichteter Graph
- $G.getEdgeSpace()=E(G)=Potenzmenge(E)$ der Kantenraum dieses Graphen
- $F\subseteq G.getEdgeSpace()$ (also $F\subseteq Potenzmenge(E)$)

Dann gilt:

- $$F.istEinfach()\Leftrightarrow \underbrace{\forall(e\in E):\left(\left|\{f|(f\in F)\wedge(e\in f)\}\right|\leqslant 2\right)}_{\text{Jede Kante aus Graph } G \text{ tritt höchstens 2 mal in } F \text{ auf.}}$$

Bemerkung

Sei

- $G=(V,E)$ ein ungerichteter Graph

Dann gilt:

- $G.getCycleSpace()\subseteq G.getEdgeSpace()$

Satz von McLane (1936)

Sei

- $G=(V,E)$ ein ungerichteter Graph

Dann gilt:

- $G.istPlanar()\Leftrightarrow\exists(F\subseteq G.getCycleSpace()):(F.istEinfach())$

Beweis

- $G.istPlanar()\Rightarrow\exists(F\subseteq G.getCycleSpace()):(F.istEinfach())$:

- „Diesen Beweis belassen wir mal wegen folgender Intuition“:
  - Jeden planaren Graphen (der bereits auf Zyklen irgendwie reduziert ist) kann man aus einzelnen Zyklen zusammensetzen, wobei jede Kante im (reduzierten) zusammengesetzten Graphen maximal zwei mal verwendet wurde (eventuell, um sie wieder auszulöschen).
- $G.istPlanar() \Leftarrow \exists (F \subseteq G.getCycleSpace()) : (F.istEinfach())$:
  - [...verlorengegangen wegen OpenOffices exzessiver Speicher-Zeit...]

Graphenfärben

Definition: Färbung

Sei
- $G = (V, E)$ ein Graph
- S eine Menge (die Menge der Farben)
- $c : V \to S$ eine Funktion (Färbungsfunktion)
- $\forall (\{x, y\} \in E) : ((x = y) \vee (c(x) \neq c(y)))$ (Benachbarte Knoten haben unterschiedliche Farben)

Dann gilt:
- c heißt **Färbung** für G.

Definition: k-färbbar

Sei
- $G = (V, E)$ ein Graph
- $k \in \mathbb{N}$

Dann gilt:
- $G.istKFärbbar(k) \Leftrightarrow \exists (c : V \to S) : (c.isColoringFor(G) \wedge (|S| = k))$

Satz: 4-Färbbarkeit planarer Graphen

Sei
- $G = (V, E)$ ein Graph,
- $G.istPlanar()$.

Dann gilt:
- $G.istKFärbbar(4)$.

Beweis

Der Beweis ist umstritten, da er ein „Computerbeweis“ und nicht schön aufzuschreiben ist.

Satz: 5-Färbbarkeit planarer Graphen

Sei
- $G = (V, E)$ ein Graph,
- $G.istPlanar()$.

Dann gilt:

- $G.istKFärbbar(5)$.

Beweis

- über Induktion
- Induktionsanfang:
  - Alle planaren Graphen mit $|V| \leq 1$ sind 5-färbbar.
- Induktionsschritt:
  - Voraussetzung:
    - Alle planaren Graphen mit $|V| \leq n-1$ sind 5-färbbar.
  - Behauptung:
    - Alle planaren Graphen mit $|V| \leq n$ sind 5-färbbar.
  - Beweis:
    - Wir betrachten den planaren Graphen $G_0 = (V_0, E_0)$ mit $|V_0| = n$
    - Man finde (aus dem Korollar von MacLane) x mit
      - $(x \in V_0) \wedge (x.getGrad() \leq 5)$
    - Wir betrachten nun $G_1 = G_0.ohneKnoten(\{x\})$.
    - Wir wissen:
      - $G_1.ist5Färbbar()$ (laut Annahme)
      - $\exists (c \in G_1.getKFärbungen(5)) : (c : V_1 \to S)$
    - Fall: $(S \setminus c.getBildFür(x.getNachbarn())) \neq \emptyset$ (Die Nachbarn von x lassen eine Farbe übrig) (einfacher Fall)
      - Dann sei $c(x) = s$ mit $s \in (S \setminus c.getBildFür(x.getNachbarn()))$
    - anderer Fall: (Die Nachbarn von x lassen keine Farbe übrig) (interessanter Fall)
      - [...]

Datum: 22.06.2006

[heute wieder gehalten von Prof. Stadler]

Das Thema heute ist Graphenfärbungen und der Konnex davon zu RNA-Strukturen.

Dabei können wir RNA-Wasserstoffbrückenbindungen so als Graphenfärbproblem auffassen, dass nur solche (Basen-)Paare entstehen dürfen:

- A..U
- U..G
- G..C
- U..A
- G..U
- C..G

Folgende Paare dürfen nicht entstehen:

- A..C
- A..G
- U..C
- G..A
- C..A
- A..A
- C..C
- G..G
- U..U

[...Foto fehlt, da kein Akku...]

Wir können versuchen, das Problem vereinfachen, indem wir

- A auf 1
- U auf 0
- G auf 1
- C auf 0

abbilden. Für jeden korrekten RNA-Sekundärstruktur-Graphen gilt, dass, wenn die RNA-Basen so durch 0 oder 1 ersetzt werden, der resultierende Graph ebenfalls gültig gefärbt ist, sodass benachbarte Basen (nur benachbart bezüglich der Wasserstoffbrückenbindungen) ebenfalls unterschiedlich gefärbt sind.

Sei

- Ψ ein gegebener Strukturgraph einer RNA-Sekundärstruktur.
- $C(\Psi)$ die Menge aller legalen A-U-G-C-Färbungen der Knoten von Ψ
- $\tilde{C}(\Psi)$ die Menge aller legalen 0-1-Färbungen der Knoten von Ψ
- die Knotenmenge geordnet $(1,2,\ldots,n)$

Für 2 Strukturen Ψ, Φ auf $(1,2,\ldots,n)$ ist die Frage: $\tilde{C}(\Phi)\cap\tilde{C}(\Psi)$, also: Welche Färbungen sind für

Satz

Sei

- Ψ_0 ein RNA-Struktur-Graph,
- Ψ_1 ein RNA-Struktur-Graph,

Dann gilt:

- $\left(\left(C\left(\Psi_0\right)\cap C\left(\Psi_1\right)\right)=\emptyset\right)\Leftrightarrow\left(\left(\tilde{C}\left(\Psi_0\right)\cap\tilde{C}\left(\Psi_1\right)\right)=\emptyset\right)$

Bemerkung

Es ist klar, dass man von $C\left(\Psi\right)$ auf $\tilde{C}\left(\Psi\right)$ schließen kann, indem die Zuordnung von $\left\{A,U,C,G\right\}$ auf $\left\{0,1\right\}$ wie oben ausführt.

Man kann auch von $\tilde{C}\left(\Psi\right)$ auf $C\left(\Psi\right)$ schlussfolgern, indem man von 0 auf A und von 1 auf U abbildet (C und G verwenden wir einfach nicht)

Wenn wir nun eine RNA-Kette betrachten, dann kann es sein, dass diese zwei verschiedene RNA-Sekundärstrukturen ausbildet
[...Foto fehlt, da kein Akku...]

„gesucht: Färbung auf der Vereinigung der Kantenmengen“ [?]

Satz

Für jeden Graphen gilt: Der Graph ist bipartit genau dann, wenn man ihn mit 2 Farben färben kann.

Satz

„Für zwei RNA-Sekundärstrukturen kann man immer eine biologische Sequenz finden, die diese Struktur ausbilden kann.“

Sei

- eine RNA-Sequenz gegeben
- Ψ_0 ein RNA-Sekundärstruktur-Graph für diese RNA-Sequenz,
- Ψ_1 ein RNA-Sekundärstruktur-Graph für diese RNA-Sequenz,
- (es gilt also: $\Psi_0.\,getKnotenMenge\left(\right)=\Psi_1.\,getKnotenMenge\left(\right)$)
- Ψ_0 , Ψ_1 Matching

Dann gilt:

- $\left(\tilde{C}\left(\Psi_0\right)\cap\tilde{C}\left(\Psi_1\right)\right)=\emptyset$

Beweis

Sei

- $\Psi_2=\left(V,\Psi_0.\,getKantenMenge\left(\right)\cup\Psi_1.\,getKantenMenge\left(\right)\right)$ wobei
  - $V=\Psi_0.\,getKnotenMenge\left(\right)=\Psi_1.\,getKnotenMenge\left(\right)$
- Für jeden Knoten $x\in V$ gilt:
  - $\left(\Psi_0.\,getDegreeFor\left(x\right)\leqslant1\right)\wedge\left(\Psi_1.\,getDegreeFor\left(x\right)\leqslant1\right)\Rightarrow\left(\Psi_2.\,getDegreeFor\left(x\right)\leqslant2\right)$

        - Da die Voraussetzung in unserem Fall wahr ist, ist auch die Folge wahr.
        - Es gilt also $\forall(x\in V):\left(\Psi_2.getDegreeFor(x)\leqslant 2\right)$
- Wir charaktierisieren den Graphen mit $\forall(x\in V):\left(\Psi_2.getDegreeFor(x)\leqslant 2\right)$
    - Betrachten wir einen Knoten x.
    - Dann kann folgendes sein:
        - $\Psi_2.getDegreeFor(x)=0$: Der Knoten ist ohne Nachbarn.
            - Dann ist dessen Farbe egal.
        - $x=x_0$ und x_0 hat genau einen Nachbarn x_1 und x_1 hat genau einen Nachbarn, nämlich x_0.
            - Dann lässt sich leicht eine Färbung finden.
        - x ist Teil einer Kette $\{(x_0,x_1),(x_1,x_2),(x_2,x_3),(x_3,x_4),\ldots\}$
            - Hat diese Kette 2 Enden, dann lässt sich eine 0-1-Färbung finden, indem einfach links mit 0 angefangen wird, dann mit 1, dann wieder 0, usw.
            - Ist diese Kette ein Kreis, so lässt sich eine 0-1-Färbung finden, wenn der Kreis aus einer geraden Anzahl von Knoten besteht. (Genau genommen lassen sich genau 2 0-1-Färbungen in diesem Fall finden.)
                - Ein solcher Kreis hat aber keine ungerade Anzahl von Knoten, denn dann müsste es vorher einen Knoten gegeben haben, der in seinem Ursprungs-Graphen den Knotengrad 2 gehabt hatte. Aber jeder Knoten im Ursprungs-Graph hatte vorher nur maximal Knotengrad 1 (weil eine RNA-Base nur mit maximal einer weiteren RNA-Base Wasserstoffbrückenbindungen aufbauen kann).
- Also können wir für Ψ_2 in jedem Fall eine 0-1-Färbung finden.
- Damit ist Ψ_2 auch bipartit.

Satz

Seien

- Ψ_0 , Ψ_2 , … RNA-Sekundärstruktur-Graphen über der selben RNA-Sequenz (damit auch mit gleicher Knotenmenge)

Dann gilt:

- $$\left(\underbrace{\bigcap_{\forall(i)}\left(C\left(\Psi_i\right)\right)}_{\text{Die Menge der Färbungen, die für alle Graphen gelten}}\neq\emptyset\right)\Leftrightarrow\left(\underbrace{\bigcup_{\forall(i)}\left(\Psi_i\right)}_{\text{vereinigter Graph}}\right).istBipartit()$$

Produkte von Graphen

Sei

- $G_0=(V_0,E_0)$ ein ungerichteter Graph
- $G_1=(V_1,E_1)$ ein ungerichteter Graph

[...Foto fehlt, da kein Akku...]

[...12 Minuten zu spät...]

Fragestellung

Sei

- ein Graph G_0 gegeben

Gilt dann:

- $\exists(G_1 \in Graphen) : \exists(G_2 \in Graphen) : \left((G_1 \, square \, G_2) \underbrace{=}_{\text{Isomorphie}} G_0 \right)$?

Definition: Djoković-Winkler-Relation

Djoković-Winkler-Relation Θ :

Sei

- $e = \{x, y\}$ eine Kante
- $f = \{v, w\}$ eine Kante

[...abgewischt von der Tafel...]

Eigenschaften

- $\Theta.istReflexiv()$
- $\Theta.istSymmetrisch()$
- $\neg\Theta.istTransitiv()$ (im Allgemeinen)
- Θ^* ist der transitive Abschluss von Θ
- $\Theta \subseteq (E \times E)$

Aus dem Buch „Product Graphs“ von „Imrich + Klavžor“

Satz

Sei

- $G = (V, E)$,
- P ein kürzester Pfad in G.

Dann gilt:

- $\forall(e \in P.getKnotenMenge()) : \forall(f \in P.getKnotenMenge()) : \left(\left((e, f) \in \Theta \right) \Rightarrow (e = f) \right)$

Beweis

[...Foto 0481...]

Satz

Sei

- $G = (V, E)$
- $\Theta = G.getDjokovićWinklerRelation()$,
- $d = G.getVertexDistanceFunction()$,

- $G.istBipartit()$,
- $e \in E$,
- $f \in E$,
- $e = \{x, y\}$,
- $f = \{u, v\}$,
- $(e, f) \in \Theta$

Dann gilt:

- $d(u, x) = d(v, y) = d(u, y) - 1 = d(v, x) = -1$

Beweis

[...Foto...]

Satz

Sei

- $G = (V, E)$ ein ungerichteter Graph,
- $\Theta = G.getDjokovićWinklerRelation()$,
- $W \in G.getWalks()$ ein Walk (also ein Lauf)
- $e \in E$
- $\neg(e \in W.getEdgeSet())$

Dann gilt:

- $\exists(f \in W.getEdgeSet()) : ((e, f) \in \Theta)$
- $f.istEindeutig() \Rightarrow ((e \cap f) = \emptyset)$

Grafik

[...Foto 0483...]

Beweis

- Wir definieren $\mu : E \times E \to \mathbb{N}$ wie folgt:
  - Sei
    - $p_0 = (x, y)$
    - $p_1 = (u, v)$
  - dann gilt:
    - $\mu(p_0, p_1) = d(x, v) - d(x, u) - d(y, v) + d(y, u)$
- $S = \sum_{i=1}^{m} (\mu(e, e_i)) = [...abgewischt...] = 2$
- [...abgewischt...]
- [...Foto 0484...]

Definition:

Sei

- $G = (V, E)$ ein ungerichteter Graph,
- $d = G.getVertexDistanceFunction()$,
- $\{u, v\} \in E$
- $W(u, v) = \{w \mid (w \in G.getWalks()) \wedge (d(w, u) < d(w, v))\}$

Dann gilt:

- W heißt .... von G.

Eigenschaften

- $(W(u,v) \cap W(v,u)) = \emptyset$
- $(W(u,v) \cup W(v,u)) = G.getWalks()$

Grafik

[...Foto 0486...]

Satz

- $G = (V, E)$ ein ungerichteter Graph,
- $G.istBipartit()$

Dann gilt:

- $\neg \exists (x) : (d(x,u) = d(x,v))$

Beweis

- Wir definieren $F : V \times V \to E$
  - $F(u,v) = \{ f \mid (f \in E) \wedge (f, \{u,v\}) \in \Theta \}$
- Wir behaupten:
  - $G \setminus F(u,v)$ ergibt genau 2 Zusammenhangskomponenten G_1, G_2, wobei
    - $G_1 = G[W(u,v)]$
    - $G_2 = G[W(u,v)]$

[...Foto 0488...]

- Sei
  - P ein kürzester Pfad von u nach w
- dann müsste $P.prepend((v,u))$ ebenfalls ein kürzester Pfad sein von v nach w.
- u ist aber in einer anderen Zusammenhangskomponente von $G \setminus F(u,v)$ als v

Erkennung von Hyperwürfeln

Satz

Jeder Hyperwürfel ist bipartit.

Korollar

Wenn ein Graph nicht bipartit ist, dann ist er auch kein Hyperwürfel.

Satz

Es gibt einen schnellen Algorithmus („linear in space and time“), der überprüfen kann, ob ein Graph bipartit ist.

Satz

Sei

- $G = (V, E)$ ein Hyperwürfel

Dann gilt:

- $|E|=\frac{1}{2}\cdot 2^n\cdot n$
- $n=\log_2(N)$
- $\frac{1}{2}\cdot N\cdot\log_2(N)=M$

Bemerkung

Wenn also ein Graph nicht die richtige Knotenanzahl-Kantenanzahl-Relation hat oder wenn er nicht bipartit ist, dann können wir aufhören, darauf zu testen, ob dieser Graph ein Hyperwürfel ist.

Satz

Jeder Hyperwürfel ist isomorph zu einem Graph-Produkt eines Hyperwürfels mit einer Kante (also einem Graphen, der nur aus einer Kante besteht, also ein K_2).

Bemerkung

Wir können schnell überprüfen, ob ein Graph G_0 sich zerlegen lässt in $G_0=G_1\boxdot K_2$. Wir können dann wieder prüfen, ob $G_1=G_2\boxdot K_2$, $G_2=G_3\boxdot K_2$, … so lange, bis $G_n\cong K_2$, also bis wir wissen, dass $G_0\cong K_2\boxdot K_2\boxdot K_2\boxdot\ldots\boxdot K_2$, also dass $G_0.istHyperwürfel()$

$G_t=G[W(u,v)]\cup G[W(v,u)]$
[...Foto 0491...]

Primfaktorzerlegung von Graphen

Definition: prim

Wir betrachten:
- G ein ungerichteter Graph,
- $G=A\boxdot B$,
- $|A.getVertices()|\geq 2$,
- $|B.getVertices()|\geq 2$ (A, B sind also nicht trivial, haben also jeweils mindestens 2 Knoten)

Sei
- G ein ungerichteter Graph

Dann gilt:
- $G.istPrim()\Leftrightarrow\neg\exists(A\in Graphen):\exists(B\in Graphen):$
  $\left((G=A\boxdot B)\wedge(|A.getVertices()|\geq 2)\wedge(|B.getVertices()|\geq 2)\right)$

Definition: Primfaktorenzerlegung

Sei
- G ein ungerichteter Graph
- $A_0, A_1, A_2, \ldots, A_{n-1}$ Graphen
- $\forall(i\in([0,n[\cap\mathbb{N})):(A_i.istPrim())$
- $G=A_0\boxdot A_1\boxdot A_2\boxdot\ldots\boxdot A_{n-1}$

Dann gilt:

- $(A_0, A_1, A_2, \ldots, A_{n-1})$ heißt **Primfaktorenzerlegung** von G.

Bemerkung

[Primfaktorenzerlegungen heißen auch dann gleich, wenn die einzelnen Primfaktoren nur umsortiert oder nur isomorph zueinander sind.]

Satz

Es gibt höchstens $\log_2(|G.getVertices()|)$ Primfaktoren.

Frage: Ist die Primfaktorenzerlegung eindeutig?

$$\left(\underbrace{K_1+K_2+\underbrace{(K_2 \boxdot K_2)}_{=K_2^{\,2}}}_{7 \text{ Knoten}}\right) \boxdot \left(\underbrace{K_1+\underbrace{(K_2 \boxdot K_2 \boxdot K_2)}_{=K_2^{\,3}}}_{9 \text{ Knoten}}\right) = \left(\underbrace{K_1+K_2^{\,2}+K_2^{\,4}}_{21 \text{ Knoten}}\right) \boxdot \left(\underbrace{K_1+K_3}_{3 \text{ Knoten}}\right)$$

$K_1+K_2^{\,2}+K_2^{\,4}+K_2+K_2^{\,3}+K_2^{\,5}$

[...Foto 0492...]

[...Foto 0493...]

... Wir bekommen heraus: Wenn der Graph nicht zusammenhängend ist, dann ist die Primfaktorenzerlegung nicht eindeutig.

Was wir haben wollen, ist zu zeigen, dass, wenn der Graph zusammenhängend ist, dass dann die Primfaktorenzerlegung eindeutig ist.
Das können wir auch beweisen, aber der Beweis „ist eine ziemlich grausige Index-Orgie“. Im Wesentlichen bauen wir ein Färbungsschema für Kanten entwickeln, sodass wir die einzelnen Kanten des Graphen auf die ursprünglichen Faktor-Graphen abbilden,

[...Foto 0494...]

Datum: 06.07.2006

[...heute hat Prof. Stadler ersst um 11:27 Uhr gestartet...]

Zufallsgraphen

gehen zurück auf Béla Bollobás aus Ungarn.

Einführung

Beim Würfeln weiß man vorher, dass für jede Seite eines Würfels gilt, dass bei einem Wurf die Wahrscheinlichkeit, ausgewählt zu werden, gleich $\frac{1}{6}$. Aber welche „intrinsischen“ a-priori-Wahrscheinlichkeiten haben Graphen?

Wir können zum Beispiel ausgehen von einem Graphen $G=(V,E)$, für den gilt, dass $n=|V|$, wobei n fest vorgegeben ist. Dann ist die Wahrscheinlichkeit, dass ein bestimmter Graph „zufällig“ entsteht, gleich $p(G)=\frac{1}{\#\text{ Mögliche Graphen auf } n \text{ Vertices}}$. Aber was ist $\#$ Mögliche Graphen auf n Vertices ? Sind isomorphe Graphen hier gleich?

Beispiel

Wir betrachten den Graph, der lediglich aussieht, wie ein Quadrat. Er hat 4 Ecken.

- Wenn wir die Ecken nummerieren, dann gibt es $\frac{4!}{4\cdot 2}=\frac{4\cdot 3\cdot 2\cdot 1}{4\cdot 2}=3$ verschiedene solche Graphen.
- Wenn wir die Ecken aber nicht nummerieren, dann gibt es nur einen einzigen solchen Graphen.

Wir betrachten den Graph, der lediglich aussieht, wie ein Ypsilon. Er hat ebenfalls 4 Ecken.

- Wenn wir die Ecken nummerieren, dann gibt es 4 verschiedene solche Graphen.
- Wenn wir die Ecken aber nicht nummerieren, dann gibt es nur einen solchen Graphen.

Weil solche Gleichverteilungsmaße „abartig“ sind bei Graphen, hat Béla Bollobás ein anderes Maß entwickelt:

Definition: binomisches Modell für Zufallsgraph

Sei

- $G=(V,E)$
- $n\in\mathbb{N}$ vorgegeben (Knotenzahl)
- $p\in[0,1]$ vorgegeben (Kantenwahrscheinlichkeit)
- $\forall(x\in V):\forall(y\in V):(p((x,y)\in E)=p)$
  - Die Wahrscheinlichkeit, dass eine bestimmte Kante bei dem Graphen gesetzt ist, ist gleich p.

Dann gilt:

- G ist ein Zufallsgraph nach dem **binomischen Modell** mit Kantenwahrscheinlichkeit p.

Beispiel

Sei $f_{2\text{colorable}} : Graphen \to \{0,1\}$ eine Funktion, die für jeden Graph sag, ob er 2-färbbar ist oder nicht: $\forall (G \in Graphen) : \left(f_{2\text{colorable}}(G) = \begin{cases} 1 & G.ist2Färbbar() \\ 0 & \text{sonst} \end{cases} \right)$

Wir wollen nun für eine gegebene Zufallsgraphenwahrscheinlichkeit p wissen: $\lim_{n\to\infty} \left(Prob\left(f\left(\underbrace{G_n}_{\text{Zufallsgraph mit } n \text{ Knoten mit Kantenwahrscheinlichkeit } p} \right) = 1 \right) \right)$, also: Wie verhält sich die Wahrscheinlichkeit, dass ein Zufallsgraph 2-färbbar ist, wenn wir die Anzahl der Knoten im Zufallsgraphen immer weiter erhöhen?

Im konkreten Fall gilt:

- Für $p \ll \frac{1}{n}$ ist der Graph bipartit (=2färbbar) für $n \to \infty$.
- Für $p = O(1)$ [?] (also eigentlich für $p > 0$) ist der Graph nicht bipartit (=2färbbar) für $n \to \infty$.
  - Denn: Sobald ein Dreieck im jeweiligen Graphen existiert, ist dieser Graph nicht mehr 2färbbar (=bipartit). Die Wahrscheinlichkeit, dass für 3 betrachtete potentielle Dreiecks-Kanten ein tatsächliches Dreieck existiert (also alle 3 Kanten gesetzt sind), ist p^3. Die Wahrscheinlichkeit, dass für 3 betrachtete potentielle Dreiecks-Kanten kein tatsächliches Dreieck existiert, ist also $1-p^3$. Die Anzahl solcher zu betrachtender potentieller Dreiecke steigt aber immer weiter mit n. Wenn wir $(1-p^3)^{\frac{1}{3}\cdot n}$ betrachten, dann sehen wir, dass $\lim_{n\to\infty} \left((1-p^3)^{\frac{1}{3}\cdot n} \right) = 0$. Damit gilt $\forall (p \in]0,1]) : \left(\lim_{n\to\infty} \left(Prob\left(f\left(\underbrace{G_n}_{\text{Zufallsgraph mit } n \text{ Knoten mit Kantenwahrscheinlichkeit } p} \right) = 1 \right) \right) = 0 \right)$.

wichtig für diese Vorlesung

- Es gibt diese Zufallsgraphentheorie.
- Man muss genau spezifizieren, wann zwei Graphen als gleich gelten.
- Man muss sein Zufallsgraphenmodell genau spezifizieren.

Ein Problem aus dem Nähkästchen (von Stadler)

„Wir sind einmal auf folgendes Problem gestoßen:“

Wir betrachten einen Hyperwürfel $(K_2)^n$. Wir betrachten den durch ausgewählte Knoten induzierten Teilgraphen. Dabei ist die Wahrscheinlichkeit, dass für einen ausgewählten Knoten ein Nachbar ebenfalls ausgewählt ist, gleich λ.

Dies ist ein anderes Zufallsgraphenmodell. Für solche Zufallsgraphen kann man auch Fragen stellen, zum Beispiel: Wieviel Zusammenhangskomponenten gibt es im betrachteten Teilgraphen abhängig von λ?

Das λ lässt sich zum Beispiel interpretieren als: Wieviel Mutation ist zulässig, ohne dass wesentliche Struktur-Änderungen passieren?
Resultate sind:

- Für große λ ist der Graph zusammenhängend.
- Für kleinere λ ist der Graph nicht zusammenhängend, aber dafür gibt es immer noch eine große Zusammenhangskomponente für einen recht großen Bereich von λ.