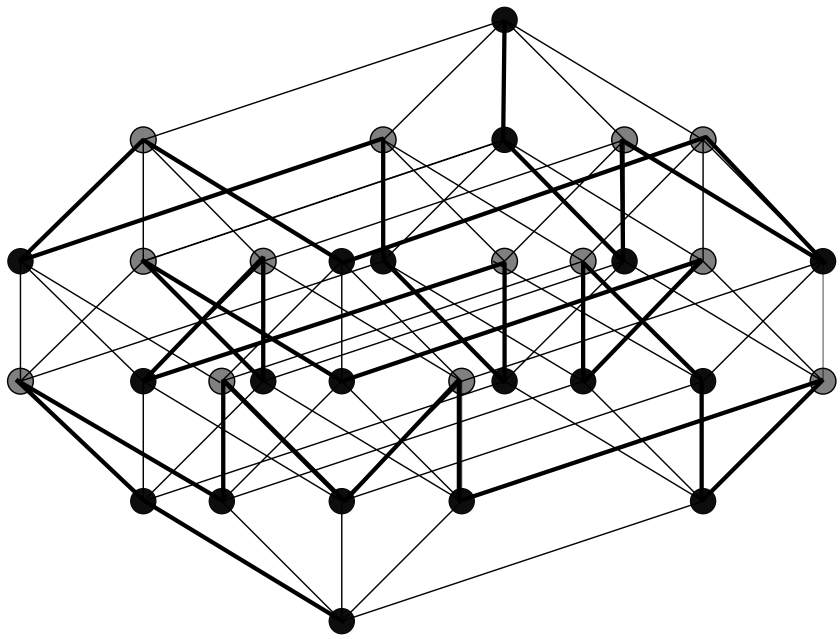
Vorlesung

Diskrete Mathematik

Stefan Felsner

felsner@inf.fu-berlin.de

21. Oktober 1998



Ausgearbeitet für das Schülerseminar 1998 am Fachbereich Mathematik der Freien Unversität Berlin.

Inhalt

1 Rundreiseprobleme und Graphen

1.1 Einleitung

Diese Vorlesung ist gedacht als Führung durch einige Teilgebiete der Diskreten Mathematik. Den roten Faden für unsere Tour soll ein populäres Problem aus der kombinatorischen Optimierung liefern, das *Traveling Salesman Problem* (Problem des Handlungsreisenden).

Ein Vertreter soll Kunden in n verschiedenen Städten besuchen. Die Fahrzeit zwischen je zwei Städten ist bekannt. Verständlicherweise möchte der Vertreter die Gesamtfahrzeit minimieren, um bald wieder Zuhause zu sein. Sein Problem ist nun, die Städte in der 'richtigen' Reihenfolge anzufahren, also eine optimale Rundreise festzulegen.

Die Eingabe für das Problem können wir uns als $n \times n$ Matrix $C = (c_{i,j})$ denken, der Eintrag $c_{i,j}$ gibt die Fahrzeit zwischen Stadt i und Stadt j an. Wir nennen C die *Kostenmatrix* für das Problem und werden immer annehmen, daß C eine positive Kostenmatrix ist, d.h. $c_{i,j} \geq 0$ für alle i, j. Gesucht ist nun eine Permutation π von $\{1, 2, .., n\}$, die folgenden Ausdruck minimiert:

$$\mathsf{cost}(\pi) := \sum_{i=1}^{n-1} c_{\pi(i),\pi(i+1)} + c_{\pi(n),\pi(1)}$$

Beispiel 1.

$$C = \begin{pmatrix} 0 & 212 & 171 & 203 \\ 212 & 0 & 186 & 247 \\ 171 & 186 & 0 & 139 \\ 203 & 247 & 139 & 0 \end{pmatrix}$$

Die Auswertung der Kosten einiger Rundreisen ergibt:

$$\mathsf{cost}\binom{1,2,3,4}{1,2,3,4} = 740$$
$$\mathsf{cost}\binom{1,2,3,4}{1,2,4,3} = 769$$
$$\mathsf{cost}\binom{1,2,3,4}{1,3,2,4} = 807$$

Wenn n etwas größer wird, dann läßt sich der naive Ansatz, der im Ausprobieren aller Permutationen besteht, nicht mehr durchführen:

$$20! = 2432902008176640000$$

Ein heutiger Rechner kann die Kosten von 50000 Permutationen von 20 Städten in einer Sekunde auswerten. So ein Rechner wäre $123 \cdot 10^6$ Jahre beschäftigt, bis er alle möglichen Permutationen durchprobiert hat.

In Anwendungen müssen TSPs oder ähnliche Probleme dennoch gelöst werden. Im konkreten Beispiel heißt das, der Vertreter muß sich für eine Rundreise durch die 20 Städte entscheiden, auch wenn er etwas länger fährt als mit einer optimalen Tour. Wir werden gegen Ende der Vorlesung Algorithmen kennenlernen, die in 'kurzer' Zeit 'gute' Touren berechnen.

1.2 Graphen

Definition 1. *Ein* Graph $G = (V, E)$ *besteht aus einer Menge* $V = \{v_1, v_2, \ldots, v_n\}$ *von* Knoten *und einer Menge* $E \subseteq \binom{V}{2}$ *von* Kanten.*

*Die Wahl der Buchstaben V und E erklärt sich durch die englischen Begriffe *vertex* und *edge*.

Es ist oft nützlich, Graphen durch Zeichnungen zu veranschaulichen. Das folgende Beispiel deutet an, daß Zeichnungen auch irreleiten können.

Beispiel 2.

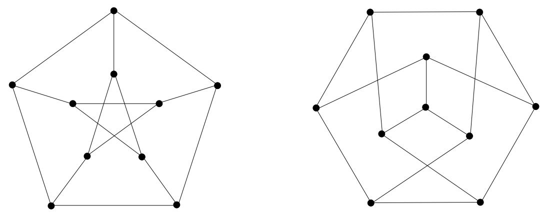


Abbildung 1. Zwei Zeichnungen des berühmten Petersen Graphen

Einige spezielle Graphen

Ein Graph, in dem jedes Paar von Knoten durch eine Kante verbunden ist, heißt *vollständiger Graph.* Den vollständigen Graphen auf n Knoten bezeichnen wir mit K_n. Der vollständige Graph hat $\binom{n}{2} = \frac{1}{2}n(n-1)$ Kanten.

Der *vollständig bipartite Graph* $K_{m,n}$ hat Knotenmenge $V = M \cup N$ mit $|M| = m$, $|N| = n$ und alle möglichen Kanten zwischen Knoten in M und N, d.h. $E = M \times N$. dieser Graph hat $m + n$ Knoten und $m \cdot n$ Kanten.

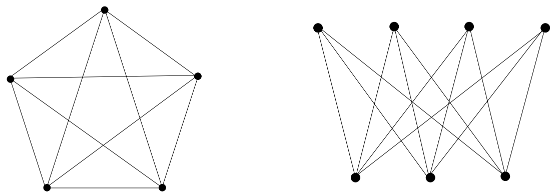


Abbildung 2. Die Graphen K_5, $K_{4,3}$

Skelettgraphen von Polytopen haben die Ecken eines Polytops als Knoten und die Kanten als Kanten. Besonders interessante Beispiele für Skelettgraphen sind die Graphen der Platonischen Körper: Tetraeder, Würfel, Oktaeder, Dodekaeder und Ikosaeder.

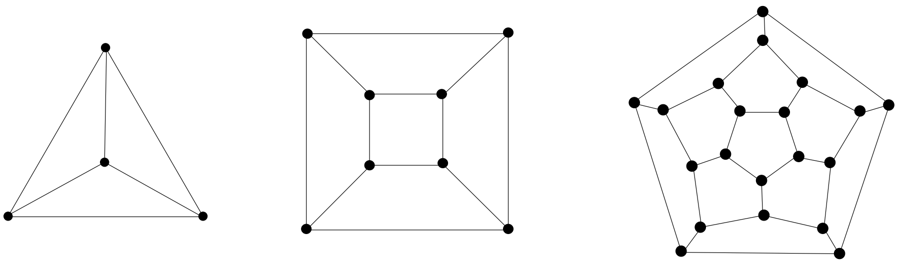


Abbildung 3. Die Skelettgraphen von Tetraeder, Würfel und Dodekaeder

Der n dimensionale *Hyperwürfel* Q_n hat als Knoten alle möglichen 0-1 Vektoren der Länge n, das sind 2^n Stück. Zwei Knoten sind durch ein Kante verbunden, wenn sie sich in genau einer Koordinate unterscheiden. Der gewöhnliche Würfel ist Q_3, auf dem Titelbild befindet sich eine Zeichnung von Q_5.

Definition 2. *Der* Grad $d(v)$ *eines Knotens* $v \in V$ *in* $G = (V, E)$ *ist die Anzahl der Kanten, die* v *enthalten, man sagt auch die zu* v inzident *sind).*

Satz 1. *Für jeden Graphen* $G = (V, E)$ *gilt:* $\sum_{v \in V} d(v) = 2|E|$.

Beweis. Man wendet das *Prinzip des doppelten Abzählens* auf die Paare (v, e) mit $v \in e$ an. Zählt man nach Knoten geordet, so erhält man $\sum d(v)$. Zählt man nach Kanten geordnet erhält man $2|E|$. □

Folgerung 2. *Jeder Graph hat eine gerade Anzahl Knoten ungeraden Grades.*

Die *Inzidenzmatrix* $M(G)$ macht das Prinzip des doppelten Abzählens anschaulich. Man zählt die Einsen der Matrix einmal Zeilen- und einmal Spaltenweise.

Beispiel 3. $G = (V, E)$ mit $V = \{v_1, v_2, v_3, v_4, v_5\}$ und Kanten $e_1 = \{v_1, v_2\}$, $e_2 = \{v_1, v_5\}$, $e_3 = \{v_2, v_3\}$, $e_4 = \{v_2, v_4\}$, $e_5 = \{v_3, v_4\}$, $e_6 = \{v_3, v_5\}$, $e_7 = \{v_4, v_5\}$.

$$M(G) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Ein Graph heißt r-*regulär* wenn alle Knoten den gleichen Grad r haben. Vollständige Graphen, Hyperwürfel und die Skelettgraphen der Platonischen Körper sind regulär.

Übung 1.

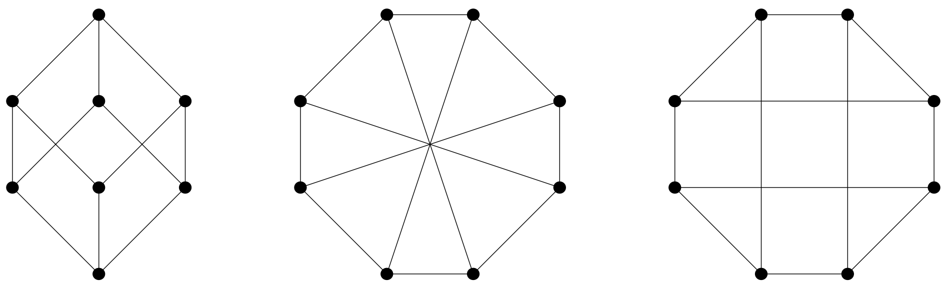
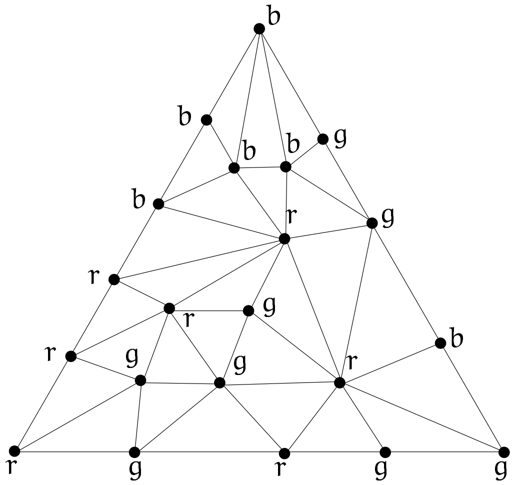


Abbildung 4. Drei 3-reguläre Graphen mit 8 Knoten

Bestimme 4 verschiedene 3-reguläre Graphen mit 8 Knoten.

Übung 2. Belege die Würfelkanten so mit den Zahlen 1 bis 12 (alle verwenden), daß die Summe der Zahlen, die bei mit v inzidenten Kanten stehen, für jeden Knoten gleich ist.

Projektaufgabe.
Ein großes Dreieck D habe eine rote, eine grüne und eine blaue Ecke. Nun sei D in kleinere Dreiecke zerlegt und die Knoten dieser Zerlegung seien rot, grün oder blau gefärbt, so daß auf jeder der Seiten des großen Dreiecks nur zwei Farben auftreten. Zeige: es gibt ein kleines Dreieck mit drei verschieden gefärbten Ecken.



1.3 Wege und Bäume

Definition 3. *Ein* Weg *in* G *ist eine Folge* $v_0v_1\dots v_l$ *von Knoten mit* $v_{i-1}v_i \in E$ *für* $i = 1,..,l$. *Der Weg führt von* v_0 *nach* v_l, *seine* Länge *ist* l. *Sind alle* v_i *verschieden, so sprechen wir von einem* einfachen Weg *oder auch einem* Pfad. *Wenn* $v_0 = v_l$, *dann ist der Weg ein* geschlossener Weg. *Sind alle übrigen Knoten verschieden, dann ist er ein* Kreis.

Definition 4. G *heißt* zusammenhängend, *wenn je zwei Knoten durch einen Weg verbunden sind. Eine Kante* e *in einem zusammenhängenden Graphen* $G = (V, E)$ *heißt* Brücke, *wenn* $G' = (V, E\backslash e)$ *nicht zusammenhängend ist.*

Beispiel 4.

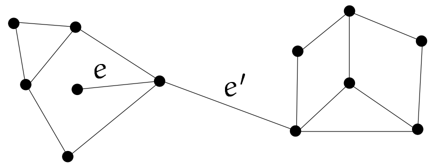


Abbildung 5. Ein Graph mit Brücken e und e'.

Lemma 3–7 ist eine Zusammenstellung von elementaren Tatsachen. Die Beweise sind in dieser Ausarbeitung ausgelassen.

Lemma 3. *In einem zusammenhängenden Graphen sind je zwei Knoten durch einen Pfad verbunden.*

Lemma 4. *Jeder geschlossene Weg ungerader Länge enthält einen ungeraden Kreis.*

Lemma 5. *Sei* G *zusammenhängend. Eine Kante* e *ist eine Brücke von* G *genau wenn* e *in keinem Kreis von* G *vorkommt.*

Definition 5. *Ein Graph* G, *der keine Kreise besitzt, heißt* Wald. *Ein zusammenhängender Wald ist ein* Baum.

Beispiel 5.

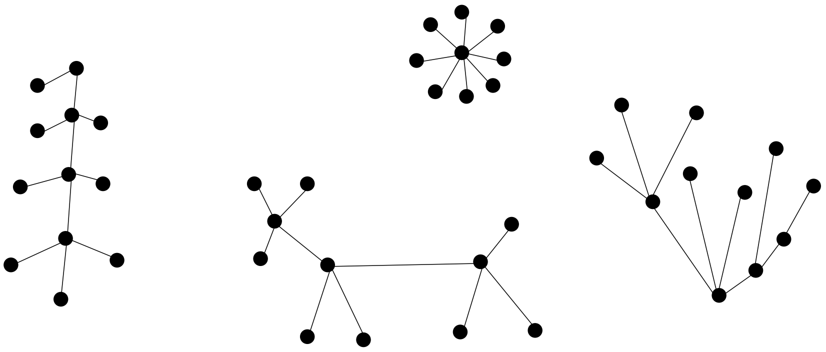


Abbildung 6. Ein aus vier Bäumen bestehender Wald.

Lemma 6. *Jeder Baum mit zumindest zwei Knoten besitzt einen Knoten vom Grad* 1.

Lemma 7. *Sei* G *zusammenhängend.* G *ist ein Baum genau wenn jede Kante von* G *eine Brücke ist.*

Satz 8. *Für* $G = (V, E)$ *ist äquivalent*

(1) G *ist ein Baum*

(2) G *ist kreisfrei und* $|E| = |V| - 1$

(3) G *ist zusammenhängend und* $|E| = |V| - 1$

Beweis. Wir zeigen, daß je zwei der drei Eigenschaften kreisfrei, zusammenhängend und $|E| = |V| - 1$ die dritte zur Folge haben. Daraus folgt die behauptete Äquivalenz unmittelbar.

(Zusammenhängend und kreisfrei $\Longrightarrow$ $|V| - 1$ Kanten)

Sei v ein Knoten von Grad 1 in G (existiert wegen Lemma 6). Entfernen wir v aus G, so bleibt ein Baum $G' = (V', E')$. Offenbar gilt $|V'| - |E'| = |V| - |E|$. Wir entfernen weiter Knoten vom Grad 1, bis nur noch eine Kante übrig ist, also ist $|V| - |E| = 1$.

(Zusammenhängend und $|V| - 1$ Kanten $\Longrightarrow$ kreisfrei)

Entferne Kanten aus G, die in Kreisen vorkommen, bis der erzeugte Graph G' kreisfrei ist. Da wir keine Brücke entfernt haben (Lemma 5) ist G' zusammenhängend. Wegen Teil 1 hat G' genau $|V| - 1$ Kanten, also ist $G = G'$.

(Kreisfrei und $|V| - 1$ Kanten $\Longrightarrow$ zusammenhängend)

Seien $G_1, \dots, G_k$ die Zusammenhangskomponenten von G. Wegen Teil 1 hat jedes G_i genau $|V(G_i)| - 1$ Kanten, zusammen haben sie $|V| - k$ Kanten, also ist $k = 1$ und G ist zusammenhängend. $\square$

Übung 3. Ein Baum mit zumindest zwei Knoten besitzt mindestens zwei Knoten vom Grad 1.

Eine *zulässige Färbung* von G ist eine Färbung (Benennung, Numerierung) der Knoten, so daß benachbarte Knoten verschiedene Farben erhalten. Wir sagen, G ist r-*färbbar*, wenn es eine zulässige Färbung von G mit r Farben gibt.

Übung 4. Kreise gerader Länge, vollständig bipartite Graphen, Hyperwürfel und Bäume sind alle 2-färbbar.

Übung 5. Ein Graph G ist genau dann 2-färbbar, wenn es keinen Kreis ungerader Länge in G gibt.

2 Euler–Wege

In diesem Abschnitt beschäftigen wir uns mit Wegen , die jede Kante eines gegebenen Graphen genau einmal benutzen. Solche Wege heißen *Euler–Wege*. Der Name geht auf Eulers Lösung des Königsberger Brückenproblems zurück. Eine populäre Variante ist “Das Haus vom Nikolaus”.

Beispiel 6.

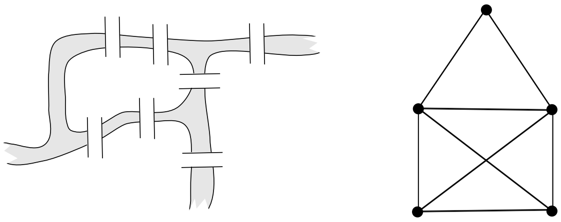


Abbildung 7. Zwei berühmte Euler–Probleme.

Beobachtung. Wenn G einen geschlossenen Euler–Weg (kurz *Euler–Kreis* genannt) besitzt, dann sind alle Knotengrade gerade. Hat G einen offenen Euler–Weg, dann besitzt G genau zwei Knoten ungeraden Grades.

Folgerung. Das Königsberger Brückenproblem ist nicht lösbar.

Satz 9. *Ein Graph G besitzt einen Euler–Kreis $\Longleftrightarrow$ G ist zusammenhängend und alle Grade sind gerade.*

Beweis. Es bleibt nur “$\Leftarrow$” zu zeigen. Da alle Grade ≥ 2 sind, können wir bei Start in v_0 immer auf neuen Kanten weitergehen, bis wir wieder in v_0 ankommen. Sei C der so erzeugte geschlossene Weg. Sei $G' = G \backslash C$ und seien $G_1 \ldots G_z$ die zusammenhängenden Komponenten von G'. Jedes G_i ist zusammenhängend und alle Grade sind gerade. Durch Induktion nach der Kantenzahl schließen wir, daß jedes G_i einen Euler–Kreis besitzt. Alle diese Kreise können mit C verknüpft werden. So erhält man einen Euler–Kreis für G. $\square$

Folgerung. Ein Graph G enthält einen Euler–Weg $\Longleftrightarrow$ G ist zusammenhängend und hat genau 2 Knoten ungeraden Grades.

Beweis. Wenn G einen Euler–Weg enthält, dann muß G sicherlich zusammenhängend sein und genau zwei Knoten ungeraden Grades besitzen. Seien u, v die beiden Knoten ungeraden Grades. Betrachte $G' = G \cup \{uv\}$. Wegen Satz 9 G' besitzt Euler Kreis C. Nach Entfernen von uv aus C wird daraus ein Euler Weg in G. □

2.1 Ein Algorithmus für Euler–Kreise

Zwar ist es möglich, den Beweis von Satz 9 zu einem Algorithmus für Euler–Kreise umzuformulieren. Der folgende Algoritmus von Fleury ist jedoch interessanter:

- Wähle beliebigen Startknoten v_0.
- Sind $v_0, v_1, \ldots, v_{i-1}$ schon gegeben, so wähle als v_i einen Nachbarn von v_{i-1} in $G_i = G \backslash \{e_1 \ldots e_{i-1}\}$, so daß (wenn möglich) $e_i = v_{i-1}v_i$ keine Brücke in G_i ist.

Satz 10. *Ist G eulersch, dann berechnet der Algorithmus von Fleury einen Euler–Kreis.*

2.2 Anwendungen von eulerschen Wegen

1. Chinese Postman Problem

Ein Postbote muß, ausgehend vom Postamt, alle Straßen seines Bezirkes durchlaufen und wieder zum Postamt kommen. Falls der Graph (Plan des Bezirks) keinen Euler–Weg besitzt, muß er Kanten mehrfach durchlaufen. Das Ziel des Postboten ist es, einen möglichst kurzen Weg zu laufen.

Wenn alle Knoten gerade sind, dann ist G eulersch und die Länge des Weges ist die Summe der Kantenlängen. Gesucht ist nun eine längenminimale Kantenmenge F, so daß die Vereinigung der Kantenmengen $G + F$ eulersch ist. (Zwar hat $G + F$ doppelte Kanten –ist ein *Multigraph*–, aber Satz 9 gilt auch für Multigraphen).

Das folgende Verfahren löst das Problem

(1) Bestimme die Menge U der Knoten ungeraden Grades, $|U| = 2m$.

(2) Für je zwei Knoten $u, v, \in U$ berechne die Länge $\delta(u, v)$ eines kürzesten Weges von u nach v (*kürzeste Wege Problem*).

(3) Ein *Matching* M auf U ist eine Menge von m Paaren, die jedes $u \in U$ genau einmal erfaßt. Bestimme ein längenminimales Matching, d.h. ein M mit $\delta(M)$ minimal, wobei
$$\delta(M) = \sum_{uv \in M} \delta(u, v)$$

(4) Ist M_0 ein minimales Matching, so verdoppeln wir für alle $uv \in M$ einen kürzesten Weg von u nach v.

(5) Berechne einen Euler Kreis in dem so entstandenen eulerschen Multigraphen.

Drei der fünf Teilprobleme dieser Lösung sind algorithmisch interessant. Wichtig ist, daß sich alle Teilprobleme effizient (d.h. auch für große Eingaben, z.B. $|V| = 100000$) lösen lassen.

Beispiel 7. Wendet man den Algorithmus für das Chinese Postman Problem auf den in der Abbildung dargestellten Graphen an, so erhält man eine Tour der Länge $\delta(G)+10 = 62$.

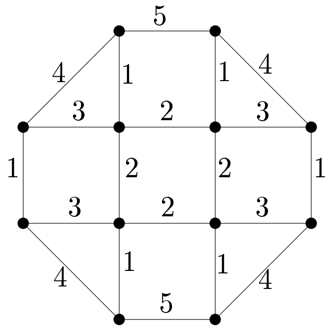


Abbildung 8. Ein Graph mit längengewichteten Kanten.

2. DeBruijn Kreise

Es gibt 2^n binäre Vektoren der Länge n. Gibt es eine zyklische (kreisförmige) Anordnung von 2^n binären Ziffern, so daß jeder der Vektoren als Abschnitt der Länge n auftritt? Für $n = 2, 3, 4$ sind solche zyklischen Anordnungen in der folgenden Abbildung dargestellt.

Beispiel 8.

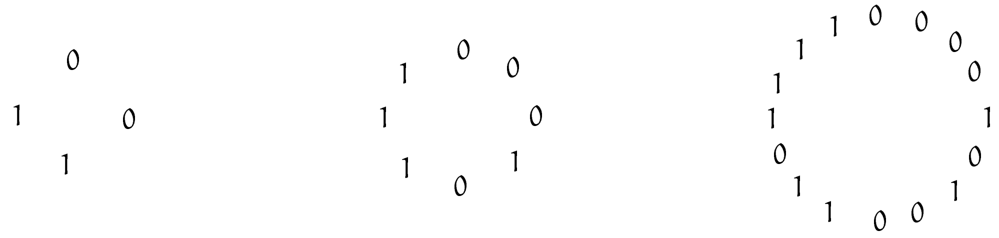


Abbildung 9. Drei DeBruijn Kreise.

Dieses Problem läßt sich auf ein gerichtetes Euler Kreis Problem zurückführen: Sei D_n ein gerichteter Graph, dessen Knoten die binären $n-1$ Vektoren sind. Wir verbinden a und b durch eine Kante $a \to b$, wenn die hinteren $n-2$ Ziffern von a mit den vorderen von b übereinstimmen. Die Kante $a \to b$ wird mit der letzten Ziffer von b gezeichnet. Jeder Knoten a des Graphen besitzt zwei ausgehende und zwei eingehende Kanten. Aus der gerichteten Version von Satz 9 folgt, daß D_n einen Euler Kreis besitzt. Die Ziffern auf den Kanten eines Euler Kreises ergeben eine zyklische Anordnung mit den gewünschten Eigenschaften.

Beispiel 9.

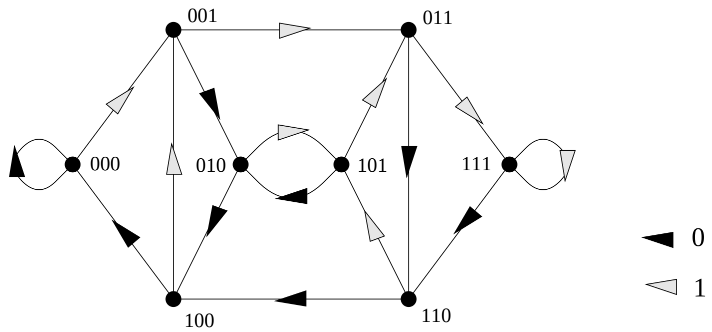


Abbildung 10. Der deBruijn Graph D_4.

Übung 6. Sei G ein eulerscher Graph mit maximalem Grad $\Delta > 2$. Zeige: Die Anzahl der Euler Kreise von G ist gerade.

Übung 7.

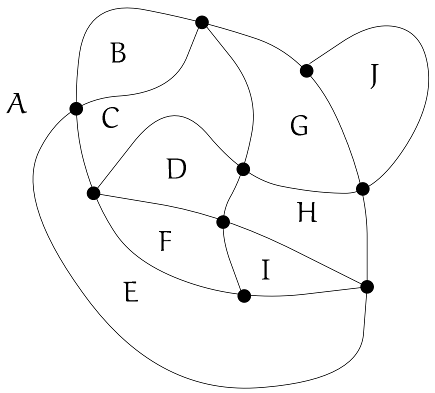


Abbildung 11. Eine Landkarte mit Ländern $A, \ldots, J$ und einem Gerüstgraphen mit 9 Knoten.

Eine Landkarte ist genau dann 2-färbbar, wenn ihr Gerüstgraph (=Grenzengraph) eulersch ist.

3 Hamilton–Kreise

In diesem Abschnitt wollen wir Wege und Kreise in Graphen betrachten, die jeden Knoten genau einmal besuchen. Solche Pfade und Kreise sollen *hamiltonsch* heißen. Ein Graph heißt *hamiltonscher Graph*, wenn er einen Hamilton–Kreis besitzt.

- K_n ist hamiltonsch für $n \geq 3$.
- $K_{m,n}$ ist nur für $n = m \geq 2$ hamiltonsch.

Es ist keine "gute" Charakterisierung von hamiltonschen Graphen bekannt. Daher sucht man nach notwendigen und hinreichenden Bedingungen.

Beispiel 10. Sir Hamilton beschrieb 1856 ein Spiel, bei dem es darum ging, einen Weg $v_1 \dots v_5$ im Dodekaedergraphen zu einer Rundreise zu ergänzen.

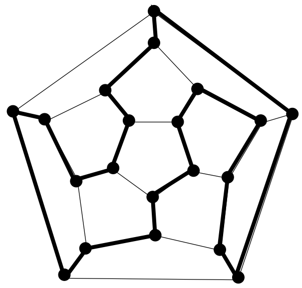


Abbildung 12. Ein Hamilton–Pfad im Dodekaedergraphen.

Satz 11 (Notwendige Bedingung). *Ein hamiltonscher Graph* $G = (V, E)$ *erfüllt für alle Knotenmengen* $A \subseteq V$*:*

$$\textit{Anzahl Komponenten von } G - A \leq |A|.$$

Dabei ist $G - A$ *der Graph, der durch Entfernen der Knoten in* A *und ihren inzidenten Kanten aus* G *entsteht.*

Beweis. Wir geben dem Kreis eine Orientierung und betrachten Kanten $x \to y$ mit $x \notin A$ und $y \in A$. Jede Komponente von $G - A$ wird über so eine Kante verlassen und die Zielknoten dieser Kanten in A sind alle verschieden. □

Beispiel 11. Die Abbildung zeigt den kleinsten polyedrischen Graphen der keinen Hamilton–Kreis, aber einen Hamilton–Weg besitzt. (Die Menge der grauen Knoten verletzt die bedingung aus dem Satz).

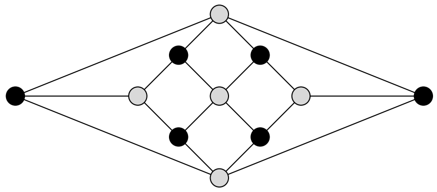


Abbildung 13. Ein nicht hamiltonscher Graph

Der Petersen Graph (Abbildung 1) zeigt, daß die in Satz 11 gegebene Bedingung nicht hinreichend ist. Jede Teilmenge A seiner Knoten erfüllt die Bedingung, aber der Graph ist dennoch nicht hamiltonsch.

Satz 12 (Hinreichende Bedingung). *Ist* $G = (V, E)$ *ein Graph mit*

$$d(u) + d(v) \geq |V| \quad \textit{für alle} \quad uv \notin E$$

dann ist G *hamiltonsch.*

Beweis. (indirekt) Unter allen Gegenbeispielen mit n Knoten sei G eines mit einer maximalen Anzahl von Kanten. Es folgt $G + uv$ ist hamiltonsch für alle $uv \notin E$, also besitzt G einen hamiltonschen Weg $u = v_1 \dots v_n = v$, der in u beginnt und in v endet. Wenn ein Nachbar von v unmittelbar vor einem Nachbarn von u liegt, d.h. uv_{i+1} und $v_i v$ sind beides Kanten, dann ist $v_1 \dots v_i v_n v_{n-1} \dots v_{i+1}$ ein hamiltonscher Kreis für G.

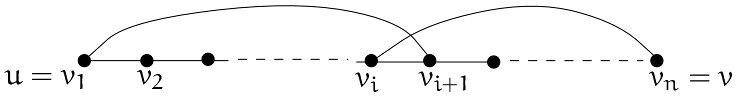


Abbildung 14. Der Austauschschritt.

Wir zeigen nun, daß so ein Paar (v_i, v_{i+1}) existiert. Sei $S = \{i : uv_{i+1} \in E\}$ und $T = \{i : v_i v \in E\}$. Es gilt

$$|S \cup T| + |S \cap T| = |S| + |T| = d(u) + d(v) \geq n$$

Da weder S noch T den Index n enthalten, muß $|S \cup T| < n$ und also $|S \cap T| \geq 1$ sein. Widerspruch. $\square$

Übung 8. Betrachte einen $3 \times 3 \times 3$ Würfel aus 27 Käsestückchen. Eine Maus sucht einen Weg von einem Eckstück über alle anderen Stücke, der im Mittelstück endet. Ist dies möglich? Gleiche Frage für $5 \times 5 \times 5$ und allgemeines ungerades n.

Übung 9. Eine Orientierung eines vollständigen Graphen heißt *Turnier**. Zeige, daß jedes Turnier einen gerichteten Hamilton–Pfad enthält.

4 Kombinatorische Optimierung

Ein Optimierungsproblem ist gegeben durch eine Menge X und eine Funktion $f : X \to \mathbb{R}$. Gesucht ist eine Lösung $x \in X$ die $f(x)$, maximiert oder minimiert. In kombinatorischen Optimierungsproblemen ist X eine kombinatorische Menge.
Beispiele für kombinatorische X:

- Menge der Hamilton Kreise in G (TSP).
- Menge der Pfade von u nach v in G (kürzeste Wege Problem).
- Gegeben n Pakete und ein Rucksack. $X = \{$ Menge von Paketen, die im Rucksack Platz haben $\}$ (Rucksackproblem).

In der kombinatorischen Optimierung beschäftigt man sich mit Methoden (Algorithmen) zur Lösung solcher Probleme. Ein Beispiel haben wir schon im Kapitel über Euler–Wege gesehen, das Chinese Postman Problem. Ein 'guter' Algorithmus für ein kombinatorisches Optimierungsproblem hat einige Forderungen zu erfüllen:

(1) Er *muß korrekt* sein (d.h. die Ausgabe x muß in X liegen)

(2) Er *soll optimieren* (d.h. $f(x)$ soll maximal oder minimal sein)

*Der Name wird damit begründet, daß ein Turnier-Graph ein mögliches Ergebnis für einen Wettkampf, in dem jeder gegen jeden einmal spielt und unentschieden unmöglich sind, ist

(3) Er *soll effizient* sein (d.h. die Laufzeit soll als Funktion in der Eingabegröße n polynomial sein, also z.B. $n, n^2, 8n^3 - n$ aber *nicht* $2^n, n!$)

Wenn sich die berühmte $\mathsf{P} \neq \mathsf{NP}$ Vermutung der Komplexitätstheorie als richtig erweist, dann kann es für viele wichtige Probleme (wie z.B. das TSP), keinen Algorithmus geben, der alle drei Forderungen erfüllt. Da auf "Schnelligkeit" nicht verzichtet werden kann, schwächt man Forderung (2) ab und verlangt nur noch eine *Approximation* des Optimums. Wir sagen, x ist eine α-*approximierende Lösung* eines Minimierungssproblems, wenn

$$f(x) \leq \alpha \cdot f(x_{\text{opt}}).$$

Der Faktor α beschreibt, um wieviel die Kosten $f(x)$ der gefundenen Lösung x die minimalen Kosten $f(x_{\text{opt}})$ übersteigen können. Gesucht werden effiziente Algorithmen, die einen möglichst kleinen Approximationsfaktor $\alpha > 1$ gewährleisten.

Ist für einen Algorithmus keine Approximationsgarantie bekannt, aber die Intuition sagt, daß die Lösung brauchbar ist (d.h. nicht zu schlecht) sein wird, so spricht man von einer *Heuristik.*

Bevor wir uns mit Heuristiken und approximierenden Algorithmen für das TSP befassen, wollen wir ein paar Beispiele für Algorithmen sammeln.

- **Euklidischer Algorithmus**
  Gegeben: $a_1, a_z \in \mathbb{Z}$ Gesucht: $\mathrm{ggt}(a_1, a_z)$

- **Weitere zahlentheoretische Probleme**
  Polynom-Multiplikation/Division; Faktorisierung.

- **Such- und Sortierprobleme**
  Gegeben: $A = \{a_1, a_2 \ldots, a_n\}$ aus total geordneter Grundmenge.

  Sortierproblem: Gesucht Umsortierung $a_1^*, a_2^* \ldots a_n^*$ von A, so daß $a_1^* < a_2^* < \ldots < a_n^*$

  Median: Gesucht ein Element $x \in A$, so daß (fast) gleichviele Elemente aus A kleiner/größer als x sind.

  Maximum: Gesucht $x \in A$, so daß alle Elemente aus A kleiner x sind.

Ein Algorithmus für **Maximum**

```
x ← a_1
for i = 2 to n
    x ← max(x, a_i)
return x
```

Dieses Verfahren benötigt $n - 1$ Vergleiche. Kann das Maximum x mit weniger als $n - 1$ Vergleichen ermittelt werden? Nein, um x einwandfrei festzulegen, muß es für jedes $a \neq x$ ein b geben, so daß a und b verglichen wurden und $b > a$.

4.1 Algorithmische Probleme aus der Graphentheorie

Kürzeste Wege.

Gegeben: $G = (V, E)$, eine Längenfunktion $\omega : E \to \mathbb{R}^+$ und zwei Knoten $u, v \in V$.
Gesucht: ein kürzester Weg von u nach v.

Matching.

Gegeben: K_{2n} und Kostenfunktion $\omega : E \to \mathbb{R}^+$.
Gesucht: Billigstes vollständiges Matching, d.h., billigste Kantenmenge $M \subset E$, in der jeder Knoten Grad 1 hat.

Minimum-Spanning-Tree (MST)

Gegeben: $G = (V, E)$ zusammenhängend mit Kostenfunktion $\omega : E \to \mathbb{R}^+$.
Gesucht: Billigster zusammenhängender Subgraph $T = (V, F)$ d.h. $F \subseteq E$ und

$$\sum_{e \in F} \omega(e) \leq \sum_{e \in F'} \omega(e)$$

für alle zusammenhängenden Subgraphen $T' = (V, F')$ von G.
Beobachtung: T ist ein Baum.

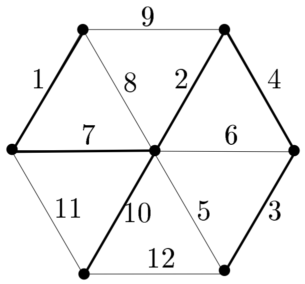


Abbildung 15. Ein Graph mit einem MST

4.2 MST und der Greedy-Algorithmus

Mit dem Namen “Greedy*-Algorithmus” bezeichnet man Algorithmen, die Entscheidungen so treffen, daß der augenblickliche Gewinn möglichst groß ist. Greedy-Algorithmen können für gewisse Probleme (z.B. TSP) beliebig schlechte Ergebnisse liefern. In anderen Fällen sind sie, wie wir gleich sehen werden, sogar optimal. Ein Greedy Algorithmus für MST etwa kann nach dem folgenden Prinzip vorgehen:

Nimm immer die billigste Kante, die möglich ist, d.h. die keinen Kreis erzeugt.

Der Algorithmus sieht lässt sich in einer an Programmiersprachen angelehnten Notation (Pseudocode) auch so aufschreiben:

* *greedy* = gierig, gefräßig

Sortiere E, so daß
$\omega(e_1) \leq \omega(e_2) \leq \ldots \leq \omega(e_m)$
$F \leftarrow \emptyset$
for $i = 1$ to m
    if $F \cup e_i$ kreisfrei, then
        $F \leftarrow F \cup e_i$
return F

Satz 13. *Der Algorithmus berechnet einen MST* $T = (V, F)$ *in jedem zusammenhängenden gewichteten Graphen* $G = (V, E)$.

Beweis. Offenbar ist T zusammenhängend und kreisfrei, also ein aufspannender Baum für G. Sei T^* ein MST für G, der möglichst viele Kanten mit T gemeinsam hat. Angenommen, $T \neq T^*$. Sei $e = e_i$ die erste Kante, die in T, aber nicht in T^* ist. In $T^* + e$ gibt es einen Kreis und dieser Kreis enthält eine Kante $e' = e_j$, die nicht in T ist. Aus der Wahl von e folgt $i < j$, also $\omega(e_i) \leq \omega(e_j)$. Nun ist $T^* + e - e'$ ein Baum, dessen Gewicht um $\omega(e_j) - \omega(e_i)$ kleiner ist als das von T^*. Also ist $T^* + e - e'$ ebenfalls ein MST, hat aber mehr Kanten mit T gemeinsam als T^*. Widerspruch. □

Übung 10. Ein *Minimax Spanning Tree* ist ein aufspannender Baum T in einem zusammenhängenden kantengewichteten Graphen, dessen maximales Kantengewicht minimal ist. Finde einen Algorithmus für das Minimax Spanning Tree Problem.

Übung 11. Sechs Personen wissen jeder eine Neuigkeit. Sie führen eine Reihe von Telefonaten durch und bei jedem Gespräch erzählen sich die Teilnehmer alle Neuigkeiten, die sie bis dahin erfahren haben. Wie viele Telefonate sind nötig, bis alle alles wissen.

Übung 12. Zeige, daß folgender Algorithmus (Prims Alg.) ebenfalls einen MST berechnet

$R \leftarrow \{r\}, \quad S \leftarrow V \setminus \{r\}, \quad F \leftarrow \emptyset$
while $S \neq \emptyset$
    bestimme leichteste Kante $e = (r', s') \in R \times S$
    $R \leftarrow R \cup \{s'\}, \quad S \leftarrow S \setminus \{s'\}, \quad F \leftarrow F \cup e$
return F

5 Heuristiken und Approximationen für TSP

Zur Erinnerung: Ein TSP Problem ist durch eine $n \times n$ Kostenmatrix $C = (c_{ij})$ gegeben. Ist C symmetrisch (d.h. $c_{ij} = c_{ji}$), dann läßt sich das TSP Problem auffassen als Problem, einen minimal gewichteten aufspannenden Kreis im vollständigen Graphen K_n mit Kantengewichten $\omega(v_i, v_j) = c_{ij}$ zu bestimmen.

5.1 Greedy-Methoden

NÄCHSTER NACHBAR (NN)
Starte in einem beliebigem Knoten $s \in V$. Verlängere den Weg schrittweise, um billigste Kante zwischen dem Endknoten des Weges und einem noch nicht besuchten Nachbarn.

Beispiel 12. Sei $c_{ii+1} = 0$ für $i = 1, \ldots, n-1$, $c_{1n} = B > 2$ und $c_{ij} = 1$ für alle übrigen Paare i, j. Bei Start in v_1 findet NN eine Tour mit Kosten B, während eine optimale Tour nur 2 kostet.

BILLIGSTES EINFÜGEN (NI NEAREST INSERTION)
Starte in beliebigem Knoten $s \in V$. Vergrößere den Kreis K schrittweise: Finde $u \notin K$ und $v \in K$ so daß $c(u, v)$ minimal ist. Füge v nach u in den Kreis ein (siehe Abbildung).

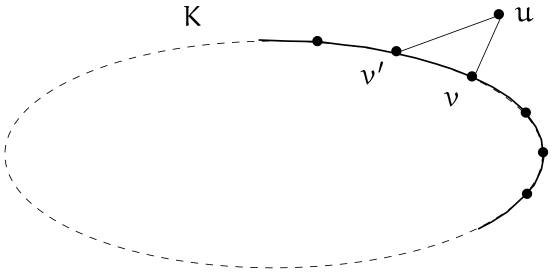


Abbildung 16. Kreis K wird bei v um neuen Knoten u verlängert.

Satz 14 (Sahni-Gonzales 1976). *Wenn $P \neq NP$, dann gibt es kein $\alpha > 2$ zu dem ein effizienter α-approximierender Algorithmus für das TSP Problem existiert.*

Bemerkung. Im allgemeinen ist NI besser als NN. Aber im obigen Beispiel erzeugt NI bei Start in v_1 dieselbe schlechte Lösung.

5.2 Euklidisches TSP

Eine Kostenmatrix C heißt d-dimensional realisierbar, wenn es Punkte $p_1, p_2, \ldots, p_n$ im $\mathbb{R}^d$ gibt, so daß $\mathrm{dist}(p_i p_j) = c_{ij}$, wobei dist der euklidische Abstand $\|p_i - p_j\|$ ist, d.h., die Länge der Strecke mit Endpunkten p_i und p_j. Das Problem mit dem sich der Handlungsreisende aus der ersten Vorlesung quält ist 2-dimensional. Können wir solche Probleme leichter lösen?

Satz 15 (Arora 1996). *Für jedes $\epsilon > 0$ gibt es effiziente $(1 + \epsilon)$-approximierende Algorithmen für euklidisches TSP in jeder Dimension.*

Wir wollen uns hier mit einer Eigenschaft befassen, die weniger restriktiv ist als "euklidisch". Eine Kostenmatrix erfüllt die Dreiecksungleichung, wenn für alle i, j, k gilt

$$c_{ik} \leq c_{ij} + c_{jk}$$

Satz 16. *Billigstes Einfügen (NI) ist ein 2-approximierender Algorithmus für TSP Probleme, die die Dreiecksungleichung erfüllen.*

Die wesentliche Beobachtung ist

> Die Kanten (u, v) mit $u \notin K$, $v \in K$ und $c(u, v)$ minimal, die NI benutzt, sind genau die Kanten, die Prims Algorithmus in den MST einfügt.

Ein Hamilton–Kreis H, aus dem wir eine Kante löschen, bleibt ein aufspannender Weg, also ein spezieller aufspannender Baum. Daraus folgt für die optimalen Kosten von MST und TSP

$$\mathsf{cost}(\mathsf{MST}) \leq \mathsf{cost}(\mathsf{TSP})$$

Nun beweisen wir die 2-Approximation durch Induktion: Für die Knotenmenge $V_1 = V(K)$ setzen wir voraus:

$$\mathsf{cost}(K) \leq 2 \cdot \mathsf{cost}(\mathsf{MST}(V_1))$$

Sei K^* der Kreis der durch den Einbau von u in K entsteht und $V_2 = V_1 \cup \{u\}$. Nun gilt

$$\begin{aligned}
\mathsf{cost}(K^*) &= \mathsf{cost}(K) + c(v, u) + c(u, v') - c(v, v') \\
&\leq \mathsf{cost}(K) + 2c(v, u) \\
&\leq 2\mathsf{cost}(\mathsf{MST}(V_1)) + 2c(v, u) = 2\mathsf{cost}(\mathsf{MST}(V_2)) \\
&\leq 2\mathsf{cost}(\mathsf{TSP})
\end{aligned}$$

□

Ähnlich wie im Beweis des vorigen Satzes überlegt man sich

Lemma 17. *Ist C eine Kostenmatrix, die die Dreiecksgleichung erfüllt, und H ein eulerscher aufspannender Subgraph von K_n, dann gilt:* $\mathsf{cost}(\mathsf{TSP}) \leq \mathsf{cost}(H)$

Das folgende zweistufige Verfahren zur Konstruktion eines billigen eulerschen aufspannenden Subgraphen wurde von Christofides vorgeschlagen.

(1) Berechne ein MST T und die Menge U der Knoten ungeraden Grades von T.

(2) Berechne ein kostenminimales Matching M von U.

Wir vergleichen nun die Kosten für das Matching mit den Kosten einer optimalen TSP Tour K. Sei $u_1 u_2 \ldots u_{2k}$ die Reihenfolge, in der die Knoten aus U in K auftreten. Wir betrachten zwei Matchings

$$M_1 = \{(u_1 u_2), (u_3 u_4), \ldots (u_{2k-1}, u_{2k})\} \quad \text{und} \quad M_2 = \{(u_2, u_3), (u_4, u_5) \ldots (u_{2k}, u_1)\}.$$

Natürlich gilt $\mathsf{cost}(M) \leq \mathsf{cost}(M_1)$ und $\mathsf{cost}(M) \leq \mathsf{cost}(M_2)$. Wegen der Dreiecksungleichung gilt außerdem $\mathsf{cost}(M_1) + \mathsf{cost}(M_2) \leq \mathsf{cost}(K) = \mathsf{cost}(\mathsf{TSP})$. Zusammengefaßt folgt also $\mathsf{cost}(M) \leq \frac{1}{2}\mathsf{cost}(\mathsf{TSP})$.

Betrachten wir nun die Kantenmenge $T + M$. Sie bildet einen aufspannenden eulerschen Subgraphen von G. Das Gewicht dieses Subgraphen ist höchstens $\mathsf{cost}(\mathsf{TSP}) + \frac{1}{2}\mathsf{cost}(\mathsf{TSP})$. Mit Lemma 17 haben wir damit folgenden Satz bewiesen.

Satz 18 (Christofides 1976). *Es gibt einen effizienten 3/2-approximierenden Algorithmus für TSP mit Dreiecksungleichungen.*

Übung 13. Wende Christofides Algorithmus auf die 9 Punkte aus der Abbildung an. Finde ein einfaches Argument dafür, daß das Ergebnis nicht optimal ist.

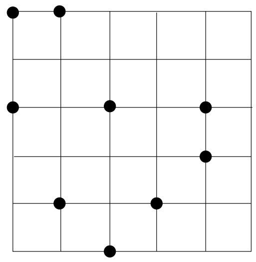


Abbildung 17. 9 ganzzahlige Punkte in $\mathbb{R}^2$.

Bemerkung. Inzwischen ist bekannt, daß man TSP mit Dreiecksungleichung nicht beliebig genau approximieren kann (andernfalls ist $P = NP$). Die untere Schranke ist aber weit von der noch immer besten bekannten Schranke 3/2 für das Problem entfernt.

6 Weitere Übungsaufgaben

Übung 14. Für welche natürlichen Zahlen n gibt es 3-reguläre Graphen mit n Knoten?

Übung 15. In einem Graphen mit n Knoten sind $0, 1, 2, \ldots, n-1$ die möglichen Grade. Zeige: Es gibt keinen Graphen, dessen n Knoten alle n verschiedene Grade haben. Aber es gibt Graphen mit n Knoten und $n-1$ verschiedenen Graden.

Übung 16. Interpretiere die Einträge der Produktmatrix MM^T.

Übung 17. Seien $d_1 \ldots d_n$ natürliche Zahlen. Zeige, daß ein Baum mit Graden $d_1 \ldots d_n$ genau dann existiert wenn $\sum d_i = 2n - 2$.

Übung 18. Sei S eine n elementige Menge, $A_1, A_2, \ldots, A_n$ seien verschiedene Teilmengen von S. Zeige, daß es ein Element $x \in S$ gibt, so daß $A_1 \cup \{x\}, A_2 \cup \{x\}, \ldots, A_n \cup \{x\}$ verschieden sind.

Übung 19. Formuliere und beweise Euler–Sätze für (a) gerichtete Graphen, (b) Multigraphen und (c) gerichtete Multigraphen.

Übung 20. Sei G ein Graph und D eine Kantenorientierung, in der es zwischen je zwei Knoten u, v einen gerichteten $u \to v$ Weg gibt. Zeige: Wenn G einen ungeraden Kreis enthält, dann enthält D einen ungeraden gerichteten Kreis.

Übung 21. Wie viele Euler Kreise besitzt der vollständige Graph K_5?

Übung 22. Ein König in einem Turnier ist ein Knoten u, so daß für alle v ein gerichteter Weg $u \to v$ mit höchstens zwei Kanten existiert. Zeige: Jedes Turnier hat einen König.

Übung 23. Das 8×8 Schachbrett läßt sich mit 32 Dominosteinen überdecken. Entfernt man ein weißes und ein schwarzes Feld, dann gibt es noch immer eine Überdeckung mit Dominosteinen.

Übung 24. Sei $G = (V, E)$ ein Graph mit Längenfunktion $\omega : E \to \mathbb{R}^+$ und $u \in V$. Zeige: Es gibt einen Baum $T_u = (V, F)$ mit $F \subseteq E$, so daß für alle $v \in V$ der Weg von u nach v in T_u ein kürzester Weg von u nach v in G ist.

Übung 25. Betrachte die Probleme kürzeste Wege und MST für den Fall $\omega : E \to \mathbb{R}$ (d.h. die Kantengewichte sind nicht notwendig positiv.)

Übung 26. Entwickle einen Algorithmus für das Problem kürzesten Weg und schätze die Anzahl der benötigten Operationen ab.

Literatur

[1] M. Aigner, *Graphentheorie. Eine Entwicklung aus dem 4-Farben Problem*, Teubner, Stuttgart, 1984.

[2] M. Aigner, *Diskrete Mathematik*, Vieweg, Wiesbaden, 1993.

[3] N. L. Biggs, E. Lloyd, and R. J. Wilson, *Graph theory 1736-1936*, Clarendon Press; Oxford University Press, London, 1976.

[4] R. Diestel, *Graphentheorie*, Springer, Berlin, 1996.

[5] A. Gibbons, *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1985.

[6] D. Jungnickel, *Graphen, Netzwerke und Algorithmen*, B.I.-Wissenschaftsverlag, Mannheim, 1987.

[7] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, John Wiley and Sons, Chichester, 1985.

[8] D. B. West, *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, NJ, 1996.

[9] R. J. Wilson, *Introduction to Graph Theory*, Longman, London, 1975.

Erläuterung zum Titelbild

Gesucht ist die maximale Länge t einer Folge $C_0, C_1, \ldots, C_t$ von Teilmengen von $\{1, .., n\}$ die für alle $0 < i < j$ der Bedingung

$$C_j \not\subseteq C_{i-1} \cup C_i$$

genügen. Dieses Problem hat erstaunlicherweise etwas mit speziellen Hamilton-Pfaden auf dem Hyperwürfel zu tun. Das Titelbild zeigt so einen Pfad und die schwarzen Knoten können als die Teilmengen in einer Folge maximaler Länge interpretiert werden.

Der Zusammenhang zwischen den Knoten des Hyperwürfels Q_n, also 0-1 Vektoren der Länge n und Teilmengen von $\{1, .., n\}$ ist leicht beschrieben: Dem Vektor x ordnen wir die Menge $C_x = \{i : x_i = 1\}$ zu. Diese Abbildung ist offensichtlich eins-zu-eins, also bijektiv.