SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Handelsresande i Sverige

av

Nasrin Naseri

2020 - No K30

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET, 106 91 STOCKHOLM

Handelsresande i Sverige

Nasrin Naseri

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Sven Raum

2020

Abstract

The travelling salesman problem is one of the most known optimization problems in graph theory. The problem is to find the shortest possible route so that you visit each city only once and then return to the starting city. In this paper we will first study graph theory and different types of graphs. Then we will introduce the nearest neighbor algorithm and the greedy algorithm to find good approximate solutions to the travelling salesman problem. Finally, we want to find a good approximation route between the 15 largest cities in Sweden with the help of these algorithms passing each city once, without returning to the starting city.

Keywords: Graph, Weighted graph, Complete graph, Hamiltonian path, Hamiltonian cycle, Travelling salesman problem, Nearest neighbor algorithm, Greedy algorithm.

Innehåll

1 Introduction

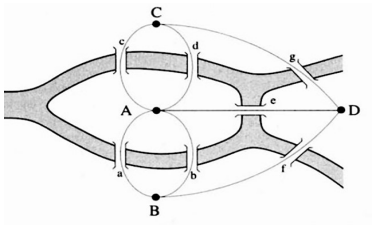
Handelsresandeproblemet (TSP) är ett av de mest kända optimeringsproblemen inom det grafteoritiska området. Problemet är att hitta den kortaste resvägen mellan olika städer. Med andra ord handlar problemet om att hitta en rutt som täcker alla städer så att totalavståndet minimeras. I dagsläget finns det ingen metod som kan ge oss den exakta optimala lösningen i TSP förutom att man provar alla möjliga lösningar och jämför dem. Men lyckligtvis finns det flera metoder som kan hitta en approximativ lösning till TSP. I det här arbetet kommer vi att beskriva två olika algoritmer som ger mycket användbara lösningar till TSP.

För att kunna analysera TSP med hjälp av grafer så kommer första delen av uppsatsen introducera de grundläggande begreppen inom grafteori, olika typer av grafer samt Hamiltonstigar och Hamiltoncykler. Sedan, i andra delen av uppsatsen, kommer vi introducera TSP och visa hur man kan hitta en approximativ lösning till TSP med hjälp av närmaste granne-algoritmen och giriga algoritmen. Slutligen kommer vi att använda oss av de ovannämnda algoritmerna och hitta en approximativ optimal rutt mellan de 15 största städerna i Sverige så att man passerar varje stad endast en gång, utan att återvända till utgångspunkten.

1.1 Grafteori

Objekten som vi kommer att kalla *graf* eller *grafer* är mycket användbara inom många områden såsom Diskret Matematik, teknik, datavetenskap, naturvetenskap, genetik och även i samhället. Några vanliga exempel på grafer som många av oss har utsatts för är buss- eller tågkartor, ett enkelt släktträd i biologi, stjärnbilder i astronomi, molekylmodeller i kemi och så vidare. Alltså är graf en matematisk modell för en diskret uppsättning vars mängd av punkter, objekt, är relaterade till varandra på något sätt. Objekt kan vara atomer i en molekyl som förbinds med varandra genom kemiska bindningar eller det kan vara olika delar av jorden som förbinds med varandra genom broar. Faktum är att betydelsen hos denna del av matematiken är obestridlig.

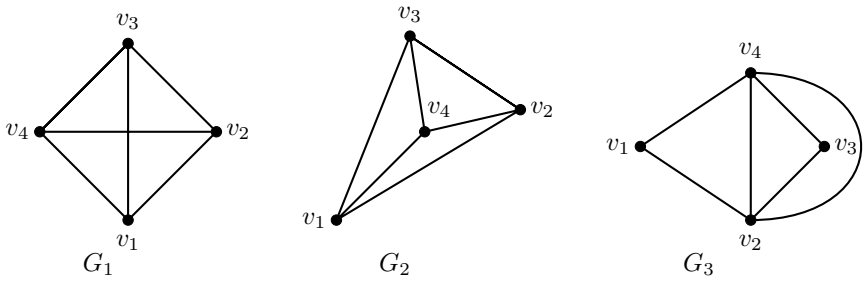
Grafteori är en gren av matematiken som har en bestämd utgångspunkt. Leonard Euler, den ryske matematikern, publicerade sin uppsats för att lösa problemet med Königsbergs sju broar år 1735 [Fig. 1]. Han visade i sin uppsats att det inte går att gå över Königsbergs sju broar, så att man går över varje bro exakt en gång, och sedan kommer tillbaks till utgångspunkten. Euler var den första personen som grundade grafteori [5, s.27-29].



Figur 1: Königsbergs sju broar.

1.1.1 Vad är en graf?

I det grafteoretiska sammanhanget är en graf en matematisk representation av ett nätverk som beskriver förhållandet mellan punkter och linjer. Längden på linjerna och positionen för punkterna spelar ingen roll i grafen. I Fig. 2. illustreras tre grafer med samma antal noder och kanter.

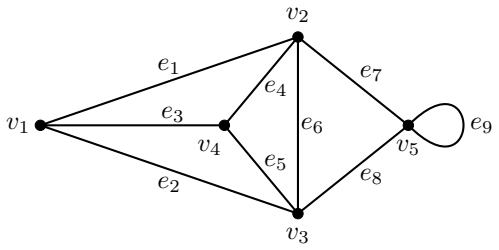


Figur 2: Grafer med samma antal noder och kanter.

Definition 1.1. En graf G är ett par (V, E), där V är en mängd av noder och E är en mängd av kanter.

Mängden $V(G)$ kallas grafens nodmängd, punktmängd eller hörnmängd och $E(G)$ kallas grafens kantmängd eller bågmängd som identifieras med ett par noder.

Om noden v_1 har en kant e_1 som är sammankopplad till samma nod, $e_1 = \{v_1, v_1\} = \{v_1\}$, säger man att den har en *ögla* eller *loop* [2, s.9-10]. Det finns även formalismer som kallas för *multigraf* och där man kan ha flera kanter mellan ett par av noder [4, s.28]. En graf utan någon ögla kallas för *enkel graf* [2, s.4]. I Fig. 3 illustrerar vi ett exempel på hur en graf ser ut.



Figur 3: Grafen G med 5 noder och 9 kanter.

I figuren ovan består grafen G av fem noder och nio kanter. Nodmängden $V(G)$ och kantmängden $E(G)$ brukar skrivas på följande sätt:

$$V(G) = \{v_1, v_2, v_3, v_4, v_5\} \quad \text{och}$$

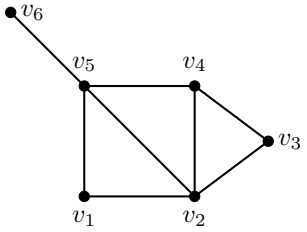
$$E(G) = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_2, v_5\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_5\}\}.$$

Vi kan även ge namn till kanter, till exempel i Fig. 3 har vi $e_1 = \{v_1, v_2\}$, $e_2 = \{v_1, v_3\}$, $e_3 = \{v_1, v_4\}$, $e_4 = \{v_2, v_4\}$, $e_5 = \{v_3, v_4\}$, $e_6 = \{v_2, v_3\}$, $e_7 = \{v_2, v_5\}$, $e_8 = \{v_3, v_5\}$ och $e_9 = \{v_5\}$.

Två noder v och w så att $v \neq w$ i en graf kallas *grannar* (adjacent) om det finns en kant som förbinder v till w det vill säga en kant på formen $\{v, w\}$. På samma sätt är två distinkta kanter i en graf, *grannar* om de har minst en gemensam nod [2, s.12]. Låt oss nu skriva detta på ett formellt sätt.

Definition 1.2. I en graf $G = (V, E)$ är två noder $v, w \in V$ grannar om $\{v, w\} \in E$.

Betrakta figuren nedan [Fig. 4]. Låt oss använda ovanstående definition och skriva en lista som visar antal grannar för varje nod i grafen . Detta kallas vi för en grannlista.



Figur 4: Graf G

Alltså har grafen, från Fig. 4, en grannlista på följande sätt

$v_1 : v_2, v_5$

$v_2 : v_1, v_3, v_4, v_5$

$v_3 : v_2, v_4$

$v_4 : v_2, v_3, v_5$

$v_5 : v_1, v_2, v_4, v_6$

$v_6 : v_5$.

För att arbeta effektivare med en grannlista låt oss presentera en så kallade *grannmatris*. Vi kan definiera grannmatrisen på följande sätt.

Definition 1.3. Grannmatrisen A_G för en enkel graf $G = (V, E)$ med nodmängden $V(G) = \{v_1, ..., v_n\}$ är en $n \times n$ matris med element a_{ij} given av

$$a_{ij} = \begin{cases} 1 & \text{om} \quad \{i, j\} \in E(G) \\ 0 & \text{annars.} \end{cases}$$

Alltså betyder detta att $a_{ij} = 1$ om, och endast om, det finns en kant mellan nod i och nod j annars $a_{ij} = 0$. Vi måste notera att en grannmatris för en enkel graf G är alltid en symmetrisk matris och har bara nollor på diagonalen. För grafen G från Fig. 4, så har vi grannmatrisen

$$A_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

1.1.2 Grad (Valens)

Nu har vi bekantat oss med en grannmatris. Med hjälp av en grannmatris kan vi bestämma antal grannar som varje nod har i en graf. Till exempel grannmatrisen A_G för grafen G från Fig. 4 visar att noden v_1 har två grannar v_2 och v_5, eller noden v_6 har bara en granne v_5. För att analysera en graf bättre, låt oss nu definiera en så kallad *grad* eller *valens*.

Definition 1.4. Låt G vara en enkel graf. Antal kanter som är kopplade till en nod v kallas nodens *grad* eller nodens *valens* och man betecknar detta med $\deg(v)$, där $v \in V(G)$. Alltså har vi

$$\deg(v) = |D_v|, \quad \text{där} \quad D_v = \{e \in E(G) \mid v \in e\}.$$

Observera att definitionen gäller endast för grafer utan loop.

För grafen G från Fig. 4 har vi då

$$\deg(v_1) = 2, \quad \deg(v_2) = 4, \quad \deg(v_3) = 2, \quad \deg(v_4) = 3, \quad \deg(v_5) = 4, \quad \deg(v_6) = 1.$$

I detta exempel kan vi märka att summan av graderna är lika med 16, medan antal kanter är 8. Således är summan av grader för en enkel graf G, lika med dubbelt antal kanter i G och därmed är ett jämnt tal [3, s.3]. Låt oss nu skriva följande satsen.

Sats 1.5. Låt $G = (V, E)$ vara en enkel graf, då har vi

$$\sum_{v \in V} \deg(v) = 2|E|.$$

Det finns också en definition av graden för en nod i en graf med ögla. Då räknar man varje ögla två gånger, till exempel i Fig. 3 har noden v_5 graden 4 medan den har 3 kanter (bara öglan, e_9, räknas två gånger).

1.1.3 Stigar och cykler i grafer

Definition 1.6. Låt $G = (V, E)$ vara en enkel graf. En sekvens av noder $v_1, ..., v_k$ i V, så att det finns kanter mellan dem, med andra ord $\{v_i, v_{i+1}\} \in E$, där $i \in \{1, ..., k-1\}$ kallas för *vandring*. Längden av vandringen är lika med $k - 1$.

Alltså en vandring specificerar en rutt i G som går från en nod till en grannod och så vidare. En vandring får även besöka vilken nod som helst en eller flera gånger. Antal kanter i en vandring kallas dess *längd*.

Definition 1.7. Låt $G = (V, E)$ vara en enkel graf. En vandring i G kallas för *stig* om $v_i \neq v_j$ för alla $i \neq j$ från $\{1, ..., k\}$.

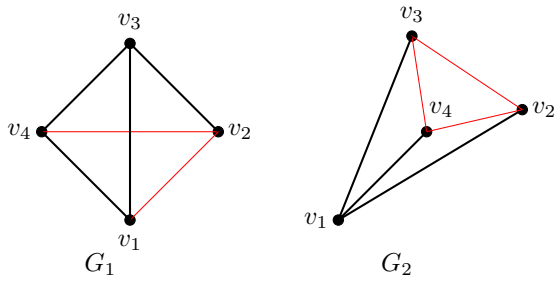
En stig besöker varje nod högst en gång. Alltså är en stig en vandring dess noder är distinkta.

Definition 1.8. En vandring i en enkel graf $G = (V, E)$ kallas för cykel om $v_i \neq v_j$ för alla $i \neq j$ från $\{1, ..., k\}$ utan $\{i, j\} = \{1, k\}$.

För att förstå bättre låt oss skriva ett exempel [3, s.7-8 & 4, s.184].

Exempel 1.9. Betrakta nedanstående figur [Fig. 5]. I grafen G_1 har vi en stig (v_1, v_2, v_4),

som är markerat med röd linje i grafen och i grafen G_2 har vi en cykel (v_2, v_3, v_4, v_2) som är markerat med röd linje i grafen.



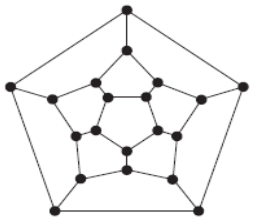
Figur 5: Ett exempel av en stig i grafen G_1 och ett exempel av en cykel i grafen G_2.

Observera att längden av en stig eller en cykel i en graf är lika med antal kanter som passeras i den cykel. I Fig. 5 har stigen (v_1, v_2, v_4) i G_1 länden två och cykeln (v_2, v_3, v_4, v_2) i G_2 har länden tre.

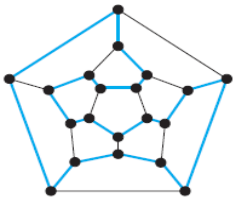
2 Hamiltongraf

2.1 Historisk bakgrund

Namnet ”Hamilton” härstammar från Sir William Rowan Hamilton, den välkända irländske matematikern. Han illustrerade ett matematiskt spel som kallades Icosian, år 1857. Spelet bestod av en regelbunden dodekaeder med 20 hörn på en träyta (se Fig. 6). Var och en av dessa hörn märktes med namnet på en stad. Syftet med spelet var att hitta en rutt, längs kanterna av dodekaedern, som passerar genom varje stad exakt en gång och sedan återgår till utgångspunkten [1, s.103]. En lösning av spelet visas i Fig. 7.



Figur 6: Hamiltons dodekaeders graf.



Figur 7: En möjlig rutt genom varje hörn av dodekaedern visas med blå.

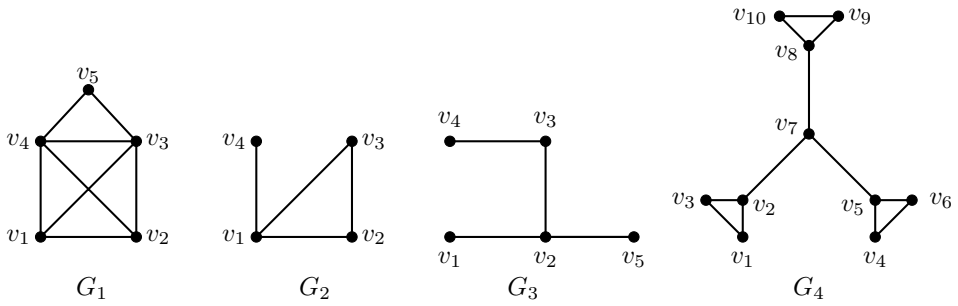
2.2 Hamiltonstigar och Hamiltoncykler

Om vi tittar på Hamiltons spel ur en grafisk aspekt kan vi märka att syftet med spelet är att hitta en cykel i dodekaedern som besöker alla noder i den. För att bekanta oss bättre med Hamiltongrafer låt oss introducera följande definition.

Definition 2.1. Låt $G = (V, E)$ vara en enkel graf. En stig i G som besöker alla noder i G kallas för en *Hamiltonstig*. En cykel i G som besöker alla noder i G kallas för en *Hamiltoncykel*. En graf som har en Hamiltoncykel kallas för en *Hamiltongraf* eller *Hamiltonisk* [1, s.103].

Exempel 2.2. Betrakta Fig. 8 som visar fyra olika grafer G_1, G_2, G_3 och G_4. Enligt ovanstående definition är G_1 en Hamiltongraf för att det finns Hamiltoncykeln $v_1, v_2, v_3, v_5, v_4, v_1$ i den. Den andra grafen, G_2, är inte en Hamiltongraf för att det finns en nod, v_4, som har graden 1. Ett villkor för att en graf skulle vara Hamiltonisk är att den måste ha minst

3 noder, där varje nod har minst graden 2. Därför finns det ingen cykel i G_2. Men den har en Hamiltonstig med sekvensen av noder v_4, v_1, v_3, v_2. Till skillnad från G_1 och G_2 har G_3 ingen av dem, varken Hamiltonstig eller Hamiltoncykel, för att det finns tre noder v_1, v_4, v_5 som har graden 1, vilket gör det omöjligt att besöka alla noder i grafen endast en gång. Sista grafen, G_4, har inte heller en Hamiltoncykel eller Hamiltonstig, trots att den har fler än 3 noder och alla noder har mer än 1 grader. Detta bekräftar att inte alla grafer har Hamiltoncykler eller Hamiltonstigar.



Figur 8: Ett exempel av en Hamiltongraf G_1 och en Hamiltonstig G_2.

Vid första anblicken kan det antas att det inte är svårt att identifiera en Hamiltongraf eller hitta en Hamiltonstig i en given graf, men faktum är att det finns ingen perfekt karakterisering av Hamiltoniska grafer. Därför är upptäckten av karakterisering av Hamiltoniska grafer, ett av de största problemen inom grafteorin [3, s.63].

I själva verket är det bara några få generella egenskaper som är kända för Hamiltoniska grafer. De flesta befintliga teorem har formen "om G har tillräckligt med kanter, då G är Hamiltonisk" [2, s.37].

En av de första och mest berömda förutsättningar som garanterar att Hamiltoncykel i en graf G existerar behandlades i satsen av Dirac (1952). Han skrev i sin sats att "om $G = (V, E)$ är en enkel graf med $n \geq 3$ noder och om $\deg(v) \geq n/2$ för varje nod v, då kan det dras slutsatsen att G är Hamiltonisk" [4, s.308].

2.3 Hamiltonstigar i kompletta grafer

2.3.1 Kompletta grafer

Definition 2.3. En enkel graf G kallas för *komplett* om $E(G) = \{\{v, w\} \in V(G) \mid v, w \in V(G), v \neq w\}$.

En komplett graf med n noder kallas för K_n.

Sats 2.4. Varje enskild nod i en komplett graf K_n har graden $n - 1$.

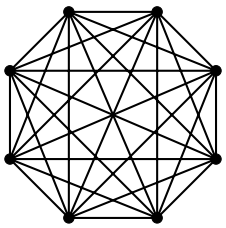
Sats 2.5. Antal kanter B_n i en komplett graf K_n är $B_n = \frac{n(n-1)}{2}$.

Bevis. Låt k_n vara en komplett graf med kanter E. I grafen har varje enskild nod, graden $n-1$ så summan av alla nodernas grad i K_n blir $n(n-1)$ för att det finns n stycken noder som har samma grad $n-1$. Enligt en tidigare sats (Sats 1.5) vet vi att summan av alla noders grad i en graf är lika med två gånger grafens kantmängd, det vill säga $2|E|$. Från detta kan vi få $n(n-1) = 2|E| \Leftrightarrow |E| = \frac{n(n-1)}{2}$. □

Exempel 2.6. Betrakta figuren nedan. K_8 är en komplett graf med 8 noder. Enligt Sats 2.4, så finns det 7 kanter för varje nod i K_8 och det totala antalet kanter är lika med 28 vilket vi kan bestämma med uträkningen

$$B_8 = \frac{8(8-1)}{2} = 28$$

.



Figur 9: En komplett graf K_8.

Sats 2.7. I alla kompletta grafer K_n, där $n \geq 1$, finns det $n \cdot (n-1) \cdot ... \cdot 2 \cdot 1 = n!$ Hamiltonstigar.

Hamiltonstigar i K_n beskrivs genom ordning av noder av K_n vilket är lika med antalet av permutationer. Alltså är antalet Hamiltonstigar i K_n lika med antalet permutationer av n, vilket är $n!$. Till exempel i en komplett graf K_5 finns det $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$ olika Hamiltonstigar.

3 Viktade grafer

Definition 3.1. En viktad graf är ett par (G, w) av en graf $G = (V, E)$ och en funktion $w : E \to \mathbb{R}_{>0}$.

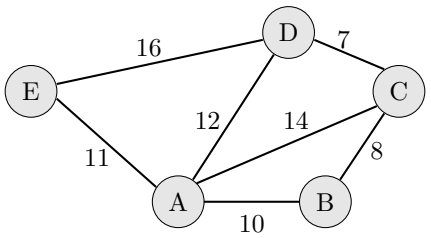
Alltså är en viktad graf en graf där varje kant i grafen har en numerisk vikt [3, s.175]. Detta tal kan representera många saker, till exempel avståndet mellan två platser på en karta eller mellan två anslutningar i ett nätverk.

Definition 3.2. Låt (G, w) vara en viktad graf och $P = (v_1, ..., v_k)$ vara en stig i G. Då är vikten av P

$$w(p) = \sum_{i=1}^{k-1} w(\{v_i, v_{i+1}\}).$$

Alltså vikten av en stig i en viktad graf är summan av vikterna längs de kanter som ingår i stigen. På samma sätt är vikten av en cykel i en viktad graf summan av vikterna längs de kanter som ingår i cykeln [3, s.67].

Exempel 3.3. Betrakta följande graf, där varje nod A, B, C, D, E representerar en stad och varje kant representerar avståndet mellan två städer i kilometer (km) .



Figur 10: En viktad graf G med 5 städer och 7 vägbanor.

Om vi åker från stad A till B och sedan till C, D, E och till slut återkommer vi till utgångspunkten, A, då har vi en cykel med diskret längd 5 och vikten 52 km som är summan av $10, 8, 7, 16, 11$. På liknande sätt om vi åker, till exempel, från B till C och sedan till A, D och E, då har vi en stig med diskret längd 4 och vikten 50 km som är summan av $8, 14, 12$ och 16.

4 Handelsresandeproblem-TSP

Idén bakom handelseresandeproblemet eller The Travelling Salesman problem (TSP) härstammar från den österrikiske matematikern Karl Menger. Ett av Mengers favoritämnen var att studera tekniker för att mäta längden på kurvor i rymden, på 1920-talet. Denna forskning gav nog honom inspiration för att presentera handelseresandeproblemet, i mitten av 1930-talet [5, s.35].

Tanken och frågan med TSP var att "hitta den billigaste möjliga rutten (stigen) genom olika städer och vägar, där varje väg har en definierad resekostnad och endast ansluter två städer. Förutsättningen med lösningen var att man borde besöka alla städer en och endast en gång och sedan återgå till utgångspunkten"[5]. Alltså är ett handelseresandeproblem en uppsättning av ett visst antal noder och kanter. Noder kan representera olika objekt, platser, städer och punkter som förbinds med varandra genom olika typer av kanter såsom vägar och linjer, där varje kant definieras med någon mätbar form av vikter såsom avstånd, tid eller pengar. Observera, resekostnaderna är symmetriska i TSP det vill säga att resa från stad A till B kostar lika mycket som att resa från stad B till A.

Definition 4.1. Låt (G, w) vara en viktad graf och låt H vara mängden av alla Hamiltoncykler i G. Handelsresandeproblemet för (G, w) är att hitta $h_0 \in H$ så att $w(h_0) = \min\{w(h) \mid h \in H\}$.

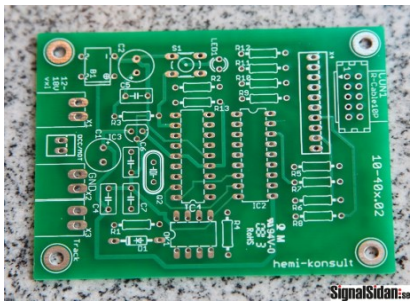
TSP, i själva verket, är ett av de mest intensivt undersökta problemen inom datalogin. Det har inspirerat från studier som gjordes av matematiker, datavetare, kemister, fysiker, psykologer och en mängd icke-professionella forskare. I dagens samhälle kan man se tillämpningar av TSP inom många områden, till exempel logistik, genetik, tillverkning, telekommunikation och naturvetenskap. Här nedan presenteras några av TSP- tillämpningar i olika områden [5].

Mahalanobis jordbrukstudie på 1930-talet är ett tidigt exempel av TSP-tillämpningar vid planering för att inspektera avlägsna platser. Denna typ av TSP-tillämpningar har även förekommit i många andra sammanhang. Till exempel tillämpade William Pulleyblank en TSP-programvara för att planera och hitta rutter för att besöka en uppsättning av 47 oljeplattformar utanför Nigerias kust. I detta exempel besöks plattformarna via en helikopter som flög från en kustbas. I ett annat exempel använde en grupp, från universitet av Maryland, av TSP för att besöka 200 stationer i Chesapeake viken vilken är den största viken i USA. De använde sig av båtar vid sina besök på alla stationer. Syftet med båtturerna var att övervaka blåkrabborna i viken. Forskarna använde sig av TSP för att hitta de snabbaste turer för att besöka alla stationer. Metoden gav de möjligheten att övervaka alla stationer i flera omgångar [5, s.47].

Utan de TSP-tillämpningar som vi normalt tillämpar vid fysiska besöker av avlägsna platser, appliceras TSP även vid observation av platser utan faktiska resor. Ett naturligt exempel av sådana platser är planeter, stjärnor och galaxer som ska observeras med någon form av teleskop. Processen att rotera en utrustning såsom ett teleskop för att göra en observation kallas Slewing. För teleskop i stor skala är Slewing en komplicerad och tidskrävande procedur som hanteras av datordrivna motorer. I detta fall kan TSP minimera den totala tiden som används vid Slewing i en uppsättning av observationer och på så sätt kan implementeras som en del av en övergripande schemalägningsprocess. I en vetenskaplig artikel

beskrev Shawn Carlson hur TSP kunde hjälpa honom att schemalägga ett äldre teleskop för att avbilda cirka 200 galaxer per natt [5, s.51].

En annan vanlig tillämpning av TSP är inom industri. I modern tillverkning används maskiner ofta för att utföra upprepade uppgifter såsom borrning av hål eller anslutning av föremål. Mönsterkort som finns i konventionella elektroniska apparater har ofta många hål för montering av datorchips eller för anslutning mellan lager [se Fig. 11]. Hål tillverkas av automatiska borrmaskiner som rör sig mellan specificerade platser för att skapa ett hål efter det andra. En klassisk tillämpning av TSP är att minimera borrhuvudets restid under tillverkningsprocessen. Tillämpningen av TSP-algoritmer har resulterat till en cirka 10% förbättring av den totala genomströmningen av mönsterkorts produktionslinjer [5, s.54].



Figur 11: Ett mönsterkort

4.1 Hur kan man lösa TSP på ett algoritmisk sätt

Det finns flera olika metoder som kan ge en approximativ optimal lösning till TSP men i denna uppsats kommer vi att fokusera bara på två metoder, vilka är:

1. Närmaste granne-algoritmen,
2. Giriga algoritmen.

4.1.1 Närmaste granne-algoritmen (NN-algoritmen)

Närmaste granne-algoritmen eller NN-algoritmen är en av de första algoritmerna som applicerades för att lösa TSP. I NN-algoritmen väljer säljaren slumpmässigt en stad, en startpunkt, från en mängd av städer sedan besöker han den närmaste granne-staden till startpunkten. Därefter väljer han närmaste granne-staden till den andra staden, från alla obesökta städer. På samma sätt besöker säljaren alla städer tills alla städer har besökts. NN-algoritmen kan snabbt ge en lösning till TSP men den är inte alltid en optimal lösning [5, s.65].
Alltså kan vi skriva algoritmen i följande steg:

1. Välj en slumpmässig stad.
2. Hitta den kortaste vägen som förbinder den aktuella staden med en obesökt stad och åk sedan dit.

3. Ange den nya staden som den aktuella staden.

4. Gå till steg 2.

5. Markera alla besökta städer.

6. Om alla städer är besökta, avsluta turen annars gå till steg 2.

Låt oss nu skriva algoritmen på ett formellt sätt:

Input: Låt (K_n, w) vara en komplett viktad graf.

Algoritmen:

1. Sätt $v_1 = 1$, $S = \{v_1\}$ och $i = 1$.

2. "Loop": Hitta första nod v_{i+1} in $V(G) \setminus S$ som minimera $w(\{v_i, v_{i+1}\})$. Bifoga v_{i+1} till S och ersätta i med $i + 1$.

3. Om $S = |\{1, ..., n\}|$ sluta turen annars gå till "loop".

Output: $(v_1, ..., v_n)$ en Hamiltonstig i G.

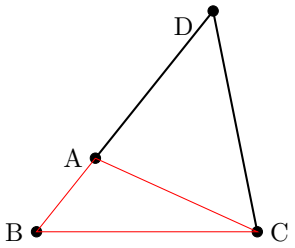
Här måste vi notera att det finns en enkel relation mellan problemen med att hitta en Hamiltonstig och en Hamiltoncykel i en viktad graf. Vi beskrev tidigare att problemet med TSP är att hitta kortaste Hamiltoncykeln i en viktad graf. Fundera på formeln för NN-algoritmen, här har vi hittat en Hamiltonstig $v_1, ..., v_n$ i grafen G, om vi bifogar en kant $\{v_1, v_n\}$ till Hamiltonstigen så får vi en Hamiltoncykel. Omvänt, om vi skriver algoritmen på ett sätt att hitta en Hamiltoncykel i en viktad graf så får vi en Hamiltonstig genom att ta bort en kant, det vill säga sista kanten $\{v_1, v_n\}$ från Hamiltoncykeln.

I samband med TSP-lösningar finns det en viktig förutsättning som den viktade grafen måste uppfylla för att algoritmer kan fungera bra, vilket kallas för triangelolikhet. I de flesta naturliga TSP är direkta rutter kortare eller snabbare än indirekta rutter, till exempel den snabbaste rutten för att nå till en nod j från i är alltid att gå från j direkt till i, snarare än att gå genom någon annan nod k. Låt oss skriva triangelolikheten på ett formellt sätt.

Sats 4.2. Triangelolikhet: I varje triangel ABC, med kantlängden AB, AC och BC då gäller

$$AB < AC + BC, \quad AC < AB + BC \quad \text{och} \quad BC < AC + AB. \tag{1}$$

Bevis. Betrakta en triangel $\triangle ABC$. Förläng $\overrightarrow{BA}$ till D så att AD blir lika med AC. Sedan anslut C till D [Fig. 12].

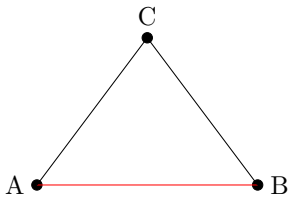


Figur 12: Euklides konstruktion för att bevisa triangelolikheten.

I $\triangle BCD$ eftersom AD är lika med AC, då är $\triangle CAD$ likbent och alltså är $\angle ACD$ kongruent med $\angle ADC$. Sedan $\angle BCD$ är större än $\angle ACD$, så $\angle BCD$ är större än $\angle ADC = \angle BDC$.

Enligt Proposition 19 i bok *Geometry Through History* "i varje triangel är sidan mot större vinkeln större än sidan mot mindre vinkeln" så i $\triangle BCD$ är BD större än BC. Sedan för att $|BD| = |BA| + |AD| = |BA| + |AC|$ så har vi $BA + AC > BC$. På samma sätt kan vi bevisa att $AB < AC + BC$ och $AC < AB + BC$ [6]. □

Låt A, B och C vara tre städer med en resekostnad mellan varje par av städerna [Fig. 13]. Så resekostnaden från A till C plus resekostnaden från C till B bör inte vara lägre än resekostnaden från A direkt till B.



Figur 13: Triangelolikheten

I själva verket kan man visa med triangelolikheten att resekostnaden för att resa till n städer i ett TSP blir inte $1 + \log_2(n)/2$ gånger mer än den optimala kostnaden om man använder NN-algoritmen. Till exempel i ett handelsresandeproblem med 50 städer garanterar NN-algoritmen att reskostnaden för att besöka alla städer inte blir mer än 4 gånger den optimala reskostnaden [5, s.66].

Sats 4.3. Låt (K_n, w) vara en komplett viktad graf som uppfyller triangelolikheten och P_1 vara resultatet av NN-algoritmen. Så gäller

$$w(P_1) \leq \left(1 + \frac{\log_2(n)}{2}\right) \cdot \min\{w(P) \mid P \quad \text{Hamiltoncykel i } K_n\}. \tag{2}$$

För att redovisningen ska bli enklare låt oss använda L_{opt} istället för $\min\{w(P) \mid P \quad \text{Hamiltoncykel i } K_n\}$. Låt oss också använda $\left(\frac{1}{2} + \frac{\lceil \log_2(n) \rceil}{2}\right)$ istället för $\left(1 + \frac{\log_2(n)}{2}\right)$ eftersom

$$\left(\frac{1}{2} + \frac{\lceil \log_2(n) \rceil}{2}\right) \leq \left(1 + \frac{\log_2(n)}{2}\right). \tag{3}$$

Beviset för sats 4.3 ges efter beviset på följande Lemma.

Lemma 4.4. Låt (G, w) vara en komplett viktad graf med noder V, där $|V| = n$, och en funktion $l : V \to \mathbb{R}_{\geq 0}$ så att följande förhållanden gäller

1. $w(\{x, y\}) \geq \min\{l_x, l_y\}$ för alla $x, y \in V$
2. $l_x \leq \frac{1}{2}\ L_{opt}$ för alla $x \in V$.

Så gäller $\sum l_x \leq \left(\frac{1}{2} + \frac{\lceil \log_2(n) \rceil}{2}\right) \cdot L_{opt}$.

Bevis. Anta en komplett viktad graf G med ovanstående förhållande. Sedan låt $V(G) = \{i \mid 1 \leq i \leq n\}$ och gäller förhållandet $l_i \geq l_j$ där $i \leq j$ för alla $i, j \in V(G)$. Nyckeln till beviset är följande olikhet:

$$L_{opt} \geq 2 \sum_{i=k+1}^{\min\{2k,n\}} l_i \tag{4}$$

för alla k från $\{1, ..., n\}$.

För att bevisa (4) låt H vara en komplett subgraf, delgraf G med nodmängden

$$\{i \mid 1 \leq i \leq \min\{2k, n\}\}.$$

Sedan låt T vara en Hamiltoncykel i H som besöker alla noder av H i samma ordning som noder besöks på en optimal tur i G. Låt L_T vara längden av T. Enligt triangelolikheten så måste varje kant b, c i T ha en längd som är mindre eller lika med någon annan stig från b till c som används i den optimala turen. Eftersom L_{opt} är summan av alla kantlängder i den optimala Hamiltoncykeln i G och L_T är summan av alla kantlängder i T så har vi

$$L_{opt} \geq L_T. \tag{5}$$

Enligt förhållandet 1 i Lemma, för alla $i, j \in T$ gäller $w(\{i, j\}) \geq \min\{l_i, l_j\}$. Därför har vi

$$L_T \geq \sum_{(i,j)\in T} \min\{l_i, l_j\} = \sum_{i \in H} \alpha_i l_i \tag{6}$$

där α_i är numret av kanten $\{i, j\}$ i T för vilken $i \geq j$ där $l_i \leq l_j$ och således $\min\{l_i, l_j\} = l_i$. Observera att varje α_i är högst 2, för att i är slutpunkten för endast två kanter i T. Alltså summan av α_i visar antal kanter i T.

Vi sa tidigare att T är en Hamiltoncykel i H som besöker alla noder av H, så T har samma nodmängd som H. Sedan vet vi att i en Hamiltoncykel är antal kanter lika med antal noder, så då finns det exakt $\min\{2k, n\}$ kanter i T. Alltså har vi

$$\sum_{i \in H} \alpha_i l_i = \sum_{i=1}^{\min(2k,n)} \alpha_i l_i.$$

Nu vill vi minska gränsen på högerleden av (6), det vill säga $\sum_{i=1}^{\min(2k,n)} \alpha_i l_i$. Låt oss använda olikheten $l_1 \geq l_i \geq l_j \geq ... \geq l_n$ där $1 \leq i \leq j \leq ... \leq n$, i summan

$$\sum_{i=1}^{\min(2k,n)} \alpha_i l_i.$$

Om vi samlar alla α_i i slutet av summan

$$\sum_{i=1}^{\min(2k,n)} \alpha_i l_i,$$

så får vi

$$\sum_{i=1}^{\min(2k,n)} \alpha_i l_i \geq 0 \cdot l_1 + 0 \cdot l_2 + 0 \cdot l_3 + ... + 2 \cdot l_{\min(2k,n)-2} + 2 \cdot l_{\min(2k,n)-1} + 2 \cdot l_{\min(2k,n)}.$$

Detta ger oss ett mindre resultat och antal termer blir högst hälften av $\sum_i \alpha_i$.

Nu låt oss kombinera detta med en uppskattning för $\sum_i \alpha_i$, som är lika med $\min(2k, n)$. Alltså vill vi ha högst $\frac{1}{2}\min(2k, n)$ termer. Vi kan jämföra detta med antal termer i summan

$$\sum_{i=k+1}^{\min(2k,n)} l_i.$$

Antal termer här är $\min(2k, n) - ((k + 1) - 1) = \min(2k, n) - k$. Detta är faktiskt mindre än $\frac{1}{2}\min(2k, n)$, för att

$$\begin{aligned} &\min(2k, n) - k \leq \frac{1}{2}\min(2k, n) \\ \Leftrightarrow\ &\frac{1}{2}\min(2k, n) \leq k. \end{aligned}$$

Om vänsterleden av sista olikhet är lika med $\min(k, \frac{n}{2})$, så är olikheten sant.
Alltså har vi

$$\sum_{i \in H} \alpha_i l_i \geq 2 \sum_{i=k+1}^{\min(2k,n)} l_i \tag{7}$$

Från (5), (6) och (7) kan vi bevisa (4).

Sedan låt oss summera olikheten (4) för alla värden på $k < n$, där k kan skrivas som $k = 2^j$. Observera att k är lika med 2^j om och endast om $\log_2(n) = j$. Alltså har vi $2^j < n$ om och endast om $j < \log_2(n)$. Eftersom j är ett heltal, gäller detta om och endast om $j \leq \lceil \log_2(n) \rceil - 1$. Alltså får vi

$$\sum_{j=0}^{\lceil \log_2(n) \rceil - 1} L_{opt} \geq \sum_{j=0}^{\lceil \log_2(n) \rceil - 1} \left(2 \cdot \sum_{i=2^j+1}^{\min\{2^{j+1}, n\}} l_i \right),$$

Vilket reduceras till

$$\lceil \log_2(n) \rceil \cdot L_{opt} \geq 2 \cdot \sum_{i=2}^{n} l_i. \tag{8}$$

Nu medför förhållandet 2 från Lemma

$$L_{opt} \geq 2 \cdot l_1. \tag{9}$$

Alltså från (8) och (9) har vi

$$L_{opt} \cdot \left(\frac{1}{2} + \frac{\lceil \log_2(n) \rceil}{2} \right) \geq \sum_{i=1}^{n} l_i,$$

vilket bevisar Lemma. □

Bevis av Sats 4.3. Anta en komplett viktad graf (K_n, w) som uppfyller triangelolikheten. Låt P_1 vara en Hamiltoncykel i K_n som är resultatet av NN-algoritmen och låt L_{opt} vara längden av den optimala Hamiltoncykeln i K_n. För varje nod x i K_n, låt l_x vara längden av kanten som valdes av NN-algoritmen. Vi vill visa att l_x uppfyller förhållandena av Lemma 4.4.

Om noden x valdes av algoritmen innan noden y, sedan om y var en kandidat för närmaste granne till x men inte valdes av algoritmen så kan kanten $\{x, y\}$ inte vara kortare än l_x. Alltså har vi

$$w(\{x, y\}) \geq l_x. \tag{10}$$

Omvänt, om y valdes innan x, så

$$w(\{x, y\}) \geq l_y. \tag{11}$$

Eftersom en av noderna valdes innan den andra, antingen (10) eller (11), så måst förhållandet 1 i Lemma 4.4 vara uppfylld.

För att bevisa förhållandet 2 räcker det att bevisa för alla kanter $\{x, y\}$ är

$$w(\{x, y\}) \leq \frac{1}{2} \cdot L_{opt}. \tag{12}$$

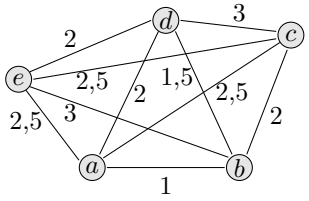
Om vi antar att den optimala turen består av två osammanhängande delar, som var och en är en stig mellan noderna x och y, så borde enligt triangelolikheten längden på någon stig mellan x och y inte vara mindre än $w(\{x, y\})$, vilket bekräftar (12).

Eftersom l_x är längden av varje par av noder i P_1, så

$$\sum l_x = w(P_1) \tag{13}$$

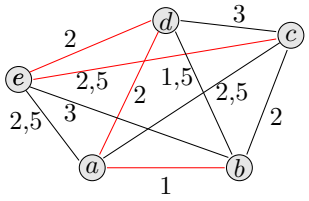
Från (13), (3) och Lemma 4.4, så kan vi bevisa satsen 4.3. □

Exempel 4.5. Låt K_5 vara en komplett viktad graf. I grafen representerar varje nod en stad och varje kant representerar vägen mellan två städer. Avståndet på varje kant anges i grafen och är i kilometer (km) [Fig. 14].



Figur 14: Vikter på grafen K_5.

Nu ska vi använda NN-algoritmen för att bestämma en Hamiltonstig i K_n. Vi antar att vi bor i staden b. Enligt NN-algoritmen måste vi välja närmaste staden till b, vilken är a med avståndet på 1 km, då åker vi till a. Nu måste vi hitta närmaste staden till a, från obesökta städer, vilket är d med avståndet på 2 km, då åker vi till d. På samma sätt väljer vi staden e och c. Alltså väljer NN-algoritmen en Hamiltonstig i K_5 som besöker alla städer i ordning b, a, d, e, c och har en längd på 7,5 km vilket är summan av 1, 2, 2 och 2,5. Vägen anges med röda färgen i Fig. 15.



Figur 15: En Hamiltonstig i grafen från Fig. 14.

För att kontrollera att lösningen verkligen är mindre än $1+\log_2(n)/2$ gånger den optimala lösningen, låt oss först hitta den optimala lösningen i K_5. Från Fig. 14 kan man se att den kortaste Hamiltonstigen i grafen är på 7 km med resvägen (e, d, b, a, c), (c, b, a, d, e) eller (a, b, d, e, c). Alltså har vi $\left(1 + \frac{\log_2(5)}{2}\right) \cdot 7 \approx 15$, så $7,5 \leq 15$. I detta exempel kunde NN-algoritmen ge oss en lösning som är mindre än ungefär två gånger den optimala lösningen.

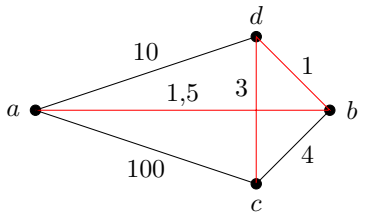
Låt oss byta startstaden i Fig. 14 och räkna avståndet för andra Hamiltonstigar med NN-algoritmen. Vi har illustrerat detta i tabellen nedan [Tabell 1].

Startpunkt	Hamiltonstigar	Totala avståndet
a	a, b, d, e, c	7 km
c	c, b, a, d, e	7 km
d	d, b, a, e, c	7,5 km
e	e, d, b, a, c	7 km

Tabell 1: 4 olika Hamiltonstigar i grafen från Fig. 14.

Tabellen ovan visar att det finns olika Hamiltonstigar beroende på startstaden i grafen från Fig. 14. Tabellen bevisar att alla Hamiltonstigar som vi fick genom NN-algoritmen har ett totalt avstånd som är mindre än $1 + \log_2(n)/2$ gånger den optimala lösningen.

Exempel 4.6. Betrakta figuren nedan. Låt K_4 vara en komplett viktad graf och låt P_1 vara den kortaste Hamiltonstigen i K_4 med avståndet på 5,5 cm. I Fig. 16 har vi markerat P_1 med röda färgen. Längden på varje kant anges även i grafen och är i cm.



Figur 16: Kortaste Hamiltonstigen i K_4.

Sedan betrakta tabellen 2 vilken visar de 4 olika Hamiltonstigar i grafen ovan som vi fick genom NN-algoritmen.

Startpunkt	Hamiltonstigar	Total längd
a	a, b, d, c	5,5 cm
b	b, d, c, a	104 cm
c	c, d, b, a	5,5 cm
d	d, b, a, c	102,5 cm

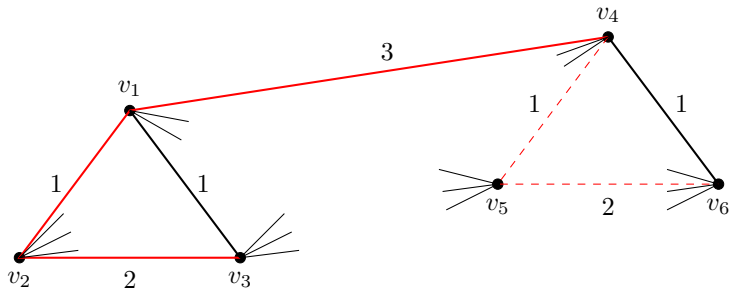
Tabell 2: 4 olika Hamiltonstigar i grafen från Fig. 16.

Från ovanstående tabell kan vi se att NN-algoritmen kunde hitta två Hamiltonstigar på längden 5,5 cm som är lika med den kortaste Hamiltonstigen i grafen. Den kunde även hitta två Hamiltonstigar med längden 104 cm och 102,5 cm som är ungefär 19 gånger större än den kortaste Hamiltonstigen i grafen. Detta visar att om en graf inte uppfyller triangelolikheten, då finns det en chans att NN-algoritmen ger oss en Hamiltonstig som inte är mindre än $1 + \log_2(n)/2$ gånger den optimala lösningen. För att förstå bättre betydelsen

av triangelolikheten i TSP, låt oss beskriva ett annat exempel.

Exempel 4.7. Betrakta Fig 17. Låt K_6 vara en komplett viktad graf. Längden på vissa kanter anges i grafen och är i centimeter (cm). Resten av kanterna som vi har ritat med korta linjer i grafen har en liknande längd på 100 cm, vilka är:

$$w(\{v_2, v_5\}) = 100$$
$$w(\{v_2, v_6\}) = 100$$
$$w(\{v_2, v_4\}) = 100$$
$$w(\{v_3, v_4\}) = 100$$
$$w(\{v_3, v_5\}) = 100$$
$$w(\{v_3, v_6\}) = 100$$
$$w(\{v_5, v_1\}) = 100$$
$$w(\{v_1, v_6\}) = 100.$$



Figur 17: Vikter på grafen K_6 utan dem långa kanter.

Sedan låt $P_1 = (v_3, v_2, v_1, v_4, v_5, v_6)$ vara den kortaste Hamiltonstigen i K_6 med längden på 9 cm. P_1 anges med röda färgen i Fig. 17. Därefter betrakta tabellen 3. I tabellen illustrerar vi 6 olika Hamiltonstigar i K_6 som vi fick med hjälp av NN-algoritmen.

Startpunkt	Hamiltonstigar	Total längd
v_1	$v_1, v_2, v_3, v_6, v_4, v_5$	105
v_2	$v_2, v_1, v_3, v_6, v_4, v_5$	104
v_3	$v_3, v_1, v_2, v_5, v_4, v_6$	104
v_4	$v_4, v_5, v_6, v_3, v_1, v_2$	105
v_5	$v_5, v_4, v_6, v_3, v_1, v_2$	104
v_6	$v_6, v_4, v_5, v_2, v_1, v_3$	104

Tabell 3: 6 olika Hamiltonstigar i grafen från Fig. 17.

Tabellen ovan visar att alla Hamiltonstigar i K_6, enligt NN-algoritmen, har en total längd som inte är mindre än $(1 + \log_2(6)/2) \cdot 9 \approx 20$ cm . I jämförelse med exempel 4.6 har vi här inte fått någon Hamiltonstig som vara mindre än $1 + \log_2(n)/2$ gånger den optimala lösningen. Detta innebär att i viktade grafer som inte uppfyller triangelolikheten kan NN-algoritmen inte hitta någon Hamiltonstig som är mindre än $1+\log_2(n)/2$ gånger den optimala lösningen. Alltså är NN-algoritmen en approximativ optimal lösning, en lösning mindre än $1+\log_2(n)/2$ gånger den optimala lösningen, bara om den viktade grafen uppfyller triangelolikheten.

4.1.2 Giriga algoritmen

Giriga algoritmen är en annan av de första algoritmerna som applicerades för att lösa TSP. I giriga algoritmen ordnar säljaren kanterna i en växande ordning. Sedan väljer den kanter med minsta vikter på ett sådant sätt att ingen cykel skapas och alla städer besöks [5, s.68]. Alltså kan vi skriva algoritmen i följande steg:

1. Ordna alla kanter i växande ordning.
2. Börja med den kant som har lägst vikt.
3. Titta på kanterna en efter en i ordningslistan och välj en kant endast om kanten, tillsammans med redan valda kanter:
   (a) inte skapar en cykel,
   (b) inte orsakar att noders grad blir tre eller mer än tre.
4. Avsluta turen när alla noder besöks.

Låt oss nu skriva algoritmen på ett formellt sätt:

Input: Låt (K_n, w) vara en komplett viktad graf med noder V och kanter E.

Algoritmen:

1. Sätt $S = \varnothing$.
2. 'Loop': Välj $e \in E$ så att $w(e)$ blir minst och grafen (V, S) får ingen cykel och dess noder får grad mindre eller lika med 2. Addera e till S.
3. Om $|S| = n - 1$ stoppa turen, annars går till 'loop'.

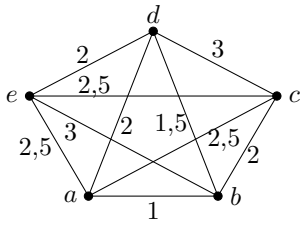
Output: S är kantmängd av en Hamiltonstig.

Sats 4.8. Låt (K_n, w) vara en komplett viktad graf som uppfyller triangelolikheten och låt P_1 vara resultatet av giriga algoritmen. Så gäller

$$w(P_1) \leq \left(\frac{1}{2} + \frac{\log_2(n)}{2}\right) \cdot \min\{w(P) \mid P \quad \text{Hamiltoncykel i } K_n\}. \tag{14}$$

Bevis. Titta på beviset för Sats 4.3.

Exempel 4.9. Titta på grafen K_5 i Fig. 18. Låt oss hitta en Hamiltonstig i den komplett viktade grafen med hjälp av den giriga algoritmen. Avståndet mellan varje par av noder räknades i centimeter.

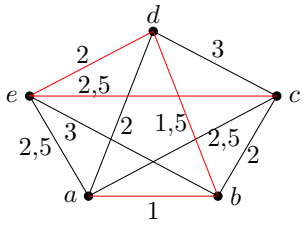


Figur 18: Vikter på grafen K_5.

Enligt giriga algoritmen ordnar vi först kanterna i en växande ordning. Alltså kommer kanterna ordnas på följande sätt

$$\{a,b\},\{b,d\},\{a,d\},\{b,c\},\{d,e\},\{a,e\},\{a,c\},\{c,e\},\{b,e\},\{c,d\}.$$

Sedan väljer vi första kanten $\{a,b\}$ med lägsta avståndet på 1 cm. Därefter väljer vi nästa kanten $\{b,d\}$ med avståndet på 1,5 cm för att den skapar ingen cykel med första kanten. Kanten $\{a,d\}$ avvisas för att $\{a,b\},\{b,d\},\{a,d\}$ skapar en cykel. Kanten $\{b,c\}$ avvisas också för att den orsakar att noden b får grad 3. Sedan väljer vi kanten $\{d,e\}$ för att $\{a,b\},\{b,d\},\{d,e\}$ skapar ingen cykel. På samma sätt avvisas kanten $\{a,e\}$ och $\{a,c\}$. Efter detta väljer vi kanten $\{c,e\}$. Nu har vi besökt alla noder, så vi är klar. Alltså ger den giriga algoritmen en resväg med avstånden på 7 cm som passerar från kanterna $\{a,b\},\{b,d\},\{d,e\},\{e,c\}$ [se Fig. 19].



Figur 19: En Hamiltonstig i grafen från Fig. 18.

Observera att den giriga algoritmen ger inte alltid en optimal lösning till TSP. I själva verket garanterar den giriga algoritmen att reseavståndet för att resa till n städer i ett TSP inte blir mer än $1/2 + \log_2(n)/2$ gånger det optimala reseavståndet, så den har bara lite bättre garanti än närmaste granne algoritmen. Problemet med både giriga och närmaste granne algoritmen uppstår när vi skulle hitta en Hamiltoncykel i TSP, då blir vi tvungen att acceptera flera långa vägar eller kanter för att göra slutliga anslutningar vilket orsakar att turlängden eller turavståndet blir stor.

5 Kortaste vägen mellan 15 största städer i Sverige

5.1 Problemmodellering

Låt K_{15} vara en komplett viktad graf. I grafen representerar varje nod, en stad av de 15 största städerna i Sverige som har mer än 70 000 invånare. Vi betecknar noderna med $v_1, v_2, ..., v_{15}$ [Tabell 4]. Problemet är att hitta kortaste Hamiltonstigen i grafen, det vill säga kortaste resvägen mellan Sveriges 15 största städerna så att man besöker varje stad endast en gång. Avståndet mellan varje par av städer illustreras i en grannmatris och beräknas i kilometer (km) [tabell 5].

Det enklaste sättet att lösa problemet är att testa alla 15! = 1307674368000 Hamiltonstigar och sedan välja den kortaste Hamiltonstigen, men sådana lösningar tar mycket tid. Således ska vi använde de två ovanstående algoritmerna för att hitta en approximativ optimal lösning till problemet.

1	Stockholm	v_1
2	Göteborg	v_2
3	Malmö	v_3
4	Uppsala	v_4
5	Upplands Väsby & Sollentuna	v_5
6	Västerås	v_6
7	Örebro	v_7
8	Linköping	v_8
9	Helsingborg	v_9
10	Jönköping	v_{10}
11	Norrköping	v_{11}
12	Lund	v_{12}
13	Umeå	v_{13}
14	Gävle	v_{14}
15	Borås	v_{15}

Tabell 4: Sveriges 15 största städer

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}
v_1	0	468	613	70,6	28,5	108	191	199	556	323	162	603	637	173	407
v_2	468	0	276	459	497	382	288	278	219	150	316	267	988	523	67,5
v_3	613	276	0	679	637	597	502	417	64,8	292	455	17,7	1245	781	285
v_4	70,6	459	679	0	54,7	77,8	171	265	622	390	228	670	571	107	473
v_5	28,5	497	637	54,7	0	102	196	223	580	348	186	628	621	157	431
v_6	108	382	597	77,8	102	0	95,2	187	540	308	150	588	608	144	354
v_7	191	288	502	171	196	95,2	0	121	446	213	116	493	700	236	260
v_8	199	278	417	265	223	187	121	0	361	129	42,6	409	832	368	212
v_9	556	219	64,8	622	580	540	446	361	0	235	398	55,1	1188	724	228
v_{10}	323	150	292	390	348	308	213	129	235	0	167	283	956	492	82,7
v_{11}	162	316	455	228	186	150	116	42,6	398	167	0	447	795	330	250
v_{12}	603	267	17,7	670	628	588	493	409	55,1	283	447	0	1236	772	276
v_{13}	637	988	1245	571	621	608	700	832	1188	956	795	1236	0	470	1039
v_{14}	173	523	781	107	157	144	236	368	724	492	330	772	470	0	576
v_{15}	407	67,5	285	473	431	354	260	212	228	82,7	250	276	1039	576	0

Tabell 5: Avståndet mellan varje par av 15 största städerna i Sverige. Avstådet beräknas i km.

5.2 Lösning av problemet

För att lösa handelsreandeproblemet i Sverige ska vi använda av:

1. NN-algoritmen,
2. Giriga algoritmen.

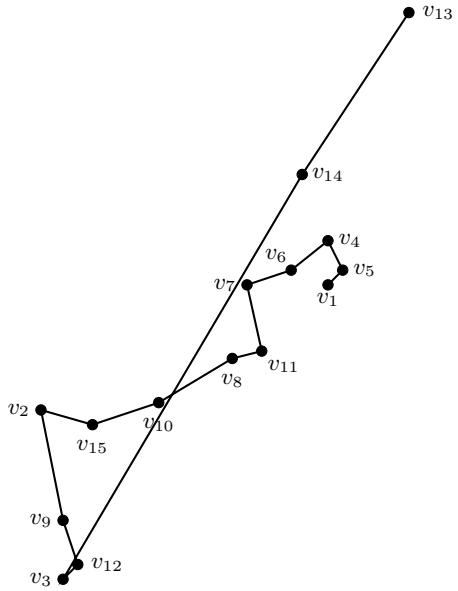
Sedan jämför vi resultaten från båda algoritmerna.

5.2.1 NN-algoritmen

Enligt NN-algoritmen väljer vi först en slumpmässigt stad från K_{15}. Låt oss välja Stockholm, v_1, som startstaden. Sedan väljer vi närmaste granne staden till Stockholm, vilken är Upplands Väsby & Sollentuna, v_5, med kortaste avståndet på 28,5 km. Vi åker till v_5. Därefter väljer vi närmaste granne staden till v_5, så väljer vi v_4 med kortaste avståndet på 54,7 km. Då åker vi till v_4. Vi upprepar metoden på samma sätt och hittar närmaste granne staden till varje aktuell stad tills alla städer besökts. Observera att vi väljer närmaste granne staden till varje aktuell stad, från obesökta städer. I detta fall är Umeå, v_{13}, sista staden som vi besöker. Således väljer NN-algoritmen en Hamiltonstig i K_{15} med sekvensen av noder

$$\{v_1, v_5, v_4, v_6, v_7, v_{11}, v_8, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3, v_{14}, v_{13}\}$$

och avståndet på 2236,8 km [se Fig. 20].



Figur 20: En Hamiltonstig i grafen K_{15} med avståndet på 2236,8 km.

Låt oss nu byta startstaden med de andra städerna i K_{15} och räkna avståndet för de andra 14 Hamiltonstigar i grafen som vi får genom NN-algoritmen. Detta har vi illustrerat i Tabell 6.

Startpunkt	Hamiltonstigar	Totala avståndet
v_1	$v_1, v_5, v_4, v_6, v_7, v_{11}, v_8, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3, v_{14}, v_{13}$	2236,8 km
v_2	$v_2, v_{15}, v_{10}, v_8, v_{11}, v_7, v_6, v_4, v_5, v_1, v_{14}, v_{13}, v_9, v_{12}, v_3$	2597 km
v_3	$v_3, v_{12}, v_9, v_2, v_{15}, v_{10}, v_8, v_{11}, v_7, v_6, v_4, v_5, v_1, v_{14}, v_{13}$	1628,8 km
v_4	$v_4, v_5, v_1, v_6, v_7, v_{11}, v_8, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3, v_{14}, v_{13}$	2267 km
v_5	$v_5, v_1, v_4, v_6, v_7, v_{11}, v_8, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3, v_{14}, v_{13}$	2252,7 km
v_6	$v_6, v_4, v_5, v_1, v_{11}, v_8, v_7, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3, v_{14}, v_{13}$	2392,6 km
v_7	$v_7, v_6, v_4, v_5, v_1, v_{11}, v_8, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3, v_{14}, v_{13}$	2282.8 km
v_8	$v_8, v_{11}, v_7, v_6, v_4, v_5, v_1, v_{14}, v_{13}, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3$	2455,8 km
v_9	$v_9, v_{12}, v_3, v_2, v_{15}, v_{10}, v_8, v_{11}, v_7, v_6, v_4, v_5, v_1, v_{14}, v_{13}$	1685,8 km
v_{10}	$v_{10}, v_{15}, v_2, v_9, v_{12}, v_3, v_8, v_{11}, v_7, v_6, v_4, v_5, v_1, v_{14}, v_{13}$	1916.8 km
v_{11}	$v_{11}, v_8, v_7, v_6, v_4, v_5, v_1, v_{14}, v_{13}, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3$	2460,8 km
v_{12}	$v_{12}, v_3, v_9, v_2, v_{15}, v_{10}, v_8, v_{11}, v_7, v_6, v_4, v_5, v_1, v_{14}, v_{13}$	1638,5 km
v_{13}	$v_{13}, v_{14}, v_4, v_5, v_1, v_6, v_7, v_{11}, v_8, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3$	1593 km
v_{14}	$v_{14}, v_4, v_5, v_1, v_6, v_7, v_{11}, v_8, v_{10}, v_{15}, v_2, v_9, v_{12}, v_3, v_{13}$	2368 km
v_{15}	$v_{15}, v_2, v_{10}, v_8, v_{11}, v_7, v_6, v_4, v_5, v_1, v_{14}, v_{13}, v_9, v_{12}, v_3$	2665,1 km

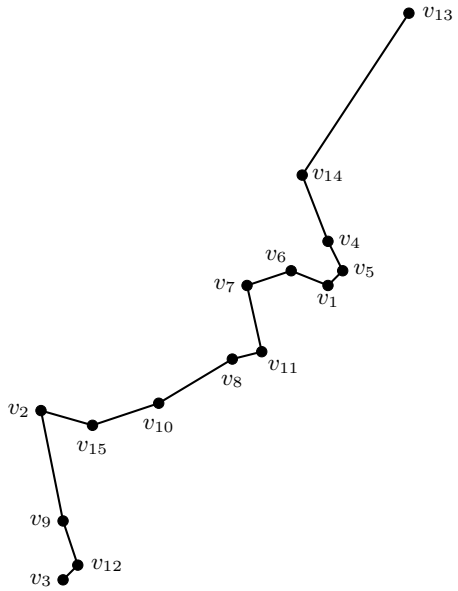
Tabell 6: 15 olika Hamiltonstigar i K_{15}, enligt NN-algoritmen.

Från Tabell 6 är den kortaste Hamiltonstigen i grafen, enligt NN-algoritmen, på 1593 km med startstaden Umeå, v_{13}, [se Fig. 21] och den längsta Hamiltonstigen är på 2665,1 km med startstaden Borås, v_{15}.

Vi visade tidigare att NN-algoritmen ger inte alltid en optimal lösning till TSP men den kan ge lösningar mindre än $1+\log_2(n)/2$ gånger den optimala lösningen med förutsättningen att viktade grafer uppfyller triangelolikheten. Den optimala lösningen eller den kortaste Hamiltonstigen, P_{opt}, borde inte vara mindre än 1336,2 km i grafen K_{15}. Eftersom minsta avståndet mellan v_{13} och v_{14} är på 470 km vilken plus de 13 andra minsta kanter i grafen blir det 1336,2 km. Alltså enligt Tabell 6 och dess data har vi $1336,2 \leq P_{opt} \leq 1593$.

Från Tabell 6 kan vi även bevisa att satsen 4.3 är sann för att den sämsta lösningen som vi fick genom NN-algoritmen har längden på 2665,1 km och $2665,1 \leq \left(1 + \frac{\log_2(15)}{2}\right) \cdot P_{opt}$.

Låt oss nu lösa problemet med hjälp av giriga algoritmen för att sedan jämföra resultaten.



Figur 21: Kortaste Hamiltonstigen i K_{15} med avståndet på 1593 km, enligt NN-algoritmen.

5.2.2 Giriga algoritmen

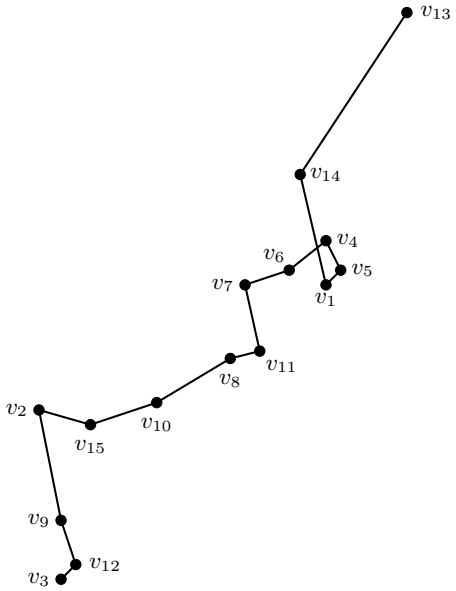
Med tanke på att K_{15} är en komplett graf och har 105 kanter, låt oss ordna alla kanter i en växande ordning. Alltså kommer kanterna ordnas på följande sätt

$\{v_3, v_{12}\}, \{v_1, v_5\}, \{v_8, v_{11}\}, \{v_4, v_5\}, \{v_9, v_{12}\}, \{v_3, v_9\}, \{v_2, v_{15}\}, \{v_1, v_4\}, \{v_4, v_6\}, \{v_{10}, v_{15}\},$
$\{v_6, v_7\}, \{v_5, v_6\}, \{v_4, v_{14}\}, \{v_1, v_6\}, \{v_7, v_{11}\}, \{v_7, v_8\}, \{v_8, v_{10}\}, \{v_6, v_{14}\}, \{v_2, v_{10}\}, \{v_6, v_{11}\},$
$\{v_5, v_{14}\}, \{v_1, v_{11}\}, \{v_{10}, v_{11}\}, \{v_4, v_7\}, \{v_1, v_{14}\}, \{v_5, v_{11}\}, \{v_6, v_8\}, \{v_1, v_7\}, \{v_5, v_7\}, \{v_1, v_8\},$
$\{v_8, v_{15}\}, \{v_7, v_{10}\}, \{v_2, v_9\}, \{v_5, v_8\}, \{v_9, v_{15}\}, \{v_4, v_{11}\}, \{v_9, v_{10}\}, \{v_7, v_{14}\}, \{v_{11}, v_{15}\},$
$\{v_7, v_{15}\}, \{v_4, v_8\}, \{v_2, v_{12}\}, \{v_2, v_3\}, \{v_{12}, v_{15}\}, \{v_2, v_8\}, \{v_{10}, v_{12}\}, \{v_3, v_{15}\}, \{v_2, v_7\},$
$\{v_3, v_{10}\}, \{v_6, v_{10}\}, \{v_2, v_{11}\}, \{v_1, v_{10}\}, \{v_{11,v_{14}}\}, \{v_5, v_{10}\}, \{v_6, v_{15}\}, \{v_8, v_9\}, \{v_8, v_{14}\},$
$\{v_2, v_6\}, \{v_4, v_{10}\}, \{v_9, v_{11}\}, \{v_1, v_{15}\}, \{v_8, v_{12}\}, \{v_3, v_8\}, \{v_5, v_{15}\}, \{v_7, v_9\}, \{v_{11}, v_{12}\},$
$\{v_3, v_{11}\}, \{v_2, v_4\}, \{v_1, v_2\}, \{v_{13}, v_{14}\}, \{v_4, v_{15}\}, \{v_{10}, v_{14}\}, \{v_7, v_{12}\}, \{v_2, v_5\}, \{v_3, v_7\},$
$\{v_2, v_{14}\}, \{v_6, v_9\}, \{v_1, v_9\}, \{v_4, v_{13}\}, \{v_{14}, v_{15}\}, \{v_5, v_9\}, \{v_6, v_{12}\}, \{v_3, v_6\}, \{v_1, v_{12}\},$
$\{v_6, v_{13}\}, \{v_1, v_3\}, \{v_5, v_{13}\}, \{v_4, v_9\}, \{v_5, v_{12}\}, \{v_1, v_{13}\}, \{v_3, v_5\}, \{v_4, v_{12}\}, \{v_3, v_4\},$
$\{v_7, v_{13}\}, \{v_9, v_{14}\}, \{v_{12}, v_{14}\}, \{v_3, v_{14}\}, \{v_{11}, v_{13}\}, \{v_8, v_{13}\}, \{v_{10}, v_{13}\}, \{v_2, v_{13}\}, \{v_{13}, v_{15}\},$
$\{v_9, v_{13}\}, \{v_{12}, v_{13}\}, \{v_3, v_{13}\}.$

Enligt giriga algoritmen börjar vi med kanten $\{v_3, v_{12}\}$ med minsta vikt på 17,7 km. Sedan väljer vi kanterna $\{v_1, v_5\}, \{v_8, v_{11}\}, \{v_4, v_5\}, \{v_9, v_{12}\}$ för att de tillsammans med första kanten inte skapar en cykel och orsakar inte att nodernas grad blir mer än två. Kanten $\{v_3, v_9\}$ avvisas för att $\{v_3, v_9\}, \{v_3, v_{12}\}, \{v_9, v_{12}\}$ skapar cykel. Därefter väljer vi kanten $\{v_2, v_{15}\}$ för att den skapar ingen cykel med dem redan valda kanterna. Kanten $\{v_1, v_4\}$ avvisas för att $\{v_1, v_4\}, \{v_1, v_5\}, \{v_4, v_5\}$ skapar cykel. Efter detta väljer vi kanterna $\{v_4, v_6\}, \{v_{10}, v_{15}\}$ och $\{v_6, v_7\}$. Kanten $\{v_5, v_6\}$ och $\{v_4, v_{14}\}$ avvisas för att $\{v_5, v_6\}, \{v_4, v_5\}, \{v_4, v_6\}$ skapar cykel och kanten $\{v_4, v_{14}\}$ orsakar att noden v_4 får grad 3. På samma sätt, med hänsyn till villkoren som anges i giriga algoritmen väljer vi mer kanter tills alla noder besökts i K_{15}.

Den giriga algoritmen ger oss en Hamiltonstig i K_{15} med sekvensen av kanter $\{\{v_3, v_{12}\},$ $\{v_{12}, v_9\}, \{v_9, v_2\}, \{v_2, v_{15}\}, \{v_{15}, v_{10}\}, \{v_{10}, v_8\}, \{v_8, v_{11}\}, \{v_{11}, v_7\}, \{v_7, v_6\}, \{v_6, v_4\},$ $\{v_4, v_5\}, \{v_5, v_1\}, \{v_1, v_{14}\}, \{v_{14}, v_{13}\}\}$ och avståndet på 1628,8 km [Se Fig. 22].

Det är intressant att Hamiltonstigen som vi fick genom giriga algoritmen är en av de Hamiltonstigar som vi fick genom NN-algoritmen [se Tabell 6, Hamiltonstigen med starpunkten v_3].



Figur 22: Kortaste Hamiltonstigen i K_{15} med avståndet på 1628,8 km, enligt giriga algoritmen.

5.2.3 Jämförelse mellan algoritmer

Ovanstående resultat visar att båda algoritmerna, närmaste granne-algoritmen och giriga algoritmen, kunde ge oss en snabb approximativ optimal lösning. NN-algoritmen i jämförelse med giriga algoritmen, som gav oss en lösning med avståndet på 1628,8 km, kunde ge oss en lite bättre lösning med avståndet på 1593 km. Observera att båda algoritmerna är heuristiska och man kan inte garantera vilken som är bättre. Men i detta exempel fick vi en lite bättre lösning genom NN-algoritmen.

6 Avslutning

I denna uppsats har vi försökt att analysera handelsresandeproblemt (TSP). Vi har undersökt två olika algoritmer för att lösa problemet. Det som vi har visat i uppsatsen är hur man kan hitta en approximativ optimal lösning till TSP genom den närmaste granne-algoritmen och den giriga algoritmen. Vi visade att båda algoritmerna kan ge en approximativ optimal lösning till TSP med förutsättningen att TSP i form av en viktad graf uppfyller triangelolikheten. Slutligen har vi hittat en approximativ optimal lösning, med andra ord en approximativ optimal Hamiltonstig, till handelsresandeproblemet i Sverige.

7 Tack

Ett stort tack till min handledare Sven Raum för all hans hjälp. Utan hans uppmuntran, goda råd och stöd under hela arbetet, skulle jag inte slutföra min uppsats.

8 Referenser

8.1 Litteraturer

[1] Chartrand, G. & Lesniak, L., (1996). Graphs & Digraphs (Third edition).

[2] Wilson, R. J., (1972). Introduction to graph theory.

[3] Thulasiraman, K., Swamy, M. N. S., (1992). Graphs: Theory and algorithms.

[4] Diestel, R., (2017). Graph Theory (Fifth Edition). ISBN 9783662536223.

[5] Cook, W. J., (2012). In Pursuit of the Traveling Salesman. ISBN 9781400839599.

[6] Dillon, Meighan I., (2018). Geometry Through History: Proposition I.20.
ISBN 9783319741352.

8.2 Figurer

[1] http://people.cs.georgetown.edu/ cnewport/teaching/cosc030-fall16/ (10/02-2020).

[6 & 7] https://www.skedsoft.com/books/discrete-mathematics/hamilton_paths_and_circuits (23/01-2020).

[12] http://butik.signalsidan.se/monsterkort/295-monsterkort-10-40x.html (13/02-2020).