Het Chinese Postbode Probleem

Marene Dimmendaal
s4419553

Nijmegen 2018

Het Chinese Postbode Probleem

Marene Dimmendaal
s4419553

Bachelorscriptie Wiskunde
aan de Radboud Universiteit
te Nijmegen

Geschreven door
Marene Dimmendaal
s4419553

Begeleider: Wieb Bosma
Tweede lezer: Bas Terwijn

9 oktober 2018, Nijmegen

Inhoudsopgave

Inleiding

Vrijwel alle bedrijven hebben een logistieke afdeling. Binnen die afdeling wordt er gekeken naar de planning, uitvoering en controle van het vervoeren van goederen. Belangrijk hierbij is dat grondstoffen en tijd zo goed mogelijk besteed en zo min mogelijk verspild worden. De logistieke afdeling van een waterbedrijf kan bijvoorbeeld bepalen wat de slimste manier is om water naar huizen te brengen en een distributiecentrum kan berekenen hoe producten het snelste in de supermarkten terecht kunnen komen.

Ook postbodes hebben te maken met een logistiek vraagstuk. Zij willen de post zo snel mogelijk hebben bezorgd. Om dit te bereiken gaan ze de route die ze moeten afleggen in een wijk minimaliseren. Dit is een optimalisatieprobleem. Dit probleem staat ook wel bekend als het Chinese Postbode Probleem: wat is de kortste route die de postbode in zijn wijk kan lopen waarbij hij elke straat minstens één keer doorloopt?

Dit Chinese Postbode Probleem kent vele varianten. De varianten komen voort uit restricties die op de routes worden gelegd. Denk hierbij aan het niet in mogen lopen van bepaalde wegen, enkel rechtsaf mogen slaan etc. Deze varianten zullen in deze scriptie besproken worden, maar ik zal op enkel een drietal varianten verder ingaan. Het interessantste aan de varianten die ik ga bespreken is het verschil in moeilijkheid. We zullen alle problemen reduceren tot een lineair-programmeer/optimalisatie-probleem. Hoewel de varianten erg veel op elkaar lijken, zal het oplossen van deze problemen in moeilijkheid verschillen. Het verschil in moeilijkheid zal in deze scriptie uitgelegd en geïllustreerd worden.

Hoofdstuk 1

Het Chinese Postbode Probleem

Voordat we varianten van het Chinese Postbode Probleem gaan bekijken, dienen we te weten wat het probleem precies inhoudt en waar het vandaan komt.

1.1 Het probleem

Het Chinese Postbode Probleem is een wiskundige vertaling van een erg praktisch probleem. Men neme een postbode. Deze postbode zal in een aangewezen wijk zijn post rondbrengen. In deze wijk geldt dat je vanuit iedere straat in een willekeurige andere straat van de wijk terecht kan komen door via straten in de wijk te lopen. De postbode rijdt met zijn fiets of busje naar zijn eigen wijk toe. Vanaf hier begint zijn wandelroute door de wijk. Hij moet weer bij zijn fiets of busje eindigen zodat hij terug naar het postkantoor kan komen.

We willen nu een route vinden zodanig dat de tijd waarin de postbode zijn rondje door de wijk loopt minimaal is. In deze route moet hij iedere straat minstens één keer doorlopen. We gaan er vanuit dat de postbode bij het doorlopen van een straat alle huizen in die straat van post voorziet, en dat hij het snelst is wanneer hij de route met de kortste totale afstand loopt. Het oplossen van het probleem is dus equivalent aan het minimaliseren van de lengte van de route in de wijk.

Dat dit probleem een postbode probleem is, is na de voorgaande uitleg overduidelijk. Maar waarom heet dit dan het Chinese Postbode Probleem? De naam is bedacht door Alan Goldman, medewerker bij het National Institute of Standards and Technology (NIST) [10]. De naam verwijst naar de Chinese wiskundige Mei Ko Kwan, geboren in 1934, die in als eerste over dit probleem publiceerde in het Chinees in 1960. In 1962 verscheen de vertaalde Engelse versie [7].

1.2 Varianten

Op dit probleem zijn erg veel varianten te bedenken. We kunnen restricties leggen op de route, door bijvoorbeeld te eisen dat je bepaalde straten in de wijk maar in één richting mag doorlopen. Dit staat bekend als de gerichte of gemengde variant. Ook kan je variëren in het aantal postbodes dat je door een wijk laat lopen. Deze variant staat bekend als het k-Chinese Postbode Probleem. Ook kan je denken aan varianten waarbij je juist optimaliseert op de snelheid door het aantal stoplichten dat je tegenkomt te minimaliseren. Een andere interessante variant waarbij je op een soortgelijke manier op de snelheid optimaliseert is het Hoover probleem.

Het Hoover probleem gaat over John Edgar Hoover, de man die van 1935 tot aan zijn dood in 1972 directeur van de FBI was. Door ooit een traumatische ervaring bij het kruisen van het verkeer mee te maken zou hij zijn chauffeurs opdragen om enkel rechts af te slaan op hun route. Later geloofde hij dat hij op die manier, door kruisend verkeer te mijden op splitsingen, op de snelst mogelijke wijze bij zijn bestemming zou komen. [1]

Op deze manier zijn er oneindig veel varianten te bedenken op het Chinese Postbode Probleem. In mijn scriptie ga ik er echter maar drie bekijken: het ongerichte probleem, het gerichte probleem en de gemengde variant van het probleem. Op het eerste gezicht lijken deze problemen veel met elkaar te maken te hebben: de gemengde variant is een combinatie van het ongerichte en het gerichte probleem. Vind je een oplossingsmethode voor de laatste twee, dan vind je ook een oplossingsmethode voor het gemengde probleem. Dit blijkt niet zo te zijn, en dat maakt deze drie varianten juist zo interessant. Een belangrijk verschil is de complexiteit van de varianten.

Het Chinese Postbode Probleem lijkt erg veel op het bekende handelsreizigersprobleem. Bij dit probleem krijg je een n aantal steden, waarbij de afstand tussen ieder paar steden gegeven is. Nu dient men de kortst mogelijke route te vinden die minstens één keer langs elke stad komt. De steden in het handelsreizigersprobleem zijn equivalent aan de kruispunten in de wijk van de postbode. De postbode moet echter zorgen dat hij alle straten minstens één keer doorloopt. Het totaal aantal keren dat hij hierbij over een kruispunt komt is niet van belang.

Het vinden van een route waarbij je minstens één keer elke stad passeert is gelijk aan het vinden van een Hamiltoncykel. Wil je een route vinden waarbij je minstens één keer elke straat passeert, dan ben je op zoek naar een Eulercykel. Het verschil in de probleemstelling kan klein lijken, maar het heeft grote gevolgen. Het grootste verschil tussen deze twee problemen is de complexiteit.

1.3 Complexiteit

De complexiteitstheorie bestudeert of problemen oplosbaar zijn of niet en als ze wel oplosbaar zijn, hoe moeilijk ze dan zijn om op te lossen. Een probleem is oplosbaar als er een algoritme bestaat die een oplossing voor het probleem geeft.

Er zijn drie complexiteitsklassen die in deze scriptie van belang zijn. Voor ik deze beschrijf hebben we eerst een aantal definities nodig.

Definitie 1.3.1. *Een **beslissingsprobleem** is een vraagstuk wat beantwoord kan worden met 'ja' of 'nee'. Een beslissingprobleem is **beslisbaar** als er een algoritme bestaat waarmee het probleem kan worden opgelost. Deze oplossing hoeft niet het meest efficiënt te zijn; het bestaan van een algoritme is voldoende.*

Definitie 1.3.2. *Een algoritme kan in **polynomiale tijd** worden uitgevoerd als de benodigde tijd, als functie van de grootte van de invoer, begrensd wordt door een polynoom.*

De drie complexiteitsklassen zijn de volgende:

1. **P**. De complexiteitsklasse P is de klasse die alle beslissingsproblemen bevat die in polynomiale tijd door een algoritme kunnen worden opgelost. Een oplossing voor problemen die in deze klasse vallen is 'gemakkelijk' te vinden.

2. **NP**. De complexiteitsklasse NP is de klasse die alle beslissingsproblemen bevat die in polynomiale tijd door een niet-deterministische Turingmachine kunnen worden opgelost. NP staat dan ook voor niet-deterministisch polynomiaal. Een oplossing voor problemen in deze klasse hoeft niet gemakkelijk te vinden te zijn. Wel is het gemakkelijk om van een gegeven oplossing te verifiëren dat het werkelijk een oplossing voor dit probleem is.

3. **NP-volledig**. Problemen die in de klasse van NP-volledig vallen zijn beslissingsproblemen die minstens zo moeilijk zijn als alle andere beslissingsproblemen in NP. Vindt men een oplossing in polynomiale tijd voor een van de problemen die NP-volledig is, dan zijn alle problemen in NP ook in polynomiale tijd op te lossen. Wederom geldt voor deze problemen dat een oplossing niet gemakkelijk te vinden hoeft te zijn, maar van een gegeven oplossing kan wel geverifieerd worden of het een oplossing is.

Het geven van deze complexiteitsklassen heeft in deze scriptie enkel het doel om te laten zien dat de problemen in die NP-volledig zijn veel ingewikkelder zijn om op te lossen dan problemen in P.

De werking van een Turingmachine en een diepgaande definitie van polynomiale tijd wordt buiten beschouwing gelaten.

We gaan de complexiteit van drie varianten van het probleem vergelijken. Om dit te kunnen doen moeten we dus telkens naar de beslissingsvariant van het probleem kijken. De manier waarop we het Chinese Postbode Probleem formuleren wordt als volgt: 'gegeven een afstand k, is er een route in de graaf te vinden die alle lijnen minstens één keer doorloopt en die een afstand van minder of exact k heeft?'. De beslissingsvariant van het ongerichte en het gerichte Chinese Postbode Probleem vallen in P en de beslissingsvariant van het gemengde Chinese Postbode Probleem is een NP-volledig probleem.
De beslissingsvariant van het handelsreizigersprobleem (gegeven de onderlinge afstanden tussen steden, bestaat er een oplossing die korter is dan een gegeven maximum afstand?) is NP-volledig.

1.4 Vertaalslag naar de wiskunde

Het oplossen van het Chinese Postbode Probleem gebeurt bij iedere variant altijd in twee stappen. Ten eerste vertalen we de gegevens naar de wiskunde via een graaf. In een graaf zijn de kruispunten en splitsingen in een wijk de knopen in de graaf. Loopt er van een kruispunt naar een ander kruispunt een straat, dan zullen de knopen in de graaf verbonden worden door een lijn. Wiskundig definiëren we een graaf en zijn eigenschappen als volgt:

Definitie 1.4.1. *Een **graaf** $G = (V, E)$ is een geordend paar, waarbij $V = \{v_1, ..., v_n\}$ de verzameling knopen in de graaf is en E de verzameling lijnen in de graaf. E bestaat uit alle paren $\{u, v\}$ uit V waar een lijn tussen loopt. We noemen u en v de **eindpunten** van de lijn.*

Definitie 1.4.2. *De **graad** van een knoop $v_i \in V$ is het aantal lijnen waarvoor geldt dat v_i een eindpunt van die lijn is.*

Definitie 1.4.3. *Een **wandeling** in een graaf $G = (V, E)$ is een rij knopen $(v_1, ..., v_n)$, $n \geq 2$, uit V waarbij v_{i-1} en v_i verbonden zijn voor $i = 2, ..., n$. Als $v_1, ..., v_n$ verschillend zijn, dan heet de wandeling een **pad**. Een wandeling heet **gesloten** als $v_1 = v_n$.*

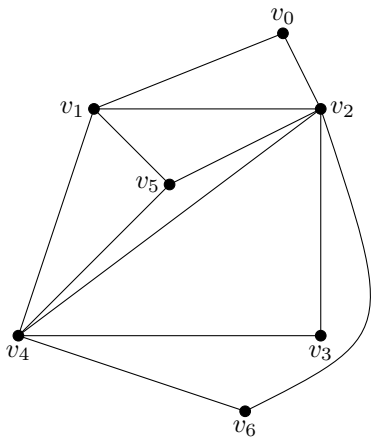
Definitie 1.4.4. *Een graaf heet **samenhangend** als er tussen ieder tweetal knopen een pad loopt.*

Definitie 1.4.5. *Een **Euler-cykel** in een graaf $G = (V, E)$ is een gesloten wandeling $W = (v_1, ..., v_n)$ met de eigenschap dat W elke lijn precies eenmaal doorloopt. Een graaf G heet een **Euler-graaf** als G een Euler-cykel bevat.*

Merk op dat iedere Euler-cykel die je in een Euler-graaf G vindt dezelfde lengte heeft.

Definitie 1.4.6. *Een **multigraaf** is een graaf waarbij er tussen twee knopen uit de graaf meerdere lijnen mogen lopen.*

Om de definities kort toe te lichten maken we gebruik van de volgende voorbeeld-graaf:



In dit voorbeeld heeft knoop v_2 graad 6. Een wandeling in deze graaf is (v_4, v_2, v_0, v_1, v_4, v_6); een gesloten wandeling in deze graaf is $(v_4, v_5, v_1, v_0, v_2, v_4)$. Een pad in deze graaf is (v_4, v_5, v_2, v_0). Deze graaf is zeker samenhangend: tussen elke twee knopen in de graaf bestaat een pad. Deze graaf is geen multigraaf: tussen elk tweetal knopen is er hoogstens één lijn om ze te verbinden (en soms geen).

Wanneer we de gegevens hebben vertaald naar een graaf, dienen we twee stappen te voltooien om zo het probleem op te lossen:

1. Breid de gegeven graaf G uit tot een Euler-graaf G', waarbij we toestaan dat G' een multigraaf is, op een optimale manier.

2. Vind in de graaf G' de Euler-cykel.

Het algoritme om de graaf uit te breiden tot een Euler-graaf G' is per variant verschillend. In dit algoritme moeten we zorgen dat de Euler-cykel die we in de uitgebreide graaf G' zullen vinden de kortst mogelijke totale afstand heeft: een Euler-cykel in een andere uitbreiding zal een langere totale afstand hebben. De uitbreiding die deze eigenschap bevat is de Euler-graaf die we zoeken in stap 1.

Met deze informatie hebben we genoeg kennis om de gerichte, de ongerichte en de gemengde variant van het Chinese Postbode Probleem te bestuderen. [4]

Hoofdstuk 2

Het ongerichte Chinese Postbode Probleem

De ongerichte variant van het Chinese Postbode Probleem houdt in dat de postbode de post rondbrengt in een wijk waar alle wegen tweerichtingswegen zijn. Als we deze informatie omzetten in een graaf $G = (V, E)$ dan wil dit zeggen dat alle lijnen ongericht zijn. Een lijn $e = \{v_{i-1}, v_i\}$ kun je dus zowel doorlopen van v_{i-1} naar v_i als andersom. Omdat de postbode de post in een aangewezen wijk rondbrengt mogen we aannemen dat hij overal kan komen. Als we de plattegrond van de wijk omzetten naar een graaf zal deze dus zeker samenhangend zijn. Wanneer de postbode een straat éénmaal doorloopt, heeft hij alle huizen in die straat van post voorzien. Als we er namelijk vanuit gaan dat de postbode alleen aan één kant van een straat post bezorgt als hij er doorheen loopt, is het probleem op een erg flauwe manier op te lossen. Dit zullen we later zien.

2.1 Algoritme

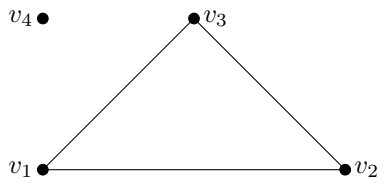
We beschikken nu over een graaf $G = (V, E)$, verkregen uit de plattegrond van de wijk van de postbode. We moeten deze graaf nu uitbreiden tot een Euler-graaf. Hiervoor maken we gebruik van de volgende stelling:

Stelling 2.1. *Een graaf $G = (V, E)$ zonder geïsoleerde knopen is een Euler-graaf dan en slechts dan als G samenhangend is en alle knopen even graad hebben.*

Bewijs. $\Longrightarrow$: Stel dat G een Euler-graaf is. Dan loopt er dus een Euler-cykel door de graaf. De wandeling begint bij het startpunt, zeg v_0. Bij het uitlopen van dit punt krijgt v_0 graad 1. Iedere keer dat deze cykel langs een volgende knoop komt is er een lijn waarover de wandeling de knoop binnenkomt en een andere lijn waarover hij vertrekt. De graad van die knoop zal dus met twee worden verhoogd. Dit gebeurt bij iedere knoop waar de cykel langsloopt. Hij eindigt weer in v_0, waardoor de graad van v_0 nu even is. Omdat deze cykel een Euler-cykel is, hebben we aan het eind van de wandeling alle lijnen precies één keer gehad. De graad van de knopen zijn dus precies de graden die we nu hebben geteld. Alle knopen hebben een even graad, zoals gewenst. Daarnaast is G zeker samenhangend, omdat er een gesloten wandeling door de graaf loopt waar elke knoop minstens één keer in zit. Tussen elk willekeurig tweetal knopen kunnen we uit de Euler-cykel een pad construeren.
$\Longleftarrow$: Zij G een ongerichte graaf zonder geïsoleerde knopen, die samenhangend is en waarvan alle knopen een even graad hebben. We gaan nu de Euler-cykel in deze graaf construeren. Heeft de graaf een Eulercykel, dan is de graaf een Euler-graaf.
Constructie: Kies een willekeurig startpunt, zeg v_0. Telkens als we een lijn hebben gepasseerd, halen we hem weg uit de graaf. Zo is goed bij te houden welke lijnen we al hebben gehad. Heeft een knoop na verwijdering van een lijn nog graad nul, dan halen we die knoop ook weg uit de graaf. Daarnaast moeten we met de volgende eigenschap rekening houden: de graaf mag niet

onsamenhangend worden na het weghalen van die lijn uit de graaf, waarbij we niet meer hoeven te kijken naar lijnen die we al doorgelopen zijn. Verder mag, wanneer het startpunt v_0 nog maar graad 1 heeft, de lijn naar v_0 alleen doorlopen worden wanneer dit de laatste lijn is die nog over is.
Op deze manier vinden we inderdaad een Euler-cykel: elke knoop waar we ingaan heeft een even graad, dus er is minstens één manier om er weer uit te lopen. Is er nog maar precies één manier om uit deze knoop te lopen, dan verwijderen we zowel de lijn als het punt uit de graaf. De graaf zal na deze verwijdering niet onsamenhangend worden: de knoop is een geïsoleerde knoop geworden. Deze verwijderen we uit de graaf. Het is niet zo dat de graaf nog lijnen bevat waar we nu niet meer naar terug kunnen keren, omdat de knoop nog maar één lijn bevatte om weer uit de knoop te komen: de graaf blijft samenhangend.
Wanneer we een knoop inlopen en hier meerdere manieren zijn om eruit te komen, is het van belang te kijken of de graaf onsamenhangend wordt na verwijdering van deze lijn. Als dit zo is betekent dat dat we eerst een andere lijn moeten doorlopen. Dit is een lijn die de graaf niet onsamenhangend maakt. Dit kan altijd. Stel namelijk dat dit niet zou kunnen, en laten we zeggen dat we in knoop u staan. Dan zou dat betekenen dat de graaf na verwijdering van iedere lijn die we vanuit het u kunnen doorlopen onsamenhangend zou worden. Dat kan alleen als we nu in een knoop staan dat op geen enkele manier verbonden is met het startpunt. Maar dat zou betekenen dat de graaf nu al onsamenhangend zou zijn, en er is gegeven dat dat niet zo is. Dus we kunnen altijd een lijn kiezen om te belopen die de graaf, na verwijdering van deze lijn, niet onsamenhangend maakt. Omdat iedere knoop een even graad heeft, zullen we uiteindelijk weer terugkomen in u: u was nog eindpunt van méér dan één lijn, en we verlieten net de knoop. Dan zijn er nog minstens twee lijnen die u als eindpunt hebben die we allebei nog moeten doorlopen. Minstens één van deze lijnen zal te bereiken zijn via een route die we gaan lopen vanuit u. Als we eenmaal terugzijn in u kunnen we opniew gaan bekijken welke lijn we gaan doorlopen. Er is nu namelijk de mogelijkheid dat de lijn die de graaf eerst onsamenhangend maakte, nu een goede keuze is omdat we het deel wat anders een component zou worden nu al verwijderd is omdat we dat eerst hebben doorlopen.
De wandeling die we vinden zal een gesloten wandeling zijn omdat de laatste lijn die we doorlopen de lijn terug is naar het startpunt v_0. G is samenhangend, dus er zal geen lijn zijn die nog niet is doorlopen. De gesloten wandeling is dus een Euler-cykel. Hieruit volgt dat G een Euler-graaf is.
□

Het belang van het uitsluiten van de geïsoleerde knopen wordt duidelijk met de volgende voorbeeld-graaf:



Deze ongerichte graaf heeft een geïsoleerde knoop: v_4. Stel nu dat v_1 het startpunt is. Er is hier een Euler-cykel met startpunt v_1, namelijk de gesloten wandeling (v_1, v_2, v_3, v_1). Alle knopen hebben daarnaast ook een even graad (nul is ook even).

Laten we nu even terugblikken op de eis die we in de intro hebben gesteld. Wanneer de postbode in een wijk enkel aan één kant van een straat de post bezorgt, kunnen we dit erg gemakkelijk uitbreiden tot een Euler-graaf door iedere lijn in de graaf te verdubbelen. Op die manier zal aan beide kanten in de straat de post bezorgd worden. Nu volgt direct dat alle knopen even graad hebben, omdat we alle lijnen hebben verdubbeld. Dit is equivalent aan het verdubbelen van de graad van elke knoop. Dan is de graad dus zeker even, en zal de uitgebreide graad een Euler-graaf zijn. Dit is ook meteen de beste graaf, want de postbode moet iedere straat tweemaal doorlopen

om overal de post te bezorgen. Op deze flauwe manier hebben we dus de optimale Euler-graaf gevonden. We zeggen dus nu dat de postbode elk huis in een straat van post heeft voorzien wanneer hij die straat één keer doorloopt.

Dat de graaf $G = (V, E)$, die we verkrijgen uit de gegeven wijk, samenhangend is, ligt voor de hand. Als de graaf niet samenhangend zou zijn zou dit betekenen voor de postbode dat er bepaalde straten zijn waar hij niet kan komen. Op die manier is er natuurlijk geen post rond te brengen in een wijk. De postbode moet op ieder moment naar elke kruising in de wijk kunnen komen: de graaf moet dus samenhangend zijn.

We gaan nu op zoek naar de kortst mogelijke route die de postbode kan belopen in de wijk waarbij hij elke straat minstens één keer doorloopt en hij eindigt bij zijn startpunt. Als we deze route kunnen vinden, kunnen we het beslissingsprobleem zeker met een 'ja' of een 'nee' beantwoorden. Wanneer dit algoritme binnen polynomiale tijd is op te lossen volgt dat dit probleem in P valt. We gaan het algoritme als volgt construeren: we gaan lijnen in de bestaande graaf $G = (V, E)$ kopiëren om zo een optimale Euler-graaf G' te produceren. Dit staat gelijk aan het volgende probleem:

Minimaliseer

$$\sum_{\{v_i,v_j\}\in E} c_{ij} \cdot x_{ij},$$

waarbij

c_{ij} = de afstand die bij de lijn $\{v_i, v_j\}$ hoort,
x_{ij} = het aantal kopieën van de lijn $\{v_i, v_j\} \in E$ in G'.

Let op: als we dit probleem hebben opgelost, hebben we de optimale Euler-graaf gevonden. Dit is stap 1 die in Hoofdstuk 1 is besproken. De Euler-cykel moet dan nog gevonden worden. Hoe dit gebeurt laat ik zien nadat we de optimale Euler-graaf hebben gevonden.

De lijnen die oorspronkelijk in de graaf G zitten zal de postbode sowieso moeten doorlopen. Het vinden van de kortst mogelijke route hangt dus af van de straten die we vaker dan één keer gaan doorlopen: welke combinatie zorgt voor een zo klein mogelijke afstand, waarbij we nog altijd willen beginnen en eindigen op dezelfde plek? Dit gaan we doen met x_{ij}; het aantal kopieën van een lijn $\{v_i, v_j\}$. Wanneer we de optimale combinatie hebben gevonden, zullen we dit aantal kopieën in G' toevoegen. Later zien we dat er nu daadwerkelijk een Euler-cykel in G' te vinden is, en dat $x_{ij} + 1$ het aantal keren is dat de postbode die straat in zijn route doorloopt.

De lijnen $\{v_i, v_j\} \in E$ waarover we in de som minimaliseren moeten voldoen aan de eis dat alle knopen op dat moment een even graad hebben. Om deze eis te formuleren maken we gebruik van de volgende definities:

$\delta(i)$ de verzameling lijnen waarvan v_i eindpunt is in G,
$T \subseteq V$, de verzameling knopen met oneven graad in G.

Met deze extra definities kunnen we het probleem als volgt formuleren:

Minimaliseer

$$\sum_{\{v_i,v_j\}\in E} c_{ij} \cdot x_{ij},$$

waarbij

$$\sum_{\{v_i,v_j\}\in\delta(i)} x_{ij} = \begin{cases} 1 \pmod 2 & v_i \in T \\ 0 \pmod 2 & v_i \in E \setminus T \end{cases}, \tag{2.1}$$

$$x_{ij} \in \{0,1\} \quad (\{v_i,v_j\} \in E). \tag{2.2}$$

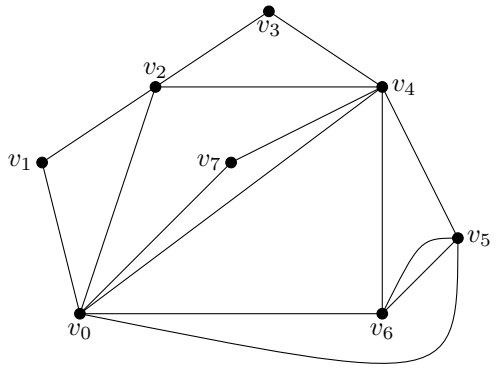
Eigenschap 2.1 garandeert dat er voldaan wordt aan het even zijn van alle graden in de graaf G'. Als $v_i \in T$, dan is zijn graad in de originele graaf G oneven. In de uitgebreide graaf G' moet die graad even worden: er zal dus in totaal 1 (mod 2) kopie van al bestaande lijnen met v_i als eindpunt (lijnen in $\delta(i)$) toegevoegd moeten worden. Als $v_i \in E \setminus T$, dan is zijn graad even in G. Om deze graad even te houden in G' moet het aantal kopieën van de toegevoegde lijnen die v_i als eindpunt hebben even zijn. Met andere woorden, de som moet gelijk zijn aan 0 (mod 2).

Eigenschap 2.2 zorgt voor een grens op de zoekopdracht naar het aantal mogelijke kopieën van een lijn. Bij het uitbreiden van de originele graaf is het mogelijk om een lijn heel vaak toe te voegen. Omdat we naar een route van minimale lengte zoeken, heeft het toevoegen van zo veel kopieën geen zin. Bij het tweemaal kopieëren van een lijn $\{v_i, v_j\}$ kun je de gesloten wandeling (v_i, v_j, v_i) lopen in de graaf. Met deze wandeling schiet de postbode in zijn route echter niks op. Dit zal dus zeker niet voorkomen in de optimale route. Om deze reden kunnen we alle veelvouden van 2 weglaten in het aantal kopieën, en zal x_{ij} ofwel 0 ofwel 1 zijn.

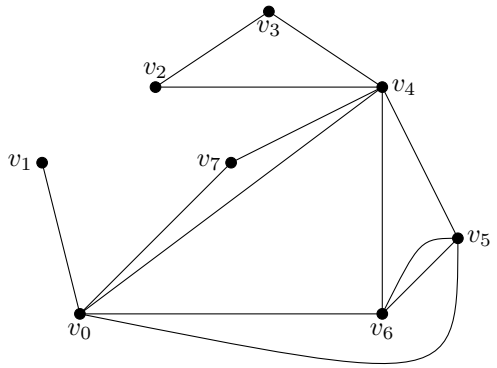
De oplossingsmethode die deze twee eigenschappen toepast op de originele graaf wordt via lineair programmeren opgelost met het Blossom Algorithm. Dit algoritme zal knopen met oneven graad op een optimale manier met andere knopen verbinden. Het algoritme is in 1961 door Jack Edmonds ontwikkeld. In 1965 is hier voor het eerst over gepubliceerd [2].
Problemen die via lineair programmeren kunnen worden opgelost zijn problemen van de makkelijkste soort. Dit ga ik niet bespreken. Leonid Khachiyan (1952 - 2005) heeft de complexiteit van lineair programmeer-problemen uitgebreid uitgelegd [6].

2.2 Voorbeeld

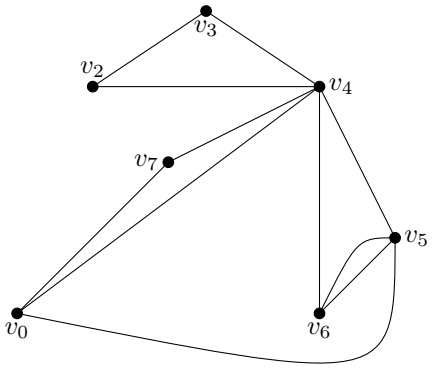
We bekijken nu een graaf die door het Blossom Algorithm zijn optimale vorm al heeft. Het vinden van de Euler-cykel is voor kleine grafen met de hand te doen. Dit doen we op de manier die in het bewijs van Stelling 2.1 uitgelegd is. Hiervoor bekijken we de volgende ongerichte Euler-graaf:



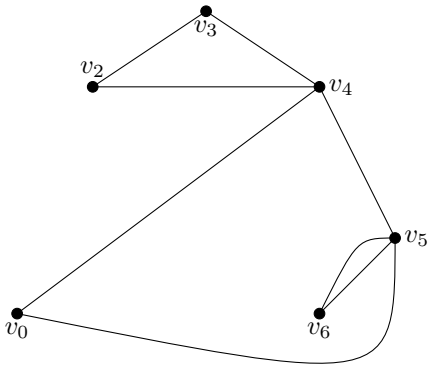
We nemen als startpunt v_0. Het enige wat we moeten doen, is checken of de lijn die we willen belopen na verwijdering uit de graaf de graaf niet onsamenhangend achterlaat. Laten we gaan lopen. De eerste lijn die we nemen mag iedere willekeurige lijn zijn. We lopen naar v_2. Deze lijn halen we weg uit de graaf. Vervolgens mogen we hier wederom elke willekeurige lijn kiezen: het weghalen van de lijn (v_2, v_1) zal de graaf niet onsamenhangend maken, zoals te zien in de afbeelding.



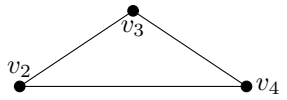
Vervolgens zien we dat we vanuit v_1 maar één keuze hebben: naar v_0 lopen. We verwijderen nu ook punt v_1 uit de graaf, omdat dit punt nu geïsoleerd is. Vanuit v_0 mogen we weer elke lijn kiezen. We lopen naar v_6. De graaf ziet er nu als volgt uit:



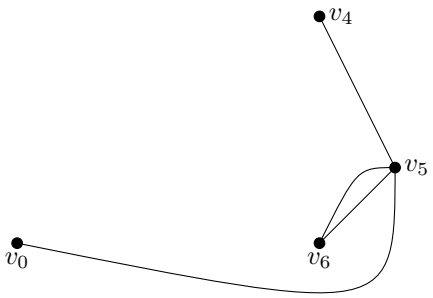
De graaf is nog steeds samenhangend. Vanuit v_6 lopen we naar v_4: dit levert ook geen problemen op. Vanuit v_4 lopen we naar v_7, en de enige mogelijkheid om vanuit v_7 weg te komen is door naar v_0 te lopen. De lijnen die overblijven zijn de volgende:



Nu wordt het interessant. Ons startpunt v_0 heeft graad 2. We gaan nu lopen naar v_4. Het startpunt heeft nu graad 1, en dus is de lijn (v_5, v_0) de laatste lijn die we mogen lopen. De graaf is nog steeds samenhangend na verwijdering van de lijn (v_0, v_4). Vanuit v_4 moeten we oppassen: als we nu naar v_5 lopen zal de graaf onsamenhangend worden:



Dit mogen we dus niet doen. We gaan vanaf v_4 naar v_2, dit kan zonder problemen, lopen dan naar v_3 en dan weer terug naar v_4. De graaf is nog steeds samenhangend.



Vanuit v_4 kunnen we maar één kant op; we lopen naar v_5. In v_5 kunnen we twee kanten op: naar v_6 of naar v_0. Echter is v_0 ons startpunt, en mag de lijn (v_6, v_0) pas als laatste belopen worden. We gaan dus naar v_6. Vanuit v_6 gaan we naar v_5, onze enige optie, en dan blijft als laatste de lijn (v_5, v_0) over. Deze mogen we nu lopen, en op deze manier zijn we terug in v_0 en hebben we alle lijnen van de graaf belopen. Een Euler-cykel in deze graaf is dus:

$(v_0, v_2, v_1, v_0, v_6, v_4, v_7, v_0, v_4, v_2, v_3, v_4, v_5, v_6, v_5, v_0)$.

Hoofdstuk 3

Het gerichte Chinese Postbode Probleem

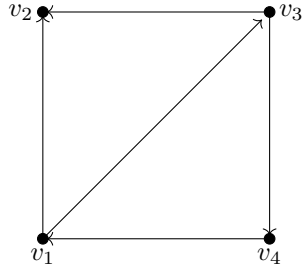
Bij de gerichte variant van het Chinese Postbode Probleem bestaat de wijk waar de postbode zijn post rondbrengt enkel uit eenrichtingswegen. De graaf G die we uit deze informatie verkrijgen zal alleen gerichte pijlen bevatten. We zullen in dit hoofdstuk zien wat een gerichte graaf allemaal inhoudt. Wat we al wel mogen vaststellen is dat de graaf samenhangend moet zijn: als hij dit niet is dan is er in de wijk waarin de postbode rondloopt soms geen route tussen twee kruispunten, en we mogen er vanuit gaan dat dit altijd zo is. Het verschil met het samenhangend zijn van een ongerichte graaf is dat we nu tussen elk tweetal kruispunten een gericht pad hebben lopen. We beschikken nu dus over een gerichte graaf $G = (V, A)$. Hoe gaan we hierin de korst mogelijke route vinden waarbij de postbode elke weg minstens één keer passeert?

3.1 Algoritme

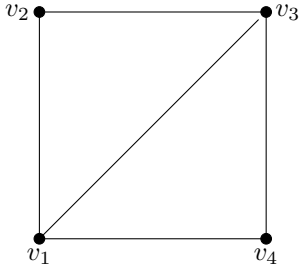
Hoewel we nu een idee hebben van wat een gerichte graaf is in de praktijk is de wiskundige manier van noteren nog niet bekend. Er zijn veel (kleine) details die verschillen met de graaf zoals we hem hiervoor kenden:

Definitie 3.1.1. *Een **gerichte graaf** $G = (V, A)$ is een geordend paar, waarbij $V = \{v_1, ..., v_n\}$ de verzameling knopen in de graaf is en A de verzameling gerichte lijnen in de graaf. A bestaat uit alle paren (u, v) waar een lijn tussen loopt, waarbij deze lijn een richting heeft: deze lijn kan alleen doorlopen worden van u naar v, en niet andersom. Deze gerichte lijnen noemen we **pijlen**. u en v zijn het **begin- en eindpunt** van de pijl.*

We zullen het in dit hoofdstuk ook gaan hebben over 'de onderliggende graaf'. Dit is een graaf die afgeleid is uit de gegeven gerichte graaf, waarbij we alle gerichte lijnen omzetten naar lijnen: de richting wordt dus buiten beschouwing gelaten. Ter illustratie geef ik een klein voorbeeld:



De gerichte graaf $G = (V, A)$



De onderliggende graaf

Definitie 3.1.2. *Een **Euler-cykel** in een gerichte graaf $G = (V, A)$ is een gesloten wandeling $W = (v_1, ..., v_n)$ met de eigenschap dat W elke pijl precies eenmaal in de richting van de pijl doorloopt. Een gerichte graaf G is een **Euler-graaf** als G een Euler-cykel bevat.*

Voor een gerichte graaf is Stelling 1 uit paragraaf 2.1 niet meer voldoende. De eis dat elke knoop een even graad moet hebben moet anders worden geformuleerd, omdat we nu te maken hebben met pijlen. We eisen daarnaast dat de gerichte graaf die we verkrijgen een sterk samenhangende graaf is.

Definitie 3.1.3. *Een **sterk samenhangende** gerichte graaf is een graaf waarbij er een gericht pad loopt tussen elk tweetal knopen.*

We mogen er vanuit gaan dat een wijk altijd sterk samenhangend is. Wanneer hij dat niet zou zijn krijgt de postbode het erg moeilijk om in die wijk post te bezorgen: dan kan het voorkomen dat hij vanaf een bepaalde kruising niet naar een andere kruising kan komen en dat is ondenkbaar in een wijk. In dit hoofdstuk maken we gebruik van de volgende stelling:

Stelling 3.1. *Een gerichte, sterk samenhangende graaf G is een Euler-graaf dan en slechts dan in elke knoop het aantal inkomende pijlen gelijk is aan het aantal uitgaande pijlen.*

Bewijs. $\Longrightarrow$: Stel dat G een Euler-graaf is. Dan loopt er dus een Euler-cykel door de graaf. De wandeling begint bij het startpunt, zeg v_0. Dit startpunt lopen we uit, dus v_0 bevat een uitgaande pijl. Iedere keer dat deze cykel langs een volgende knoop komt is er een zijde waarover de wandeling de knoop binnenkomt en een zijde waarover hij vertrekt. Het aantal inkomende en uitgaande pijlen is dus gelijk. Hij eindigt weer in v_0, waardoor ook v_0 een even aantal inkomende en uitgaande pijlen bevat. Omdat deze cykel een Euler-cykel is, hebben we aan het eind van de wandeling alle pijlen precies één keer gehad. Alle knopen hebben dus een even aantal inkomende en uitgaande pijlen, zoals gewenst.
$\Longleftarrow$: Zij G een sterk samenhangende gerichte graaf, waarbij elke knoop een gelijk aantal inkomende en uitgaande pijlen bevat. We gaan nu de Euler-cykel in deze graaf construeren: dan is de graaf een Euler-graaf. Kies een willekeurig startpunt, zeg v_0. Telkens als we een pijl hebben gepasseerd, halen we hem weg uit de graaf. Zo is goed bij te houden welke pijlen we al hebben gehad. Daarnaast moeten we met de volgende eigenschap rekening houden: de graaf mag niet onsamenhangend worden na het weghalen van die pijl uit de graaf, waarbij we voor het controleren van onsamenhangendheid de richting van de lijnen even buiten beschouwing laten (we kijken hiervoor naar de onderliggende ongerichte graaf). Verder mag, wanneer het startpunt v_0 nog maar één inkomende pijl bevat, de pijl naar v_0 alleen doorlopen worden wanneer dit de laatste lijn is die nog over is.
Op deze manier vinden we inderdaad een Euler-cykel: elke knoop waar we ingaan heeft een gelijk aantal inkomende en uitgaande, dus er is minstens één manier om er weer uit te lopen.
Is er nog maar precies één manier om eruit te lopen, dan verwijderen we zowel de lijn als het punt uit de graaf. De graaf zal na deze verwijdering niet onsamenhangend worden: de knoop is een geïsoleerde knoop geworden. Deze halen we uit de graaf. Het is niet zo dat de graaf nog pijlen bevat waar we niet meer naar terug kunnen keren, omdat de knoop alleen nog één uitgaande pijl bevatte: de graaf blijft samenhangend.
Wanneer we een knoop inlopen en hier meerdere manieren zijn om eruit te komen, is het van belang te kijken of de graaf onsamenhangend wordt na verwijdering van deze pijl. Als dit zo is betekent dat dat er een wandeling is die vanuit dit punt gelopen kan worden waardoor één deel waarin de graaf anders gescheiden zou worden nu al gelopen is. Vervolgens kunnen we de lijn doorlopen die we eerder wilden doorlopen, waardoor de graaf nu niet onsamenhangend wordt. De wandeling die we vinden zal een gesloten wandeling zijn omdat de laatste pijl die we doorlopen de lijn terug is naar het startpunt v_0. G is sterk samenhangend, dus er zal geen pijl zijn die nog niet is doorlopen. De gesloten wandeling is dus een Euler-cykel. Hieruit volgt dat G een Euler-graaf is.

□

Laat nu I de verzameling knopen $v_i \in V$ zijn die meer inkomende dan uitgaande pijlen bevatten. Het verschil noemen we s_i. Zij J de verzameling knopen $v_j \in V$ zijn die meer uitgaande dan ingaande pijlen bevatten. Dit verschil noteren we met d_j. We gaan kopieën van bestaande pijlen in de graaf G toevoegen waarbij we gaan minimaliseren op de totale lengte. Hierbij is c_{ij} de lengte van de pijl (v_i, v_j) en x_{ij} het aantal kopieën van de pijl (v_i, v_j) in G. Het vinden van een uitbreiding van G waarvan de totale lengte minimaal is én aan de eisen van een gerichte Euler-graaf voldoet kunnen we nu als volgt formuleren:

Minimaliseer

$$\sum_{v_i \in I} \sum_{v_j \in J} c_{ij} \cdot x_{ij},$$

waarbij

$$\textstyle\sum_{v_j \in J} x_{ij} = s_i \quad (v_i \in I) \tag{3.1}$$

$$\textstyle\sum_{v_i \in I} x_{ij} = d_i \quad (v_j \in J) \tag{3.2}$$

$$x_{ij} \in \mathbb{Z}, x_{ij} \geq 0 \quad (v_i \in I, v_j \in J) \tag{3.3}$$

Met eis 3.1 zorgen we dat aan elke $v_j \in J$ het aantal kopieën van ingaande pijlen wordt toegevoegd die hij nog mist om evenveel inkomende als uitgaande pijlen te bezitten. Eis 3.2 zorgt voor een vergelijkbaar resultaat, maar dan voor de $v_i \in I$, dus voor de knopen die op dit moment meer inkomende dan uitgaande pijlen bezitten. Tot slot is het aantal kopieën dat wordt toegevoegd aan de graaf G' altijd positief, zoals vermeld in eis 3.3.

De optimale waarden voor de x_{ij} die uit het oplossen van dit probleem volgen, representeren het aantal extra keren dat een lijn (v_i, v_j) moet worden doorlopen in de kortst mogelijke route. Het vinden van deze optimale Euler-graaf is een lineair-programmeer probleem. Zo'n optimalisatieprobleem is een 'gemakkelijk' probleem. Het is op te lossen binnen polynomiale tijd en behoort tot de gemakkelijke problemen. Ondanks dat het gemakkelijk is, ga ik er niet dieper op in omdat het toch technische details bevat [6].

Wanneer we dit probleem hebben opgelost en G hebben uitgebreid tot een Euler-graaf G' moeten we de Euler-cykel in deze graaf vinden. Dit doen we met de volgende prodecure, aangedragen door van Aardenne-Ehrenfest en de Bruijn, in 1951.

3.2 Het Van Aardenne-Ehrenfest en De Bruijn algoritme

Tatjana Pavlovna Van Aardenne-Ehrenfest (1905 – 1984) en Nicolaas Govert De Bruijn (1918 – 2012) waren Nederlandse wiskundigen. Ze hebben bijgedragen aan ontdekkingen in de grafentheorie. Eén van die ontdekkingen is een algoritme dat naar hen beiden vernoemd is. Dat algoritme maakt gebruik van een maximale opspannende boom. Om te begrijpen wat dit inhoudt hebben we eerst een aantal andere definities nodig:

Definitie 3.2.1. *Een **circuit** is een gesloten wandeling $(v_0, ..., v_k)$ met $k \geq 3$ en alle v_i zijn verschillend voor $i = 1, ..., k$ (zogenaamd een gesloten pad).*

Definitie 3.2.2. *Een **boom** is een samenhangende graaf zonder circuits.*

Definitie 3.2.3. *Een **opspannende boom** in een gerichte graaf $G = (V, A)$ met een wortel v_r is een boom in de onderliggende ongerichte graaf van G met extra eisen. Er geldt namelijk voor elke knoop $v_n, n \neq r$, in de boom, dat deze knoop in de gerichte graaf precies één gerichte pijl vanaf v_n in de boom heeft zitten. Alle pijlen richting de wortel v_r zitten in de boom en er zit geen pijl in de boom die gericht is vanaf v_r. Bijgevolg zal er voor ieder punt precies één manier zijn om via de boom bij de wortel te komen.*

In een opspannende boom mag het niet meer mogelijk zijn om een pijl toe te voegen aan deze boom zonder dat er een circuit ontstaat. Voor het bestaan van een circuit kijken we naar de onderliggende graaf van de boom. Daarin geldt het volgende: er is een circuit als er een gesloten wandeling van 3 of meer lijnen in de boom te maken is. Je kan van een boom dus een opspannende boom maken door dit soort pijlen toe te voegen tot het niet meer kan. Van dit soort opspannende bomen maakt dit algoritme gebruik.

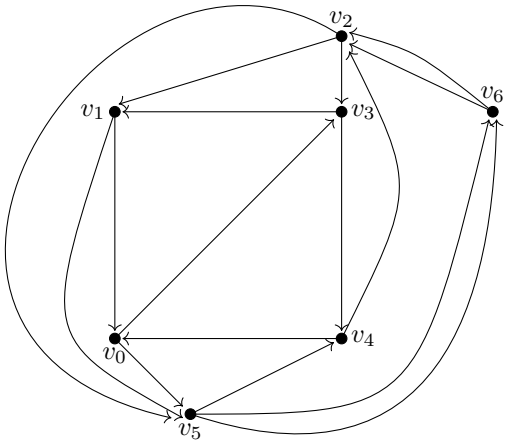
Zij $G' = (V, A')$ de uitbreiding die we van $G = (V, A)$ gemaakt hebben. Deze graaf is zeker een sterk samenhangende graaf, want het is een uitbreiding van de sterk samenhangende graaf G. Het algoritme zegt nu het volgende:

Vind in de graaf G' een maximaal opspannende boom. Nummer nu alle pijlen als volgt: bekijk in iedere knoop de uitgaande pijlen. Nummer deze van 1 t/m k, met k het aantal uitgaande pijlen. De pijl die in de maximaal opspannende boom zit krijgt de hoogst mogelijke index, de andere uitgaande pijlen mogen op willekeurige manier genummerd worden. De Euler-cykel vinden we dan op de volgende manier: Zij v_0 het startpunt. We verlaten het startpunt via de laagst genummerde pijl. Wanneer we een knoop v_n bereiken, verlaten we deze knoop via de laagst genummerde, nog ongebruikte pijl. Zo doorlopen we de hele graaf en we eindigen door de manier van nummeren weer in v_0. Deze gesloten wandeling is de Euler-cykel die we zoeken.

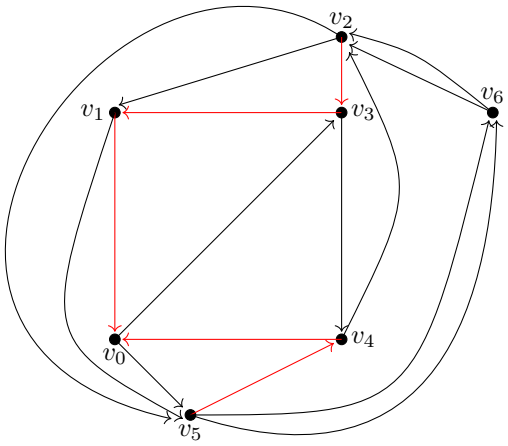
Het bewijs dat deze procedure inderdaad een Euler-cykel geeft is na te lezen in het artikel *'Matching, Euler tours and the Chinese postman; Edmonds and Johnson.* [3]

3.3 Voorbeeld

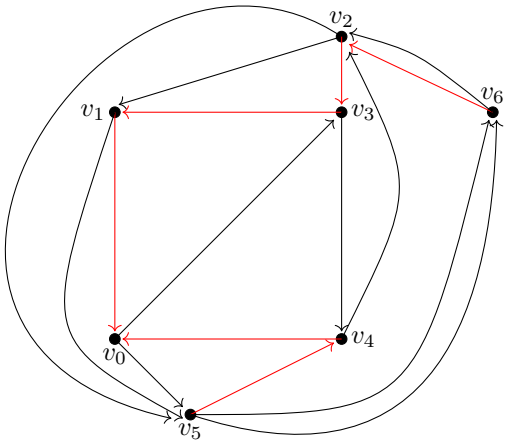
Het algoritme om een Euler-cykel te vinden is erg abstract. Dit gaan we verduidelijken met een voorbeeld. We kijken naar de volgende graaf, welke al een Euler-graaf is:



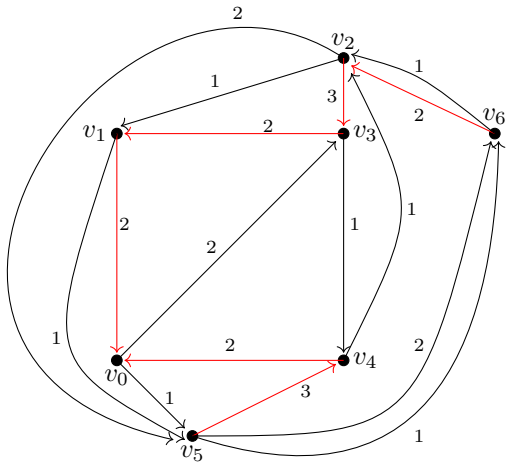
We gaan nu een opspannende boom in de graaf maken. We zeggen dat v_0 de wortel is. We beginnen met de boom aangegeven met rood:



Dit is nog zeker geen opspannende boom. Er is een pijl van een punt dat nog niet in de opspannende boom zit naar een punt dat wel in de opspannende boom zit, welke geen circuit laat ontstaan: (v_6, v_2). Elke andere pijl die we toevoegen zal een circuit laten ontstaan. Met deze pijl erbij hebben we een opspannende boom geconstrueerd.



Dan gaan we nu de pijlen nummeren volgens de nummering die eerder geïntroduceerd is: 'Nummer deze van 1 t/m k, met k het aantal uitgaande pijlen. De pijl die in de maximaal opspannende boom zit krijgt de hoogst mogelijke index, de andere uitgaande pijlen mogen op willekeurige manier genummerd worden'. Met deze nummering kunnen we bijvoorbeeld de volgende graaf maken:



Deze graaf is zeker niet uniek: er zijn veel manieren om de graaf te nummeren, en dus ook veel verschillende Euler-cykels in deze graaf. Met deze nummering vinden we, als we vanuit v_0 vertrekken en telkens de pijl met de laagste index pakken die we nog niet hebben doorlopen, een Euler-cykel. Volgen we het algoritme in deze graaf, dan vinden we de volgende Euler-cykel:
$(v_0, v_5, v_6, v_2, v_1, v_5, v_6, v_2, v_5, v_4, v_2, v_3, v_4, v_0, v_3, v_1, v_0)$.

Hoofdstuk 4

Het gemengde Chinese Postbode Probleem

Het gemengde CPP wil zeggen dat in de wijk waar de postbode rondloopt er zowel eenrichtingswegen als tweerichtingswegen voorkomen. Het probleem om in deze wijk een zo kort mogelijke route voor de postbode te vinden blijft gelijk. Van dit probleem, hoewel het een combinatie is van twee beslisbare problemen die binnen polynomiale tijd opgelost kunnen worden, is niet bekend of het binnen polynomiale tijd is op te lossen.

Van een gemengde graaf mogen we stellen dat hij samenhangend is. Wederom kan het in een wijk niet voorkomen dat de postbode op een moment op een kruispunt staat en de postbode het kruispunt waar hij daarna naartoe moet niet kan bereiken. Om de samenhang van een gerichte graaf na te gaan, behandelen we alle lijnen als wegen die je via twee richtingen kan doorlopen. De pijlen kunnen alleen doorlopen worden in de richting die ze aangeven. Het moet dus mogelijk zijn om tussen ieder willekeurig tweetal kruispunten een weg te vinden.

De beslissingsvariant van het gemengde CPP houdt het volgende in: gegeven een graaf $G = (V, E, A)$, waarbij E bestaat uit alle paren $\{u, v\}$ uit V waar een lijn tussen loopt, en A bestaat uit alle paren (u, v) uit V waar een pijl tussen loopt. De lengte per (gerichte) lijn is c_{ij}. Zij k een willekeurig geheel getal. Bevat G een route van lengte k of minder? Dit probleem is NP-volledig, oftewel een van de moeilijkst oplosbare problemen in NP.

4.1 NP-volledig

Definitie 4.1.1. *Een **gemengde graaf** noteren we als $G = (V, E, A)$. V is de verzameling knopen in de graaf, E is de verzameling lijnen in de graaf en A is de verzameling gerichte lijnen in de graaf.*

Definitie 4.1.2. *Een **Euler-cykel** in een gemengde graaf $G = (V, E, A)$ is een gesloten wandeling $W = (v_1, ..., v_n)$ met de eigenschap dat W iedere lijn en iedere gerichte lijn in de graaf precies eenmaal heeft doorlopen. Een gemengde graaf G heet een **Euler-graaf** als G een Euler-cykel bevat.*

Wanneer er van een samenhangende gemengde graaf gegeven is dat het een Euler-graaf is, is met een paar aanpassingen op het Van Aardenne-Ehrenfest – De Bruijn algoritme een Euler-cykel te vinden. Het vinden van een meest optimale uitbreiding van de graaf waarbij deze uitbreiding ook nog eens een Euler-graaf is, is het lastige deel. Dit gaan we zien in de volgende stelling:

Stelling 4.1. *Een gemengde graaf $G = (V, E, A)$ is een Euler-graaf dan en slechts dan als elke knoop in de onderliggende graaf een even graad heeft. Iedere knoop mag eindpunt zijn van zowel lijnen als pijlen. Daarnaast, als $S \subseteq V$ moet het verschil tussen aantallen pijlen van S naar $V \setminus S$ en van $V \setminus S$ naar S gelijk of minder zijn dan het aantal ongerichte lijnen die $V \setminus S$ en S verbinden. Deze eigenschap heet de **balanced set condition**.*

Wanneer we nu weer de meest optimale uitbreiding van de graaf gaan zoeken, gaan we de vergelijking die we moeten minimaliseren weer schrijven zoals in secties 2.1 en 3.1, waarbij de x_{ij} aan bepaalde eisen moet voldoen (zoals bijvoorbeeld de eisen (3.1), (3.2) en (3.3)). Het oplossen van dit lineair-programmeer probleem wordt uitgebreid uitgelegd in *An Optimal Algorithm for the Mixed Chinese Postman Problem, Nobert and Picard* [8]
.

Om te bewijzen dat we te maken hebben met een NP-volledig probleem, gaan we het volgende doen: We stellen dat we een oplossing hebben voor dit probleem. Dan gaan we bewijzen dat we, met gebruik van deze oplossing voor dit probleem, een oplossing kunnen vinden voor een probleem waarvan al bekend is dat het een NP-volledig probleem is. Van dit probleem is het zeer onwaarschijnlijk dat hier een oplossing voor gevonden wordt in polynomiale tijd, en daaruit volgt dat het gemengde Chinese Postbode Probleem minstens zo moeilijk is als dit NP-volledige probleem, en zelf dus ook in de klasse NP-volledig valt. Het NP-volledige probleem dat we gaan bekijken is het 3-vervulbaarheidsprobleem.

4.1.1 3-SAT

Het 3-vervulbaarheidsprobleem (3-SAT, afkorting van het Engelse 3-satisfiability) is een probleem uit de logica. Het verwijst naar het bepalen of een logische propositie vervuld kan worden. Dit wil zeggen dat er een waarde 'waar' of 'onwaar' aan elke literaal in de propositie kan worden toegekend, op zo'n manier dat de hele propositie waar is. Zo'n propositie kan er als volgt uit zien:

$$(p_1 \vee p_2 \vee \neg p_4) \vee (p_3 \vee \neg p_5 \vee \neg p_1) \vee ... \vee (p_2 \vee p_6 \vee \neg p_6)$$

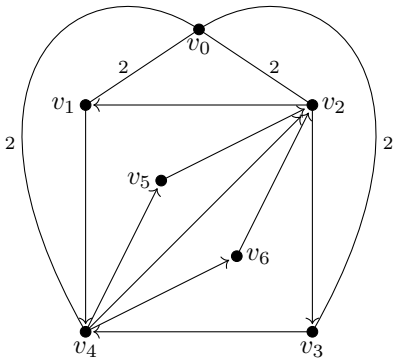
Waarbij de literalen de variabelen zoals p_1 en $\neg p_1$ zijn en elke clausule een term is waar drie literalen in staan. In deze variant kijken we dus naar de conjunctieve normaalvorm van de propositie. In het geval van het 3-vervulbaarheidsprobleem staan er in elke clausule exact drie literalen, waarbij er nooit twee dezelfde literalen in een clausule mogen zitten.
Dit probleem is een van de oudst bekende NP-volledige problemen. Het is een van de problemen uit Karp's '21 NP-volledige problemen' en het wordt vaker gebruikt om te illustreren dat een ander probleem (minstens) net zo moeilijk is als dit probleem [5].

We zullen nu het bewijs bekijken dat het gemengde Chinese Postbode Probleem NP-volledig is.

4.1.2 Bewijs

We gaan een gemengde graaf G op een optimale manier uitbreiden tot een Euler-graaf. Dit gaan we weer doen door kopieën van al bestaande lijnen en pijlen in de graaf toe te voegen en zo een graaf G' te krijgen. De postbode moet elke straat van de wijk minstens één keer doorlopen dus we kunnen bij het minimaliseren op de totale lengte net zo makkelijk de totale lengte van de originele graaf weglaten. We gaan dus minimaliseren op de totale lengte van de lijnen en pijlen die we gaan toevoegen om een Euler-graaf van de graaf te maken. Het gemengde probleem is moeilijker dan het gerichte of ongerichte probleem, en daarom maken we gebruik van een hulpgraaf. Deze beschrijven we in het volgende lemma:

Lemma 4.1. *In de gemengde graaf van Figuur 2 zijn de lengtes van de (gerichte) lijnen tenzij anders vermeld gelijk aan 1. De minimale lengte van de lijnen die we toevoegen om het er een Euler-graaf van te maken is 2. Voor de route van minimale lengte geldt dat we de ongerichte lijnen* (v_0, v_1)*,* (v_0, v_2)*,* (v_0, v_3) *en* (v_0, v_4) *in de volgende richting doorlopen: ofwel de lijnen* (v_0, v_1) *en* (v_0, v_3) *lopen we richting* v_0 *en de lijnen* (v_0, v_2) *en* (v_0, v_4) *doorlopen we vanuit* v_0*, of vice versa.*



Figuur 2

Bewijs. Het is duidelijk dat als we op zoek gaan naar de meest optimale route in deze graaf, twee van de lijnen (v_0, v_1), (v_0, v_2), (v_0, v_3), (v_0, v_4) v_0 moeten verlaten en de andere twee moeten v_0 ingaan. Is dit namelijk niet het geval, dan moeten minstens twee extra kopieën van sommige van deze vier lijnen worden toegevoegd in de route, en dat resulteert in een minder optimale route dan de eerdergenoemde.
Als gevolg zijn er dus zes mogelijke orientaties van de nog ongerichte lijnen. Door de gevallen apart te bekijken volgt het bewijs. De zes mogelijke orientaties met de corresponderende lengtes van de optimale route zijn in tabel 4.1.2 te vinden. □

Lijnen die v_0 binnenkomen	(v_0, v_1) (v_0, v_2)	(v_0, v_3) (v_0, v_4)	(v_0, v_1) (v_0, v_4)	(v_0, v_2) (v_0, v_3)	(v_0, v_1) (v_0, v_3)	(v_0, v_2) (v_0, v_4)
Minimale lengte toegevoegde lijnen	3	4	4	3	2	2

Tabel 4.1

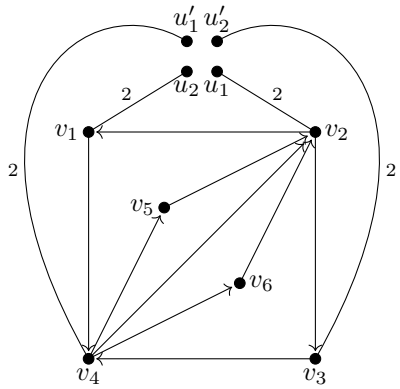
Ik zal van de eerste kolom hierboven uitleggen hoe ze aan de waarde voor de minimale lengte van de toegevoegde lijnen komen en wat in deze graaf dan een Euler-cykel kan zijn.

De bovenste rij van de tabel geeft aan welke ongerichte lijnen we in de richting naar v_0 zullen lopen. Geef deze lijnen voor het gemak een pijl naar v_0 toe. De twee ongerichte lijnen die overblijven geven we een richting van v_0 weg, dus gericht naar v_3 en v_4. Is dit gedaan, dan zien we dat er een paar knopen zijn die nog geen gelijk aantal inkomende en uitgaande pijlen bevatten. Dit is het geval bij v_1 en v_3. Er is één inkomende pijl tekort in v_1, en één uitgaande pijl tekort in v_3. We gaan nu het kortst mogelijke pad van v_3 naar v_1 vinden. Als we die toevoegen aan de bestaande graaf, zullen alle knopen een gelijk aantal inkomende en uitgaande pijlen bevatten, en kunnen we een Euler-cykel vinden. Het kortst mogelijke pad is de volgende: (v_3, v_4, v_2, v_1), dit heeft een lengte 3 (het onderste getal in de tabel!). Stel nu dat v_4 het startpunt is. De Euler-cykel die we nu kunnen lopen is de volgende:

$$(v_4, v_2, v_0, v_3, v_4, v_6, v_2, v_3, v_4, v_5, v_2, v_1, v_4, v_2, v_1, v_0, v_4)$$

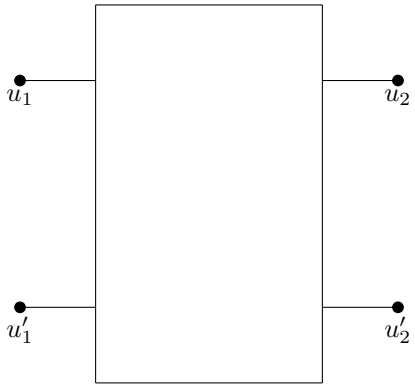
Merk op dat er ook andere Euler-cykels in de verkregen graaf te vinden zijn, maar dat ze allemaal dezelfde lengte hebben. De lengte van deze route is dus de totale route van de originele graaf opgeteld bij de lengte van de kopieën, dus $17 + 3 = 20$.

We gaan nu de tekening in Figuur 2 vereenvoudigen. We verwijderen de knoop v_0. Alle lijnen waarvan v_0 een eindpunt was, krijgen nu een nieuw eindpunt. Dit zijn knopen die worden toegevoegd. In de tekening zie je deze knopen als u_1, u_1', u_2 en u_2'.



Figuur 2(a)

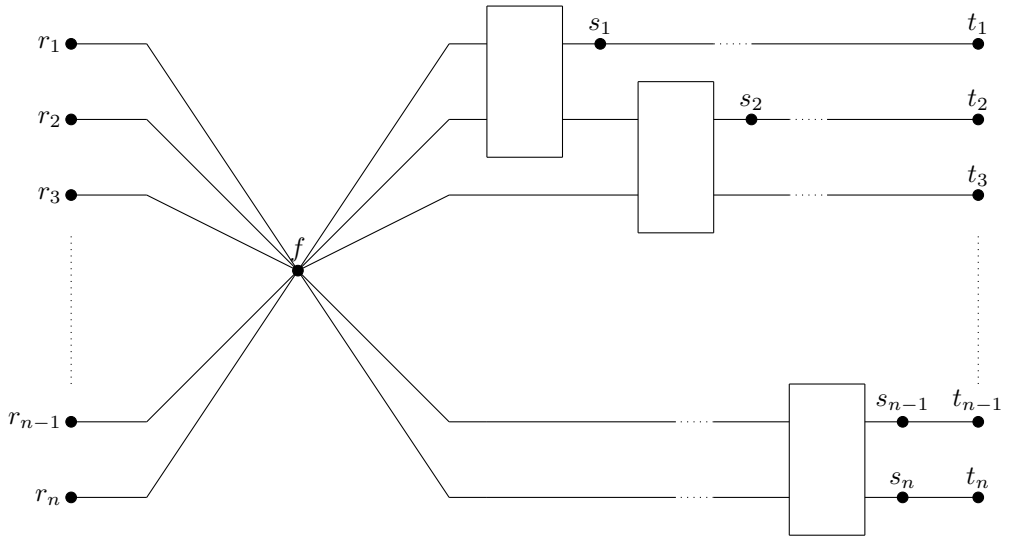
Van deze figuur zijn alleen de lijnen (v_1, u_2), (v_2, u_1), (v_3, u_2') en (v_4, u_1') belangrijk, dus we gaan het vanaf nu weergeven met de volgende schematische weergave:



Figuur 2(b): Graaf C

Deze weergave noemen we graaf C. In deze schematische weergave zal de orientatie van de lijnen ofwel naar links zijn ofwel naar rechts. Dit volgt uit het lemma dat we net bewezen hebben. Stel nu dat we een gemengde graaf M hebben die m kopiën van deze graaf C heeft, dan heeft de graaf een optimale route van lengte minimaal $2m$.

We definiëren nu de uitwaaigraaf $F(n)$ als de gemengde graaf in Figuur 3. Als $F(n)$ nu deel is van een graaf M, en $F(n)$ draagt enkel $2(n-1)$ bij aan de lengte van de optimale route in M, dan dwingt de aanwezigheid van de $n-1$ kopieën van C de lijnen (s_1, t_1), (s_2, t_2), ..., (s_n, t_n) ofwel allemaal $F(n)$ te verlaten, of in te gaan. Het omgekeerde geldt voor de lijnen (f, r_1), (f, r_2), ..., (f, r_n).



Figuur 3

Zij nu D een willekeurig 3-SAT probleem met variabelen $x_1, x_2, ..., x_n$ en clausules $L_1, L_2, ..., L_m$. We zeggen dat c_j het aantal keren is dat x_j voorkomt in D, en d_j is het aantal keren dat $\neg x_j$ voorkomt in D. Zij $g_j = max(c_j, d_j) > 0$ en $k = 2\sum_{j=1}^{n}(g_j - 1)$. We gaan nu een gemengde graaf M construeren met lengtes zodanig dat M een route van lengte k of minder heeft dan en slechts dan als D vervulbaar is. We maken hierbij gebruik van de uitwaaigraaf voor iedere clausule, en gaan de waardes voor de literalen omzetten naar richtingen. Dit wil zeggen dat we elke ongerichte pijl in de graaf uit *Figuur 2(b)* toekennen aan een literaal uit de clausule. Vervolgens geven we de pijl een richting ten aanzien van de waarde die aan de literaal is gegeven in de oplossing voor het 3-SAT probleem. Dit zal een Euler-graaf geven. Anderzijds geven we de literalen, als we een Euler-graaf in deze gemengde graaf gevonden hebben, de waarde die we toekennen aan de richting van de lijn. Op die manier kunnen we alle literalen een waarde toekennen zodat de gehele propositie waar is, en hebben we een oplossing voor het 3-SAT probleem gevonden. De details van het bewijs zijn te vinden in het artikel van Papadimitriou, paragraaf 3 [9].

Bibliografie

[1] K. Chen en T.L. Saaty. "Hoover's Problem". In: *Mathematics Magazine* 51 (1978), p. 288–292.

[2] J. Edmonds. "Paths, trees, and flowers". In: *Can. J. Math,* 17 (1965), p. 449–467.

[3] J. Edmonds en E.L. Johnson. "Matching, Euler tours and the Chinese postman". In: *Mathematical Programming* 5 (1973), p. 115–123.

[4] H. A. Eiselt, M. Gendreau en G. Laporte. "Arc Routing Problems, Part 1: The Chinese Postman Problem". In: *Operations Research 43* 3 (1995), p. 231–242.

[5] R.M. Karp. *Reducibility among combinatorial problems.* Berkeley: University of California at Berkeley, 1992.

[6] L. G. Khachiyan. "A Polynomial Algorithm in Linear Programming". In: *Soviet Math. Dokl.* 20 (1979), p. 191–4.

[7] M. Kwan. "Graphic programming using odd or even points". In: *Chinese Mathematics* 1 (1962), p. 273–277.

[8] Y. Nobert en J. Picard. "An Optimal Algorithm for the Mixed Chinese Postman Problem". In: *NETWORKS* 27 (1996), p. 95–408.

[9] C.H. Papadimitriou. "On the Complexity of Edge Traversing". In: *Association for Computing Machinery* 23 (1976), p. 544–554.

[10] V. Pieterse en P.E. Black. *Dictionary of Algorithms and Data Structures.* CRC Press LLC, 1999.