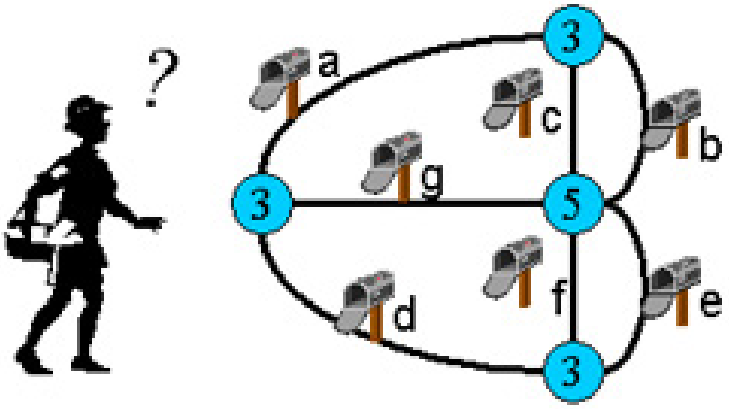
Chinese postbodeprobleem

Dorthe Van Waarden

9 juli 2010

Eindverslag Bachelorproject

Begeleiding: dr. Marcel van de Vel



KdV Instituut voor wiskunde
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Universiteit van Amsterdam

Samenvatting

Het Chinese postbodeprobleem is er op gericht om, gegeven een samenhangende en gewogen graaf, de kortste gesloten wandeling te vinden die elke lijn tenminste één keer bevat. Om dit op te lossen, splitsen we het probleem op in twee gevallen. Allereerst bekijken we de eenvoudige situatie, dat we te maken hebben met een Eulergraaf. Vervolgens richten we ons meer algemeen op het oplossen van het probleem voor een willekeurige graaf. Hiervoor bespreken we verschillende algoritmes die nodig zijn om tot een oplossing te komen, waaronder het kortstepad-algoritme van Dijkstra en Edmonds' bloesem-algoritme. Daarnaast bespreken we de oplossing van het Chinese postbodeprobleem aan de hand van lineair programmeren. Vervolgens worden enkele variaties op het postbodeprobleem gegeven. Als afsluiting komen nog een paar andere mooie toepassingen van Eulergrafen aan bod.

Gegevens

Titel: Chinese postbodeprobleem

Auteur: Dorthe Van Waarden, djvw@science.uva.nl, 5801974
Begeleider: dr. Marcel van de Vel
Tweede beoordelaar: Prof. dr. Alexander Schrijver

Einddatum: 9 juli 2010

Korteweg de Vries Instituut voor Wiskunde
Universiteit van Amsterdam
Science Park 904, 1098 XH Amsterdam
`http://www.science.uva.nl/math`

Inhoudsopgave

1 Inleiding **2**

2 Eulergrafen **4**
2.1 Euler 5

3 Korstepad-algoritmes **8**
3.1 Het algoritme van Dijkstra 8
3.2 Het algoritme van Floyd en Warshall 11

4 Postbodegrafen **12**
4.1 Voorbeeld 14

5 Matchings **16**

6 Lineair Programmeren **21**

7 Vergelijkbare problemen **28**
7.1 Postbodeprobleem voor gerichte grafen 29
7.2 Postbodeprobleem voor gemengde grafen 29
7.3 Handelsreizigersprobleem 30

8 Toepassingen Eulergrafen **31**
8.1 Domino 31
8.2 RNA-ketens 32
8.3 Teleprintersprobleem 35

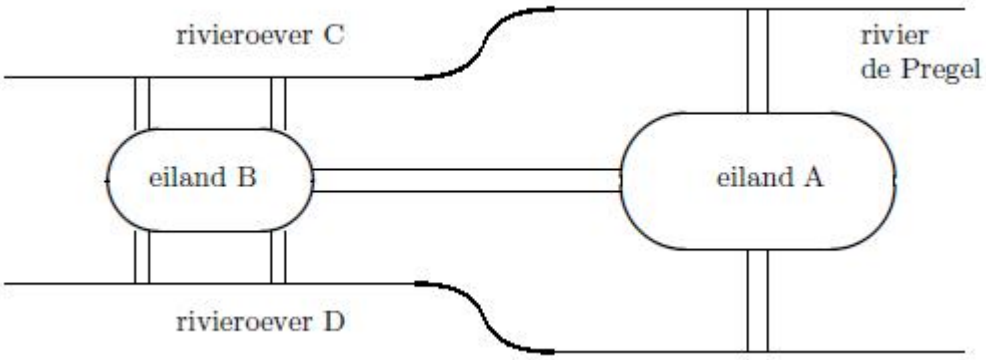
9 Populaire samenvatting **38**

10 Appendix **42**
10.1 Grafentheorie 42
10.2 Lineair Programmeren 44

Hoofdstuk 1

Inleiding

In Koningsbergen (tegenwoordig Kaliningrad) stroomt de rivier de Pregel. Midden in de rivier liggen twee eilandjes. De stad en de eilandjes zijn met elkaar verbonden door zeven bruggen.



Figuur 1.1: Plattegrond van de stad Koningsbergen.

De inwoners vroegen zich af of het mogelijk was om een rondwandeling te maken, waarbij elke brug precies eenmaal gepasseerd werd. In 1736 beschreef Leonard Euler (1707-1783) dit zogenoemde 'Koningsberger bruggenprobleem' in de jaarlijkse uitgave van de St-Petersburgse Academie van Wetenschappen, en dat betekende -achteraf gezien- de geboorte van de grafentheorie. Euler ontwikkelde geen algoritme om het bruggenprobleem op te lossen, maar wel ontstonden hieruit begrippen als 'Eulergraaf' en 'Eulertoer'. Juist deze begrippen vormen de kern tot de oplossing van het Chinese postbodeprobleem, waar deze scriptie over gaat.

Het Chinese postbodeprobleem is vernoemd naar de Chinese wiskundige Kwan Mei-ko, die het probleem in 1962 als eerste bestudeerde. Het pro-

bleem komt er op neer dat een postbode brieven moet bezorgen in een bepaalde wijk. Hij vertrekt daarbij vanuit het postkantoor, gaat vervolgens alle straten langs om de post te bezorgen en keert uiteindelijk weer terug bij het postkantoor. Hierbij is het natuurlijk de bedoeling om de totaal afgelegde afstand minimaal te houden.

In deze scriptie zal ik dit probleem omzetten naar grafentheorie. Een belangrijk onderdeel van de oplossing is het vinden van een kortste pad tussen twee punten in een graaf. Hiervoor zal ik dan ook verschillende algoritmes bespreken. Ook een belangrijke rol is weggelegd voor Edmonds' bloesemalgoritme. Om dit algoritme te kunnen bespreken, zal ik me gaan verdiepen in de theorie achter matchings.
Dit is echter niet de enige manier waarop het Chinese postbodeprobleem opgelost kan worden. Met behulp van lineair programmeren zal ik nog een andere methode laten zien om tot een oplossing te komen.

Aan het eind van deze scriptie komen nog enkele variaties van het Chinese postbodeprobleem aan bod. Ook geef ik nog wat andere mooie voorbeelden, die het belang van Eulergrafen illustreren.

Tot slot wil ik van de gelegenheid gebruik maken om mijn begeleider bij deze scriptie, dr. Marcel van de Vel, te bedanken. Zonder zijn ondersteuning en kennis zou deze scriptie nooit geworden zijn tot wat hij nu is. Ook ben ik mijn dank verschuldigd aan Prof. dr. Alexander Schrijver, die met zijn open aanmerkingen de kwaliteit van mijn scriptie verbeterd heeft.

Hoofdstuk 2

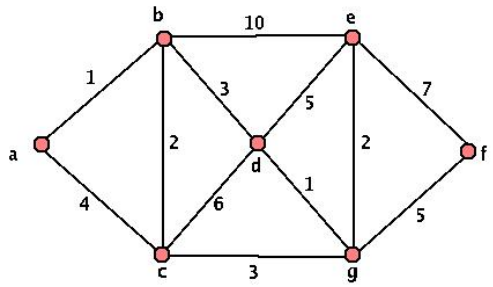
Eulergrafen

In dit hoofdstuk zullen we het Chinese postbodeprobleem introduceren en oplossen voor een simpel geval. Hierbij worden de basisbegrippen uit de grafentheorie bekend verondersteld. Als dit niet het geval is, verwijzen we naar sectie 10.1, voor een beknopte inleiding.

Voordat we het Chinese postbodeprobleem kunnen formuleren, hebben we eerst enkele definities nodig.

Definitie 2.1. Een *gewogen graaf* is een graaf G waarin aan elke lijn e een niet-negatief getal $w(e)$ (ook wel w_e) wordt toegekend. Dit getal wordt wel het *gewicht* van de lijn e genoemd.

Voorbeeld 2.2. Stel dat de graaf een voorstelling geeft van een bepaalde wijk; de punten zijn de kruisingen en de lijnen stellen de wegen voor. Dan is het gewicht van de lijn representatief voor de lengte van de bijbehorende weg. De lengte van $(v_1, e_1, v_2, e_2, ..., v_k, e_k, v_{k+1} = v_1)$ wordt dan $\sum_{i=0}^{k} w_{e_i}$. We bekijken onderstaande graaf. Hierin geldt dus dat de lengte van de wandeling (a, b, e, d, c) gelijk is aan $1 + 10 + 5 + 6 = 22$.



Figuur 2.1: Een gewogen graaf.

Definitie 2.3. Een *toer* of *gesloten wandeling* in een graaf G is een rij $(v_1, e_1, v_2, e_2, ..., v_k, e_k, v_{k+1} = v_1)$ van punten v_i en lijnen e_i zodanig dat e_i een lijn is tussen de verschillende punten v_i en v_{i+1}.

Er zijn verschillende soorten toeren:

- Een *simpele toer* bevat elke lijn $e \in E$ maximaal één keer.
- Een *postbodetoer* bevat elke lijn $e \in E$ minimaal één keer.
- Een *Eulertoer* bevat elke lijn $e \in E$ precies één keer. Een graaf die een Eulertoer bevat wordt ook wel een *Eulergraaf* genoemd.

De naam postbodetoer is natuurlijk niet voor niets zo gekozen. Het Chinese postbodeprobleem is er namelijk op gericht om in een gegeven graaf de optimale postbodetoer te vinden.

Probleem 2.4. [Chinese postbodeprobleem].
Gegeven een samenhangende, gewogen graaf, vind de kortste gesloten wandeling die elke lijn minstens één keer bevat.

We merken op dat als een graaf een Eulertoer bevat, deze toer ook een postbodetoer moet zijn. Als we dus te maken hebben met een Eulergraaf, dan hoeven we enkel een Eulertoer te vinden om zo ons probleem op te lossen. Hoe dit in zijn werk gaat, bespreken we in de volgende sectie.

2.1 Euler

In de volgende stelling gaan we er van uit dat we een graaf bekijken zonder geïsoleerde punten.

Stelling 2.5. *Een graaf G bevat een Eulertoer $\Leftrightarrow$ G is samenhangend en elk punt heeft even graad.*

Bewijs. $\Rightarrow$ Stel $G = (V, E)$ bevat een Eulertoer. Kies nu twee verschillende punten $a, b \in V$ en neem het stuk van de Eulertoer tussen a en b. Dit is een pad tussen a en b, waaruit volgt dat G samenhangend is. Als we nu de Eulertoer vanuit a volgen, dan komen we uiteindelijk weer terug in a. Dus elke keer als je in een punt $x \neq a$ aankomt, vertrek je er weer via een nog niet doorlopen lijn. De graad van x is dus even. Voor het beginpunt a geldt dat de graad gelijk is aan 1 plus een aantal keer 2 plus 1, dus even.
$\Leftarrow$ Gegeven is nu dat G samenhangend is en dat elk punt even graad heeft. We beginnen in een willekeurig punt $x \in V$ en doorlopen steeds de lijnen die

we nog niet gebruikt hebben. Als we in een punt $y \in V$ aankomen en $y \neq x$, dan kunnen we nog verder lopen omdat de graad van y even is en er dus nog een vrije lijn moet zijn. Omdat er maar eindig veel lijnen zijn, komen we op een gegeven moment terug in x. Er zijn dan twee mogelijkheden:
Geval 1: we hebben alle lijnen doorlopen. Dan zijn we klaar, want we hebben een gesloten wandeling gemaakt die alle lijnen van G precies één keer bevat.
Geval 2: er zijn nog lijnen die nog niet doorlopen zijn. Als al deze lijnen niet uitkomen op de al geconstrueerde wandeling, dan kun je een willekeurig punt op die wandeling blijkbaar niet verbinden met een punt erbuiten. Dit is echter in tegenspraak met de samenhang van G. Er moet dus een lijn $e = yz$ zijn, die nog niet in de wandeling zit, maar waarvan y wel op die wandeling ligt. We beginnen nu in y en lopen naar z en vervolgen de wandeling zoals boven beschreven (steeds lijnen doorlopend waarover nog niet gelopen is). Omdat de restgraaf (zonder de eerder geconstrueerde wandeling) evengraads is, kom je na het doorlopen van een aantal lijnen weer terug in y. Als we deze twee wandelingen nu combineren, krijgen we een langere gesloten wandeling die alle tot nu toe doorlopen lijnen precies één keer aandoet. Zo kunnen we doorgaan tot we uiteindelijk een Eulertoer geconstrueerd hebben. □

We weten nu dus dat als we een graaf hebben waarin elk punt evengraads is, er een Euler- en dus een postbodetoer bestaat. Wat we nu nog nodig hebben is een efficiënte manier om deze toer te vinden. Hiervoor kunnen we onderstaand algoritme van Fleury gebruiken. Dit algoritme construeert een spoor in een graaf G.

Algoritme 2.6. [Fleury].
Begin met een graaf $G = (V, E)$.

1. Kies een willekeurig punt $v_0 \in V$ en definiëer de wandeling $W_0 = v_0$
2. Neem aan dat de wandeling $W_i = (v_0, e_1, v_1, ...e_i, v_i)$ gekozen is. Kies dan een lijn $e_{i+1} \in E \backslash \{e_1, e_2, ..., e_i\}$ z.d.d
   - e_{i+1} een lijn is vanuit punt v_i
   - e_{i+1} geen brug is van $G_i = G \backslash \{e_1, e_2, ..., e_i\}$, tenzij er geen alternatief is.
3. Doorloop deze lijn. Zo ontstaat de wandeling $W_i = (v_0, e_1, ..., v_i, e_{i+1}, v_{i+1})$
4. Herhaal stap 2 en 3 tot alle lijnen doorlopen zijn en je weer terug bent in v_0.

Stelling 2.7. *Als G een Eulergraaf is, dan is elk spoor geconstrueerd door het algoritme van Fleury een Eulertoer van G.*

Bewijs. Zij G een Eulergraaf en $W_n = (v_0, e_1, v_1, ...e_n, v_n)$ een spoor in G, geconstrueerd door bovenstaand algoritme. In de deelgraaf $G_n = G \backslash \{e_1, e_2, ..., e_n\}$ moet nu gelden dat $d(v_n) = 0$, want anders zou het algoritme nog niet klaar zijn. Hieruit volgt dat $v_n = v_0$, dus W_n is gesloten.
Stel nu dat W_n geen Eulertoer van G is. Zij S de verzameling van alle punten in G_n met positieve graad en $\bar{S} = V \backslash S$. Dan is S niet leeg en $v_n \in \bar{S}$. Zij m het grootste gehele getal zodanig dat $v_m \in S$ en $v_{m+1} \in \bar{S}$. Omdat W_n eindigt in $\bar{S}$, is $e_{m+1} = \{v_m, v_{m+1}\}$ de enige lijn uit $[S, \bar{S}]$ in G_m, waaruit volgt dat het een brug is van G_m.
Zij e een andere lijn in G_m die verbonden is met v_m. Uit stap 2 van het algoritme volgt dat e dan ook een brug van G_m moet zijn en dus van de geïnduceerde deelgraaf $G_m[S]$. Maar omdat $G_m[S] = G_n[S]$, heeft elk punt in $G_m[S]$ even graad. En een evengraadse graaf kan helemaal geen brug hebben. Tegenspraak. $\square$

Hiermee is het Chinese postbodeprobleem opgelost voor het geval dat we te maken hebben met een Eulergraaf.

Hoofdstuk 3

Korstepad-algoritmes

Zoals de titel al zegt, zullen we in dit hoofdstuk enkele kortstepad-algoritmes bespreken. Deze algoritmes hebben we later nodig om het algemene geval van het Chinese postbodeprobleem op te lossen. We beginnen met een definitie.

Definitie 3.1. Een *kortste pad* tussen twee punten in een gewogen graaf is een pad tussen die twee punten met het minste gewicht.

3.1 Het algoritme van Dijkstra

Het eerste algoritme dat we bespreken is van de Nederlander Edsger Dijkstra. Dit algoritme vindt het kortste pad tussen twee vooraf gekozen punten, bijvoobeeld A en B. Als we het algoritme herhaaldelijk toepassen, dan geeft het uiteindelijk de korste paden van A naar alle andere punten van de graaf. Als er geen pad bestaat, dan wordt dit ook duidelijk door het algoritme.

Dijkstra's kortstepad-algoritme komt er op neer om aan elk punt $v \in V(G)$ een geordend paar (x, d) toe te kennen, waarin d de kortste afstand is tussen A en v en xv de laatste lijn van dit kortste pad. Dus als B uiteindelijk het label (y, t) krijgt, dan is het kortste pad van A naar B precies t eenheden lang en de laatste lijn van dit pad is yB. Ook het punt y heeft een label en zo kunnen we dus terugwerken om het kortste pad van A naar B te construeren.

Nu kan het echter ook voorkomen dat het punt B niet gelabeld kan worden. Als dit het geval is, dan bestaat er geen kortste pad tussen A en B; de graaf is niet samenhangend.

Algoritme 3.2. [**Dijkstra**].
Om in een gewogen graaf een korste pad van punt A naar punt B te vinden, kan de volgende procedure gevolgd worden:

1. Geef A het label $(-, 0)$
2. Totdat B gelabeld is of er geen labels meer toegekend kunnen worden, doe het volgende:
   - Voor elk gelabeld punt $u(x, d)$ en elk ongelabeld punt v dat daarmee verbonden is, bereken $d + w(uv)$.
   - Voor elk ongelabeld punt v, geef v het label (u, d') als gegeven is dat $d' = min\{d + w(uv)\}$, waarbij dit minimum loopt over alle punten $u \in V(G)$ die met v verbonden zijn. Als een punt gelabeld kan worden met (x, d') voor verschillende punten x, maak dan een willekeurige keuze.

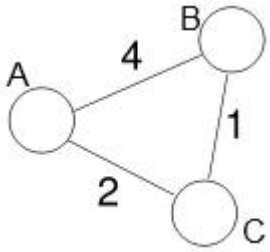
Dit algoritme is echter niet ideaal. Met een paar aanpassingen kan de efficiëntie enorm verhoogd worden. Dit leidt tot het verbeterde algoritme van Dijkstra.

Algoritme 3.3. [**Verbeterde versie van Dijkstra**].

1. Stel $v_1 = A$ en ken aan dit punt het permanente label 0 toe. Ken aan alle andere punten het tijdelijke label ∞ toe.
2. Totdat een permanent label aan B is toegekend of er geen tijdelijke labels meer veranderd kunnen worden, doe het volgende:
   - Neem het punt v_i, dat het meest recent een permanent label gekregen heeft, zeg d. Voor elk punt v dat verbonden is met v_i en die nog geen permanent label gekregen heeft, verander het tijdelijke label van v in $d + w(v_i v)$, maar alleen als dit label kleiner is dan het huidige label van v.
   - Neem een punt v, die het kleinste tijdelijke label heeft en maak dit tijdelijke label permanent. Als er meerdere punten v zijn met een kleinste tijdelijk label, maak dan een willekeurige keuze.

De verbeterde versie van Dijkstra's algoritme is dan wel efficiënter dan de eerste, hij heeft één belangrijk nadeel. Hij geeft namelijk alleen aan wat de lengte van het kortste pad tussen twee punten is, maar hij geeft niet aan via welke punten dit pad loopt. Dit laatste doet de eerste versie van Dijkstra's algoritme juist wel.

Voorbeeld 3.4. We passen de verbeterde versie van Dijkstra's algoritme toe op onderstaande graaf.



1. We geven punt A het permanente label 0 en de punten B en C allebei het tijdelijke label ∞.

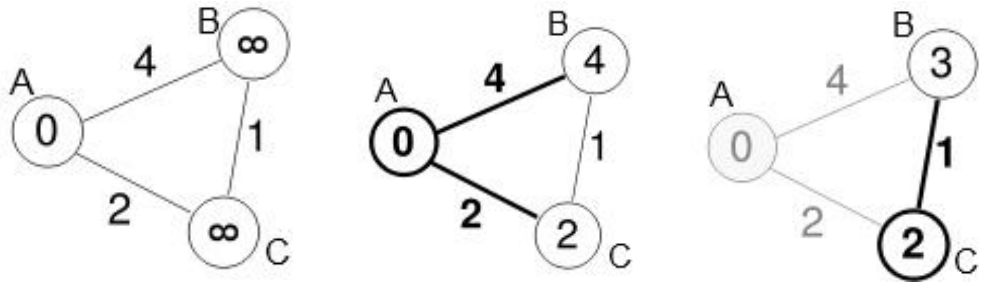
2. Punt A heeft het meest recent een permanent label gekregen. We veranderen nu het tijdelijke label van B in $0 + 4 = 4$ en het tijdelijke label van C in $0 + 2 = 2$.

3. Punt C heeft nu het kleinste tijdelijke label, dus we maken dit label permanent.

4. Hierdoor is C het punt dat meest recent een permanent label gekregen heeft. We veranderen het tijdelijke label van B in $2 + 1 = 3$, want $3 < 4$.

5. Nu heeft B het kleinste tijdelijke label, dus wordt ook dit label permanent.

Er volgt dat het kortste pad van A naar B lengte 3 heeft en het kortste pad van A naar C lengte 2.



Figuur 3.1: Illustratie van bovenstaande procedure

3.2 Het algoritme van Floyd en Warshall

Het algoritme van Dijkstra is niet het enige dat gebruikt kan worden om een kortste pad te vinden. Ook R. W. Floyd en S. Warshall hebben een algoritme ontwikkeld. Deze procedure kan gebruikt worden om de kortste afstanden te vinden tussen alle paren van punten in een gewogen graaf en is in veel toepassingen sneller dan bovenstaande versies van Dijkstra's algoritme.

Algoritme 3.5. [Floyd-Warshall].
Zij $G = (V, E)$ een graaf met $V = \{v_1, v_2, ..., v_n\}$

1. Zeg $d(i, i) = 0$ voor $i = 1, 2, ..., n$.
   Als $i \neq j$ en $v_i v_j \in E$ laat dan $d(i, j)$ gelijk zijn aan het gewicht van deze lijn. Als het geen lijn is, dan zeggen we $d(i, j) = \infty$.

2. We definiëren nu voor $k, i, j = 1, 2, ..., n$:

$$\text{d(i,j)=min\{d(i,j), \quad d(i,k)+d(k,j)\}.}$$

Hierbij wordt stap 2 iteratief toegepast. De uiteindelijke waarde van $d(i, j)$ is de kortste afstand van v_i naar v_j.

Hoofdstuk 4

Postbodegrafen

In hoofdstuk 2 hebben we laten zien hoe we een postbodetoer kunnen vinden als de graaf die we bekijken een Eulergraaf is. In de praktijk zullen we echter veel vaker te maken hebben met grafen die geen Eulertoer bevatten. In dit hoofdstuk zullen we bespreken hoe we dan tot de oplossing van het Chinese postbodeprobleem kunnen komen.

Stel dat we een graaf $G = (V, E)$ hebben met een postbodetoer. Elke lijn $e \in E$ komt dan minstens één keer in de toer voor. Zij nu $1 + x(e)$ het aantal keer dat de lijn e in de toer voorkomt. We construeren een nieuwe graaf $G' = (V, E')$, door aan de oorspronkelijke graaf G precies $x(e)$ kopieën van e toe te voegen. Op deze manier wordt de postbodetoer in G een Eulertoer in G'. Van deze laatste weten we al hoe we hem kunnen vinden.

Echter, in dit geval gaan we er van uit dat we al weten hoe vaak een bepaalde lijn in de postbodetoer voorkomt. In de praktijk is dit echter nog onbekend. Hoe moeten we dan wel te werk gaan? We beginnen met een gewogen, samenhangende graaf G. Door bepaalde lijnen te kopiëren willen we een pseudograaf creëren met een Eulertoer van minimaal gewicht. Maar welke lijnen moeten we dan kopiëren? We weten al dat we een Eulertoer kunnen vinden dan en slechts dan als alle punten van even graad zijn. Het ligt dus voor de hand om ons te richten op de punten die roet in het eten gooien; die met oneven graad.

We definiëren hiertoe de verzameling $G^- = \{v \in G : d(v) \text{ is oneven}\}$.

Stelling 4.1. *Het aantal elementen van G^- is even.*

Bewijs. Zij het aantal lijnen van G gelijk aan m. Dan moet gelden dat

$$\sum_{v \in G} d(v) \;=\; 2m.$$

Zij G^+ de verzameling van punten in G met even graad. Dan volgt

$$\sum_{v \in G^-} d(v) \;+\; \sum_{v \in G^+} d(v) \;=\; \text{even}.$$

Maar voor elke $v \in G^+$ is $d(v)$ per definitie even. De som van even getallen is weer even, dus $\sum_{v \in G^+} d(v)$ is even, waaruit volgt dat $\sum_{v \in G^-} d(v)$ ook even moet zijn. Maar $d(v)$ is oneven voor alle $v \in G^-$, dus moet het aantal punten in G^- even zijn. □

Uit bovenstaande stelling volgt dat we de oneven punten in een graaf op kunnen delen in paren.

Opmerking 4.2. Stel dat een graaf precies $2k$ oneven punten bevat. Dan kunnen we de collectie lijnen opdelen in precies k sporen, die elk een paar punten van oneven graad verbinden en verder alleen punten van even graad bevatten. Daarnaast kunnen natuurlijk nog lijncollecties optreden van componenten van even graad.

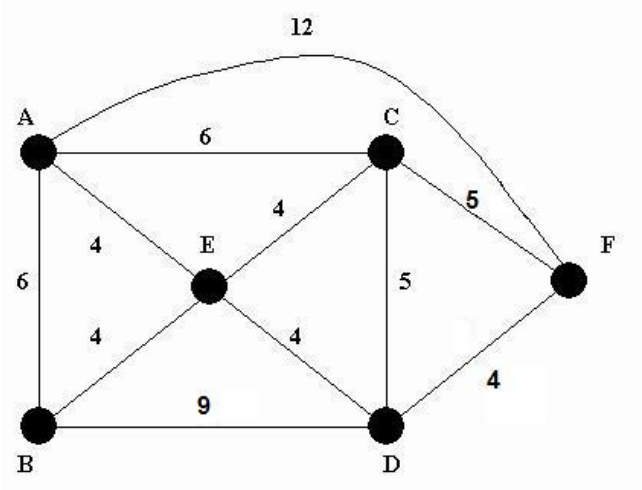
Het gegeven dat we de oneven punten in een graaf op kunnen delen in paren hebben we nodig voor het algoritme dat de oplossing geeft voor het Chinese postbodeprobleem.

Algoritme 4.3. [**Chinese postbode-algoritme**].
Zij $G = (V, E)$ een gewogen, samenhangende graaf.

1. Vind alle punten in G met oneven graad
2. Maak alle mogelijke partities van deze punten in paren: $\{v_1, w_1\}, ..., \{v_n, w_n\}$
3. Bereken voor elke partitie de lengte van een kortste pad tussen elke v_i en w_i en tel deze lengtes bij elkaar op.
4. Neem de partitie uit stap 3 waarvoor de som van de lengtes minimaal is en kopieër, voor elk paar $\{v_i, w_i\}$ uit deze partitie, de lijnen langs het kortste pad van v_i naar w_i.

4.1 Voorbeeld

Om bovenstaand algoritme te illustreren, bekijken we onderstaande gewogen graaf:

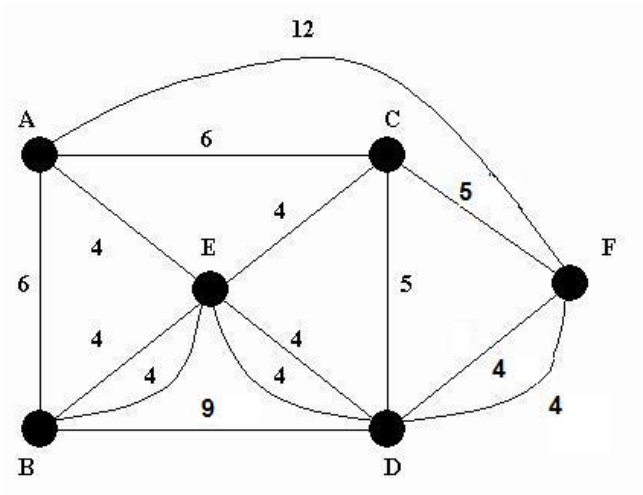


Figuur 4.1: Graaf ter illustratie van het Chinese postbode-algoritme.

Er zijn twee punten met oneven graad, namelijk B en F. Er is dus slechts één mogelijke partitie: $\{B, F\}$. Met behulp van bijvoorbeeld het algoritme van Dijkstra kunnen we nu het kortste pad tussen deze twee punten vinden. Omdat we echter te maken hebben met een kleine graaf, kunnen we ook zelf snel zien welke verschillende paden mogelijk zijn:

- $B - A - F$ met gewicht 18
- $B - D - F$ met gewicht 13
- $B - E - D - F$ met gewicht 12
- $B - E - C - F$ met gewicht 13
- $B - A - C - F$ met gewicht 17

Het kortste pad is dus $B - E - D - F$. Zoals voorgeschreven in algoritme 4.3 voegen we in onze nieuwe graaf een kopie toe van de lijnen BE, ED en DF. Dit resulteert in de volgende nieuwe graaf:



Figuur 4.2: Door het algoritme verkregen Eulergraaf.

In onze nieuwe graaf hebben alle punten even graad. Nu kunnen we dus het algoritme van Fleury gebruiken om een Eulertoer te creëren. Dit leidt dan tot een optimale postbodetoer in onze originele graaf: $ACDEDBEBAECFDFA$.

In bovenstaand voorbeeld was het algoritme erg simpel, omdat we te maken hadden met slechts twee punten van oneven graad. In de praktijk is het aantal punten dat oneven graad heeft echter vaak veel groter, waardoor het aantal mogelijke partities snel toeneemt. Als er namelijk m oneven punten zijn, dan zijn er $\prod_{i=1}^{m/2}(2i-1)$ mogelijke partities. Het is duidelijk dat stap 2 en 3 dan steeds gecompliceerder worden. Om dit op te lossen, zullen we ons in het volgende hoofdstuk gaan verdiepen in matchings.

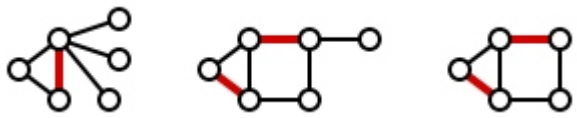
Hoofdstuk 5

Matchings

Definitie 5.1. Zij $G = (V, E)$ een graaf. Een *matching* in G is een verzameling lijnen zonder gedeelde eindpunten. We noemen een punt *verzadigd* als hij een lijn heeft die in de matching zit.

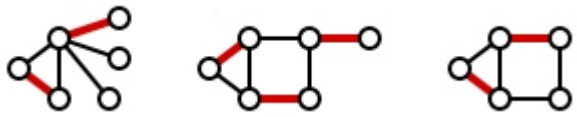
Er zijn verschillende soorten matchings:

- Een *maximale* matching is een matching die niet groter gemaakt kan worden door een lijn toe te voegen.



Figuur 5.1: Voorbeelden van maximale matchings.

- Een *maximum* matching is de grootst mogelijke matching in de graaf.



Figuur 5.2: Voorbeelden van maximum matchings.

Voordat we verder kunnen, hebben we nog twee definities nodig:

Definitie 5.2. Zij M een matching.

- Een *M-alternerende wandeling* is een wandeling waarvan de lijnen afwisselend wel en niet tot de matching M behoren.
- Een *M-verbeterende wandeling* is een M-alternerende wandeling die begint en eindigt in een onverzadigd punt.

Stelling 5.3. ***[Lemma van Berge]***.
Zij $G = (V, E)$ een graaf en M een matching. Dan geldt:
M is maximum in $G \Leftrightarrow$ Er bestaat geen M-verbeterend pad in G.

Bewijs. We zullen dit bewijzen door contrapositie, oftewel:
M is geen maximum matching $\Leftrightarrow$ Er bestaat een M-verbeterend pad in G.
$\Leftarrow$ Stel G heeft wel een M-verbeterend pad. We noemen dit pad P. We kunnen de de lijnen van P die in M zitten vervangen door precies de andere lijnen van P. Zo krijgen we een matching M' die precies één lijn meer heeft dan M. Dus M is geen maximum matching.
$\Rightarrow$ Stel M is geen maximum matching. Er bestaat dus een matching M' die groter is dan M. We definiëren nu $F = M \triangle M'$. Omdat M en M' allebei matchings zijn, heeft elk punt $v \in V(G)$ maximaal één lijn uit iedere matching. Dus F heeft voor ieder punt maximaal twee lijnen. Hieruit volgt dat elke component van F een pad of cykel moet zijn. Elk pad en elke cykel alterneert tussen lijnen uit $M \backslash M'$ en uit $M' \backslash M$. Dus elke cykel moet even lengte hebben, met een gelijk aantal lijnen uit M en M'. Omdat $|M'| > |M|$, moet F een component hebben die meer lijnen van M' dan van M bevat. Omdat de cykels even zijn, kan dit alleen een pad zijn die begint en eindigt met een lijn uit M'. Dit is een M-verbeterend pad in G. $\square$

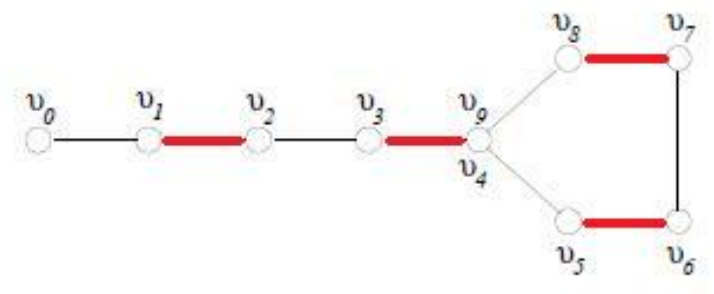
Definitie 5.4. Zij M een matching in een graaf $G = (V, E)$ en zij $v_0 \in V$ een M-onverzadigd punt. Een *M-bloem* is een M-alternerende wandeling $(v_0, v_1, ..., v_j)$ met:

- $v_0, v_1, ..., v_{j-1}$ verschillende punten in $V(G)$.
- $v_i = v_j$ voor een zekere even i en oneven j.

We noemen de oneven cykel $(v_i, v_{i+1}, ..., v_{j-1}, v_j)$ ook wel de *bloesem* en het even pad $(v_0, v_1, ..., v_i)$ wordt ook wel de *steel* genoemd.

We zullen deze definitie illustreren aan de hand van een voorbeeld.

Voorbeeld 5.5. We beschouwen onderstaande graaf, waarin de dikke lijnen de matchinglijnen voorstellen.



Figuur 5.3: Een voorbeeld van een M-bloem.

De steel wordt gegeven door $(v_0, v_1, v_2, v_3, v_4)$ en de punten $(v_4, v_5, v_6, v_7, v_8, v_9)$ vormen samen de bloesem van de bloem.

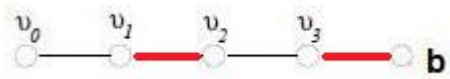
Als B een M-bloesem is, dan kunnen we B samentrekken tot één punt, namelijk b. We definiëren nu de nieuwe graaf

$$G/B := ((V \backslash V(B)) \cup \{b\}, E/B)$$

waarbij

$$E/B := \{e \in E : e \cap V(B) = \emptyset\} \cup \{\{u, b\} | \{u, v\} \in E, v \in V(B), u \in V \backslash V(B)\}$$

Voorbeeld 5.6. We bekijken de graaf uit figuur 5.3. We kunnen de bloesem $B = (v_4, v_5, v_6, v_7, v_8, v_9)$ nu samentrekken tot het punt b. Zo krijgen we de volgende graaf:



Figuur 5.4: De bloesem B samengegrokken tot één punt b.

Omdat iedere lijn van M ofwel disjunct is van B, ofwel bevat is in B, is M/B nu een matching in G/B.

Stelling 5.7. *De matching M is een maximum matching in $G \Leftrightarrow M/B$ is een maximum matching in G/B.*

Bewijs. We zullen gaan bewijzen M is niet maximum $\Leftrightarrow M/B$ is niet maximum.
$\Rightarrow$ Zij $P = (v_0, ..., v_i)$ de steel van de M-bloem en laat $N := M \triangle E(P)$ de matching zijn die verkregen is uit M door langs P te wisselen. Nu is v_i onverzadigd door N en B is onverzadigd door N/B. Omdat $|N| = |M|$ en $|N/B| = |M/B|$ is het voldoende om aan te tonen dat als er een N-verbeterend pad in G bestaat, er ook een N/B-verbeterend pad bestaat in G/B. Zij $Q = (v_0, ..., v_k)$ een N-verbeterend pad in G. Zonder verlies van algemeenheid mogen we aannemen dat $v_0 \neq v_i$ (anders nummeren we Q in omgekeerde richting). Als $Q \cap V(B) = \emptyset$, dan is Q een N/B-verbeterend pad in G/B. Als Q wel een punt gemeen heeft met B, laat l dan de kleinste index zijn waarvoor $v_l \in V(B)$. Nu is $(v_0, ..., v_{l-1}, b)$ een N/B-verbeterend pad in G/B.
$\Leftarrow$ Zij M' een matching in G/B met $|M'| > |M/B|$. Door eventueel een lijn $\{b, v\} \in M'$ te vervangen door een lijn $\{y, v\} \in E$, geeft dit een matching M'' in G die alle punten van B onverzadigd laat, op hoogstens één punt $y \in V(B)$ na. Toevoegen van lijnen uit de perfecte matching van $B \backslash y$ aan M'' geeft een matching in G die groter is dan M. M is dus niet maximum. $\square$

We hebben nu genoeg informatie over matchings besproken om Edmonds' bloesem-algoritme te behandelen. Gegeven een graaf en een matching, resulteert dit algoritme of in een grotere matching of in niks als de matching waarmee je begon al maximum was.

Algoritme 5.8. [Edmonds' bloesem-algoritme].
Zij G een graaf, M een matching in G en u een onverzadigd punt. Zij S de verzameling van u en alle onverzadigde punten in $V(G)$ die bereikt kunnen worden door middel van een M-alternerende wandeling. Er zijn twee mogelijkheden:

1. $S \backslash u = \emptyset$. Er is geen M-verbeterend pad vanuit u. Het algoritme stopt.

2. $S \backslash u \neq \emptyset$. Kies . Bekijk de M-alternerende wandeling W van u naar $v \in S \backslash u$, met v zo gekozen dat W zo kort mogelijk is. Er zijn nu twee situaties denkbaar:

   a) W is een pad. $N := M \triangle E(W)$ is een matching met $|N| > |M|$.

   b) W is geen pad. We hebben te maken met een M-bloem, met steel Q en bloesem B. Voor de matching $N' := M \triangle E(Q)$ geldt nu dat $|N'| = |M|$. Herhaal het algoritme met graaf G/B en matching N'/B

We sluiten dit hoofdstuk af met een uitleg over hoe matchings gebruikt kunnen worden om de oplossing van het Chinese postbodeprobleem te vinden. In algoritme 4.3 moesten we alle punten van oneven graad vinden. M.b.v één van de algoritmes uit hoofdstuk 3 kunnen we nu voor elk paar oneven punten een kortste pad vinden. We maken nu een nieuwe graaf G', bestaande uit alle oneven punten van de originele graaf G, met tussen alle punten precies één lijn. Hierbij geldt dat het gewicht van een lijn in G' gelijk is aan het gewicht van het kortste pad tussen die twee punten in G. We willen nu een maximum matching van miminaal gewicht in G' vinden. Deze matching geeft namelijk aan welke partitie van paren oneven punten we moeten gebruiken in algoritme 4.3.

Hoofdstuk 6

Lineair Programmeren

In het vorige hoofdstuk hebben we een manier besproken om het Chinese postbodeprobleem op te lossen. In dit hoofdstuk zullen we nog een andere methode behandelen, namelijk met behulp van lineair programmeren. Voor een beknopte inleiding op deze oplossingsmethode verwijzen we naar sectie 10.2. Voor een gedetailleerder beeld van wat lineair programmeren precies inhoudt, kan het handig zijn om [9] te bestuderen.

Het uitgangsprincipe is hetzelfde als in het vorige hoofdstuk. Gegeven een postbodetoer in een graaf G, komt elke lijn $e \in E(G)$ minstens één keer in de toer voor, maar misschien ook wel vaker. Zij nu $1 + x_e$ het aantal keer dat lijn e in de toer voorkomt. We definiëren de graaf G' die gevormd wordt vanuit G door x_e kopieën van e toe te voegen. Dus als de lijn e één keer in G voorkomt, dan komt hij $1 + x_e$ keer in G' voor. De postbodetoer in G wordt zo dus een Eulertoer in G'.

Het probleem van het vinden van een optimale postbodetoer reduceert nu dus tot het vinden van de aantallen x_e voor alle $e \in E(G)$. Als we het gewicht van de lijnen e aangeven met c_e, dan is dit equivalent met het vinden van een geheel getal $x_e \geq 0$ voor elke $e \in E(G)$ zodanig dat $\sum c_e x_e$ minimaal is. De som loopt hierbij over alle lijnen e vanuit een punt $v \in V(G)$, voor elk punt in G. Hierbij moeten we wel rekening houden met een aantal voorwaarden, zoals we straks zullen zien. Deze voorwaarden komen door het feit dat we willen eindigen met een evengraadse graaf. Als we deze x_e gevonden hebben, dan heeft de graaf G' een Eulertoer die gelijk is aan de optimale postbodetoer in de originele graaf G.

Het Chinese postbodeprobleem kan dus gescheiden worden in twee delen: de optimale x_e vinden voor bovenstaand probleem en vervolgens een Eulertoer vinden in de resulterende graaf G'. Het tweede deel hebben we al behandeld in hoofdstuk 2. Hoe we het eerste, en moeilijkste, gedeelte kunnen oplossen, zullen we in dit hoofdstuk bespreken.

Alleerst definiëren we *punt-lijn incidentie matrix* (a_{ve}) voor $v \in V$ en $e \in E$ als:

$$a_{ve} = \begin{cases} 1 & \text{als lijn } e \text{ in punt } v \text{ komt,} \\ 0 & \text{anders} \end{cases}$$

Het probleem waar we nu naar kijken is het vinden van gehele getallen $x_e \geq 0$, $e \in E$ zodanig dat $\sum_{e \in E} a_{ve}(1 + x_e) \equiv 0 \pmod 2$ en $\sum c_e x_e$ minimaal is. Dit kunnen we ook schrijven als:

$$\sum_{e \in E} a_{ve} x_e \equiv \sum_{e \in E} a_{ve} \pmod 2$$

De som $\sum_{e \in E} a_{ve}$ is de graad van punt v.

We definiëren nu:

$$b_v \equiv \sum_{e \in E} a_{ve} \pmod 2 = \begin{cases} 0 & \text{als } d(v) \text{ even is,} \\ 1 & \text{als } d(v) \text{ oneven is} \end{cases}$$

We kunnen het Chinese postbodeprobleem oplossen als we x_e, $e \in E$ vinden z.d.d voor $v \in V$ geldt:

(1) x_e is een geheel getal

(2) $x_e \geq 0$

(3) w_v is een geheel getal

(4) $w_v \geq 0$

(5) $\sum_{e \in E} a_{ve} x_e - 2w_v = b_v$

(6) $z = \sum c_e x_e$ is minimaal

Opmerking 6.1. We merken op dat, als aan (1), (2), (3) en (5) is voldaan, in (5) automatisch is voldaan aan de niet-negativiteit van w_v, . Dit komt doordat

$$2w_v = \sum_{e \in E} a_{ve} x_e - b_v$$

waar b_v gelijk is aan 0 of 1. Hieruit volgt dat $2w_v$ waarden aanneemt in ofwel $\{0, 1, ...\}$ ofwel $\{-1, 0, 1, ...\}$. Als w_v een geheel getal is, dan kan $2w_v$ niet gelijk zijn aan -1, waaruit volgt dat $w_v \geq 0$.
Dus (1), (2), (3) en (5) beschrijven dezelfde verzameling van gehele getallen als (1), (2), (3), (4) en (5).

Voordat we verder kunnen, hebben we de volgende definities nodig:

Definitie 6.2. Zij S een verzameling punten. We zeggen dat de lijn e de verzameling S *raakt* als e een lijn is tussen een punt $s \in S$ en een punt $t \notin S$. We noemen S een *oneven verzameling* als S een oneven aantal punten met oneven graad bevat.

Met behulp hiervan kunnen we de *bloesem-ongelijkheden* introduceren, als:

$$\sum \{ x_e \ : e \text{ raakt } S\} \geq 1 \qquad \text{voor } S \text{ oneven} \tag{6.1}$$

Uit stelling (10.6) volgt nu dat er ongelijkheden bestaan die het convex omhulsel van de oplossingen x_e, $e \in E$, w_v, $v \in V$ voor het Chinese postbodeprobleem bepalen. Zonder bewijs vermelden we dat deze gelijk zijn aan het veelvlak van de lineaire restricties (2), (5) en de bloesem-ongelijkheden (6.1).

We kunnen dit probleem reduceren tot een equivalent matchingprobleem. We beginnen met het vinden van een kortste pad tussen elk paar oneven punten (bijvoorbeeld m.b.v. één van de algoritmes uit hoofdstuk 3). Vervolgens construeren we een nieuwe graaf G_p, bestaande uit de oneven punten van G met tussen alle punten precies één lijn. We hebben dus te maken met een volledige graaf. Hierbij geldt dat het gewicht $w(m)$ van een lijn $m = \{v_1, v_2\}$ in G_p gelijk is aan het gewicht van het kortste pad tussen v_1 en v_2 in G. We willen nu een optimale matching van G_p vinden. In andere woorden: we willen een verzameling $M \subset E(G_p)$ vinden zodanig dat geldt:

- elk punt $v \in V(G_p)$ raakt precies één lijn $m \in M$.
- $\sum_{m \in M} w(m)$ is minimaal.

Stel dat we zo'n optimale matching gevonden hebben. Als we nu de oplossing x_e willen vinden voor bovenstaande vergeljkingen (1), (2) en (5) dan moeten we voor elke lijn $m \in M$ kijken naar het kortste pad in G dat hiermee correspondeert en $x_e = 1$ stellen voor elke lijn e op dit kortste pad. Voor alle overige lijnen in G stellen we $x_e = 0$.

Stelling 6.3. *Bovenstaande procedure lost (1), (2) en (5) op.*

Bewijs. Allereerst gaan we bewijzen dat er geen lijn $e \in E(G)$ is die in twee kortste paden zit die met matchtinglijnen van G_p corresponderen. Neem hiervoor aan dat dit wel het geval is, dus dat er een lijn e bestaat die in het kortste pad P_i van i_1 naar i_2 zit en ook in het kortste pad P_j van j_1 naar j_2. Neem ook aan dat deze paden corresponderen met matchinglijnen van G_p. We mogen er nu van uit gaan dat beide paden lijn-simpel zijn, omdat ze kortste paden zijn in een graaf met niet-negatieve gewichten bij elke lijn. Als we e verwijderen, dan houden we twee paden over, namelijk van i_1 naar j_1 (of j_2) en van i_2 naar j_2 (of j_1). Zonder verlies van algemeenheid kiezen we steeds de eerste van de twee mogelijke paden en we noemen ze respectievelijk P_1 en P_2. Nu geldt dat $w(P_1 + P_2) < w(P_i + P_j)$. Dus in G_p geldt dat $w(i_1, j_1) + w(i_2, j_2) < w(i_1, i_2) + w(j_1, j_2)$. Hierdoor kunnen de lijnen van (i_1, j_1) en (i_2, j_2) de lijnen (i_1, i_2) en (j_1, j_2) vervangen in de matching M van G_p zodat we een betere matching M' krijgen en zodanig dat de kortste paden in G die corresponderen met matchinglijnen van G_p lijn e twee keer minder bevatten terwijl ze de andere lijnen even vaak bevatten. Op deze manier kunnen we de matching veranderen zodat er geen lijn $e \in E(G)$ is die in meer dan één kortste pad zit dat correspondeert met matchinglijnen. We zeggen nu $x_e = 1$ voor elke lijn $e \in E(G)$ die in een kortste pad zit dat correspondeert met een matchinglijn en $x_e = 0$ voor de overige lijnen.
Deze x_e voldoet duidelijk aan (1) en (2). De enige vraag die we nu nog moeten beantwoorden is of er nog een geheel getal w_v gevonden kan worden zodanig dat ook aan (5) voldaan is. De definitie van b_v in herinnering gebracht, zien we dat dit kan als de som over alle lijnen e die v raken $\sum x_e$ even is als $d(v)$ even is en oneven als $d(v)$ oneven is. Deze som over x_e in een kortste pad dat correspondeert met een matchinglijn is even voor alle punten met hierop uitgezonderd de twee oneven punten op het einde van het pad. Omdat elk oneven punt een eindpunt is van precies één zo'n kortste pad, volgt dat w_v gevonden kan worden. □

We hebben nu laten zien dat als we een optimale matching M in G_p hebben, we een oplossing x_e voor (1), (2) en (5) kunnen vinden. We moeten echter nog laten zien dat deze oplossing ook optimaal is voor het probleem van het minimaliseren van $z = \sum c_e x_e$ t.o.v (1), (2) en (5). Hiervoor beschouwen we

een optimale oplossing. We mogen aannemen dat $x_e \in \{0, 1\}$, want anders kunnen we x_e verminderen met 2 en omdat $c_e \geq 0$ neemt het gewicht dan af of het blijft gelijk. Een oneven aantal lijnen e met $x_e = 1$ raakt oneven punten en een even aantal lijnen even punten. We bekijken nu de deelgraaf G' van de gewogen graaf G_p, bestaande uit alleen die lijnen e waarvoor $x_e = 1$. Volgens opmerking (4.2) kunnen we de lijnen van G' opdelen in (onder andere) een aantal disjuncte sporen die paren punten van oneven graad verbinden via punten van even graad. Wat resteert zijn hoogstens een paar Eulertoeren in evengraadse componenten. Voor dergelijke lijnen e kunnen we $x_e = 1$ terugbrengen tot $x_e = 0$ zonder (1), (2) en (5) aan te tasten en zonder z te verhogen met $c_e \leq 0$. De sporen tussen onevengraadse punten kunnen worden uitgedund tot paden tussen die punten. Voor de niet gebruikte lijnen e kunnen we weer $x_e = 0$ stellen. We weten al dat onze optimale matching het minimum z voor zo'n verzameling paden verzorgde, dus er volgt dat de matching een optimale oplossing voor (1), (2) en (5) geeft.

We keren nu weer terug naar het Chinese postbodeveelvlak, gegeven door (2), (5) en de bloesem-ongelijkheden (6.1). We gaan ons resultaat aanscherpen door het veelvlak te geven in termen van alleen x_e, dus zonder de w_v. Hiervoor bekijken we de verzameling x_e, $e \in E$, die voldoet aan (1), (2) en

$$\sum_{e \in E} a_{ve} x_e \equiv b_v \, (mod\, 2) \qquad v \in V \tag{6.2}$$

Stelling 6.4. *Het convex omhulsel van (1), (2) en (6.2) is gelijk aan het veelvlak van oplossingen van (2) en de bloesem-ongelijkheden (6.1).*

Bewijs. We zagen al eerder dat (1), (2), (3) en (5) dezelfde verzameling van gehele getallen beschrijven als (1), (2), (3), (4) en (5). We kunnen nu dus de restricties $w_v \geq 0$ in de ongelijkheden (5) en (6.1) laten vallen zonder dat we de veelvlakken veranderen.
Als we nu (5), (6.1) en $x_e \geq 0$ bekijken als een systeem van lineaire restricties, dan kunnen we de variabelen w_v laten vallen om een equivalent en gereduceerd systeem te krijgen. Dit is mogelijk omdat we voor elke $x_e \geq 0$ die voldoet aan (6.1), vergelijking (5) kunnen gebruiken om w_v te bepalen. Deze w_v en de resulterende x_e voldoen aan (5) en (6.1) en $x_e \geq 0$. Dus het convex omhulsel van oplossingen van (1), (2) en (6.2) is gelijk aan het veelvlak dat gegeven wordt door: $x_e \geq 0$, $e \in E$ en

$$\sum \{ x_e \, : e \text{ raakt } S\} \geq 1 \qquad \text{voor } S \text{ oneven}$$

□

Het probleem van het minimaliseren van $z = \sum c_e x_e$ over alle x_e die voldoen aan (1), (2) en (6.2) is nu hetzelfde als het minimaliseren van z over (2) en (6.1). Bij dit laatste minimalisatieprobleem komt lineair programmeren om de hoek kijken. Het heeft een duaal probleem in de variabelen y_s gegeven door:

Maximaliseer de functie

$$\sum \{y_s : \text{ oneven verzameling S}\} = v \tag{6.3}$$

ten opzichte van de beperkingen

$$y_s \geq 0, \qquad \text{S een oneven verzameling} \tag{6.4}$$

$$\sum \{y_s : S : e \text{ raakt } S\} \leq c_e, \qquad e \in E. \tag{6.5}$$

Lemma 6.5. *De functie v gegeven door (6.3) is kleiner dan of gelijk aan $z = \sum c_e x_e$ m.b.t. (2) en (6.1).*

Bewijs. We vermenigvuldigen elke ongelijkheid in (6.1) met $y_s \geq 0$ en trekken aan beide kanten z er van uit. Zo krijgen we:

$$\sum_{e \in E} \left(c_e - \sum \{y_s : S : e \text{ raakt } S\} \right) x_e \leq z - \sum_S y_s. \tag{6.6}$$

Wegens (6.5) is elke coëfficiënt van x_e in (6.6) niet-negatief. Dus er volgt dat $z \geq \sum_S y_s = v$ □

Met behulp van bovenstaand bewijs kunnen we nu voldoende voorwaarden afleiden zodat $z = v$. Elk paar $(x_e, e \in E), (y_s,\ S \text{ oneven})$ waarvoor geldt dat $z = v$ zal optimaal zijn voor hun respectievelijke lineaire programma's. Uit (6.6) volgt dat we nodig hebben dat:

$$x_e > 0 \Rightarrow \sum \{y_s : S : e \text{ raakt } S\} = c_e \tag{6.7}$$

Echter, als de ongelijkheid in (6.6) geen gelijkheid is, dan geldt nog steeds slechts dat $v \leq z$ en niet dat $z = v$.
Uit (6.1) volgt echter dat de ongelijkheid een gelijkheid is als:

$$y_s > 0 \Rightarrow \sum \{x_e : e \text{ raakt } S\} = 1 \tag{6.8}$$

Als zowel (6.7) als (6.8) geldt, dan geldt dus $z = v$ en er volgt dat $(x_e, e \in E)$ een optimale oplossing voor het lineaire programma probleem met beperkingen (2) en (6.1). Daarnaast is $(y_s, S$ oneven) een optimale oplossing van het duale probleem (6.3), (6.4), (6.5).

Om dergelijke paren te vinden, hebben we een variant van Edmonds' bloesemalgoritme (5.8) nodig. Omdat deze variant erg uitgebreid is, zullen we het hier niet behandelen, maar verwijzen we naar [3]. Dit algoritme produceert een paar $(x_e, e \in E)$ en $(y_s, S$ oneven) waarvoor x_e een geheel getal is en voldoet aan (6.2). Op dezelfde manier vindt het bloesemalgoritme een matching voor het algemene matchingprobleem en een oplossing voor het duale probleem. Deze oplossingen zijn optimale paren voor het lineaire programma over het matchingveelvlak en zijn duale lineaire programma.

Hoofdstuk 7

Vergelijkbare problemen

In de voorgaande hoofdstukken hebben we steeds gekeken naar ongerichte grafen, oftewel naar wijken waarin elk weg vanaf beide kanten doorgereden mag worden. In werkelijkheid bestaan er echter ook veel woonwijken met éénrichtingsverkeer. In dit hoofdstuk zullen we daarom naar enkele variaties van het Chinese postbodeprobleem gaan kijken. Allereerst zullen we gerichte grafen gaan bekijken, waarvoor we de volgende definitie en stelling nodig hebben:

Definitie 7.1. Zij $G = (V, E)$ een graaf.

- De *ingraad* van $v \in V$ is het aantal lijnen dat naar v gericht is.
- De *uitgraad* van $v \in V$ is het aantal lijnen dat vanuit v gericht is.

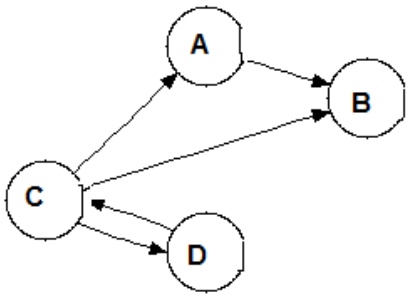
Definitie 7.2. Een gerichte graaf heet *samenhangend* als er voor iedere partitie $V = V_1 \cup V_2$ een lijn loopt van V_1 naar V_2 (of omgekeerd).
Als er altijd een dergelijke lijn in beide richtingen loopt (d.w.z. als er tussen iedere twee punten een wandeling is in beide richtingen), dan heet de graaf *sterk samenhangend*.

Stelling 7.3. *Een gerichte graaf bevat een Euler-wandeling $\Leftrightarrow$ voor alle punten in de graaf zijn de in- en uitgraad gelijk.*

Bewijs. Het bewijs hiervoor is vrijwel hetzelfde als dat voor de simpele Eulergraaf, zie stelling (2.5). $\square$

7.1 Postbodeprobleem voor gerichte grafen

Een gerichte graaf kunnen we vergelijken met een wijk waarin alle straten éénrichtingsverkeer zijn. In tegenstelling tot het standaard probleem, heeft het probleem voor gerichte grafen niet altijd een oplossing. Bekijk bijvoorbeeld onderstaande graaf:



Figuur 7.1: Een gerichte graaf zonder oplossing.

Als de postbode in A of B aankomt, kan hij niet meer terug naar C of D. Er bestaat dus geen toer in deze graaf.
In het algemeen geldt dat er geen oplossing voor het gerichte probleem bestaat als de graaf $G = (V, E)$ een deelverzameling van punten S heeft met de eigenschap dat er geen lijnen vanuit een punt $s \in S$ naar een punt $t \notin S$ gaan. Als dat niet het geval is, dus als G sterk samenhangend is, dan is het altijd mogelijk om een postbodetoer op te stellen. Voor een uitgebreide uitleg hierover verwijzen we naar [4].

7.2 Postbodeprobleem voor gemengde grafen

Het Chinese postbodeprobleem in gemixte grafen is misschien wel het meest realistische van de drie. In dit geval bestaat de graaf uit enkele gerichte en enkele ongerichte lijnen. Dit staat dus voor een combinatie van éénrichtings- en tweerichtingsverkeer.
Net als bij het gerichte geval, zijn er ook nu grafen te bedenken die geen oplossing hebben. Dit komt bijvoorbeeld voor als de graaf een deelverzameling punten S bevat met de eigenschap dat alle lijnen die verbonden zijn met een punt $s \in S$ en een punt $t \notin S$ gericht zijn naar s.

7.3 Handelsreizigersprobleem

Het handelsreizigersprobleem, ook wel Traveling Salesman Problem genoemd, is één van de grootste open problemen in de grafentheorie en informatica. Het komt er op neer dat een handelsreiziger zijn producten in een aantal steden wil verkopen. Om dit zo efficiënt mogelijk te doen, wil hij de kortste weg vinden die alle steden ten minste één keer aandoet.

Naast bovenstaand verkoopprobleem, heeft het handelsreizigersprobleem talloze directe toepassingen in de praktijk. Enkele voorbeelden hiervan zijn het aanleggen van wegen en het bepalen van de route van een boorkop bij het boren van gaatjes in printplaten.

Net als het Chinese postbodeprobleem, kan ook het handelsreizigersprobeem worden vertaald naar grafen: Construeer een graaf $G = (V, E)$ waarin de punten corresponderen met de steden en de lijnen met de wegen hiertussen. Het gewicht wat deze lijnen hebben moet dan evenredig zijn met de reisduur langs die weg.

Een rechtstreekse, maar omslachtige, manier om dit probleem op te lossen is door alle mogelijke routes langs deze steden te construeren en de lengte hiervan te bepalen. Als we echter kijken naar het geval dat de handelsreiziger 10 steden wil bezoeken, dan wordt het aantal routes dat nagegaan moet worden gelijk aan $10! = 3.628.800$. Dit wordt dus al erg snel ondoenlijk. Daarom wordt er veel gebruik gemaakt van computers. Om een idee te geven van de complexiteit van het probleem: In 1998 berekenden wiskundigen van de Universiteit van Princeton de oplossing voor 15.112 steden in Duitsland. Dit kostte hen echter wel 22,6 jaar computertijd, terwijl het op een groot aantal samenwerkende processors tegelijk werd berekend.

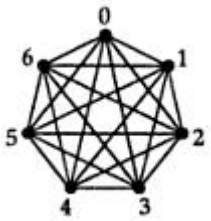
Hoofdstuk 8

Toepassingen Eulergrafen

We herinneren ons uit hoofdstuk 2 dat een Eulergraaf een graaf is die een Eulertoer bevat. Eulergrafen hebben vele mooie toepassingen. Één hiervan hebben we al gezien, namelijk het Chinese postbodeprobleem. In dit hoofdstuk zullen we nog enkele andere verrassende toepassingen bekijken. Voor meer voorbeelden, zie ook [5].

8.1 Domino

Bijna iedereen kent het en heeft het wel eens gespeeld: domino. Wij zullen ons hier richten op de standaard domino. Hierin hebben de helften nul tot zes ogen en komen alle 28 combinaties precies één keer voor. De vraag die we hier willen beantwoorden is of het mogelijk is om alle stenen zo neer te leggen dat ze een correcte gesloten kring vormen.
We maken een graaf $G = (V, E)$ met $V = \{0, 1, 2, 3, 4, 5, 6\}$ en $E = \{(a, b) : a, b \in V$ en ab is een dominosteen$\}$. Omdat elke combinatie precies één keer voorkomt, krijgen we zo - als we de stenen waarvan beide kanten hetzelfde zijn niet meenemen - de volledige graaf op 7 punten: K_7. We zien dat alle

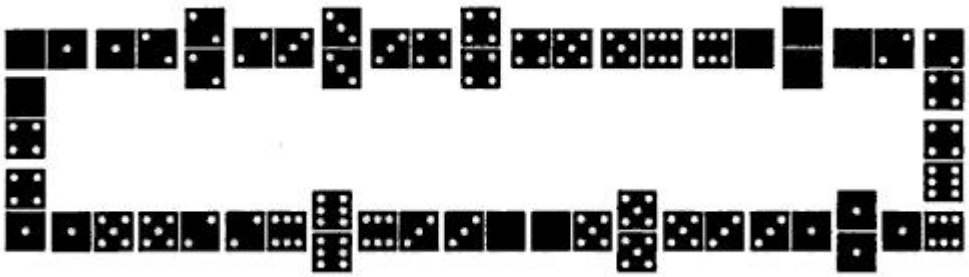


Figuur 8.1: K_7, de volledige graaf op 7 punten.

punten van even graad zijn. K_7 is dus een Eulergraaf en bevat een Eulertoer. Hieruit volgt dat het mogelijk is om de stenen in een gesloten kring neer te leggen. Een mogelijke Eulertoer is bijvoorbeeld:

01, 12, 23, 34, 45, 56, 60, 02, 24, 46, 61, 13, 35, 50, 03, 36, 62, 25, 51, 14, 40

Al deze lijnen corresponderen met een dominosteen. De dubbele stenen kunnen we nu op elke willekeurige (maar natuurlijk wel correcte) plek in de rij leggen. Zo krijgen we de volgende manier om alle stenen neer te leggen:



Figuur 8.2: Alle dominostenen in een gesloten kring.

8.2 RNA-ketens

Desoxyribonucleïnezuur, oftewel DNA, is de belangrijkste chemische drager van erfelijke informatie in alle bekende organismen. Grofweg kunnen we stellen dat DNA een keten is die bestaat uit vier verschillende basen: Guanine (G), Cytosine (C), Adenine (A) en Thymine (T).
RNA dient voor het kopiëren van genetische informatie die is opgeslagen in het DNA. Het lijkt qua chemische structuur op DNA, alleen bevat RNA de base Uracil (U) in plaats van Thymine.
Nu willen we graag kunnen bepalen uit welke rij van U's, C's, A's en G's een zekere RNA-keten bestaat. Dit is echter niet direct mogelijk. Met behulp van enkele enzymen kunnen we echter een heel eind komen. Dit zullen we illustreren met een voorbeeld.

Als mogelijke RNA-keten bekijken we:

GAUGGACC

In totaal zijn er $\frac{8!}{3!2!1!2!} = 1680$ mogelijke ketens die allemaal 3 G's, 2 A's, 1 U en 2 C's hebben.

Er bestaan twee verschillende soorten enzymen die de verbindingen in zo'n keten kunnen verbreken. De eerste, het zogenaamde G-enzym, verbreekt de keten na elke G-base. Onze keten zou dan uiteenvallen in de G-fragmenten:

G, AUG, G, ACC

Het tweede is het U,C-enzym. Dit enzym verbreekt de keten na elke U-base en na elke C-base. Zo zouden we dan dus de volgende U,C-fragmenten krijgen:

GAU, GGAC, C

Stel nu dat we niet weten in welke volgorde deze G-fragmenten voorkomen. Dan zijn er $4!/2! = 12$ mogelijke ketens die deze fragmenten in willekeurige volgorde bevatten. We merken nu op dat elk G-fragment eindigt met een G-base op één na: de base ACC. Dit fragment wordt *abnormaal* genoemd en moet wel het eind van de keten zitten. Hierdoor blijven er nog $3!/2! = 3$ mogelijke ketens over die leiden tot onze G-fragmenten, namelijk:

GAUGGACC
GGAUGACC
AUGGGACC

Maar we weten ook in welke U,C-fragmenten onze keten uitvalt. Alleen de eerste keten zal resulteren in onze U,C-fragmenten. De tweede heeft namelijk het fragment GGAU en de derde het fragment AU, die niet in onze rij voorkomen. We zijn nu dus in staat geweest om uit de 1680 ketens met dezelfde bases die keten terug te vinden waarmee we begonnen.

In het algemeen kan deze procedure volbracht worden door gebruik te maken van gesloten Eulerpaden en multigrafen. Ook dit zullen we illustreren aan de hand van een voorbeeld:
Stel dat we, na toepassing van beide splitsingsmethoden, de volgende fragmenten overhouden:

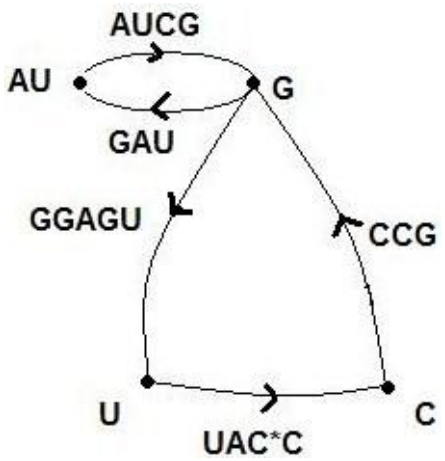
G-fragmenten: AUCG, G, CCG, AG, UAC
U,C-fragmenten: C, C, C, GAU, GGAGU, AC

Allereerst merken we op dat Het G-fragment UAC abnormaal is. Onze originele keten moet dus eindigen met dit fragment. Onze tweede stap is nu om het G-enzym toe te passen op de U,C-fragmenten en het U,C-enzym toe te passen op de G-fragmenten. Het fragment AUCG wordt dan dus opgesplitst in de fragmenten AU, C en G. Deze delen van de keten worden *verlengde basen* genoemd. De verlengde basen die niet in het het begin of het eind van

een fragment zijn, worden *intern* genoemd. Dus een fragment dat opsplitst in $n \geq 3$ verlengde basen, heeft $n - 2$ interne verlengde basen.
We maken nu een twee rijen. Één bestaande uit alle interne verlengde basen en één bestaande uit alle fragmenten die niet opgesplitst kunnen worden:

Interne verlengde basen: C, C, G, AG
Ongesplitste fragmenten: G, AG, C, C, C, AC

We zien dat deze twee rijen veel overeenkomsten hebben. Voor het grootste deel bevatten ze dezelfde basen. Alleen de derde C en AC komen niet in de eerste rij voor, maar wel in de tweede. Dit zijn de fragmenten waarmee de originele keten eindigt en begint. We hadden al opgemerkt dat de keten eindigt met het fragment UAC, waaruit volgt dat de originele keten begin met C en eindigt met AC.
Nu komen de grafen in beeld. Als we een normaal fragment hebben met meer dan één verlengde base, dan tekenen we de eerste en laatste verlengde base als punt. Tussen deze punten trekken we een lijn die we labelen met de naam van het corresponderende fragment. Zo krijgen we bijvoorbeeld een lijn van het punt AU naar G, gelabeld met de naam van het G-fragment AUCG. Als we dit voor alle basen gedaan hebben, voegen we nog één lijn toe. Deze wordt verkregen door een lijn te identificeren met de (langste) abnormale base - in ons geval UAC- en die te tekenen tussen de eerste verlengde base in dit abnormale fragment en met de eerste verlengde base in de keten. Deze lijn krijgt wel een verschillend label, nl X*Y, waarin X het langste abnormale fragment aangeeft, * dat het om een speciaal fragment gaat en Y de eerste verlengde base van de keten is. Onderstaand figuur laat het resultaat zien voor onze fragmenten:

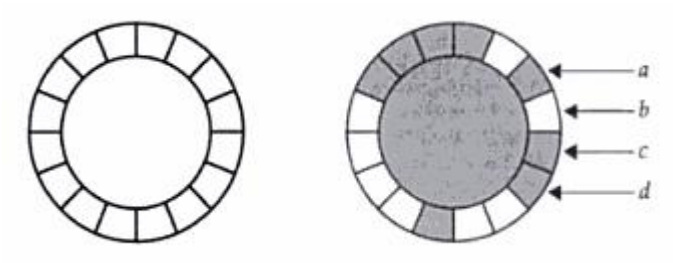


Figuur 8.3: De graaf die hoort bij onze RNA-fragmenten.

Een RNA-keten correspondeert nu met een Eulertoer in onze graaf die eindigt met het speciale fragment UAC*C. In ons voorbeeld hebben we maar één Eulertoer, namelijk (C, CCG, G, GAU, AU, AUCG, G, GGAGU, U, UAC*C, C). Als we nu de labels van de lijnen achter elkaar plakken krijgen we de reeks: CCGGAUAUCGGGAGUUAC*C. De bijbehorende RNA-keten krijgen we door de namen van de punten, die nu twee keer voorkomen, steeds één keer te verwijderen. Dit resulteert in: CCGAUCGGAGUAC
Men kan nagaan dat deze keten inderdaad resulteert in bovenstaande G- en U,C-fragmenten.

8.3 Teleprintersprobleem

Een probleem waar men tegen aanliep in de telecommunicatie is het zogenaamde 'rotating drum problem' of 'teleprintersprobleem'. Om dit probleem te illustreren bekijken we eerst de volgende afbeelding:



Figuur 8.4: Schematisch overzicht van een draaiende trommel.

Het oppervlak van een roterende trommel is verdeeld in 16 delen, de donkere delen bestaan uit geleidend materiaal en de lichte delen uit niet-geleidend materiaal. Nu is de vraag: kunnen we de positie van de trommel bepalen zonder er naar te kijken?

Om hier antwoord op te kunnen geven, representeren we de positie met vier binaire cijfers a, b, c en d, zoals in de afbeelding te zien is. Afhankelijk van de positie van de trommel zijn de eindpunten die gerepresenteerd worden door a, b, c en d geaard of juist geïsoleerd van de aarde. Zo zijn in bovenstaande afbeelding de eindpunten a, c en d geaard, terwijl b geïsoleerd is. De geaarde eindpunten geven een signaal door dat 1 representeert, terwijl de geïsoleerde punten een signaal voor 0 doorgeven.

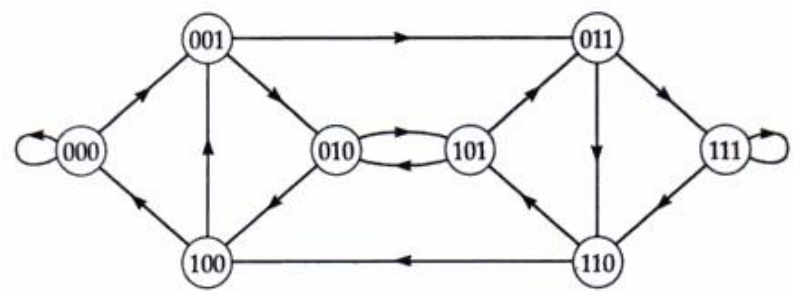
We willen nu dat elk van de zestien mogelijke posities van de trommel uniek gerepresenteerd wordt door een 4-cijferig binair woord $abcd$. Om dit voor

elkaar te krijgen, moeten de geleidende en niet-geleidende materialen op zo'n manier over de zestien delen verdeeld worden dat alle binaire woorden *abcd* voorkomen. Is dit mogelijk?

Om deze vraag te kunnen beantwoorden, construeren we een graaf met acht punten die corresponderen met de uit drie getallen bestaande binaire woorden:

$$000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111.$$

We tekenen lijnen tussen elk punt abc en de punten $bc0$ en $bc1$. Zo krijgen we de volgende graaf, die ook wel de *De Bruijngraaf* genoemd wordt:

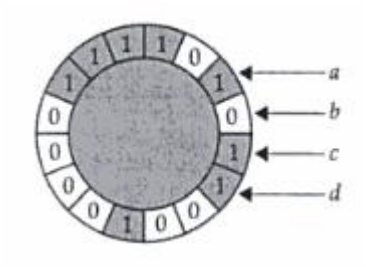


Figuur 8.5: De Bruijngraaf.

Omdat zowel de in- als de uitgraad van alle punten gelijk is aan 2 hebben we te maken met een Eulergraaf. Elke Eulertoer die we vinden is nu een oplossing voor ons probleem. Als voorbeeld nemen we de toer:

$$101 \to 011 \to 110 \to 100 \to 001 \to 010 \to 100 \to 000 \to$$
$$000 \to 001 \to 011 \to 111 \to 111 \to 110 \to 101 \to 010 \to 101.$$

We kunnen nu opeenvolgende termen als volgt samennemen: de drie termen $101 \to 011 \to 110$ nemen we samen tot 10110. Uit bovenstaande Eulertoer volgt dan de rij 1011001000011110, die ons onderstaande posities van geleidende en niet-geleidende delen geeft:



Figuur 8.6: Posities van alle (niet-)geleidende delen.

Als we naar bovenstaande afbeelding kijken, zien we dat de positie die we daar zien, correspondeert met het woord 1011. We draaien nu de trommel tegen de klok in. Zo krijgen we achtereenvolgens de volgende woorden:

$$0110, 1100, 1001, 0010, 0100, 1000, 0000, 0001,$$
$$0011, 0111, 1111, 1110, 1101, 1010, 0101, 1011.$$

Dit zijn precies alle zestien mogelijke binaire woorden bestaande uit vier getallen. Het is dus mogelijk om de positie van de trommel bepalen zonder er naar te kijken.

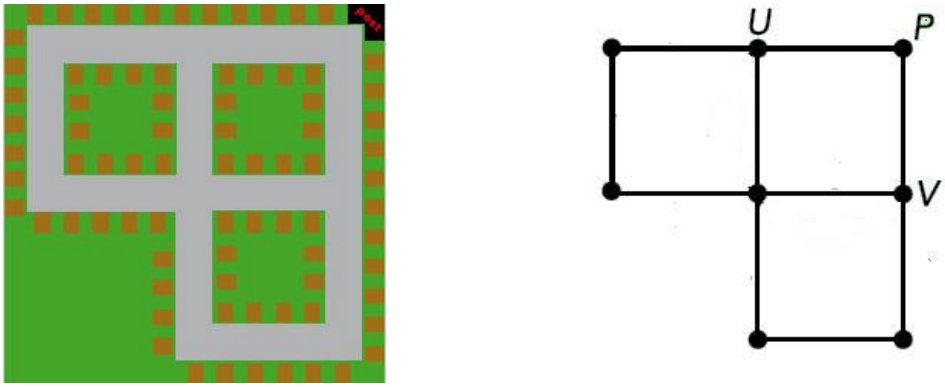
Hoofdstuk 9

Populaire samenvatting

Stel je bent postbode en je haalt de post op bij het postkantoor om die vervolgens te gaan bezorgen. Hierbij heb je brieven voor elke straat in je wijk en wil je uiteindelijk weer bij het postkantoor terugkomen. Je wilt nu natuurlijk de route zo uitstippelen, dat je zo min mogelijk hoeft te lopen. Hoe doe je dit? Dit probleem is beter bekend als het Chinese postbodeprobleem sinds het in 1962 voor het eerst werd beschouwd door de Chinese wiskundige Kwan Mei-Ko. Het is echter duidelijk dat het oplossen hiervan niet alleen nuttig is voor postbodes, maar ook voor bijvoorbeeld vuilnismannen, patrouillerende politiemannen en sneeuwschuivers.

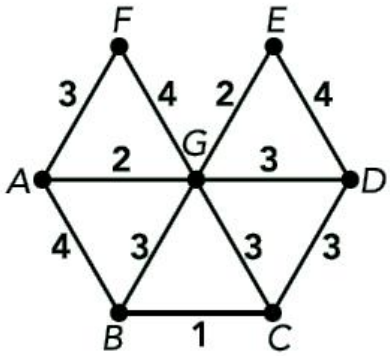
De hoofdvraag die ik in deze scriptie behandeld heb, is hoe we dit probleem op kunnen lossen. Een optie zou natuurlijk zijn om alle mogelijke routes in de wijk te bekijken en uit te rekenen welke daarvan de kortste is. Dit is echter, zeker als we naar wat grotere wijken kijken, een nogal omslachtige en tijdrovende methode. We zullen dus verder moeten zoeken naar een betere oplossing. Hierbij is het handig om een schematisch overzicht van een wijk te gebruiken. Dit kunnen we doen door gebruik te maken van grafentheorie

Grafentheorie is een tak van de (discrete) wiskunde die zich bezighoudt met de eigenschappen van grafen. Een graaf bestaat uit een verzameling punten, een verzameling lijnen en een relatie die elke lijn met twee punten associeert. We kunnen nu als we de plattegrond van een wijk hebben, de bijbehorende graaf maken. Dit doen we door elk kruispunt te tekenen als een punt en een lijn tussen punten te tekenen als de bijbehorende kruispunten verbonden worden door een straat. Dit wordt geïllustreerd door onderstaande afbeeldingen.



Figuur 9.1: Een plattegrond met bijbehorende graaf.

Omdat in de praktijk de meeste straten niet even lang zijn en we de kortste route zoeken, is het handig om de lengte van de straat aan te geven in onze graaf. We kunnen aan elke lijn een (niet-negatief) getal toekennen, dat correspondeert met de lengte van de bijbehorende straat. Zo krijgen we een zogenaamde 'gewogen graaf':



Figuur 9.2: Een gewogen graaf.

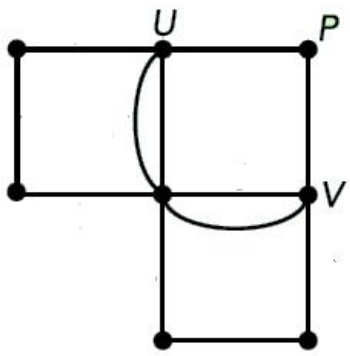
We kunnen het Chinese postbodeprobleem nu herformuleren als: "gegeven een gewogen graaf, vind de kortst mogelijke gesloten wandeling die alle lijnen van de graaf minstens één keer doorloopt."

Om dit probleem op te lossen, onderscheiden we twee gevallen. In het eerste geval hebben we te maken met een Eulergraaf. Dit houdt in dat elk punt van de graaf van even graad is, d.w.z dat er een even aantal lijnen in dat punt samenkomen. Als we dus via een ongebruikte lijn dat punt bereiken, dan

kunnen we altijd een andere lijn vinden die nog niet gebruikt is om dat punt te verlaten. Als ieder punt evengraads is, dan is het dus mogelijk om een gesloten wandeling te construeren die alle lijnen in de graaf precies één keer doorloopt. Deze wandeling geeft meteen de gewenste postbodetoer, waarmee het probleem is opgelost.

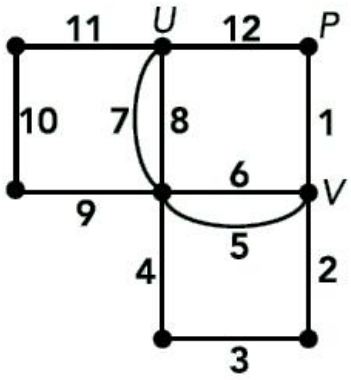
In de praktijk komt het echter vaker voor dat we niet te maken hebben met een Eulergraaf. In dat geval moeten we kijken naar de punten die ervoor zorgen dat een Eulertoer onmogelijk is. Dit zijn de punten waarin een oneven aantal lijnen samenkomen. Deze punten delen we op in paren. Vervolgens bepalen we voor elke mogelijke opdeling de kortste paden tussen deze paren punten en die tellen we bij elkaar op. We kiezen nu de partitie waarvoor de som van kortste paden het minst is. We verdubbelen dan de lijnen langs deze kortste paden. Zo krijgen we een graaf waarin alle punten even graad hebben. In zo'n graaf kunnen we dus een Eulertoer vinden. Dit is ook de optimale postbodetoer in onze originele graaf.

Om dit te illustreren keren we terug naar het voorbeeld uit het begin van deze samenvatting. We hebben hier twee punten van oneven graad, namelijk U en V. We kopiëren nu dus de lijnen langs het kortste pad tussen deze twee punten. Zo krijgen we de volgende graaf.



Figuur 9.3: Onze graaf met gekopiëerde lijnen.

In deze graaf zijn alle punten van even graad. We kunnen dan de volgende Euler- en dus postbodetoer vinden:



Figuur 9.4: Een optimale postbodetoer in onze graaf.

Hiermee is het Chinese postbodeprobleem opgelost.

Dit is echter niet de enige manier om tot een oplossing te komen. Een andere mogelijkheid om een optimale postbodetoer te vinden, is met behulp van lineair programmeren. Deze methode komt neer op het maximaliseren of minimaliseren van een lineaire functie ten opzichte van lineaire beperkingen. Ook deze methode heb ik in mijn scriptie behandeld.

Verder heb ik in mijn scriptie nog enkele variaties op het Chinese postbodeprobleem besproken, zoals het handelsreizigersprobleem en gerichte versies van het postbodeprobleem. Ook heb ik gekeken naar een paar andere toepassingen van Eulergrafen, zoals RNA-ketens en het teleprintersprobleem.

Hoofdstuk 10

Appendix

10.1 Grafentheorie

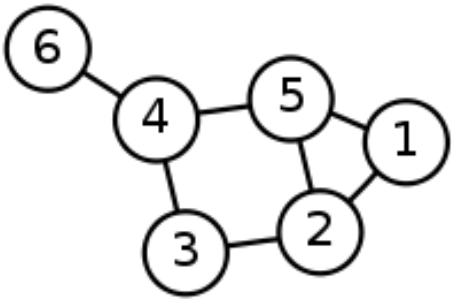
Grafentheorie is een tak van de (discrete) wiskunde die zich bezighoudt met de eigenschappen van grafen. Een *graaf* $G = (V, E)$ bestaat uit een verzameling punten $V = V(G)$, een verzameling lijnen $E = E(G)$ en een relatie die elke lijn met twee punten associeert. Deze twee punten hoeven niet verschillend te zijn.

Definitie 10.1. Zij $G = (V, E)$ een graaf.

- Een *simpele graaf* is een graaf waar tussen elk tweetal punten maximaal één lijn loopt. Daarnaast bevat een simpele graaf ook geen lijnen waarvoor geldt dat begin- en eindpunt hetzelfde zijn.
- Een *pseudograaf* is een graaf die niet simpel hoeft te zijn. Hij kan dus loops en dubbele lijnen bevatten
- De *graad* $d_G(x)$ van een punt x in de graaf G is het aantal lijnen in dat punt.
- Een *wandeling* in een graaf $G = (V, E)$ is een alternerende rij $W = (v_0, e_1, v_1, ..., e_k, v_k)$ van punten en lijnen zodanig dat de lijn e_i de punten v_{i-1} en v_i verbindt voor elke $i = 1, ..., k$. Het getal k heet de *lengte* van de wandeling
- Als de lijnen $e_1, e_2, ..., e_k$ allemaal verschillend zijn, dan wordt de wandeling ook wel een *spoor* genoemd.
- Als de punten $v_0, v_1, ..., v_k$ allemaal verschillend zijn, dan wordt de wandeling ook wel een *pad* genoemd.

- G heet *samenhangend* als er een pad is tussen elk paar $u, v \in V$.
- Een *brug* is een lijn die bij weglating het aantal componenten van de graaf verhoogt. Bij een samenhangende graaf is een brug dus een lijn die de graaf in twee losse stukken doet uiteenvallen.
- Het *complement* van G is de graaf $\bar{G} = (V, F)$, waarbij F uit alle lijnen bestaat die niet tot E behoren.
- $H = (V', E')$ is een *deelgraaf* van $G = (V, E)$ als $V' \subseteq V$ en $E' \subseteq E$ en elke lijn in E' eindpunten heeft in V'.
- Zij $U \subseteq V$. Dan noemen we $G[U] = (U, \{e \in E | e \subseteq U\})$ een *geïnduceerde deelgraaf*. Oftewel: Je kiest een aantal punten uit je graaf en tekent alle lijnen tussen deze punten die ook in je oorspronkelijke graaf staan.
- Een *volledige graaf* K_n op n punten is een graaf waarin alle punten onderling verbonden zijn.

Voorbeeld 10.2. We beschouwen onderstaande graaf $G = (V, E)$



Figuur 10.1: Een voorbeeld van een graaf.

Voor deze graaf geldt:

- $V = \{1, 2, 3, 4, 5, 6\}$.

- $E = \{(1, 2), (1, 5), (2, 3), (2, 5), (3, 4), (4, 5), (4, 6)\}$.

- G is samenhangend.

- G is simpel.

- $d(1) = d(3) = 2, \qquad d(2) = d(4) = d(5) = 3, \qquad d(6) = 1.$

10.2 Lineair Programmeren

Een lineair programmeren probleem kan gedefiniëerd worden als het probleem van het maximaliseren of minimaliseren van een lineaire functie ten op zichte van lineaire beperkingen. Deze beperkingen kunnen gelijkheden of ongelijkheden zijn.
Met elk probleem correspondeert een duaal lineair programmeren probleem. Zij c en x n-vectoren, b en y m-vectoren en A een $(m \times n)$-matrix.

Definitie 10.3. Het *duale probleem* van het standaard maximalisatie probleem:

$$\text{Maximaliseer } c^T x$$

$$\text{t.o.v. de beperkingen } Ax = b \text{ en } x = 0$$

is gedefiniëerd als het standaard minimalisatie probleem:

$$\text{Minimaliseer } y^T b$$

$$\text{t.o.v. de beperkingen } y^T A = c^T \text{ en } y = 0$$

Voorbeeld 10.4. We bekijken het standaard maximalisatie probleem: Vind x_1 en x_2 zodanig dat $x_1 + x_2$ gemaximaliseerd wordt ten op zichte van de beperkingen:

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_1 + 2x_2 &\leq 4 \\ 4x_1 + 2x_2 &\leq 12 \\ -x_1 + x_2 &\leq 1 \end{aligned}$$

Het hierbij behorende duale probleem is het standaard minimalisatie probleem: Vind y_1, y_2 en y_3 zodanig dat $4y_1 + 12y_2 + y_3$ geminimaliseerd wordt ten op zichte van de beperkingen:

$$\begin{aligned} y_1 &\geq 0 \\ y_2 &\geq 0 \\ y_3 &\geq 0 \\ y_1 + 4y_2 - y_3 &\geq 1 \\ 2y_1 + 2y_2 + y_3 &\geq 1 \end{aligned}$$

Definitie 10.5. Een *H-veelvlak* is een intersectie van gesloten halfruimten. Het is een verzameling $P \subseteq \mathbb{R}^d$ van de vorm:

$$P = P(A, z) = \{x \in \mathbb{R}^d \,:\, Ax \leq z\} \qquad \text{voor } A \in \mathbb{R}^{m \times d},\ z \in \mathbb{R}^m.$$

(Hierbij is $Ax \leq z$ verkorte notatie voor het systeem van ongelijkheden gegeven door $a_1 x \leq z_1, ..., a_{,} x \leq z_m$).

Stelling 10.6. *(Hoofdstelling voor polytopen)*
Zij $P \subseteq \mathbb{R}^d$, $V \in \mathbb{R}^{d \times n}$, $A \in \mathbb{R}^{m \times d}$ en $z \in \mathbb{R}^m$. Dan geldt:
P is het convex omhulsel van een eindige verzameling punten $P = conv(V)$

$$\Leftrightarrow$$

P is een begrensde intersectie van halfruimten $P = P(A, z)$.

Bewijs. Voor een bewijs van deze stelling verwijzen we naar [8]. □

Bibliografie

[1] Douglas B. West, *Introduction to Graph Theory*, second edition, Prentice Hall, 2001.

[2] Edgar G. Goodaire & Michael M. Parmenter, *Discrete Mathematics with Graph Theory*, second edition, Prentice Hall, 2002.

[3] Jack Edmonds & Ellis Johnson, *Matching, Euler Tours and the Chinese Postman*, North-Holland Publishing Company, 1973.

[4] Edward Minieka, *Optimization Algorithms for Network and Graphs*, Marcel Dekker inc, 1978.

[5] Fred S. Roberts, *Graph Theory and its Applications to Problems of Society*, Society for Industrial and Applied Mathematics, 1978.

[6] J. A. Bondy & U. S. R. Murty, *Graph Theory with Applications*, The MacMillan Press ltd, 1976.

[7] Joan M. Aldous & Robin J. Wilson, *Graphs and applications: an introductory approach*, fourth edition, Springer-verlag, 2004.

[8] Günter M. Ziegler, *Lectures on Polytopes*, Springer-Verlag New York, 1995.

[9] Thomas S. Ferguson, *Lineair Programming, A Concise Introduction* http://www.usna.edu/Users/weapsys/avramov/Compressed

[10] http://staff.science.uva.nl/∼depaepe/Grafen2007.pdf