Samenvatting

Optimalisering - Collegejaar 2014-2015

Algemene samenvatting

OPTIMALISERING - SAMENVATTING

1. Grafen en netwerken

Ongerichte graaf: $G = (V, E)$
Gerichte graaf: $G = (V, A)$
Multigraaf: graaf waarin meerdere kanten kunnen bestaan tussen 2 knopen.
Boom: samenhangende graaf zonder cykels (heeft $|V| - 1$ kanten).
Bos: verzameling knoop-disjuncte bomen.
Netwerk: $N = (s, t, V, A, b)$ (gerichte graaf met begin, eind en capaciteiten).

Een **wandeling** in een graaf is een aaneenschakeling van knopen, waartussen de kanten in de graaf liggen. De wandeling is **gesloten** als de laatste knoop dezelfde als de eerste is (en meer dan één knoop bevat).
Als alle knopen in een wandeling maar één keer voorkomen, heet deze wandeling een **pad**. Een gesloten pad is een **kring/cykel**. Als er tussen elk tweetal knopen een pad bestaat, is de graaf **samenhangend**.
Een graaf is **bipartiet** als er een partitie van de knopen (V_1 en V_2) bestaat zodat elke kant $e \in E$ grenst aan één knoop in V_1 en aan één knoop in V_2.
Een multigraaf is **Euleriaans** als er een gesloten wandeling bestaat waarbij elke knoop minstens één keer voorkomt en elke kant precies één keer voorkomt.

Een **opspannende boom** bevat alle knopen in een graaf.

2. Probleemformulering

Standaardvorm LP:

$$\begin{aligned} \min \quad & c^T x \\ \text{o.d.v.} \quad & Ax = b \text{ (met b } > 0) \\ & x \geq 0 \end{aligned}$$

Geef bij het formuleren van een LP duidelijk aan wat de **beslissingsvariabelen** zijn. Vergeet ook de voorwaarden die aangeven welke waarden deze variabelen aan kunnen nemen niet (zoals $x \geq 0$, $x \in \{0, 1\}$ of $x \in \mathbb{Z}^n$). Zorg verder dat alle voorwaarden **lineair** zijn.

3. Basisoplossingen

Om het Simplexalgoritme te beginnen, zet je $n - m$ variabelen gelijk aan 0 ($n =$ aantal beslissingsvariabelen, $m =$ aantal voorwaarden). Vervolgens los je $Ax = b$ op voor de overige m variabelen. Een (unieke) oplossing hiervan is een **basisoplossing**. Als deze basisoplossing toegelaten is, is het een **bfs (basic feasible**

solution).
Een bfs waarin één of meerdere basisvariabelen ook 0 zijn, is **gedegenereerd**.

Stellingen:

- Als meerdere basissen horen bij dezelfde bfs x, dan is x gedegenereerd.
- Voor elke instantie van LP (met $F \neq \emptyset$) bestaat er een bfs.

4. Buurruimtes

Als $f \in F$ een toegelaten oplossing is van een bepaald probleem, dan is de **buurruimte** van f de verzameling oplossingen die in een bepaald opzicht dichtbij f liggen. De beste oplossing in een buurruimte N is een **lokaal optimum**. Als dit lokale optimum ook een globaal optimum is, dan is N **exact**.

5. Simplexalgoritme

Om het Simplexalgoritme te kunnen toepassen, moet je de voorwaarden omschrijven naar een startoplossing:

- $\leq$ - voorwaarde: s erbij optellen (**slackvariabele**)
- $=$ - voorwaarde: x^a erbij optellen (**artificiële variabele**)
- $\geq$ - voorwaarde: x^a erbij optellen, s ervan aftrekken (**surplusvariabele**)

Als je een artificiële variabele hebt toegevoegd, moet je deze minimaliseren naar 0. Als dit niet mogelijk is, dan heeft het probleem geen bfs. Dit deel van het Simplexalgoritme wordt **fase 1** genoemd.

Als er geen artificiële variabelen meer zijn, begint **fase 2**. Hierbij wordt een **intredende variabele** gekozen (de variabele waarvan de gereduceerde kosten het hoogst (bij maximaliseren) of het laagst (bij minimaliseren) zijn). In de kolom die bij deze variabele hoort, wordt vervolgens met behulp van de **min-ratiotest** de **uittredende variabele** gekozen (degene waarbij b_i/a_{ij} het kleinst is, maar $a_{ij} > 0$).
Conclusies uit het Simplexalgoritme:

- Als x^a niet geminimaliseerd kan worden naar 0, is het probleem niet toegelaten.
- Als alle a_{ij} in de kolom van de intredende variabele kleiner dan/gelijk aan 0 zijn, is het probleem onbegrensd.
- Als $c_j = 0$ voor een niet-basisvariabele, dan is het optimum niet uniek (de optimale oplossing is dan een convexe combinatie van de verschillende optima).
- Als $x_j = 0$ voor een basisvariabele, dan is het probleem gedegenereerd (er horen meerdere basissen bij dezelfde oplossing).

6. Dualiteit

max	$c^T x$	min	$b^T \pi$
	$a_i x \leq b_i$		$\pi_i \geq 0$
	$a_i x = b_i$		$\pi_i \in \mathbb{R}$
	$a_i x \geq b_i$		$\pi_i \leq 0$
	$x_j \leq 0$		$\pi^T A_j \leq c_j$
	$x_j \in \mathbb{R}$		$\pi^T A_j = c_j$
	$x_j \geq 0$		$\pi^T A_j \geq c_j$

Stellingen:

- **Zwakke dualiteitsstelling**: Als $\hat{x}$ toegelaten is voor (P) (met doelfunctie z) en $\hat{\pi}$ is toegelaten voor (D) (met doelfunctie w), dan geldt $z(\hat{x}) \geq w(\hat{\pi})$ (voor minimalisatie) of $z(\hat{x}) \leq w(\hat{\pi})$ (voor maximalisatie).
- **Sterke dualiteitsstelling**: Als (P) een toegelaten optimum x^* heeft, dan heeft (D) ook een toegelaten optimum π^*. Hierbij geldt $z(x^*) = w(\pi^*)$.
- **Complementary slackness**: Als $\hat{x}$ en $\hat{\pi}$ toegelaten oplossingen zijn voor (P) en (D), dan geldt:
  $\hat{x}$ en $\hat{\pi}$ zijn optimaal $\iff \hat{x}_j(\hat{\pi}^T A_j - c_j) = 0$ en $\hat{\pi}_i(b_i - a_i\hat{x}) = 0$ voor $j = 1, ..., n$ en $i = 1, ..., m$.
- **Farkas' lemma**: Precies één van de volgende dingen geldt:
  - $\{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\} \neq \emptyset$
  - $\{\pi \in \mathbb{R}^m \mid \pi^T A \geq 0, \pi^T b < 0\} \neq \emptyset$

Uit het optimale Simplextableau kun je de optimale waarden van de duale variabelen terugvinden. Dit zijn de gereduceerde kosten van de initiële basisvariabelen (je moet wel zelf opletten of ze positief of negatief moeten zijn).
Mogelijke primaal-duale combinaties:

- (P) begrensd en (D) begrensd
- (P) onbegrensd en (D) niet-toegelaten
- (P) niet-toegelaten en (D) onbegrensd
- (P) niet-toegelaten en (D) niet-toegelaten

7. Duale Simplexalgoritme

Het Duale Simplexalgoritme begint met een tableau waarin alle $\bar{c}_j \geq 0$ (minimalisatie) of alle $\bar{c}_j \leq 0$ (maximalisatie), ofwel met een primale basisoplossing (niet noodzakelijk toegelaten). Vervolgens wordt een uittredende variabele gekozen waarvoor $\bar{b}_i = \min\{\bar{b}_i \mid \bar{b}_i < 0\}$ (als alle $\bar{b}_i \geq 0$, dan is de oplossing toegelaten en optimaal).

De intredende variabele wordt dan gekozen met de min-ratiotest, maar dan met $\bar{c}_j$ in plaats van $\bar{b}_j$: je kiest de kolom waarvoor $\frac{\bar{c}_j}{\bar{a}_{ij}} = \min\left\{\left|\frac{\bar{c}_j}{\bar{a}_{ij}}\right| \mid \bar{a}_{ij} < 0\right\}$. Als alle $\bar{a}_{ij}$ in de rij groter dan/gelijk aan 0 zijn, dan heeft het probleem geen toegelaten

oplossing.

8. Orde-notaties

Als $f(n)$ en $g(n)$ functies van $\mathbb{N}$ naar $\{x \in \mathbb{R} \mid x > 0\}$ zijn, dan geldt:

- $f(n) = O(g(n))$ als er een $c > 0$ bestaat zodat vanaf bepaalde n geldt dat $f(n) \leq cg(n)$
- $g(n) = \Omega(g(n))$ als er een $c > 0$ bestaat zodat vanaf bepaalde n geldt dat $f(n) \geq cg(n)$
- $f(n) = \Theta(g(n))$ als er $c, c' > 0$ bestaan zodat vanaf bepaalde n geldt dat $cg(n) \leq f(n) \leq c'g(n)$

Stellingen:

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Longrightarrow f(n) \in O(g(n))$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c > 0 \Longrightarrow f(n) \in \Theta(g(n))$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Longrightarrow f(n) \in \Omega(g(n))$

9. De klassen P en NP

De **klasse P** is de verzameling van alle beslissingsproblemen die in polynomiale tijd oplosbaar zijn. De **klasse NP** is de verzameling van alle problemen die in polynomiale tijd verifieerbaar zijn, oftewel alle beslissingsproblemen waarbij er voor iedere ja-instantie I een certificaat $c(I)$ bestaat, waarvan de geldigheid met een polynomiaal algoritme kan worden nagegaan.

Een probleem Π_2 is **NP-volledig** als $\Pi_2 \in$ NP en er voor alle $\Pi_1 \in$ NP geldt dat $\Pi_1 \propto \Pi_2$ (dit betekent dat Π_1 een speciaal geval is van Π_2, dus Π_1 kan niet moeilijker zijn dan Π_2). Om NP-volledigheid te bewijzen, moet je bewijzen dat het probleem in NP zit en dat er een NP-volledig probleem is dat naar jouw probleem reduceert (dan geldt dit automatisch voor alle problemen in NP).

Problemen in de klasse P:

- Kortste pad (Dijkstra's algoritme)
- Max flow (Ford-Fulkerson en Max Flow-Min Cutstelling)
- Minimum Spanning Tree (Prim, Greedy en Kruskal)

10. Approximatie-algoritmen

Een algoritme A is een ρ **- approximatie-algoritme** voor een probleem Π als A voor iedere toegelaten instantie I van Π in polynomiale tijd een oplossing van waarde $z_A(I)$ produceert zodat:

- $z_A(I) \leq \rho \cdot z_{OPT}(I)$ (minimalisatie) ($\rho \geq 1$)
- $z_A(I) \geq \rho \cdot z_{OPT}(I)$ (maximalisatie) ($\rho \leq 1$)

Het getal ρ hierin is de prestatiegarantie. Als deze gelijk is aan 1, dan zit Π in de klasse P.

Om te bewijzen dat iets een approximatie-algoritme is, moet je 3 vragen beantwoorden:

- Heeft het algoritme polynomiale rekentijd?
- Retourneert het een toegelaten oplossing?
- Wat is de prestatiegarantie?

11. Integer Lineair Programmeren

LP's waarbij de beslissingsvariabelen gehele getallen aan moeten nemen (**ILP's**) zijn NP-volledig. Een ILP heeft een **LP-relaxatie**, dit is hetzelfde probleem, maar zonder de voorwaarde dat de beslissingsvariabelen integers zijn. Als z_{IP} de oplossing van het ILP is en z_{LP} de oplossing van de LP-relaxatie, dan geldt voor alle toegelaten oplossingen z_{toeg}:

- $z_{toeg} \leq z_{IP} \leq z_{LP}$ (maximalisatie)
- $z_{LP} \leq z_{IP} \leq z_{toeg}$ (minimalisatie)

Een vierkante geheeltallige matrix B is **unimodulair (UM)** als $\det(B) = \pm 1$. Een geheeltallige matrix A is **totaal unimodulair (TUM)** als elke vierkante deelmatrix determinant -1, 0 of 1 heeft (dus singulier of unimodulair is).
Dit is het geval als elke kolom maximaal 2 getallen ongelijk aan 0 bevat en als de rijen van A in twee verzamelingen I_1 en I_2 kunnen worden verdeeld zodat:

- Als een kolom 2 elementen van hetzelfde teken heeft, dan behoren de bijbehorende rijen tot verschillende verzamelingen.
- Als een kolom 2 elementen met verschillend teken heeft, dan behoren de bijbehorende rijen tot dezelfde verzameling.

Stellingen:

- Als A een TUM matrix is, dan geldt dat de hoekpunten van $\{x \mid Ax = b, x \geq 0\}$ en $\{x \mid Ax \leq b, x \geq 0\}$ geheeltallig zijn voor iedere geheeltallige b. De LP-relaxatie geeft dat de oplossing voor het ILP.

ILP's kunnen op 2 manieren worden opgelost.
De eerste is **branch-and-bound**. Hierbij splits je steeds in 2 deelproblemen, waarbij elke toegelaten oplossing zich in exact één deelprobleem bevindt. Over het algemeen wordt gesplitst voor de variabele waarvan het fractionele deel het dichtst bij 0,5 ligt. Vervolgens los je de bijbehorende LP-relaxatie op. Intussen houd je de boven- en ondergrens van de oplossing bij. Bij minimalisatie is de bovengrens voor de hele boom en geldt de ondergrens alleen lokaal (bij maximalisatie andersom).
Een deelprobleem hoeft niet meer verder gesplitst te worden als:

- Het deelprobleem niet toegelaten is (ofwel geen enkele geheeltallige oplossing heeft).
- Het deelprobleem een geheeltallige LP-relaxatie heeft (dit is dus de nieuwe bovengrens (minimalisatie) of ondergrens (maximalisatie)).

- De ondergrens groter dan/gelijk aan de bovengrens is (dus $z_{LP} \geq z - toeg$ of $z_{toeg} \geq z_{LP}$, afhankelijk van minimalisatie of maximalisatie), aangezien de oplossing die je uiteindelijk vindt nooit beter is dan die van de LP-relaxatie.

Een tweede manier om ILP's op te lossen is het gebruik van **Gomorycuts / Gomorysneden**. Hierbij schrijf je de gelijkheid die in een rij van het Simplextableau staat om naar een ongelijkheid door de gehele delen naar 1 kant te halen, en de rest naar de andere kant, en te gebruiken dat dit kleiner dan/gelijk aan 0 moet zijn (omdat het geheeltallig moet zijn en de variabelen allemaal positief zijn). Door deze ongelijkheid toe te voegen, wordt het gebied met toegelaten oplossingen kleiner. De oplossing van de LP-relaxatie gaat er dan uit, maar er worden geen geheeltallige punten afgesneden. Door na het toevoegen Duale Simplex toe te passen, vind je een nieuwe oplossing die dichter bij de oplossing van het ILP ligt.

12. Algoritmen

12.1. **Dijkstra's algoritme.** Het algoritme van Dijkstra wordt gebruikt om het kortste pad tussen twee punten s en t te vinden. Het algoritme kent aan iedere knoop x een label $p(x)$ toe, dat aangeeft wat de kortste lengte is van een pad van s naar x, via alleen knopen in W. De verzameling W is de verzameling van alle punten die tot nu toe een label $p(x)$ hebben gekregen. Het algoritme begint met $W = \{s\}$ en $p(s) = 0$. Alle punten die direct te bereiken zijn vanaf s hebben nu een label $p(x)$, het punt waarvoor deze $p(x)$ het laagst is wordt toegevoegd aan W. Alle labels van punten die niet in W zitten worden dan geupdate: $p(i) = \min\{p(i), p(x) + c_{(x,i)}\}$. Bij de punten wordt steeds het label gezet, met daarvoor het punt waarvandaan dit kortste pad komt. Het algoritme stopt als $W = V$.

12.2. **Min cut, max flow stelling.** Een **s,t snede** in een netwerk is een snede waarbij het netwerk in tweeën wordt gedeeld, waarbij s in het ene deel zit en t in het andere deel. De capaciteit van de snede is de som van alle pijlen die van het eerste deel naar het tweede deel lopen (dus alle voorwaartse pijlen waar doorheen wordt gesneden). De kleinste capaciteit van een s,t snede in een netwerk is gelijk aan de maximale flow in dat netwerk.

12.3. **Ford-Fulkerson algoritme.** Het Ford-Fulkerson algoritme wordt gebruikt om de maximale stroom in een netwerk te bepalen. Dit wordt gedaan door een residuaal-netwerk bij te houden, waarin de nettocapaciteiten van alle pijlen staan. Als hierin een pad gevonden kan worden, wordt hierover de hoogst mogelijke stroom gestuurd. In het nieuwe residuaal-netwerk, komt er op alle plekken waar deze stroom loopt een pijl in de andere richting bij, met de waarde van de stroom. Verder vermindert de capaciteit van de gebruikte pijlen met deze stroom. Dit gaat door tot er geen pad meer te vinden is in het residuaal netwerk, de maximum flow is dan de som van alle gevonden paden, of de som van de capaciteiten van de pijlen waarop het algoritme vastloopt (dus waar de stroom ophoudt).

12.4. **Algoritme van Prim.** Het algoritme van Prim geeft een minimum spanning tree in een graaf. Je kiest 1 knoop, dit is de verzameling U in het begin. Je zoekt steeds de kortste kant die van een knoop in U naar een knoop buiten U gaat. De knoop waar deze kant heen gaat voeg je toe aan U, en de betreffende kant voeg je toe aan T. Als alle knopen in U zitten, is T de MST.

12.5. **Snellere variant van algoritme van Prim.** Bij dit algoritme verdeel je de graaf in componenten. In het begin bestaat ieder component uit precies één knoop. Voor ieder component kies je de kortste kant die hieruit gaat, deze kant voeg je toe aan de tree T. Als je dit voor ieder component hebt gedaan, en de boom is nog niet opspannend, dan neem je als nieuwe componenten de verschillende (disjuncte) bomen in de graaf, hiermee doe je hetzelfde. Als T de graaf opspant, dan heb je de MST gevonden.

12.6. **Algoritme van Kruskal.** Het algoritme van Kruskal is een greedy algoritme voor het MST-probleem, dit wil zeggen dat in iedere iteratie de lokaal optimale keuze wordt gemaakt. In het begin is F leeg, E is de verzameling van alle kanten. Je kiest steeds de kortste kant in E, voegt deze aan F toe (als dit geen cykel geeft) en verwijdert hem uit E. Als E leeg is, is F de MST.