Inhoud van de cursus Numerieke technieken en optimalisatie

9 Februari 2010

Samenvatting De cursus zal bestaan uit 7 lessen grafentheorie (optimalisatie in netwerken) en 3 lessen optimalisatie in $\mathbb{R}^n$. Voor het deel grafentheorie baseren we ons op het boek "Introduction to Graph Theory" (second edition), van Douglas. B. West. Voor het deel over optimalisatie wordt het boek "Inleiding tot de numerieke wiskunde", van A. Bultheel gebruikt.

In de volgende bladzijden zal besproken worden wat de leerstof van de cursus omvat. Het boek van grafentheorie is lijvig (ca 600 bladzijden), het is onmogelijk en ook onnodig om het ganse boek te behandelen. We halen in deze cursus de meest interessante begrippen uit het boek en lichten deze nader toe.

Op de volgende bladzijden zijn de titels van de secties en subsecties weergegeven. De titels en ook de slides zijn in het engels, om geen verwarring te zaaien met de originele tekst. Er werd voor Engels gekozen omdat de meeste literatuur in deze taal geschreven is, en omdat de Nederlandse naamgeving niet volledig uniform is. Per les zullen er telkens ongeveer 20 pagina's van het boek gezien worden, dit is voorbeelden en dergelijke inclusief. Na de weergave van de behandelde delen, per sectie, is er een lijst met oefeningen, te maken in de zelfstudie.

Voor het gedeelte optimalisatie wordt hetzelfde principe toegepast. Tevens zal de laatste les van de cursus over het numerieke rekenprogramma Matlab handelen. In deze "Computer les" zullen we een algoritme behandeld in de les zelf implementeren en onze theoretische kennis staven aan de praktijk.

Examen: Normaliter wordt de volgende examenvorm toegepast.

- Theoretisch gedeelte: De theorie wordt eerst afgelegd in de vorm van een gesloten boek examen.
- Oefeningen deel: De oefeningen mogen met behulp van het boek en de oefeningen uit de lessen opgelost worden. Het oefeningengedeelte werkt met een soort tip-systeem. Zie je niet hoe aan de oefening te starten of zit je vast, dan kan je altijd voor een kleine prijs verdere informatie krijgen.
- De theorie zal mondeling verdedigd worden, het oefeningen gedeelte niet.

1 Chapter 1: Fundamental concepts

1.1 What is a Graph

- "The definition" volledig.
- "Graphs as models" volledig. (Maak je geen zorgen als sommige definities moeilijk overkomen, in latere hoofdstukken komen we op alle begrippen terug!)
- "Matrices and Isomorphism" alles behalve, op pg. 8, we slaan 1.1.23 en 1.1.24 over. We gaan verder vanaf 1.1.25.
- "Decomposition and Special Graphs", we zien enkel 1.1.32 en 1.1.33. De rest: pg. 11-14, vanaf 1.1.34 tot de oefeningen slaan we over.
- "Exercises:" 1.1.2 (Enkel het deeltje voor een 3-vertex pad, en 2 adjacency en 2 incidence matrices tekenen is voldoende.), 1.1.5 (Zeg gewoon juist of fout.), 1.1.10 (Toon aan dat als x en y twee vertices zijn die niet verbonden zijn in G, dat elke vertex in $\bar{G}$ verbonden is met ofwel x en y.), 1.1.30 (Tip bij de laatste zin "What do the entries in position (i, j)...": voor AA^T kijk je naar de gezamenlijke buren van vertex i en j, voor MM^T bekijk je ook multiple edges en zoek naar het verband tussen edges en de knopen i en j. Om dit af te leiden schrijf je best expliciet het matrix product uit voor een element (i, j) uit de matrix AA^T (d.w.z. $(AA^T)_{ij} = \sum_\ell ....$) en idem voor MM^T.), 1.1.37 (Vergelijk het aantal interne knopen en eindknopen in een pad van lengte minstens 5. Als je dan weet dat je graaf ontbonden kan worden in k paden van lengte 5 en dat elke knoop in elk pad graad 3 moet hebben dan kom je tot een contradictie.).

1.2 Paths, Cycles and Trails

- De intro slaan we volledig over (dat is basiswiskunde).
- "Connection in Graphs" volledig buiten de laatste twee items op pg. 23, namelijk 1.2.13 en het bewijs van 1.2.14 laten we vallen, de stelling 1.2.14 moet wel gekend zijn.
- "Bipartite Graphs" de items 1.2.20 t/m 1.2.23 laten we vallen.
- "Eulerian circuits" de items 1.2.29 t/m 1.2.33 laten we vallen.
- "Exercises:" 1.2.1, 1.2.5 (Toon enkel aan dat een cut-vertex graad > 1 heeft.), 1.2.8, 1.2.10 (Enkel item (b). De bewering is fout, zoek een eenvoudig tegenvoorbeeld.), 1.2.30 (Enkel het eerste deel, niet het bipartiete deel, toon dit aan per inductie op k met A^k.), 1.2.32 (Wat ontbreekt er, niet bewijzen.), 1.2.35 (Niet bewijzen, voer enkel het algoritme uit op K_5.), 1.2.38 (Toon dit aan per inductie, voor de inductiestap splits je op in twee mogelijkheden, 1 van de twee mogelijkheden gebruikt lemma 1.2.25.).

1.3 Vertex degrees and Counting

- Intro volledig.
- "Counting and bijections" We bekijken enkel 1.3.3 t/m 1.3.6 en ook nog 1.3.9 en 1.3.11, de rest valt weg.
- "Extremal Problems" We bekijken enkel 1.3.13 en 1.3.15 de rest valt weg.
- "Graphic sequences" laten we volledig vallen.
- "Exercises:" 1.3.1 (Stel dat er geen pad bestaat, dan bestaan er twee componenten, kom tot een contradictie.), 1.3.3 (Het aantal driehoeken komt overeen met het aantal gemeenschappelijke buren, tel het aantal buren.), 1.3.14 (Enkel het eerste deel, schrijf alle mogelijke graden uit voor een graaf met n knopen.), 1.3.41 (Kijk naar het bewijs van 1.3.15: Toon aan dat vanuit "een specifieke" vertex x, elke andere vertex kan bereikt worden door aan te tonen dat de doorsnede van buren van x en van y (met y geen buur van x), verschillend is van de lege. Tel het aantal elementen in de doorsnede met een ondergrens.).

1.4 Directed Graphs

Hier gaan we enkele definities van bekijken. Dit zijn onmiddellijke uitbreidingen van de traditionele gevallen.

- "Definitions and examples" We bekijken enkel de belangrijkste definities, namelijk 1.4.1, 1.4.2, 1.4.3, 1.4.6, 1.4.9, 1.4.10, 1.4.11, 1.4.12, 1.4.13 dienen gekend te zijn.
- "Vertex Degrees" Enkel de definitie 1.4.17 en Propositie 1.4.18.
- "Euleran Digraphs" De items 1.4.22, 1.4.23 en 1.4.24 worden behandeld.
- "Orientations and tournaments" laten we weg.
- "Exercises:" 1.4.4, 1.4.9 (Zoek kleine tegenvoorbeelden en construeer dan een algemeen tegenvoorbeeld. Zoek niet te lang als je het niet vindt.), 1.4.10 (Tip: De $\Rightarrow$ richting is triviaal. Voor de $\Leftarrow$ vertrek je van een knoop x. Stel S de verzameling punten die bereikt kunnen worden vanuit x. Dan vormen S en $V(G)\backslash S$ twee partities, pas nu de aanname toe.), 1.4.11 (Tip: Kijk even wat er juist bedoeld wordt met een sterke component met geen binnenkomende en geen buitengaande zijden. Voor het bewijs construeer je vertrekkende van graaf G een nieuwe georiënteerde graaf, met per sterke component 1 vertex. De vertices van de nieuwe graaf zijn verbonden via directed edges als er in de originele graaf tussen een element van één component en een element van de andere component een pijl loopt. Hieruit kan je de informatie halen (cfr. lemma 1.4.23).).

2 Chapter 2: Trees and Distance

2.1 Basic Properties

- Intro: volledig.
- "Properties of Trees" Alles tot en met item 2.1.7, item 2.1.8 valt weg.
- "Distance in Trees and Graphs" Enkel Definitie 2.1.9. en Theorema 2.1.11.
- "Disjoint Spanning Trees" Laten vallen.
- "Exercises:"
  - 2.1.6 ($n(T)$ is het aantal vertices. Je weet hoeveel zijden een boom heeft als je $n(T)$ kent. Stel dit gelijk aan het aantal zijden dat je bekomt als de gemiddelde graad a is.),
  - 2.1.7 (Om te starten veronderstel je dat er slechts 1 component is. Deze component is dus samenhangend en bevat een spanning tree. Je weet hoeveel zijden een spanning tree heeft, en dat elke extra zijde een cykel creëert (Corollary 2.1.5.b). Hieruit kan je eenvoudig de stelling bewijzen voor het geval van 1 component. Veronderstel nu dat er $k \geq 1$ componenten zijn, als je stelt dat elke component n_i ($1 \leq i \leq k$) knopen bevat kan je ook gebruikmakend van de vorige informatie het gevraagde aantonen.),
  - 2.1.9 (Kijk hoe dit werkt voor een klein voorbeeld bijvoorbeeld $n = 3,4,5,6$, en dan kan je eenvoudig zien hoe het principe werkt.)a,
  - 2.1.10, (Let op: $d(u)$ is de graad van de vertex u, $d(u,v)$ is de kortste afstand tussen vertices u en v. Als $d(u,v) > 2$, dan weet je dat de doorsnede tussen $N(u)$ en $N(v)$ leeg is, daardoor is het aantal elementen in $N(u) \cup N(v)$ gelijk aan $d(u) + d(v)$. Schat dan het aantal elementen in deze unie af. Het afschatten doe je door te onderstellen dat er een pad van lengte $d(u,v)$ tussen u en v loopt, rekening houdend met de lengte van dit pad zijn er zeker een aantal knopen die niet in $N(u) \cup N(v)$ kunnen zitten. Dit levert de gelijkheid. Zoek het tegenvoorbeeld niet te ver, het is een standaard graaf.),
  - 2.1.13 (Als de diameter d is dan is er tussen twee knopen u en v een pad van minimum lengte d, gebruik dit pad en construeer expliciet die verzameling van onafhankelijke knopen.),
  - 2.1.21 (buiten het deel over de degree sequence) (Als er een decompositie is in k spanning trees, dan kan je exact het aantal zijden van de graaf bepalen. Als dan $k \geq n/2$ is dan kan je een ondergrens zoeken voor het aantal zijden (dus $e(G) \geq ...$, dat zou $(n-1)n/2$ moeten uitkomen. Dat aantal zijden kan enkel bij een heel bijzondere graaf (simple graph). Bij deze graaf zal de eigenschap $\Delta(G) = \delta(G) + 1$ tot een contradictie leiden.) ,
  - 2.1.33($\Rightarrow$ als je een zijde van de cykel wegdoet dan krijg je een geconnecteerde graaf zonder cykels, dit is een ... , en dus kan je het aantal zijden berekenen. $\Leftarrow$ Gebruik 2.1.5 c en dan 2.1.5 b.),
  - 2.1.35 (We tonen enkel de richting $\Rightarrow$ aan. Odd order betekent dat de componenten een oneven aantal knopen bevatten. Als je veronderstelt dat e de twee vertices u en v verbindt, knip dan e door, dan krijg je twee componenten. Tevens weet je dat u en v beiden een .... graad moeten hebben. Dan gebruik je de sum-degree formule, je weet dat de som van alle graden van de knopen deelbaar door twee moet zijn, rekening houdend met het feit dat in elke component alle knopen van oneven graad zijn, buiten u of v, kan je het gevraagde afleiden.),
  - 2.1.48 (Toon aan dat het minimum van $2n-4$ bereikt wordt in de graaf $K_{2,n-2}$. Verder toon je aan dat er minimum $2n-4$ nodig zijn voor een graaf waarbij y en x geen buren zijn en waarbij de knoop x van graad $n-2$ is. Je gaat nu gewoon het aantal zijden beginnen tellen, waarbij je rond x reeds $n-2$ zijden hebt, en je weet dat y met elke andere knoop verbonden is over een pad van lengte minstens 2 (wegens de diameter beperking).

2.2 Spanning Trees and Enumeration

We bekijken de Prüfer code en Cayley's formule.

- Alles tot en met 2.2.3. Van 2.2.3 laten we het bewijs achterwege. Van de rest van deze sectie moet niets gekend zijn.
- Bij wijze van oefening kan je misschien zelf de Prüfer code van een boom opstellen.

2.3 Optimization and Trees

- Bij deze items ook de extra theorie uit de les (DFS en BFS), kennen (cfr de toegevoegde bladen pag. 582-591).
- "Minimum Spanning Tree" Volledig kennen (2.3.4 mag je eens lezen maar is niet belangrijk).
- "Shortest Paths" Volledig kennen, maar het bewijs van 2.3.7(Dijkstra) laten we vallen (de applicatie in 2.3.9 kan je eens lezen).
- "Trees in Computer Science" Valt volledig weg.
- Extra: Van op de slides weten wat het traveling salesman problem is, en ook Prim's algoritme voor het berekenen van een minimum spanning tree is gekend.
- "Exercises:"
  - Van de extra bladen oefening 1 (Pas BFS toe op (i) en (ii). Pas DFS toe op (iii) en (iv)),
  - Van de extra bladen oefening 4 (Als je de DFS en de BFS hebt toegepast, dan kom je altijd een specifieke graaf uit, welke?)
  - uit het boek, 2.3.2,
  - 2.3.3 (Construeer een minimum weight spanning tree. Bij wijze van oefening kan je dit via zowel Kruskal's algoritme als Prim's algoritme proberen.),
  - 2.3.5 (Gebruik Dijkstra's algoritme. Let op, het is een georiënteerde graaf. Het is onnodig om dit voor elke i en j te doen, probeer $i = 1$ en zoek de kortste afstand tot elk ander punt.),
  - 2.3.12 (Zoek 1 voorbeeld van 4 knopen waarvoor het beschreven algoritme faalt.).

3 Chapter 3: Matchings and Factors

Vaak staan er in de tekst vermeldingen van dat het betreffende theorema (propositie,...) ook op een andere wijze kan bewezen worden. Steek er geen tijd in om dat te zoeken, de nieuwe manier is niet noodzakelijk eenvoudiger dan het oorspronkelijke bewijs. De bewijzen in dit deel worden moeilijker, als er problemen zijn met een specifiek bewijs, dan kan daar ook tijdens het responsiecollege aandacht aan besteed worden.

3.1 Matching and Covers

- "Intro", laat voorbeeld 3.1.3 en de commentaar over de Peterson graph vallen.
- "Maximum Matchings" volledig.
- "Hall's matching condition" volledig.
- "Min-Max theorems" volledig.
- "Independent sets and covers" volledig.
- "Dominating sets" laten we vallen.
- "Exercises:"
  - Een losse vraag: Is een boom altijd bipartiet?
  - 3.1.1 (Uit 3.1.16 weet je dat een maximum matching even groot is als een minimum vertex cover. Zoek dus een maximum matching en een minimum vertex cover. Als ze beiden evenveel elementen bevatten is je oplossing dus goed. Doe dit voor de linkse en middelste graaf. Toon tevens aan dat de gevonden maximum matching niet uniek is, door voor de linkse graaf twee verschillende matchings te zoeken. Tip: voor de linkse en middelste is de grootte van de cover en de matching gelijk aan 3.)
  - 3.1.5 (We gaan expliciet zo een onafhankelijke set creëren. Neem een vertex uit de graaf, en verwijder uit de graaf G deze vertex en al zijn buren. De vertex zelf steken we in de verzameling van onafhankelijke knopen S. Aangezien we in elke stap maximum $\Delta G + 1$ knopen verwijderen uit G, kunnen we berekenen hoeveel stappen we minstens nodig hebben om alle knopen uit G te verwijderen. In elke stap steken we immers een knoop bij in S dus...)
  - 3.1.6 (Gebruik 3.1.21 en het König-Egerváry theorema.)
  - 3.1.8 (Veronderstel dat je twee matchings M en M' hebt, toon dan aan dat ze gelijk zijn. Het bewijs gebruikt technieken vanuit het bewijs van Lemma 3.1.9. Construeer het symmetrisch verschil tussen M een M', noem dit F. Wat weet je over de graden van de knopen van F (deze kunnen enkel $\{0,1,2\}$ zijn). Let op je weet iets meer dan in Lemma 3.1.9, namelijk dat de matchings perfect zijn, dit legt een extra beperking op aan de graden van de knopen, je houdt slechts twee mogelijkheden over. Als je dan de componenten van F bekijkt en rekening houdt met de twee mogelijkheden, kom je omdat F een deel van de boom is ook nog tot een beperking. In feite hebben alle knopen dus dezelfde graad. Er blijft dus slechts 1 mogelijkheid over, als je dit weet kan je iets zeggen over het symmetrisch verschil en dus ook over M en M'.)
  - 3.1.9 (Toon eerst aan dat als M een maximal matching is dat de vertices 'saturated' by M een vertex cover vormen, dit is anders in contradictie met de maximaliteit. Deze vertices (van de vertex cover) hun aantal schat je dan af in termen van $\beta(G)$ en $\alpha'(G)$. Lees misschien nogmaals de tekst na definitie 3.1.14.)
  - 3.1.11 (Deze zou zo moeten lukken. Anders is hier de info. Probeer eerst eens te kijken naar het symmetrisch verschil van twee cykels met 1 gemeenschappelijke zijde. Uiteindelijk pas je Propositie 1.2.27 toe. Eerst ga je het aantal zijden rakend aan een knoop v in C en in C' tellen dit is steeds een .... getal. Tel dan het aantal zijden in het symmetrisch verschil tussen C en C'. Er verdwijnen dan misschien een aantal zijden, maar let op een gemeenschappelijk zijde komt zowel in cykel C als in cykel C' voor.)
  - 3.1.21 Toon aan dat voor welke zijde je ook kiest dat er steeds een matching is zodat X gesatureerd is. (Neem een zijde xy uit de graaf G. Toon dan aan dat Hall's conditie nog steeds geldig is voor de Graaf $G' = G - x - y$, dus de graaf G waarin we de vertices x en

y hebben verwijderd. Let op de strikte ongelijkheid in de opgave! Als Hall's conditie nog voldaan is kan je eenvoudig een matching voor G construeren die *xy* bevat en perfect is.)

3.2 Algorithms and Applications

- "Maximum Bipartite Matching" praktisch volledig. We laten Definitie 3.2.3 en Opmerking 3.2.4 vallen. Tevens moet het bewijs van Theorema 3.2.2 niet gekend zijn, maar wel het idee om een maximum bipartiete matching te construeren.
- "Weighted Bipartite Matching" praktisch volledig. We laten het bewijs van 3.2.11 vallen. Tevens valt 3.2.12, 3.2.13 en 3.2.14 ook weg.
- "Stable Matchings" valt weg.
- "Faster Bipartite Matching" valt weg.
- "Exercises:"
  - 3.2.1,
  - 3.2.2 (Het Hungarian algoritme is enkel toepasbaar op $K_{n,n}$, hier wordt dit gevraagd voor een willekeurige bipartiete graaf. Je moet deze dus inbedden in $K_{n,n}$. Ken nu aan de zijden in de oorspronkelijke graaf een gewicht toe, en ken ook aan de zijden niet in de oorspronkelijke graaf een gewicht toe. Als je het Hungarian algoritme dan toepast weet je dat er een perfect matching bestaat als en slechts als je iets kan zeggen over het gewicht van de maximum matching! Als het je niet onmiddellijk duidelijk is pas dan het Hungarian algoritme eens toe op een bipartitie graaf met 4 knopen, waarvan elke partiete set 2 knopen heeft.)
  - 3.2.5 a) (Het aantonen dat de transversal maximale lengte heeft doe je gewoon door tegelijkertijd het minimum weighted cover probleem op te lossen, zoals in Example 3.2.10.) Geef tevens ook het gewicht van je maximale 'transversal'.
  - 3.2.6 (Wees creatief!! Als je weet hoe het probleem te transformeren is het moeilijkste deel opgelost. Je kan bij wijze van spreken het algoritme nogmaals toepassen om te zien of je het echt begrepen hebt, maar dit heb je in voorgaande oefening reeds gedaan.)

3.3 Matchings in General Graphs

- Valt volledig weg.

4 Chapter 4: Connectivity and Paths

4.1 Cuts and Connectivity

- "Connectivity", we laten 4.1.3 (over de hypercube) weg. Vergeet het stukje tekst na het voorbeeld 4.1.3 (dit meldt dat $\kappa(G) \leq \delta(G)$) niet! De opmerking 4.1.6 over het bewijs van 4.1.5 valt ook weg. Tevens wordt er in het bewijs 4.1.5 gemeld dat Exercise 12 een stukje oplost, dit moet je niet nagaan. Enkel het even geval kennen is voldoende.
- "Edge Connectivity", zien we helemaal. Een tip bij het bewijs van Theorema 4.1.9: De ongelijkheid (in het geval alle knopen in S buren zijn van $\bar{S}$)

  $$|[S,\bar{S}]| = |S||\bar{S}| \geq n(G) - 1,$$

  toon je als volgt aan:

  $$\begin{aligned} |[S,\bar{S}]| &= |S||\bar{S}| \\ &= |S|(|\bar{S}| - 1) + |S| \\ &\geq (|\bar{S}| - 1) + |S| \text{ want } |S| \geq 1 \\ &= n(G) - 1 \end{aligned}$$

- "Blocks", vanaf 4.1.20 (ook de opmerking na 4.1.19 valt weg) valt alles uit het deeltje weg.
- "Exercises:"
  - 4.1.1 a) Gebruik de definitie.
    4.1.1 b) Gebruik de definitie.
    4.1.1 c) Gebruik de ongelijkheid tussen de connectivity $\kappa(G)$ en de edge-connectivity $\kappa'(G)$.
    4.1.1 d) Neem aan dat $k = 3$. Om dit te behandelen kan je gebruik maken van de volgende graaf bestaande (als samenstelling) uit 2 maal de complete graaf K_4, waarbij de twee complete grafen 1 gemeenschappelijke knoop hebben. Het totaal aantal knopen in de graaf is dus 7. (Het algemene geval voor k moet je niet aantonen, maar dan beschouw je twee maal de volledige graaf met $k+1$ knopen namelijk K_{k+1}, waarbij de twee grafen weer 1 gemeenschappelijke knoop hebben.)
  - 4.1.2 (Om aan te tonen dat de bewering vals is, moet je een zeer eenvoudig tegenvoorbeeld zoeken met 3 knopen. Denk eraan dat een cut-edge en een cut-vertex het aantal componenten verhogen. Zoek naar een eigenschap zodat het wel klopt, maar toon dit niet aan.)
  - 4.1.5 (Dit tonen we aan d.m.v. contradictie. Onderstel dat de graaf G' toch connectivity 1 heeft. Bijgevolg heeft de graaf G' een cut-vertex v. Bijgevolg heeft $G' - v$ twee componenten (we gaan nu aantonen dat dit onmogelijk is). Als $G' - v$ twee componenten heeft, wat weet je dan van $G - v$. Bijgevolg kan je een statement maken over op zijn minst 2 buren, zeg x en y, van v in de graaf G met betrekking tot hun onderlinge afstand. Dit zal een contradictie opleveren met de initiele opbouw van de graaf G' en het veronderstelde aantal componenten in $G' - v$.)
  - 4.1.8 (Zoek enkel $\kappa(G), \kappa'(G)$ en $\delta(G)$ voor de linkse graaf. Geef ook aan wat de corresponderende "separating set of vertices" and "disconnecting set of edges" zijn. De opmerking over het gebruik van Propositie 4.1.12 mag je achterwege laten.)
  - 4.1.14 (Teken eerst eens twee cykels met 1 gemeenschappelijke zijde, en kijk wat er gebeurt als je de gemeenschappelijke zijde verwijdert. Laat ons aantonen dat het verwijderen van 2 zijden nog steeds een geconnecteerde graaf overlaat en dus zal de graaf zeker 3-edge-connected zijn.
    Stel dat je twee verschillende zijden e en e' gaat verwijderen uit de graaf G.
    1) Is $G - e'$ nog steeds samenhangend?
    2) In de nieuwe graaf $G - e'$ is er zeker nog 1 van de oorspronkelijke cykels waartoe e behoorde aanwezig. Als er een zijde uit een cykel verwijderd wordt, wat betekent dat dan voor de samenhangendheid van de graaf? Dit levert dan het gevraagde. Kijk na dat de Petersen graaf zoals weergegeven op pg. 13 aan deze eis voldoet.

- 4.1.20 (We tonen aan dat als $\delta(G) \geq \left(\frac{n+k-2}{2}\right)$, de graaf altijd k-connected is.)
  Laat ons nog even herhalen dat k-connected betekent dat er minstens k knopen weggehaald moeten worden om de graaf te splitsen in twee of meer componenten. Dit is ook wat we zullen aantonen.
  Veronderstel dat je twee knopen hebt die geen buren van elkaar zijn, dan gaan we nu kijken hoeveel knopen er op zijn minst verwijderd moeten worden om de ene knoop onbereikbaar te maken van de andere knoop. Uit de condities op de graad $n/2-1 \leq \delta(G) \leq n-2$, kan je eenvoudig opmaken dat er zeker een knoop x en een knoop y bestaan zodat ze geen buren zijn van elkaar. Bereken nu zoals we reeds gedaan hebben in voorgaande oefeningen een ondergrens voor het gemeenschappelijk aantal buren, met behulp van de formule:

  $$|N(x) \cap N(y)| = |N(x)| + |N(y)| - |N(x) \cup N(y)|.$$

  Gebruik daarbij wat je weet over $\delta(G)$ en het feit dat x en y zelf geen buren zijn. Als je weet dat ze minstens ℓ gemeenschappelijke buren hebben, dan weet je dat er minstens ℓ knopen verwijderd moeten worden om x en y onbereikbaar te maken via de graaf. Dit betekent dat de graaf ... connected is.
  De tweede vraag betekent hetvolgende: toon aan voor een eenvoudig voorbeeld met $n/2-1 \leq \delta$ dat de grens $k = 2\delta + 2 - n$ bereikt wordt. (Tip: gebruik een graaf met 5 knopen en minimum graad 2.)

4.2 k-Connected Graphs

Dit stuk bevat interessante leerstof, maar we kunnen niet alles zien, dus we zien enkel twee definities and Menger's theorema, de belangrijkste stelling van dit deel.

- "2-Connected Graphs" valt volledig weg, buiten de definitie 4.2.1, die "internally disjoint paths" definieert.
- "Connectivity of digraphs" laten we volledig weg.
- "k-connected and k-edge connected graphs", alles valt weg buiten de definitie van een x,y-cut in Definitie 4.2.15 en een voorbeeld van een x,y-cut in 4.2.16. Lees eerst Theorema 4.2.17 (het bewijs niet kennen), en kijk dan nogmaals naar voorbeeld 4.2.16 om de gelijkheid zoals vermeld in het theorema te verifiëren. De twee laatste paragrafen van het voorbeeld, startend met "Consider also the pair w,z", laat je vallen.
- "Applications of Menger's Theorem" laten we volledig vallen.
- "Exercises:"4.2.1 We berekenen enkel $\kappa(u,v)$, dus de minimum grootte van een u,v-cut. Geef ook het corresponderende aantal disjuncte paden weer.

4.3 Network Flow Problems

- Introductie tekst na de titel volledig kennen.
- "Maximum Network Flow" praktisch volledig kennen. Het bewijs van de stelling 4.3.7 is niet verduidelijkend en mag achterwege gelaten worden.
- "Integral Flows" laten we vallen, maar weet dat als al de capaciteiten natuurlijke getallen zijn, dat er dan ook een maximale flow kan gevonden worden waarbij de flow doorheen elke edge ook natuurlijk is. Dit is in feite Corollary 4.3.12, de eerste regel.
- "Supplies and Demands" laten we vallen.
- "Exercises:"
  - 4.3.1 Volg niet de opgave maar bereken gewoon een maximum flow en een source/sink cut met minimum capacity. Vergeet niet dat als je een flow f hebt waarvan $val(f)$ gelijk is aan de capaciteit van een zekere source/sink cut, dat je een optimale oplossing hebt gevonden. Let op: je moet dit niet exact doen zoals beschreven in de theorie van de les, probeer gewoon op het zicht een f-augmenting pad te vinden. Als je denkt dat je de oplossing gevonden hebt, pas dan het Ford-Fulkerson algoritme toe en zie of je een source-sink cut

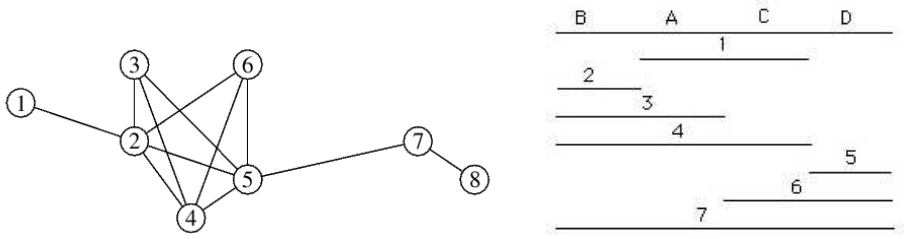
uitkomt, zo ja, dan heb je de oefening opgelost, zo nee, dan krijg je een nieuw f-augmenting pad en kan je verder gaan.

- 4.3.2 Doe deze oefening enkel om jezelf te testen, deze is niet verplicht! De opgave is dezelfde als in 4.3.1.
- 4.3.11 Het "augmenting path algorithm" voor bipartiete grafen kunnen we ook bekijken als een bijzondere versie van het "Ford-Fulkerson labeling algorithm". In feite moet je van de X,Y-bigraaf een georiënteerde graaf maken. Laat ons even aantonen hoe we starten. Verbind alle punten uit X met de source s en de oriëntatie loopt van s naar $x \in X$. Verbind alle punten uit Y met de sink t en de oriëntatie is van $y \in Y$ naar t. Als laatste beschouw je dan de oorspronkelijke zijden uit de X,Y-bigraaf en je voegt een oriëntatie toe van X naar Y. Veronderstel dat de capaciteit van alle zijden gelijk is aan 1.
  Traditioneel gaan we vertrekkende van een gegeven matching in de X,Y-bigraph het augmenting pad algoritme toe passen om dan te komen tot een grotere matching.
  Nu moet je zelf, veronderstellende dat je een matching hebt, een flow toekennen aan je nieuw geconstrueerde graaf (deze flow is natuurlijk gekoppeld aan de matching), dan het Ford-Fulkerson labeling algoritme toepassen op dit netwerk om zo te komen tot een grotere flow. Tevens moet dan deze grotere flow (via het f-augmenting pad algoritme) ook aanleiding geven tot een grotere matching.
  Zoek dus nu zelf wat de initiële flow gekoppeld aan de matching en ook gekoppeld aan je nieuwe geconstrueerde georiënteerde graaf is, opdat het Ford-Fulkerson algoritme aanleiding zal geven tot een grotere matching in het originele probleem.

5 Chapter 5: Coloring of Graphs

5.1 Vertex Coloring and Upper Bounds

- "Definitions and Examples" voorbeeld 5.1.3 is wel te kennen, maar niet het stukje van de Peterson graaf; 5.1.9, 5.1.10 en 5.1.11 vallen weg. De definitie van "proper subgraph" vind je op pag. 472, Definitie A.3.
- "Upper bounds" volledig kennen.
- "Brooks' Theorem" volledig kennen. Om dit theorema te bewijzen moet je wel weten wat een "block" is (zie daarvoor op de slides van vorige les, enkel de eigenschappen van blokken die noodzakelijk zijn moet je weten, je moet niets over blokken kunnen bewijzen. Kortweg is een "block" is een maximaal samenhangend deel zonder cut-vertex.) Item 5.1.23 moet je maar eens nalezen.
- "Exercises:"
  - Eerst een losse vraag. Wat is het "chromatic"number voor een boom? Bewijs dit tevens. (Tip: kijk naar de losse vraag van vorige week.)
  - 5.1.1 (Het clique number kan je vinden door de grootste volledige graaf te zoeken als deel van de gegeven graaf. Tevens is het mogelijk om de graaf iets te hertekenen, zodat je hem makkelijker kan analyseren. Om $\chi(G)$ te vinden, kan je best gebruik maken van de ongelijkheid $\chi(G) \geq (n(G)/\alpha(G))$, dan weet je dadelijk dat je kleuring goed is. Om na te gaan dat de graaf color-critical is moet je kijken of een graaf met 1 zijde minder, minder kleuren nodig heeft voor een "proper" coloring. Gebruikmakend van symmetrie moet je maar een beperkt aantal zijden (zie remark 5.2.12) behandelen voor de color-critical property.)
  - Theorema 5.1.21 geeft in feite ook een algoritme weer om een graaf te kleuren. Pas dit algoritme toe op de graaf uit 5.1.1, denk eraan dat dit algoritme niet noodzakelijk de beste kleuring geeft. Maar, theorema 5.1.21 zegt dat het ook mogelijk is om de oriëntatie zodanig te kiezen dat de kleuring wel optimaal is, geef ook deze oriëntatie weer, vertrek van je eigen oplossing en zoek dan de oriëntatie! Hiervoor pas je in feite het laatste paragraafje van Theorema 5.1.21 toe, let op je moet niet dezelfde kleuring uitkomen enkel een kleuring met hetzelfde aantal kleuren.
  - 5.1.7, (Zonder uitleg, zou moeten lukken. Neem een graaf met 4 knopen.)
  - Losse vraag: als je met het greedy algoritme zo weinig mogelijk kleuren wil gebruiken, met welke knoop kan je dan best beginnen: degene met de hoogste of de laagste graad? En als je zoveel mogelijk kleuren wil gebruiken? (Let op dit is geen algemeen geldende waarheid, maar in vele gevallen kan je van dit idee gebruik maken om de opgave op te lossen, zo ook bij de volgende oefening.)
  - 5.1.34 (Construeer een iteratief procede, waarbij je gebruikmaakt van de boom in stap $k-1$ om je boom in stap k te verkrijgen. Je boom moet maximum graad gelijk aan k hebben en door een bijzondere volgorde van de knopen te kiezen zal het greedy algoritme $k+1$ kleuren gebruiken.
    Vertrek van een pad van lengte 1 en kies een willekeurige volgorde om het greedy algoritme toe te passen, je hebt dus nu enkel twee knopen en twee kleuren, dit is boom T_1. Bed nu deze boom in in een grotere boom door aan "elke" knoop een nieuw blad te hangen. Kies nu een volgorde zodat het greedy algoritme 3 kleuren zal gebruiken (houd bij de nieuwe keuze van de volgorde ook rekening met de volgorde die je gebruikte in T_1). Tevens heeft deze boom T_2 maximum graad 2.
    Pas dan je gevonden werkwijze, d.w.z. toevoegen van bladen aan alle knopen en de keuze van de volgorde, nogmaals toe om T_3 te construeren. Denk eraan, T_2 is ingebed in T_3 gebruik dus ook de volgorde die je kende van T_2.
    Aantonen dat dit procede blijft werken is zeer eenvoudig door constructie, dat moet je zelf niet doen, het word nu kort verklaard. Je boom T_k zit ingebed in T_{k+1} en door de gekozen volgorde (die heb je zelf moeten bedenken) zal de kleur van elke knoop in het ingebedde



Figuur 1. Interval graaf en een interval graaf representatie.

deel T_k van T_{k+1} met 1 moeten stijgen bij toepassing van het greedy algoritme. Bijgevolg zal het aantal kleuren als je van T_k naar T_{k+1} gaat met 1 stijgen.

- 5.1.38 Dit is een zeer interessant bewijs en een heel belangrijke oefening waarin verschillende technieken toegepast moeten worden, gebruikmakend van de theorie in hoofdstuk 3. Het bewijs delen we op in verschillende stappen. Stel G de graaf en $\bar{G}$ zijn complement dat tevens bipartiet is.
  1. Je weet dat als S een onafhankelijke set van knopen is in de ene graaf, dat dan S een clique vormt in het complement.
  2. Stel eerst dat je aan $\bar{G}$ (Let op dit is het complement!) een geïsoleerde "vertex" toevoegt, wat is de impact op $\omega(G)$ en op $\chi(G)$. Bijgevolg kunnen we dus onderstellen dat er geen geïsoleerde knopen zijn in $\bar{G}$, want ze verstoren toch de gelijkheid niet, ze wijzigen de waardes van zowel $\omega(G)$ en $\chi(G)$ in dezelfde mate.
  3. Wat is dan het verband tussen $\alpha(\bar{G})$ en $\omega(G)$ (gebruik puntje 1.)
  4. Voor het volgende deel moet je goed nadenken.
     $\chi(G)$=k, d.w.z. dat $V(G)$ de unie is van k.... sets van knopen (zie 5.1.2).
     Bijgevolg is $V(\bar{G})$ de unie van (beter geformuleerd is dit eigenlijk covered door) k ..... (gebruik puntje 1.)
     Maar de structuur van deze k ..... is zeer bijzonder want $\bar{G}$ is een bipartiete graaf, namelijk..... (je kan pogen om een clique te construeren in een bipartiete graaf, je zal merken dat een clique nooit meer dan twee knopen kan bevatten.) Bijgevolg vormen deze k ..... een ....-cover.
     Deze .....-cover heeft dus dezelfde waarde als $\chi(G)$. Door het minimaal karakter van $\chi(G)$ zijn dus $\beta'(G)$ en $\chi(G)$ ...
  5. Pas nu het König-theorema toe (3.1.24) en gebruik alle voorgaande relaties. Dat levert de oplossing.
- Stel de interval graaf representatie op van de linkse graaf in Figuur 1 en stel de graaf op van de interval representatie in Figuur 1.
- 5.1.41 Toon dit aan per inductie op $n(G)$. Voor $n(G) = 1$ is dit triviaal. Onderstel nu dat de eigenschap geldt voor $n-1$ en je onderzoekt nu $n(G) = n$. Beschouw dan de graaf $G' = G - v$ met v een willekeurige knoop uit G. Je weet nu per inductie dat $\chi(G') + \chi(\bar{G}') \leq n(G') + 1 = n - 1 + 1 = n$.
  Nu onderscheid je twee gevallen.
  1. Als je een knoop toevoegt aan een graaf kan het "chromatic" number maximum stijgen met .... . Bijgevolg als je van G' naar G gaat stijgt het "chromatic" number met maximum ... . De te bewijzen afschatting blijft dus geldig tenzij zowel $\chi(G')$ als $\chi(\bar{G}')$ met ... stijgen. Dit is geval 2.
  2. Veronderstel dat beiden $\chi(G')$ en $\chi(\bar{G}')$ met ... stijgen. Dan gaan we $\chi(G') + \chi(\bar{G}')$ iets strenger afschatten.
     Wat is het aantal buren van v in de graaf G in termen van $\chi(G')$?
     Wat is het aantal buren van v in de graaf $\bar{G}$ in termen van $\chi(\bar{G}')$?
     Tevens weet je dat de som van het aantal buren van v in G en $\bar{G}$ kleiner is dan .... d.w.z. $N_G(v) + N_{\bar{G}}(v) \leq ....$
     Rekening houdend met wat je weet over het aantal buren van v in G en $\bar{G}$ kom je dan

tot een iets strengere afschatting dan voorheen:

$$\chi(G') + \chi(\bar{G}') \leq ...,$$

bijgevolg is dus, rekening houdend met de stijging van het chromatic number als initieel ondersteld,

$$\chi(G) + \chi(\bar{G}) \leq ....$$

wat het gevraagde bewijst.

5.2 Structure of k-chromatic Graphs

- De introductie lezen.
- "Graphs with large chromatic number", punt 5.2.4 valt weg.
- "Extremal problems and Turan's theorem", dit deel behandelt de eigenschappen van een zeer specifieke klasse van grafen. We laten dit stuk achterwege, buiten de allereerste propositie over het aantal zijden in een k-chromatische graaf. Dus enkel 5.2.5 dient gekend te zijn, alsook het eenvoudige bewijs. Na 5.2.5 valt alles weg uit dit deeltje.
- "Color-critical graphs", alles tot en met 5.2.16. Dat wil zeggen dat 5.2.17 en de rest wegvalt.
- "Forced subdivisions" laten we volledig achterwege.
- "Exercises:"
  - 5.2.1 (Stel dat je twee knopen x en y hebt die niet "adjacent" zijn (dit kan enkel in een niet volledige graaf) en $G' = G - x - y$. Stel $\chi(G') = k$, dan kan je $\chi(G)$ afschatten in functie van $k = \chi(G')$, wat een contradictie zal leveren met de initiële eis namelijk $\chi(G) = \chi(G') + 2$.)
  - 5.2.7 (Veronderstel dat voor een zekere kleur i er geen enkele knoop van die kleur is die aan alle overige $k-1$ kleuren grenst. Toon dan aan dat je de kleuren kan herschikken zodat je een "proper" coloring van $k-1$ kleuren krijgt, dat is natuurlijk in contradictie met de initiële onderstellingen dat je k kleuren nodig hebt.)
  - 5.2.8 (Stel dat G' een "k-color critical subgraph" van G is. De graaf G' heeft op zijn minst ... knopen, en je weet via propositie 5.1.18 dat $\delta(G') \geq ....$ Als voor elke knoop in G' zijn graad groter is dan ..., d.w.z. $deg_{G'}(v_k) \geq ...$ dan zal dat voor dezelfde knoop in G $d_k = deg_G(v_k) \geq ...$ geven.
    Bijgevolg krijg je (we duiden met de index k de knopen aan die ook in G' zitten en met de index i alle knopen in G):

    $$\chi(G) = \chi(G') = k = \min_k\{d_k + 1, k\} \leq \max_i ....$$

    De laatste ongelijkheid bewijst het gevraagde.)

5.3 Enumerative Aspects

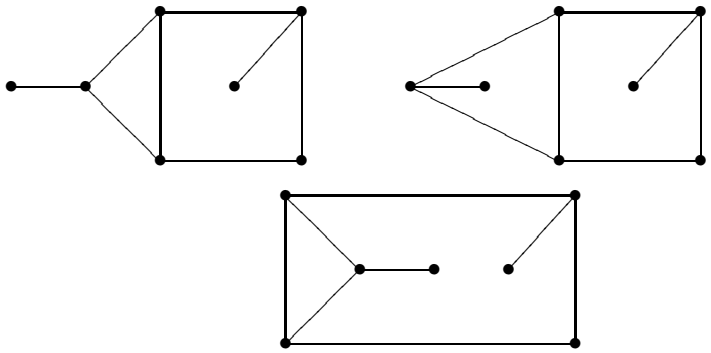
Dit deel van het hoofdstuk behandelt het tellen van het aantal mogelijke "proper colorings" van een graaf met k kleuren.

- "Counting proper colorings" we bekijken hier enkel 5.3.1, 5.3.2 en 5.3.3, de rest valt volledig weg.
- "Chordal graphs" valt volledig weg.
- "A hint of perfect graphs" valt weg.
- "Counting acyclic orientations" valt weg.

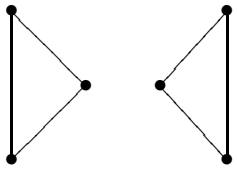
6 Chapter 6: Planar Graphs

6.1 Embeddings and Euler's formula

- "Drawings in the plane", in het bewijs van propositie 6.1.2 wordt gebruik gemaakt van een "chord" in een cykel, dit kan je vertalen als een koorde. De definitie vind je in Definition 5.3.15 (deze definitie wordt ook weergegeven op de slides). Na definitie 6.1.3 (deze definities zijn niet echt belangrijk, ze zijn enkel toegevoegd om niet te zondigen tegen de topologie en om duidelijk wiskundig te formuleren wat nu een tekening van een graaf is) staan er enkele opmerkingen over "drawings", dit zijn gewoon aannames over hoe grafen in het boek zullen getekend worden.
  Het bewijs van 6.1.6 laat je vallen.
- "Dual Graphs", bij 6.1.8 mag je het tweede paragraafje over de cube en $K_{2,2,2}$ weglaten. Om het bewijs van 6.1.14 beter te begrijpen kan je misschien nog eens kijken naar het bewijs van Propositie 4.1.15.
  We laten hier ook 6.1.17, 6.1.18, 6.1.19 en 6.1.20 vallen.
- "Euler's formula", dit is een belangrijk stuk, we laten dan ook enkel 6.1.25, 6.1.26 en 6.1.28 vallen. Opmerking 6.1.27 is belangrijk omdat ze aangeeft dat planaire embedding identiek is aan embedding op de bol.
- "Exercices:"
  - 6.1.1 Deze oefening is eenvoudig, zou moeten gaan.
  - 6.1.2 Met bewijs bedoelt men enkel dat je een planaire tekening van de graaf in kwestie moet maken.
  - 6.1.4 Teken enkel de duale van de volgende 3 tekeningen van dezelfde graaf.



  - 6.1.10 Gebruik Propositie 6.1.23. Let goed op welke van de twee afschattingen je kan gebruiken, rekening houdend met het feit dat de graaf bipartiet is. Als je dan het maximum aantal zijden neemt, dan weet je ook de maximale som van de graden (door de sum-degree formule te gebruiken). Als je de maximale som van de graden hebt, kan je de gemiddelde graad berekenen. Daaruit kan je het gevraagde concluderen.
  - 6.1.18 Het bewijs is in feite niet zo interessant, we gaan eerst een graaf maken en daarna tonen we enkel a) aan. Teken de duale van onderstaande graaf G dus G^* (is deze graaf geconnecteerd?). Teken G^{**}, is $G^{**} = G$? Teken dan G^{***}, wat kan je nu zeggen en waarom denk je dat dit zo is?



    Om de eigenschap a) aan te tonen, moet je in feite bewijzen dat er voor elke twee punten u en v in G^* een u,v-pad bestaat in G^*. Aangezien u en v faces voorstellen in G moet je

dus verscheidene edges kruisen om vanuit face u, face v te bereiken. Wat betekent dit dan in termen van de graaf G^*.
Onthoud ook de belangrijke eigenschap dat $G = G^{**}$ enkel en alleen als G geconnecteerd is.

– 6.1.20 We zullen twee richtingen moeten aantonen.
  - $\Rightarrow$ We tonen aan dat als een graaf bipartiet is dat dan de face-lengte even is. De randen van een face vormen een gesloten "walk". Stel per contradictie dat dat een oneven wandeling is, dan weet je dat dat een oneven ... bevat (tip 1.2.15). Maar de graaf is bipartiet en kan dus geen oneven ... bevatten. Dus onze onderstelling was vals en er zijn enkel even gesloten wandelingen die de face lengte bepalen.
  - $\Leftarrow$ We tonen dit aan per inductie op het aantal faces.
    · Stel 1 face. Als er maar 1 face is, wat kan je dan zeggen over de structuur van de graaf: Hij bevat cykels of hij is acyclisch? Een .... graaf is van een bijzondere vorm (cfr. Definitie 2.1.1), namelijk ... . Voor dit soort graaf weet je dat hij bipartiet is (zie vorige oefenzittingen).
    · Stel dat je > 1 faces hebt. Het Jordan Curve theorema, impliceert dat er een cykel is. Minstens 1 zijde e van de cykel behoort tot twee faces, stel F_1 en F_2. Beschouw nu de graaf $G-e$, Faces F_1 en F_2 worden verenigd in 1 nieuwe face, deze face heeft even/oneven lengte. Wat weet je dus over de lengte van de faces in $G-e$. Dus $G-e$ is sowieso ... (door inductie).
    Nu moeten we aantonen dat ook G nog ... is. F_1 heeft even randlengte. Als e verwijderd wordt uit F_1, wat weet je dan over de eindpunten van de zijde e m.b.t. hun positie in de ... graaf $G-e$. Wat gebeurt er dus met de ..... structuur van de graaf $G-e$? Inderdaad, dit bewijst het gevraagde.

– 6.1.21 Let goed op bij de formulering van de vraag. Hier wordt de constructie van T^* nog eens toegelicht.
Stel dat T een spanning tree is van de graaf G. Noteer met G^* de duale van de graaf G. Dan noteren we met T^* de subgraaf van G^* met als zijden de zijden duaal aan de zijden van de graaf $G-T$, dus duaal aan de zijden die overblijven als de zijden van de spanning tree T uit graaf G verwijderd zijn.
  - De formulering van de vraag bevat een $\Leftrightarrow$ pijl. Door het feit dat $G^{**} = G$, moeten we dus enkel 1 richting aantonen.
  - Stel dat G, n knopen, e zijden en f "faces" heeft. Hoeveel knopen heeft dan G^*? De graaf T^* is duidelijk een subgraaf van de graaf G^*. Hoeveel knopen heeft T^*? Bijgevolg is dus T^* alzeker een spanning subgraaf. Je weet echter niet of T^* geconnecteerd en acyclisch is, dit komt nog.
  - Dit is een losstaande vraag, die je 2 maal nodig hebt in hetvolgende puntje.
    Stel dat H een connected spanning subgraaf is van de graaf G, kan de graaf $G-H$ dan een "bond" (minimale edge cut) bevatten voor de volledige graaf G?
    Stel dat H een subgraaf is van G die geen "bond" bevat van de graaf G. Is dan de graaf $G-H$ connected en spanning voor G?
  - Gebruikmakend van Theorema 2.1.4 moeten we in feite slechts 2 van de volgende 3 beweringen aan tonen. Je kan proberen ze alledrie aan te tonen.
    · *T^* is connected.*
    Gebruik het vorige item om dit aan te tonen. Vertrek van het feit dat T acyclisch is en gebruik dan 6.1.14, pas dan het vorige item toe.
    · *T^* heeft $f-1$ zijden.*
    Je weet dat het totaal aantal zijden gelijk is aan $n+f-2$. Als de tree T een aantal zijden heeft dan kan je eenvoudig berekenen hoeveel zijden er nog overblijven in $G-T$, dit is per constructie gelijk aan het aantal zijden van T^*.
    · *T^* is acyclisch.*
    Analoog aan het bewijs van T^* is connected.

6.2 Characterization of planar graphs

Jammer genoeg moeten we wegens besparingen op de leerstof ook dit achterwege laten. Het deeltje bespreekt onder andere verbanden tussen “connectedness” en “planariteit”.

6.3 Parameters of Planarity

- “Coloring of planar graphs”, dit hoofdstuk geeft een overzicht over mogelijkheden en hun beperkingen om het vier-kleuren theorema aan te tonen. Wij behandelen enkel het vijf-kleuren theorema, dus Stelling 6.3.1 met bewijs, en voorts nog Stelling 6.3.6 zonder bewijs.
- “Crossing number”, dit deeltje bevat enkele interessante definities en begrippen. We bekijken echter enkel definitie 6.3.11 die het crossing number definieert.
- “Surfaces of higher Genus”, laten we vallen.
- “Exercices”
  - 6.3.2 Per inductie op het aantal knopen. Probeer deze eerst zelf vooraleer de tips te bekijken, als je de leerstof beheerst is het een eenvoudige oefening.
    Tips: een deelgraaf van k-degenerate graaf is ook k-degenerate. En als je een knoop met k buren hebt, dan moet je enkel 1 kleur toevoegen of hergebruiken om $(k+1)$-kleurbaar te bereiken.

7 Chapter 7: Edges and Cycles

7.1 Line Graphs and Edge-coloring

- "Introduction", na de definitie staan enkele opmerkingen omtrent wat er in hoofdstuk 4 en 5 behandeld werd, leest dit gewoon en besteed er niet te veel aandacht aan.
- "Edge-colorings", voorbeeld 7.1.2 begrijp het voorbeeld, maar het stuk over de oplossing met de congruentie-klassen is niet interessant. Na definitie 7.1.3 staat er een opmerking over Vizings theorem en het feit dat "Line Graphs are almost perfect", laat deze opmerkingen gewoon achterwege. Bij het bewijs van Theorema 7.1.7 wordt verwezen naar 1-factor, kijk gewoon naar de corollary 3.1.13 (een 1-factor wordt op 3.3.1 en 3.3.2 gedefinieerd, maar is niet te kennen) en vergeet wat een 1-factor betekent. Het bewijs is niet al te duidelijk, kijk misschien best naar de uitleg op de transparanten. Verder laten we van dit deel de items 7.1.8 en 7.1.9 vallen. Het bewijs van theorema 7.1.10 bevat in feite gewoon een algoritme om een gepaste kleuring te construeren, (het bewijs is interessant en kan best eens gelezen worden, het zal echter niet gevraagd worden op een examen). Verder laten we alles vallen, nl. 7.1.11, 7.1.12, 7.1.13 en 7.1.14. De bovengrens aangeduid in 7.1.13 dient wel gekend te zijn, zonder bewijs.
- "Characterization of line graphs" laten we volledig achterwege.
- Exercices:
  - 7.1.1 Enkel voor de linkse graaf. Bewijs tevens dat je kleuring optimaal is, door na te gaan dat een bepaalde ondergrens van een afschatting bereikt wordt.
  - 7.1.4 Kijk naar een analoge afschatting voor het kleuren van knopen. (Tip: Een edge-color klasse heeft maximaal ... elementen. Bijgevolg moet $\chi'(G) \geq \dots$.)
  - 7.1.9 Dit is een eenvoudig bewijs. We tonen enkel de richting $\Rightarrow$ aan, de andere richting is triviaal.
    Noteer de verzameling zijden corresponderend met de clique met S. Neem een willekeurige zijde e uit S.
    i) Ofwel raken alle andere zijden e in één eindpunt. Dit bewijst het eerste deel.
    ii) Ofwel zijn er twee zijden... Toon nu verder aan dat je dan een "triangle", dus een driehoek bekomt. Houd er rekening mee dat je zijden afkomstig zijn van een clique.
  - 7.1.11 a) Hier gaan we het aantal zijden in een lijngraaf tellen.
    Onderstel $e = uv$ als een zijde uit de graaf G.
    De graad van de knoop $e = uv$ in $L(G)$ is dan $d_G(u) + d_G(v) - 2$. (Ga dit na.)
    Om de "sum-degree" formule toe te passen moet je dus sommeren over alle zijden uv in G, dit correspondeert met alle knopen uit $L(G)$. Je hebt dan

$$2e(L(G)) = \sum_{uv \in E(G)} d_G(u) + d_G(v) - 2.$$

    Herschrijf eerst die som door de 2 af te zonderen en $e(G)$ te gebruiken. Vorm de som over de zijden dan om tot een som over de knopen. Denk hierbij aan het feit dat uit een punt u van G, $d_G(u)$ zijden vertrekken. Let ook op, want je telt sommige dingen dubbel! Uiteindelijk moet je dan dit bekomen:

$$2e(L(G)) = \sum_{u \in V(G)} d_G(u)^2 - 2e(G).$$

    Als je dan gebruik maakt van $2e(G) = \sum_{u \in V(G)} d_G(u)$, kan je de formule herschrijven. Als je dan nog weet dat

$$\binom{x}{y} = \frac{x!}{y!(x-y)!},$$

    kan je de gelijkheid bekomen.
  - 7.1.11 b) We gaan niet aantonen dat de beweerde stelling correct is. Teken echter de lijngraaf van een willekeurige cykel. Wat bekom je?
  - Een losse vraag, die nodig is voor de volgende opgaven. Vul samenhangend of onsamenhangend in.

- Als G samenhangend is dan zal $L(G)$ ... zijn.
- Als $L(G)$ onsamenhangend is dan zal G ... zijn.

– 7.1.10 Neem een zijde $e = uv \in E(G)$. Dan weet je uit een vorige oefening wat de graad van e in $L(G)$ zal zijn in termen van $d_G(u)$ en $d_G(v)$. Je hebt dus

$$d_{L(G)}(e) = ...$$

Stel nu dat je een zijde $e' = u'v \in E(G)$ neemt, dus een zijde ook grenzend aan het eindpunt v. Schrijf hier dan de graad van op in $L(G)$:

$$d_{L(G)}(e') = ...$$

Als gevolg van het feit dat $d_{L(G)}(e') = d_{L(G)}(e)$, want $L(G)$ is regulier, weet je een verband tussen $d_G(u)$ en $d_G(u')$. In feite was $u'v$ een willekeurige zijde grenzend aan v, deze eigenschap geldt dus voor alle buren van v.
Nu kan je twee conclusies trekken:

i) Ofwel hebben alle knopen uit G ... graad en dus is G regulier.
ii) Stel dat G niet regulier is, dan weet je dat als je een gesloten wandeling neemt doorheen G dat de knopen achtereenvolgens graden $d_G(v), d_G(u), d_G(v), d_G(u), \ldots$ moeten hebben. Bijgevolg is een gesloten wandeling altijd even/oneven van lengte? Daardoor is de graaf dus ... zoals gevraagd. Tevens zal door de volgorde van de graden in de gesloten wandeling, ook nog iets gezegd kunnen worden over de graden van knopen in de bipartiete sets.

– 7.1.14 Veronderstel dat G k-edge connected is.

i) Toon aan dat $L(G)$ k-connected is.
Stel dat je een 'separating' set $S \subset V(L(G))$ hebt, met $|S| = l$. Gebruikmakend van wat in de vorige losse vraag is aangetoond, wat weet je dan van de zijden in G corresponderende met de knopen uit S.
Bijgevolg: als G k-edge connected is dan impliceert dit dat $L(G)$... connected is.
ii) Dit deel doen we niet ($2k-2$-edge-connected).

– 7.1.26 Je weet dat altijd $\chi'(G) \geq \Delta(G)$, onderstel nu $\chi'(G) = \Delta(G)$, dan gaan we tot een contradictie komen.

- Aangezien G een reguliere graaf is en $\chi'(G) = \Delta(G)$, weet je dat elke kleuren klasse een ... matching definieert. (Elke kleur komt voor aan elke knoop.)
- Als gevolg hiervan zal het aantal elementen in G even/oneven zijn?
- Bijgevolg zal $G - x$ een oneven/even aantal elementen bevatten?
- Daardoor zal zeker 1 van de componenten van $G - x$, zeg component H oneven/even zijn?
- Stel nu y als een buur van x, niet in H, dan bestaat er een .... matching (dat is dus een kleurenklasse) in de originele graaf G, die die zijde xy bevat.
- Deze kleurenklasse (= ... matching), impliceert dus ook een ... matching in de component H. Maar dit is in contradictie met even/oneven aantal elementen in H.
- De contradictie impliceert dat de initiële onderstelling fout was, bijgevolg zal $\chi'(G) > \Delta(G)$.

7.2 Hamiltonian Cycles

– "Introduction" volledig te kennen.
– "Necessary conditions", alles behalve het bovenste stuk op pg. 288. Namelijk de opmerking over de Peterson graaf en item 7.2.6.
– "Sufficient conditions", item 7.2.12 laten we vallen, weet gewoon dat de sluiting goed gedefinieerd is. Theorema 7.2.13 behandelen we zonder bewijs! Verder laten we dan 7.2.14, 7.2.15, 7.2.16, 7.2.17 en 7.2.18 achterwege. De laatste stelling met bewijs is deze van 7.2.19.
– "Exercices"
  – 7.2.1, is een eenvoudige oefening.

- 7.2.6, in feite is dit een zeer triviale opgave. We gaan dit niet bewijzen, enkel via een voorbeeld nagaan of dit klopt. We gaan hierbij de perfect matching expliciet te construeren. Teken een willekeurige bipartiete graaf G met een Hamiltonian cykel H in de graaf. We gaan de perfect matching construeren, enkel door gebruik te maken van de Hamiltonian cykel. Teken dus je graaf opnieuw en laat nu alle zijden die niet in de Hamiltonian cykel zitten vallen. Je krijgt nu een bipartiete graaf met 1 spanning cykel. In feite teken je nu enkel H.
  Markeer nu 2 knopen x en y van de Hamiltonian cykel, met x en y in een verschillende partiete set.
  Teken nu de graaf $H-x-y$, je hebt nu een graaf bestaande uit 2 paden (je kan ook maar 1 pad hebben als x en y buren waren in H).
  Teken nu je perfect matching in deze graaf $H-x-y$. Je merkt dan dat deze perfect matching ook een perfect matching vormt voor de originele graaf $G-x-y$. Deze constructie bewijst het gevraagde.
  Deel 2 van het schaakbord laat je achterwege.
- 7.2.12, probeer een Hamiltonian cykel te vinden.
- 7.2.31, gebruik om dit aan te tonen Theorema 7.2.19.
  Stel dat S een deel knopen is zodanig dat $G-S$ onsamenhangend is, dan moet deze set S zeker enkele bijzondere knopen bevatten, welke? Je weet dus dat $|S|...\alpha(G)$ is. Bijgevolg kan je een uitspraak doen over het verband tussen $\kappa(G)$ en $\alpha(G)$. Theorema 7.2.19 levert je dan de oplossing.

Extra Topic: Google's PageRank

Ik heb een korte tekst afgeprint die introduceert hoe de Google matrix opgesteld wordt en hoe de power methode gebruikt wordt om hiervan de eigenvector horend bij de grootste eigenwaarde te berekenen. Jammer genoeg gebruikt men in dit artikel een iets afwijkende notatie van deze op de slides (corresponderend met de cursus). In de slides wordt aangegeven hoe men de linkse eigenvector berekent, op de afgedrukte paginas wordt de rechtse eigenvector bekomen, dit is gewoon een andere interpretatie, alle resultaten blijven volledig gelden. In feite komen de rijen van de matrix $D^{-1}A$ op de slides overeen met de kolommen van de matrix H in de tekst.

- "Introductie" is goed te begrijpen.
- "How to tell who's important" staat in de slides. Let op met de opmerking hierboven.
- "Computing I", "A probabilistic interpretation of H", "How does the power method work", "When things go wrong", deze kan je lezen als je geinteresseerd bent, maar dit wordt niet behandeld in de slides.
- "A final modification", dit komt terug naar de formule gebruikt in de slides. In de tekst werkt men met de matrix $1/n * \mathbf{1}$ die op elke plaats in de matrix het element $1/n$ bevat, dat geeft dus elke site een even belangrijke positie. In onze slides gebruiken we de werkelijke situatie, ez^T, waarbij z aan de belangrijkere sites een groter gewicht toekent.

Wat is nu te kennen van dit deeltje.

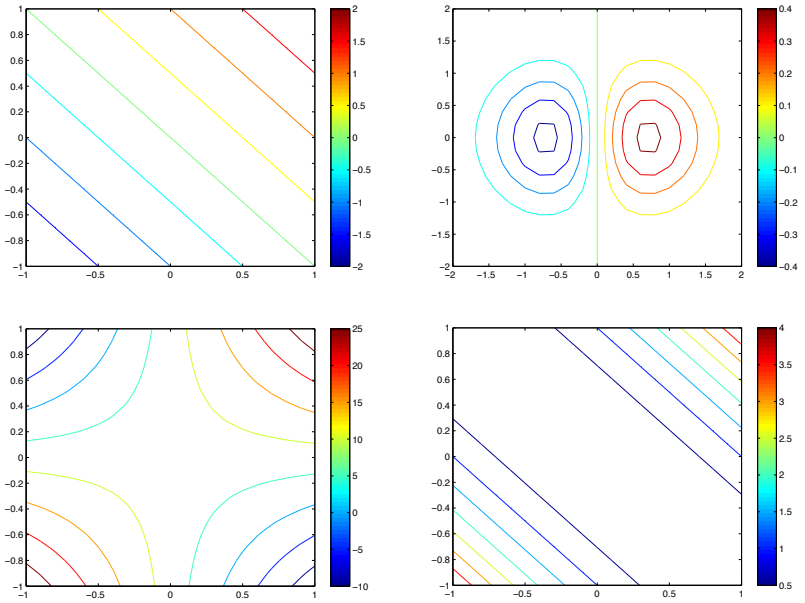
- De tekst is gewoon een leestekst, die moet je niet echt leren, als je geinteresseerd ben mag je hem lezen. Dit wordt niet ondervraagd op het examen. Beschouw dit als wat achtergrond informatie.
- Weten wat de interpretatie is van de google equation. D.w.z. de formule kunnen ontrafelen en alle componenten verklaren.
- Weten dat de eigenvector van deze vergelijking je een ranking geeft van het relatieve belang van de paginas.
- Beknopt het effect van gewijzigde inlinks en outlinks weergeven.

8 Chapter 8: Optimalisatie in meerdere veranderlijken

- "Definities en eigenschappen", alles te kennen, alleen de eigenschappen van Stelling 1.2 dienen niet allemaal gekend te zijn. Onthoud enkel de eigenschappen die noodzakelijk zijn voor de rest van de cursus, d.w.z. punt 1,2 en 3.
- "Nodige en voldoende voorwaarden voor een minimum", alles te kennen.
- "Conditie van een optimalisatie probleem", alles te kennen. Tevens zijn er enkele bladzijden uit een vroeger hoofdstuk (zie pag. 31), dit behoort niet tot de leerstof, maar is noodzakelijk om de rest te begrijpen. De formules van de relatieve en absolute conditie zijn wel belangrijk.
- "Afdalingsmethodes", alles te kennen.
- "Oefeningen (zie pagina 329)"
  - Beschouw $f(x)$ als $f(x) = [f_1(x), f_2(x), \ldots, f_n(x)]$, met $x = [x_1, x_2, \ldots, x_n]$. De Jacobiaan van deze functie in het punt x_0 is gedefinieerd als de matrix:

$$J(x_0) = \left[\frac{\partial f_i(x_0)}{\partial x_j}\right]_{i,j=1}^{n}.$$

    Geef een eenvoudig verband tussen de Jacobiaan, de gradient en de Hessiaan.
  - 8.1 Met "wat kan men zeggen over de Hessiaan" duidt men op de eigenwaarden daarvan. Let op: in de linkse figuur zie je onderaan de contourlijnen (vergeet de rechte door de topjes van die parabolen), d.w.z. dat deze figuur langzaam daalt t.o.v. het x_1, x_2-vlak.
  - 8.2 Zou zo moeten kunnen lukken, zoek eerst wat $\nabla^3 f(x)$ betekent, en druk de vector maal matrix vermenigvuldigingen uit met behulp van sommaties.
  - Teken een 3d-figuur die past bij volgende contourplots. De kleuren stemmen overeen met een specifieke hoogte zoals aangegeven in de kleurenbalk.



  - Los hetvolgende stelsel van 2 vergelijkingen op:

$$\begin{aligned} x + 2y &= 3 \\ 0.499x + 1.001 &= 1.5 \end{aligned}$$

dit komt overeen met

$$AX = B = \begin{bmatrix} 1 & 2 \\ 0.499 & 1.001 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ 1.5 \end{bmatrix}.$$

Los daarna hetvolgende licht gewijzigde stelsel op

$$\begin{aligned} x + 2y &= 3 \\ 0.5x + 1.001y &= 1.5 \end{aligned}$$

Wat betekent dit voor de conditie van dit stelsel? De conditie van een stelsel wordt vastgelegd in het conditiegetal, dit is de verhouding tussen de grootste en de kleinste singuliere waarde. (Voor een symmetrisch positief definitie matrix is dit de grootste eigenwaarde gedeeld door de kleinste.) In de matlab oefenzitting zullen we hier even dieper op ingaan en kijken wat het effect van het conditiegetal op het oplossen van een stelsel is.

– Beschouw twee getallen x en y en definieer $s = x + y$ (som) en $v = x - y$ (verschil). Stel nu dat je een kleine wijziging op je gegevens aanbrengt namelijk $\tilde{x} = x + \Delta x$ en $\tilde{y} = y + \Delta y$. Stel dan de relatieve fout op voor de bewerking $s = x + y$ en de bewerking $v = x - y$, welke is de best geconditioneerde bewerking, en welke de slechtst geconditioneerde? Wanneer kunnen er zich problemen voordoen?

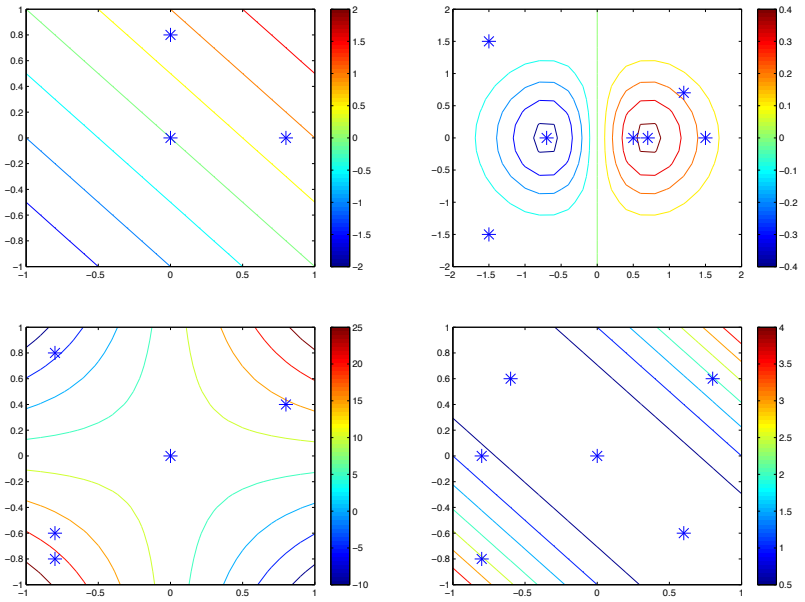
9 Chapter 9: Optimalisatie in meerdere veranderlijken

Dit is het vervolg van hoofdstuk 8.

- "De methode der steilste afdaling", alles te kennen, vooral het laatste paragraafje op pagina 322 is zeer interessant.
- "De methode van Newton Rapson". Op sommige plaatsen wordt verwezen naar andere leerstof uit het boek. Als we dat niet gezien hebben moet je ook die andere leerstof niet kennen, bv.: je moet niet weten wat de methode van Newton Rapson is voor het oplossen van een stelsel niet lineaire vergelijkingen, ook de methode van Cholesky moet niet gekend zijn.
- "Methodes zonder afgeleiden", lees dit tekstje maar eens.
- "Kleinste kwadraten problemen en stelsels", van dit deel dient alles tot bovenaan pagina 328 gekend te zijn. D.w.z. dat alles vanaf en inclusief het kader met het algoritme niet geleerd dient te worden. Het is wel aangeraden om dit deel eens te lezen.
- "Oefeningen"
  - 8.6 Met lineair in x bedoelen we dat $r(x)$ kan geschreven worden als een vermenigvuldiging tussen een matrix en een vector x. Dus gebruik de volgende vergelijking

    $$r(x) = Ax,$$

    waarbij A een matrix van m rijen en n kolommen is. Doe nu de berekeningen opnieuw, rekening houdend met deze vergelijking.
  - 8.7 Bereken gewoon J.
  - Onderzoek aan de hand van de volgende contourplots en de aangegeven startpunten of je convergentie verkrijgt of niet met de steepest descent methode. Teken dan ook de gevolgde weg.



  - We lossen hetvolgende overgedetermineerde systeem op, zodanig dat onze oplossing de best mogelijke oplossing is. We willen dus eigenlijk het kleinste kwadraten probleem oplossen. Namelijk

    $$\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 4 \\ 9 \end{bmatrix}.$$

Deze vergelijking $AX = B$ werd in je vroegere algebra lessen als volgt opgelost: $A^TAX = A^TB$. Dit geeft je een veel kleiner stelsel dat eenvoudig op te lossen is. Los dit maar eens op. Wat denk je dat er gebeurt met de conditie van de matrix A als je naar A^TA gaat? (We zullen dit probleem volgende les bekijken bij de Matlab oefenzitting. Kijk ook bij de volgende opgave.) Is dit bijgevolg een goede methode om te gebruiken op een machine.

– Om reeds aan te duiden wat er kan gebeuren kijken we naar de volgende matrix:

$$A = \begin{bmatrix} 1 & 1 \\ \varepsilon & 0 \end{bmatrix}.$$

Stel dat ε een heel klein getal is. Bereken dan A^TA. Wat gebeurt er dan met ε? Kan dit gevolgen hebben als de computer begint af te ronden?
Los nu eerst de volgende vergelijking op, waarbij je nu rekening houdt met het feit dat je computer het getal ε niet afrondt:

$$\begin{bmatrix} 1 & 1 \\ \varepsilon & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

En los nu de "theoretisch" equivalente vergelijking (door vooraan te vermenigvuldigen met A^T) op, rekening houdend met de afronding van $1+\varepsilon^2$ tot 1, omdat ε^2 nog veel kleiner is dan ε.

$$\begin{bmatrix} 1 & \varepsilon \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ \varepsilon & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & \varepsilon \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Wat merk je? Er is dus een belangrijk verschil tussen theoretische analyses en werkelijke implementaties.
Bekijk nu nogmaals je antwoorden op de voorgaande vraag.

10 Werken met Matlab

In de komende pagina's wordt het numerieke programma Matlab (Matrix Laboratory) kort toegelicht. We gaan volgende 3 items behandelen:

- leren matrices definiëren en enkele matrix bewerkingen uitvoeren;
- kijken wat numerieke wiskunde in realiteit betekent, door enkele korte experimenten uit te voeren;
- de steepest descent methode implementeren en toepassen op enkele functies (op een grafische manier).

De oplossingen van implementaties die je zelf dient te maken zijn ook weergegeven. Op deze manier kan je nadien je resultaten verifiëren. Let op: niet alle opgaves dienen opgelost te worden:

- deze opgaven zijn de basis en hier begin je mee;
* deze opgaven sla je in eerste instantie over, en als je tijd over hebt doe je deze;
** dit zijn de allerlaatste opgaven die je uitvoert.

10.1 Inloggen in C19

Inloggen in computerklas C19: Username: WETSTUD, paswoord wetstud. (Let op dat je toetsenbord wel in qwerty/azerty mode staat.)
De documenten die je moet downloaden kan je vinden op:
http://www.cs.kuleuven.be/∼raf/homepage/student/matlabpagina.php
Deze bestanden kan je opslaan onder de map documenten. Als je de bestanden die je zelf creëert wil bijhouden, kan je ze misschien best naar een schijf, usb-stick overzetten of naar jezelf toemailen.

10.2 Introductie

Matlab is vooral een numeriek rekenprogramma, gecreëerd om efficiënt met matrices te kunnen werken. De grafische interface bestaat uit verschillende componenten. Vooral de command window is de belangrijkste. In de command window worden expliciet commando's aan Matlab doorgegeven. Probeer al spelenderwijs een beetje vertrouwd te worden met Matlab.

- Definiëer zelf een matrix, als volgt:

  ```
  A=[1,2,3; 4,5,6]
  ```

  Een komma (of spatie) scheidt de getallen in dezelfde rij. Een puntkomma scheidt de verschillende rijen in een matrix. Het bovenstaand commando maakt dus een nieuwe matrix A aan.
- Als je een puntkomma weergeeft na je commando wordt er geen output op het scherm getoond.
- De vermenigvuldiging, optelling en aftrekking worden als volgt gedefiniëerd: *,+,-. Probeer zelf enkele matrices/vectoren te creëren en voer er enkele van deze bewerkingen op uit. Let op, als je bewerkingen met matrices uitvoert, dan moeten de dimensies deze bewerkingen toelaten. Als de dimensies niet kloppen, zal Matlab een foutmelding geven.
- Natuurlijk zijn er nog meerdere commando's: A', inv(A), size(A). Probeer deze commando's en kijk wat hun effect is.
- Wanneer je meer informatie over een welbepaald commando wil kan je die altijd opzoeken in de help. Ook het commando

  ```
  help inv
  ```

  geeft je normaal al voldoende informatie.
- Als je reeds verscheidene matrices gedefiniëerd hebt, kan je met

  ```
  who
  ```

kijken welke namen voor variabelen je al gebruikt hebt. Alle variabelen tegelijk wissen kan je doen met hetvolgende commando:

```
clear
```

– Matlab kan ook submatrices uit matrices halen, dat doe je als volgt:

```
A(1,2)
```

dat geeft je het element van de matrix A op de eerste rij, tweede kolom. Het commando

```
A(1:2,3:4)
```

geeft de submatrix bestaande uit rij 1 tot en met rij 2 en kolom 3 tot en met kolom 4. Deze bevelen kunnen ook eenvoudig gebruikt worden om toekenningen te doen in de matrix

```
A(1,2)=3
```

zal het element in de eerste rij tweede kolom gelijk stellen aan 3. Het commando

```
A(1:2,4)=[3;3]
```

stelt de eerste twee elementen in kolom vier gelijk aan 3. Ook volgende commando is plausiebel

```
A(1:2,4)=A(3:4,2)
```

Je kan ook matrices samenkleven

```
C=[A;B]
```

dan wordt de matrix (of vector) B onderaan bij de matrix (of vector) A bijgekleefd, of

```
C=[A,B]
```

dan wordt B achteraan A bijgekleefd. Let wel op dat de dimensies overeenkomen, anders kan Matlab de gevraagde operatie niet uitvoeren.

* Zoek het commando

```
eig
```

op en voer dan

```
[V,D] = eig(X)
```

uit voor een vierkante matrix X en controleer voor enkele matrices of de gevonden resultaten, namelijk $XV = VD$ kloppen. De matrix D bevat dus de eigenwaarden en de matrix V de eigenvectoren.

** Matlab kan zelf ook enkele eenvoudige matrices definieren. Bijvoorbeeld

```
zeros(100)
eye(10)
rand(3,4)
```

test ook wat het effect van het commando

```
diag
```

is op een matrix en wat het effect is op een vector. Probeer nu zelf wat matrices te creeren, je kan gerust een matrix van grootte 100, 1000, opstellen en bereken dan de eigenwaarde of de inverse. Denk er misschien wel aan om na je commando een ; te zetten, anders dan kan er een geweldige uitvoer over het scherm scrollen. Als een berekening te lang duurt kan je ze onderbreken, door CTRL-C in te drukken.

10.3 Werken met bestanden

Vaak is het sequentieel ingeven van commando's omslachtig. Het is eenvoudiger om een hele reeks commando's achter elkaar te typen en die dan allemaal tegelijk te laten uitvoeren. We zullen nu een matlab bestand (m-file) aanmaken, waarin verscheidene commando's kunnen weergegeven worden.

- Download het bestand bestand.m van de website. Open het dan met matlab (via file-¿ open).
- Lees de tekst van het bestand kort door en typ dan

  ```
  bestand
  ```

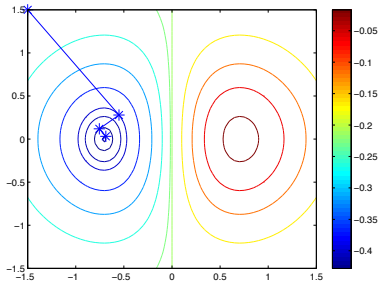
  in de command line (lees ook de volgende lijnen, mocht er een foutmelding komen). Matlab voert dan alle commando's achtereenvolgens uit. Let erop dat je Current-directory (de directory waarin de command window naar commando's zoekt), gelijk is aan de directory waarin je je bestand net hebt opgeslaan. Als dit niet dezelfde directory is zal Matlab de volgende foutmelding geven:

  ```
  ??? Undefined function or variable 'bestand'.
  ```

  De Current-directory kan je veranderen door in het gelijknamige balkje bovenaan de gepaste directory te kiezen.
- Probeer nu zelf een klein bestand te maken. Bij bestand(file), maak je een nieuwe m-file aan.
- Typ verscheidene commando's achtereenvolgens in je bestand. Elk commando op een nieuwe lijn. Save je file bijvoorbeeld als "numopt.m". Let op de extensie ".m" deze is belangrijk. Als je bestand opgeslagen is en je zit in de correcte directory, typ dan numopt in je command window. Je ziet dat Matlab nu al de berekeningen achtereenvolgens uitvoert.

10.4 Steepest descent methode

Nu we wat vertrouwd zijn met de werking van Matlab gaan we de steepest descent methode implementeren. De bedoeling is dat we uiteindelijk grafische prenten (die het verloop van de steepest descent weergeven) zoals deze hieronder gaan construeren. Het volledige implementatie



werk met functies, stop criteria, while lus enzovoort vraagt te veel tijd. De belangrijkste functies in Matlab krijg je allemaal. Enkel de basisstap van de steepest descent methode zoals geleerd uit de cursus zal je moeten implementeren.

- Om te beginnen download je de volgende bestanden, die allemaal nodig zijn om de steepest descent te implementeren: main.m; steep.m ; steepplotsurf.m ; steepplotcont.m ; xyfunc.m ; linesearchmin.m. Hier zal kort samengevat worden wat deze functies bevatten.
  - main.m: Een bestand gewoon gemaakt om zo eenvoudig mogelijk de functie steep.m aan te roepen.
  - steep.m: De steepest descent methode, **deze functie gaan we verder aanpassen**.

  - steepplotsurf.m en steepplotcont.m: Deze twee functies tekenen het oppervlak van de functie, en de contouren, met de convergentielijnen van het algoritme.
  - xyfunc.m: Dit bestand bevat de functie, waarop we de steepest descent methode gaan toepassen. **Deze functie moeten we wijzigen, of een nieuwe soortgelijke functie maken om de steepest descent op andere functies toe te passen.**
  - linesearchmin.m: Hier wordt het minimum van de parameter α in de steepest descent methode berekend.
- De belangrijkste functie is de functie steep.m, dit is de enige functie die niet volledig is. Je zal hier zelf de nodige stappen moeten toevoegen. Vervolledig daartoe de lijnen code waar

  ```
  % COMPLETE
  ```

  bij staat weergegeven. Telkens dien je slechts 1 lijn code toe te voegen. Hieronder staan achtereenvolgens de verschillende dingen die je moet toevoegen.

Vooraleer je begint te vervolledigen kan je best het programma even overlopen, zodat je het nut weet van de verschillende regels code.

- Open het bestand main.m. Je ziet dat er hier bijna niets gebeurt, enkel steep.m wordt opgeroepen en de startvector wordt gedefiniëerd. Vooral de regels in het vet zijn belangrijk. We weten niet of onze routine straks goed gaat werken, dus werken we op voorhand een stop criterium in: maximum 6 iteraties mogen uitgevoerd worden.
  Open nu het bestand steep.m. Zoek de lijn die de code

  ```
  % COMPLETE (Number of iterations)
  ```

  bevat. Voeg nu een regel code toe, zodanig dat je programma het aantal iteraties telt. Probeer nu main.m uit te voeren, en als je alles correct gedaan hebt, zou je enkel 6-7 maal op een toets mogen drukken. Werkt het niet, stop dan je programma door CTRL-C te gebruiken. Normaal gezien krijg je nu enkel een oppervlak en een contourplot te zien. Je plaatst deze prenten best naast elkaar. Op de contourplot is tevens een kruis geplaatst dat de huidige waarde $x^{(k)}$ weergeeft. In de volgende stappen zal die $x^{(k)}$ hopelijk convergeren naar het minimum.
- Definieer de nieuwe iteratie, vervolledig dus de lijn:

  ```
  % COMPLETE (Define the new iterate)
  ```

  Als eerste poging nemen we gewoon de iteratie met $\alpha = 1$, dus:

  $$x^{(k+1)} = x^{(k)} - \nabla f(x^{(k)}).$$

  Stokeer de uitkomst dan opnieuw in de variabele *xk*. Als je dit uitvoert verschijnt er nog niets op je contourplot. Je kan wel je output, dus de berekende waarden van *xk* laten afprinten op het scherm (door er geen puntkomma) achter te plaatsen.
- Vervolledig de lijn:

  ```
  % COMPLETE (Store the new value)
  ```

  Hier voeg je de nieuw berekende waarde *xk* (*xk* is een kolom vector), achteraan de matrix *intx* toe. Dit is nodig om je iteraties op het scherm te laten verschijnen in de contourplot. Je zou nu een rechte lijn moeten zien die altijd maar langer wordt. De $*$'en markeren de gevonden iteratie punten. Waarom groeit deze lijn maar in 1 vaste richting??
  Inderdaad, dat is omdat we de gradient, en functie waarde in de nieuwe *xk* nog niet aangepast (of herberekend) hebben binnen de iteratie. We zullen nu in het nieuwe punt deze waarden berekenen om zo de nieuwe gradient voor de volgende iteratie te vinden.
- Vervolledig de lijn:

  ```
  % COMPLETE (Compute the new gradient and function value)
  ```

  Je ziet dat de 'zoek'lijn op de contourplot nu van richting verandert, maar veel te ver springt. Als je dit uitvoert in Matlab kan het zijn dat je zelfs na enkele iteraties een foutmelding krijgt. Dit is niet erg, deze fout verdwijnt terug wanneer we verder aanpassingen doorvoeren.

- De zoeklijn springt telkens over het minimum. Laat ons daarom eerst en vooral de gradient normaliseren, vervang:
$$\nabla f(x^{(k)}),$$
door
$$\nabla f(x^{(k)})/\|\nabla f(x^{(k)})\|.$$
De zoekrichting zal dan steeds een vaste lengte $= 1$ hebben. De norm bereken je in matlab simpelweg met
`norm( )`
Je kan nu al eens proberen om meer dan 6 iteraties uit te voeren.
- Je ziet dat je nog steeds over het minimum springt. De parameter α is dus noodzakelijk. Zoek het lijnminimum, gebruik makend van de de functie "linesearchmin.m" en vervolledig de lijn:
`% COMPLETE (Line minimum)`
Je kan deze functie openen in een nieuw venster en nagaan hoe je het lijnminimum bepaalt. Vergeet ook niet de parameter α toe toe voegen in de volgende vergelijking:
$$x^{(k+1)} = x^{(k)} - \alpha * \nabla f(x^{(k)})/\|\nabla f(x^{(k)})\|.$$
Als je je algoritme nu laat lopen zie je dat je naar het minimum convergeert.
- Als laatste vervolledig je
`% COMPLETE (Assign your solution to x)`
- Je kan nu ook het maximum aantal iteraties in je bestand main.m optrekken, en zo kijken hoeveel dichter je tot bij het minimum raakt.

Nu heb je de functie steep.m volledig aan het werken. Nu gaan we deze methode toepassen op verschillende functies.

- De functie die je als parameter meegeeft, namelijk de functie xyfunc.m, kan je eenvoudig wijzigen. Kijk in het bestand xyfunc.m. Sla deze functie op onder een andere naam bv. xyfuncnew.m en probeer dan je functie te wijzigen. Er zijn drie plaatsen waar je je functie dient aan te passen: berekenen van de evaluatie van de functie, berekenen van de gradient en dan onderaan voor het plotten van je functie.
- Eenmaal de functie aangepast, kan je je steepest descent methode laten lopen. Wijzig in je bestand main.m (of sla een nieuw bestand main1.m op) de startvector, en verander in je functieaanroep van de steepest descent *@xyfunc* door jouw gekozen functienaam.
- Probeer nu je steepest descent methode uit op de functies die je de vorige maal in je opdracht hebt meegekregen. Je kan dan controleren of je oplossingen corresponderen met de realiteit. Ik geef hierbij de functies en de startvectoren:
  - $f(x,y) = x+y$, in de startvectoren: $x_0 = [0.8,0]^T, x_0 = [0,0]^T, x_0 = [0,0.8]^T$.
  - $f(x,y) = (x+y)^2$, in de startvectoren: $x_0 = [-0.8,-0.8]^T, x_0 = [-0.8,0.8]^T, x_0 = [0,0]^T, x_0 = [-0.8,-0.6]^T, x_0 = [0.8,0.4]^T$.
  - $f(x,y) = xe^{-x^2-y^2}$, in de startvectoren $x_0 = [0.7071,0]^T, x_0 = [-0.7,0]^T, x_0 = [-1.5,1.5]^T, x_0 = [-1.5,-1.5]^T, x_0 = [0.5,0]^T, x_0 = [1.5,0]^T, x_0 = [1.2,0.7]^T$. Let op: e^x noteer je in Matlab als $exp(x)$.
  - $f(x,y) = x*y$, in de startvectoren: $x_0 = [-0.8,-0.8]^T, x_0 = [0,0]^T, x_0 = [-0.6,0.6]^T, x_0 = [0.6,-0.6]^T, x_0 = [-0.8,0]^T, x_0 = [0.8,0.6]^T$.

10.5 Numerieke spielerei

Dit is een extra deeltje bij de cursus, om de eigenschappen van numerieke rekenprogramma's even te bekijken. Zorg eerst dat je het deel over de steepest descent afgerond hebt. Dit is EXTRA, en moet dus niet noodzakelijk gezien worden in deze lessen. Het is meer iets voor als je tijd over hebt.

Laat ons even enkele numerieke eigenschappen bekijken die ook in de lessen aangekaart werden.

- De machine epsilon geeft aan waar er afgerond wordt[1] (in verhouding tot het getal 1). De machine epsilon, ε, kan je in matlab opvragen en wordt weergegeven met het commando:

  ```
  eps
  ```

- Standaard geeft matlab slechts een beperkt aantal van de 16 bijgehouden cijfers. Als je volgende commando's ingeeft verandert de weergave van getallen op je scherm:

  ```
  format short e
  format long e
  format short
  ```

  standaard heb je format short. Format short e geeft je iets van de vorm $y.yyy10^{xx}$, en format long e geeft je hetzelfde maar meer cijfers na de komma. Verander het output format en print dan eps eens af. Kijk of het inderdaad zo is dat als je $1+x$ doet met $x < \varepsilon$, dat je dan terug 1 krijgt.
- We gaan nu eens kijken naar de afrondingsfouten van matlab. Voor de volgende bewerking herhaaldelijk uit en kijk waar het fout loopt:

  ```
  0.1+0.1-2*0.1
  0.1+0.1+0.1-3*0.1
  0.1+0.1+0.1+0.1-4*0.1
  .......
  ```

- Zoals vermeld in de lessen geeft het conditiegetal een maat aan voor het verlies van cijfers, bij bv. het oplossen van stelsels. In Matlab kan je eenvoudig het conditie getal van een matrix opvragen door hetvolgende commando te gebruiken:

  ```
  cond(A)
  ```

  Bekijk nu het conditiegetal van wat matrices en ook van een diagonaal matrix. Wat merk je bij het conditiegetal van de volgende matrices

  ```
  diag([1,2,3])
  diag([2,2,3])
  ```

  hoe komt dit denk je. Denk aan de definitie van het conditiegetal voor een positief definiete symmetrische matrix, in functie van de eigenwaarden. Probeer nu de diagonaal elementen te wijzigen om zo een zeer slechte conditie (dus een groot conditiegetal) te verkrijgen.
- Als laatste onderdeeltje gaan we kijken naar de nauwkeurigheid van het oplossen van een stelsel via invertering, rekening houdend met de conditie van de matrix.
  - Download de bestanden accuracy.m en gencond.m van de site.
  - Het bestand gencond.m genereert een matrix met een bepaald conditiegetal. Kijk even naar de help van de functie en probeer dan uit of dit inderdaad zo is.
  - De functie accuracy.m is nog niet afgewerkt. Dit moet je zelf doen, er staat al een heleboel uitleg in het bestand. Grofweg moet je gewoon, gegeven een zekere A en x de vector $b = Ax$ berekenen. Daarna bereken je opnieuw de vector $\tilde{x} = A^{-1}b$. Je zou verwachten dat $\tilde{x}$ gelijk is aan x. We zullen nu zien dat dit zal afhangen van de conditie van je matrix.
    De functie moet dan als output de vector $\tilde{x}$, de vector b en de norm: $\|\tilde{x} - x\|$ geven. Je hoopt dat $\tilde{x} = x$ en dat bijgevolg de norm ook nul zal worden. Pas deze dingen aan in het bestand accuracy.m, de dingen die je moet vervolledigen zijn duidelijk vooraf gegaan door
    ```
    % COMPLETE
    ```
  - Probeer nu je programma te testen voor verschillende keuzes van x. Probeer eerst je programma uit op een diagonaal matrix. Genereer daarna matrices met het programmaatje gencond.m dat je meegekregen hebt. Probeer dan opnieuw wat het resultaat is.
  - Wat kan je besluiten i.v.m. de nauwkeurigheid van de gevonden x, m.b.t. de grootte van de matrix en m.b.t. het conditiegetal.
- Als laatste: kijk eens wat er gebeurt met de conditie van een matrix (van gencond) als je de conditie van A^TA berekent. Wat kan je hieruit concluderen in verband met het oplossen van een kleinste kwadraten probleem via A^TA.

[1] Merk op dat dit niet als definitie mag beschouwd worden.

De oplossingen van de figuren van in de optimalisatie lessen

