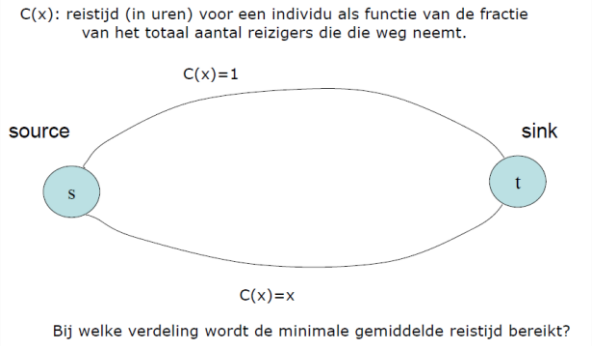
Toepassingen van Operationeel Onderzoek

Samenvatting

18-1-2011
KUL, Prof. Spieksma
Lynn.gyselen@student.kuleuven.be, indien u aanpassingen, opmerkingen, extra opgaven of oplossingen heeft, gelieve deze door te sturen, dan pas ik deze aan in het document!

Inhoud

Pigou's example



Laten we ervan uit gaan dat een fractie x langs beneden gaat, en eenfractie1-x langs boven. De gemiddelde reistijd bedraagt dan x*x + (1-x)*1, oftewel x^2–x + 1.
De gemiddelde reistijd bedraagt dus x^2–x + 1. Differentieren en nul stellen levert 2x-1=0, oftewel x = ½, met een gemiddelde reistijd van 45 minuten.

1. Netwerken

2. Inleiding

De *Euler-graaf* is een speciaal soort graaf die geïdentificeerd is door de wiskundige Leonhard Euler naar aanleiding van zijn oplossing van het probleem van de Zeven bruggen van Königsberg. Dit probleem komt neer op de vraag of het mogelijk is in een samenhangende graaf een wandeling te maken waarin alle kanten van de graaf precies één keer voorkomen (een zogeheten *Euler-wandeling*) of zelfs een dergelijke wandeling te maken zodat deze begint en eindigt in dezelfde knoop (*Euler-cykel*).
In 1736 loste Euler dit probleem op en bewees dat de oplossing heel simpel is:
Stelling
Een simpele, samenhangende graaf bevat een Euler-cykel dan en slechts dan als de graad van alle knopen even is.
Bewijs
"$\Rightarrow$": Als een graaf een Euler-cykel bevat, dan is er dus een wandeling van iedere knoop in de graaf naar diezelfde knoop waarin alle kanten van de graaf voorkomen. In een simpele graaf kan het niet anders dan dat in een dergelijke wandeling ook alle knopen voorkomen: een kant moet tussen twee knopen lopen. Verder is het ook nog zo, dat je in die wandeling bij iedere knoop waar je "aankomt" ook weer "weggaat" (in omgekeerde volgorde bij het beginpunt, uiteraard). Omdat je in een Euler-cykel iedere kant maar één keer mag gebruiken, betekent dat dat er bij iedere knoop, voor iedere "inkomende" kant ook een "uitgaande" kant moet zijn. Dus het aantal kanten waarmee een knoop incident is, is in een Euler-graaf altijd een veelvoud van twee, oftewel de graad van al de knopen is even.(wikipedia)

2.1. Wat is een netwerk precies?

Een netwerk bestaat uit punten (vertices) en verbindingen tussen punten. Die verbinden kunnen gericht zijn, en dat heten ze pijlen (arcs), of ze kunnen ongericht zijn, en dan heten ze kanten (edges). Conventies: de puntenverzameling heet V, met |V|=n; de pijlenverzameling heet A, met |A|=m; de kantenverzameling heet E, met |E|=m.

2.2. Hoe representeer ik een netwerk?

Node-nodeadjacencymatrix:

Een 0-1 (n x n) matrix A met aij= 1 asa(als en slechts als) er een arcis van knoop i naar knoop j.
Voordelen:
–Eenvoudig te implementeren
–en te manipuleren (analoge matrices voor eventuele kosten\capaciteiten)
–Efficient voor dichte netwerken (veel pijlen)
Nadelen:
–Inefficient voor netwerken met weinig pijlen
Arc-nodeincidencematrix:
Een (n x m) matrix A met aij= -1 als arcj vertrekt uit knoop i, aij= 1 als arcj binnenkomt in knoop i, en aij= 0 in alle andere gevallen.
Nadelen:
–Inefficient (veel nullen)
–Niet eenvoudig te manipuleren
Adjacencylists:
voor elke knoop i, een lijst met knopen waarmee knoop i verbonden is via een uitgaande pijl.
Voordelen
–Efficient in ruimtegebruik
–Makkelijk te manipuleren (kosten, capaciteiten)
–Voor alle soorten netwerken

Node-node incidentiematrix

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	0	1	0	0
3	0	0	0	1	1	0
4	0	0	0	0	0	1
5	0	0	0	1	0	1
6	0	0	0	0	0	0

Arc-node adjacency matrix

	a	b	c	d	e	f	g	h
1	-1	-1	0	0	0	0	0	0
2	0	1	0	0	-1	0	0	0
3	1	0	-1	-1	0	0	0	0
4	0	0	0	1	1	1	0	-1
5	0	0	1	0	0	-1	-1	0
6	0	0	0	0	0	0	0	1

Adjacency-lists

1: 2,3
2: 4
3: 4,5
4: 6
5: 4,6
6: -

3. Kortste pad

Gegeven een netwerk N=(V,A), en gegeven een getal cij≥ 0 voor alle {i,j} in A,
•vind het kortste pad tussen een gegeven punt s en een gegeven punt t, of
•vind het kortste pad tussen een gegeven punt s en elk ander punt van V, of
•vind het kortste pad tussen elk tweetal punten van V.

3.1. Toepassingen

Toepassing 1
Minimaliseer de inspectie-en productiekosten van een productielijn.
Een productielijn bestaat uit n machines. Er is een batch van B producten die achtereenvolgens de machines bezoekt. Elke machine voert een bepaalde operatie uit op elk der producten. Voor elke machine is een fractie aibekend die het percentage fout-geproduceerde items aangeeft (1 ≤ i ≤ n). Aangezien de items met een fout waardeloos zijn wilt u die uit de batch halen (inspectie) zodat volgende machines geen operaties meer hoeven uit te voeren op die items. Aan de andere kant: inspectie kost ook geld…
Hoe weegt u de kosten van nodeloze productie af tegen inspectie?
De data:
B: aantal items in de batch
ai: fractie fout-productiebij machine i

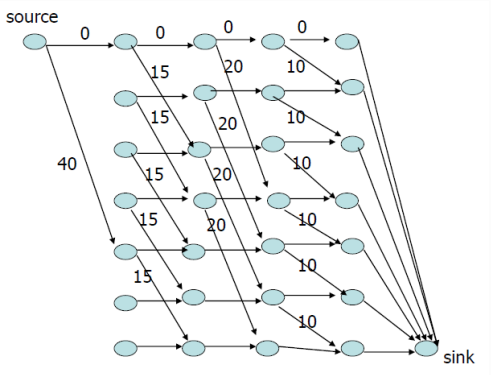
pi: productie kosten per item voor machine i
fij: vaste kosten van inspectie van de batch direct na machine j, wanneer vorige inspectie na machine i was
gij: variabele kosten (per item) van inspectie van de batch direct na machine j, wanneer vorige inspectie na machine i was
Merk op: de inspectiekosten na machine j hangen af van de laatste keer dat er inspectie plaats vond vòòr machine j, omdat een item op een defect veroorzaakt door machines i+1, i+2, …, j gecontroleerd moet worden.
We gaan dit probleem modelleren als een kortste pad probleem met de kost c per boog.

$$c_{ij} = f_{ij} + B(i)\sum^{j}_{k=i+1} p_k + B(i)g_{ij}$$

Toepassing 2
u krijgt n items, en een knapsack. De knapsack heeft capaciteit C, elk item neemt ai capaciteit in. Als u item i selecteert levert u dat wi op. Elk item kan maximaal 1 maal geselecteerd worden. We modelleren dit probleem als een langstepad probleem dat we dan omzetten in een kortste padprobleem. Het netwerk: er is een 'kolom' knopen voor elk item, en er is een 'rij' knopen voor elke eenheid ruimte in de knapsack.

i	1	2	3	4
w_i	40	15	20	10
a_i	4	2	3	1

C=6



Een langste-pad probleem kan gezien worden als een kortste-pad probleem door de kosten van elke arc met -1 te vermenigvuldigen. Dit kan een probleem veroorzaken als er vervolgens negatieve cycles ontstaan, maar dat zal hier niet gebeuren.
Een *cykel* in een graaf G is een wandeling van een knoop a naar a.=negatieve lus

Toepassingen:
In boek Operations Research: Transshipment problem en minimizing car replacement costs.

3.2. Dijkstra's algoritme

```
Algoritme Dijkstra
begin
    S:={};T:=V;
    d(i):=+∞ voor alle i ϵ V
    d(s):=0 en pred(s):=0;
    while |S|<n do
          begin
          kies i ϵ T zodanig dat d(i) = min{ d(j): j ϵ T};
          S := S + {i};
          T := T  \ {i};
          voor iedere (i,j) ϵ A(i) do
                if d(j)>d(i)+cij  then d(j):=d(i)+cij en pred(j):=i;
          end;
end.
```

Label node 1 met 0 permanent.
Label iedere node verbonden met node 1 met een tijdelijk label, oneindig voor diegene niet verbonden met node 1.
Volgende recursie:
Maak de kleinste tijdelijke permanent.
Vervang iedere andere tijdelijke door
min(tijdelijke, permanente van vorige knoop+ lengte boog tot huidige knoop)
Onthou steeds van elke knoop van welke knoop het vorige permanente kwam.

3.2.1. Geeft Dijkstra steeds een kortste pad?

Optimaliteit van pad-afstanden d is equivalent met
d(j) ≤d(i) + cij voor alle {i,j} in A
Stel u geeft mij pad-afstanden d die WEL voldoen aan de condities.
•Pak een willekeurig pad P van de source naar j: s = i1-i2-… -ik= j .

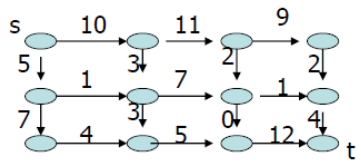
We weten
d(ik) ≤ d(ik-1) + c(ik-1,ik)
d(ik-1) ≤ d(ik-2) + c(ik-2,ik-1)
.
d(i2) ≤ d(i1) + c(i1, i2)
•Optellen levert:
d(j) = d(ik) ≤c(ik-1,ik) + c(ik-2,ik-1) + …+ c(i1, i2).
d(j) vormt een ondergrens voor de lengte van elk pad van s naar j, er bestaat geen j met d(j) >d(i) + cij

Looptijd van een algoritme is een belangrijk kenmerk.
•Looptijd hangt af van: i) grootte van de instantie, ii) toevallige data
•De *tijdscomplexiteitsfunctie* f is een functie van de probleemgrootte, en beschrijft het maximale aantal operaties nodig om een instantie van gegeven grootte op te lossen.
•**Definitie**: een algoritme loopt in O(f(n))als er een constante c bestaat zodanig dat de tijd die het algoritme nodig heeft ten hoogste c f(n) is.

3.3. Complexiteit

Hoeveel elementaire operaties heeft Dijkstra nodig? O(n^2)
Merk op: het aantal mogelijke kortste paden in een netwerk met n punten is 2^n, terwijl we het kortste pad dus in O(n^2) berekeningen kunnen vinden.

3.4. Oefeningen



Vind een kortste pad van s naar t.

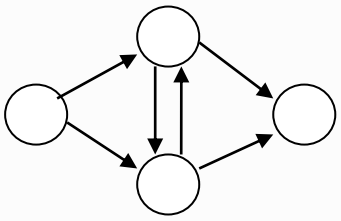
1. Oplossing Kortste pad via Dijkstra (wel cycli, geen negatieve afstanden)
2. Oplossing **Manhattan Netwerk** via dynamische programmering (geen cycli, geen negatieve afstanden gezien topologische ordening mogelijk moet zijn)

Achterwaartse recursie: f(x,y)= min {R(x,y)+f(x+1,y); U(x,y)+f(x,y+1)}

f(1,12)=min{R(11,12)+f(1,11); U(8,12)+f(1,8)}
f(1,12)=min{R(1,2)+f(2,12); U(1,5)+f(5,12)} (voorwaartse)
Tijdscomplexiteit : 3(n+1)^2 : O(n^2)

Werkt Dijkstra nog steeds indien er negatieve afstanden zijn?
Neen, algoritme van Bellman Ford:

1. Init: $d^0(1)=0$; $d^0(i)=\infty$ voor overige i∈ N; k=1
2. $\forall i \in N$: $d^k(i)$=min{voor alle voorgangers j van i: $d^{k-1}(j)+c_{ji}$;)
3. Stop indien $d^k(i)= d^{k-1}(i)$ voor alle I of indien k=n; anders k=k+1



$d^1(2)=\min\{d^0(1)+c_{12}; d^0(3)+c_{32}\}=\min\{0+3;\infty-3\}=3$
$d^1(3)=\min\{d^0(1)+c_{13}; d^0(2)+c_{23}\}=\min\{0+5;\infty+1\}=5$
$d^1(4)=\min\{d^0(2)+c_{24}; d^0(3)+c_{34}\}=\min\{\infty+10;\infty+1\}=\infty$
$d^2(2)=\min\{d^1(1)+c_{12}; d^1(3)+c_{32}\}=\min\{0+3;5-3\}=2$
$d^2(3)=\min\{d^1(1)+c_{13}; d^1(2)+c_{23}\}=\min\{0+5;3+1\}=4$
$d^2(4)=\min\{d^1(2)+c_{24}; d^1(3)+c_{34}\}=\min\{3+10;5+1\}=6$
$d^3(2)=\min\{d^2(1)+c_{12}; d^2(3)+c_{32}\}=\min\{0+3;4-3\}=1$
$d^3(3)=\min\{d^2(1)+c_{13}; d^2(2)+c_{23}\}=\min\{0+5;2+1\}=3$
$d^3(4)=\min\{d^2(2)+c_{24}; d^2(3)+c_{34}\}=\min\{2+10;4+1\}=5$
negatieve lus: 2-3-2

	1	2	3	4
d_0	0	∞	∞	∞
d_1	0	3	5	∞
d_2	0	2	4	6
d_3	0	1	3	5
d_4	0	0	2	4
d_5	0	-1	1	3

Om lus op te sporen, residueel netwerk tekenen in het negatieve lus algoritme om de capaciteit op te gebruiken.

Geef de wiskundige formulering van het het kortste pad probleem

- Definieer beslissingsvariabelen:

x_{ij} = hoeveelheid stroom over arc {i,j} voor elke {i,j} in A

$$\text{Min } \Sigma_{ij \text{ in } A}\, c_{ij}x_{ij}$$
$$\text{st } \Sigma_{j:ji \text{ in } A}\, x_{ji} - \Sigma_{j:ij \text{ in } A}\, x_{ij} = 0 \quad \text{voor i in V, i} \neq s,t$$
$$- \Sigma_{j:sj \text{ in } A}\, x_{sj} = -1$$
$$\Sigma_{j:jt \text{ in } A}\, x_{jt} = 1$$
$$x_{ij} \geq 0 \quad \text{voor elke arc } \{i,j\} \text{ in A}$$

Dwz. In ieder knooppunt is de hoeveelheid stroom die aankomt gelijk aan de hoeveelheid stroom dat vertrekt, het beginknooppunt en eindknooppunt niet meegerekend.
De hoeveelheid stroom dat vertrekt van het beginknooppunt is 1 en de hoeveelheid stroom dat toekomt in het eindpunt is 1. (geen capaciteiten)

van s naar alle andere knopen

$$\text{Min } \Sigma_{ij \text{ in } A}\, c_{ij}x_{ij}$$

$$\text{st } \Sigma_{j:ji \text{ in } A}\, x_{ji} - \Sigma_{j:ij \text{ in } A}\, x_{ij} = 1 \qquad \text{voor i in V, i} \neq s$$

$$-\Sigma_{j:sj \text{ in } A}\, x_{sj} = -(n-1) \qquad \text{voor i = s}$$

$$x_{ij} \geq 0 \qquad \text{voor elke ij in A}$$

4. Het opspannende boom probleem

(Een boom is een verbonden netwerk zonder cykels)

Gegeven een netwerk N=(V,E), en gegeven een afstand $c_{ij} \geq 0$ voor elke {i,j} in E,
Vind een verzameling kanten met minimale totale lengte, zodanig dat er een pad bestaat tussen elk tweetal punten.
Hoeveel kanten heb je *tenminste* nodig?
•Het minimale aantal kanten dat je nodig hebt om n punten met elkaar te verbinden is n-1.
•Immers, de eerste kant verbindt twee punten, en vervolgens verbindt elke volgende kant een extra punt.
Hoeveel kanten heb je *maximaal* nodig?
•Het maximale aantal kanten dat je kan hebben in een netwerk met n punten zonder een cykel te veroorzaken is n-1 (de afwezigheid van cykels is een definiërende eigenschap van een boom).

4.1. Algoritme Prim

```
Algoritme Prim
begin
    S := {v1}, E':={};
    for all i in V, i ≠ v1, do closest[i] := v1;
    while S ≠ V do
    begin
          min:=∞;
          for all i in V-S do
                    if c(i, closest[i]) < min then
                    min:=    c(i, closest[i]), next:=i;
          S:=S U {next};
          E':=E' U {next, closest[next]};
          for all i in V-S do
                     if c(i, closest[i]) > c(i,next) then
                    closest[i]:=next;
    end.
```

Kies 1 node.
Zoek een tweede node die daar het dichtstebij is
Zoek een derde node die het dichtstebij is bij 1 vd 2 nodes in S, etc.

4.2. Algoritme Kruskal

```
Algoritme Kruskal
begin
   Sorteer de kanten naar hun gewicht cij in niet-
   aflopende volgorde: e1, e2,…, em.
   begin
       F:={e1};
       for j =2, …,m do
             als het toevoegen van ej geen
             cykel veroorzaakt in F, set F := F U ej.
   end.
```

4.3. Complexiteit

Hoeveel operaties zijn er nodig voor Prim? O(n^2)
Hoeveel operaties zijn er nodig voor Kruskal? O(m log m)

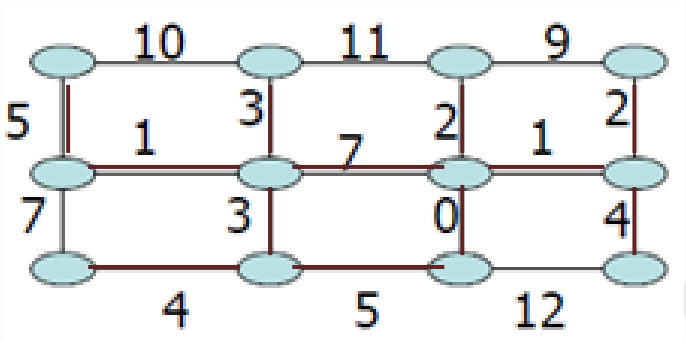
4.4. Oefeningen

Aanpassen Dijkstra's algoritme naar Max Flow Algoritme

Laat c_{ij} de capaciteit zijn van een boog, Definieer de capaciteit van een pad als de minimale capaciteit van een boog in pad P. Beschouw het probleem: vind, voor elke knoop i, een pad P van de source naar knoop i met maximale capaciteit. Hoe kunt u Dijkstra's algoritme aanpassen om bovenstaand probleem op te lossen?

Algoritme Dijkstra
begin
S:={};T:=V;
c(i):= -∞ voor alle i ϵ V
c(s):= +∞ en pred(s):=0;
while |S|<n do
begin
kies i ϵ T zodanig dat c(i) = max{ c(j): j ϵ T};
S := S + {i};
T := T \{i};
voor iedere (i,j) ϵ A(i) do
if c(j)<min{c(i); c_{ij} }then c(j):=min{c(i); c_{ij} }en pred(j):=i;

Los dit probleem op met Prim en Kruskal

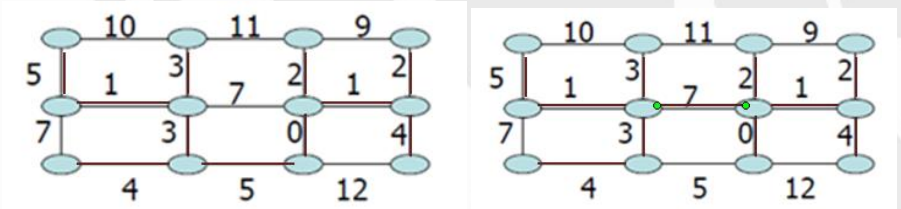


Prim: beginnen bij S
Kruskal: eerst selecteren 0, dan 1,..

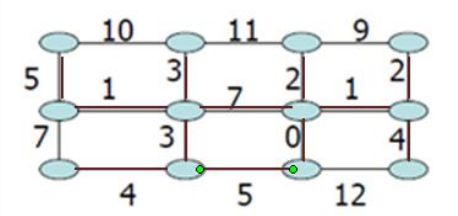
Optimaliteit van Kruskal: bewijs

Beschouw de kanten van de kortste boom, en zet ze in toenemende volgorde. Doe hetzelfde voor de "Kruskal-boom", en neem aan dat de twee bomen verschillen. Beschouw vervolgens de eerste kant in de Kruskal-boom die verschilt van een kant in de kortste boom. Wanneer we die kant toevoegen aan de kortste boom, ontstaat er een cykel. Die cykel verbreken we door het weghalen van een niet-Kruskal kant (zo'n kant moet er zijn anders had de Kruskal boom een cykel!). De boom die we dan vinden heeft dus een kleinere lengte. Dat kan niet, want het was een kortste boom, en vandaar dat de twee bomen niet verschillen.

Kruskal kortste boom



Verbeterde kortste boom: onmogelijk, dus kortste boom moest gelijk zijn aan Kruskal



Geef een wiskundige formulering van het probleem om een minimaal opspannende boom te vinden.

We gaan uit van een netwerk N = (V, E).

Definieer beslissingsvariabelen:

x_{ij}= 1 als kant {i,j} wordt gekozen, 0 anders.

$$\text{Min } \Sigma_{ij \text{ in } E}\ c_{ij}x_{ij}$$

$$\text{st } \Sigma_{i \text{ in } S}\ \Sigma_{j \text{ not in } S}\ x_{ij} \geq 1 \quad \text{voor elke S C V}$$

$$x_{ij} \text{ in } \{0,1\} \quad \text{voor elke kant } \{i,j\} \text{ in E.}$$

Gegeven is een *compleet* netwerk (i.e., een netwerk met alle mogelijke kanten). Op n=3 punten hoeveel verschillende bomen zijn er? En als n=4? Voor willekeurige n?

- Met n=3 zijn er 3 verschillende bomen.
- Met n=4 zijn er 16 verschillende bomen.
- Met n=5 zijn er 125 verschillende bomen.
- Met n=6 zijn ? verschillende bomen

5. Max Flow

Gegeven een netwerk N=(V,A), en gegeven een getal (capaciteit) $u_{ij} \geq 0$ voor alle {i,j} in A en gegeven een node s en een node t, vind een maximale stroom van node s naar node t.

Maxflow problemen zijn een speciaal geval van MCNFP

Aannames:

- Het netwerk is gericht
- Alle capaciteiten zijn geheeltallig
- Er bestaat geen pad van s naar t bestaande uit pijlen elk met een onbegrensde capaciteit
- Als (i,j) ϵ A, dan ook (j,i) ϵ A. (Dit is zonder verlies van algemeenheid…)
- ALS alle capaciteiten geheeltallig zijn
- DAN bestaat er een optimale stroom die geheeltallig is
- Dit volgt uit de structuur van de constraint-matrix (totaal unimodulair)

5.1. Wiskundige formulering

Maximaliseer v

subject to:

$$\sum_{\{j:(i,j) \in A\}} x_{ij} - \sum_{\{j:(j,i) \in A\}} x_{ji} = \begin{cases} v & \text{voor } i=s \\ 0 & \text{voor } i \in V\{s,t\} \\ -v & \text{voor } i=t \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij} \quad \text{voor alle } (i,j) \text{ in } A$$

5.1.1. Totaal UniModulair

Definitie: Een matrix A is totaal unimodulair(TUM) als elke vierkante submatrix van A determinant +1,

Eigenschap: Als matrix A TUM is, dan lost de LP relaxatie het IP op.

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

niet TUM

$$\begin{bmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

TUM

Determinant = 2

Determinant 3x3 linksboven: 0+0+1-0-0-0

5.2. Toepassingen

Toepassing 1

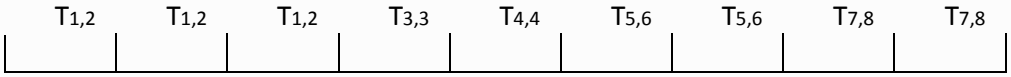
Het verhaal: u heeft een verzameling jobs (J), elk met een procestijd pj, een release tijd rj, en een deadline dj. Ook heeft u een verzameling machines om de taken uit te voeren. Een machine kan slechts 1 job tegelijkertijd uitvoeren, en aan een job kan slechts maximaal 1 machine werken. Onderbrekingen zijn toegestaan.

De vraag: bestaat er een schedule zodanig dat alle taken op tijd gereed zijn?

Stel 3 machines en:

Job	1	2	3	4
p_j	1,5	1,25	2,1	3,6
r_j	3	1	3	5
d_j	5	4	7	9

Laten we alle release dates en deadlines sorteren: 1,3,4,5,7,9. We vinden 5 tijdsperioden :



met $T_{k,l}$ een interval dat begint aan het begin van dag k en eindigt op het einde van dag l

Hoe ziet nu het netwerk eruit??

•Naast source s en sink t is er een knoop voor elke job (job-knoop) en periode (periode-knoop).

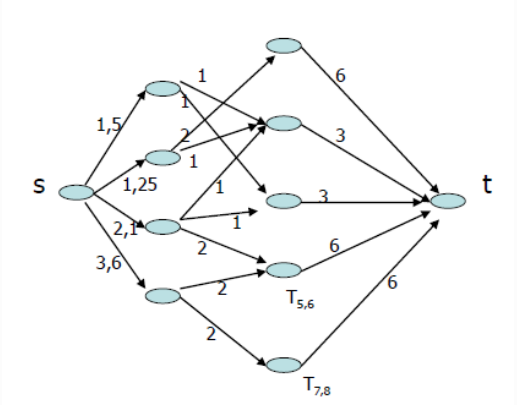
•Er is een pijl van de source naar elke job-knoop j met capaciteit pj.

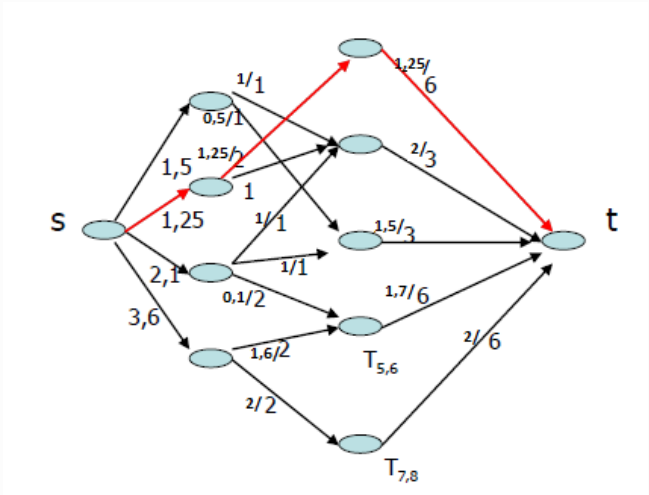
•Er is een pijl van elke periode-knoop $T_{k,l}$ naar de sink met capaciteit 3 (l –k + 1).

•Er is een pijl van job-knoop j naar periode-knoop $T_{k.l}$ als rj ≤ k EN dj ≥ l + 1. De capaciteit bedraagt dan: l –k + 1.

Merk op: binnen elk interval verandert de set van beschikbare jobs niet!

Een stroom ter waarde van p1+ p2+ p3 + p4 komt overeen met een toelaatbare productieplanning en andersom





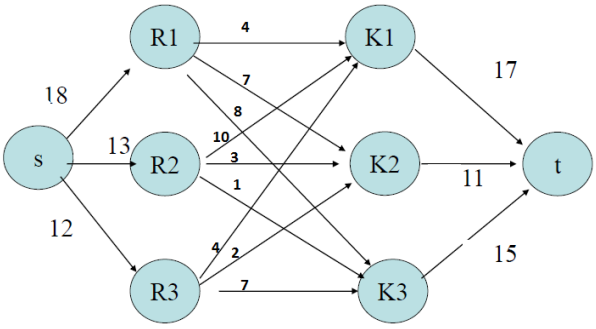
Toepassing 2

U krijgt een twee-dimensionale tabel, met rijsommen en kolomsommen. De waarde van de elementen van de tabel zijn fractioneel. Het is onze taak om elk element of naar boven of naar beneden af te ronden, maar…

Het moet zo gebeuren dat de som van de afgeronde elementen in een rij (kolom) gelijk is aan de afgeronde rijsom (kolomsom)

Vb. Naar boven afronden

3.1	6.8	7.3	17.2
9.6	2.4	0.7	12.7
3.6	1.2	6.5	11.3
16.3	10.4	14.5	



Toepassing 3

Stel een wedstrijd eindigt alleen in winst of verlies. We zeggen dat een team geëlimineerd is, wanneer er, voor elk mogelijk vervolg van de competitie, altijd een team met meer winstpartijen is.

wi: aantal winstpartijen van team i tot nu toe

gij: aantal nog te spelen wedstrijden tussen team i en team j.

Laat team 0 ons team zijn. Dan is het maximale aantal te behalen winstpartijen: W = w0+ Σj g0j.

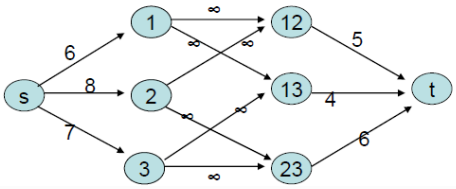
Observatie: wil team i ons **niet** uitschakelen, dan mag het aantal komende winstpartijen van team i niet meer zijn dan W -wi.

- Polen 11 gespeeld, 21 punten
- Finland 11 gespeeld, 19 punten
- Portugal 10 gespeeld, 19 punten
- Servië 9 gespeeld, 15 punten
- België 9 gespeeld, 8 punten
- Armenië 8 gespeeld, 8 punten
- Kazachstan 10 gespeeld, 7 punten
- Azerbeidjan 8 gespeeld, 5 punten

Kan Belgie zich nog kwalificeren?

Het netwerk: maak een knoop voor elk paar {i,j}, i,j ≠ 0, en een knoop voor elk team i, i ≠ 0. Verbind knoop i en knoop j met knoop {i,j}, met een arc met capaciteit ∞. Verbind s met knoop i met een arc met capaciteit W -wi; verbind knoop {i,j} met t met een arc met capaciteit gij.

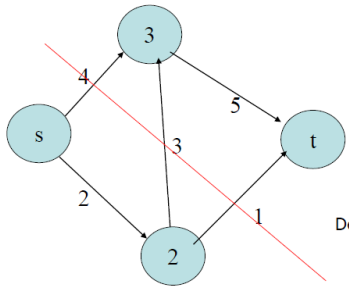
g_{ij}	0	1	2	3
0	0	3	2	1
1	3	0	5	4
2	2	5	0	6
3	1	4	6	0
$W-w_i$		6	8	7



Extra toepassingen in boek: Hoeveel olie kan er verscheept worden per uur via een pijplijn van s ->t
Hoeveel verbindingsvluchten kunnen kunnen er dagelijks gemaakt worden tussen steden.
Maximaliseren aantal overeenkomende koppels volgens compabiliteit. Pg 422
Families verdelen aan bepaalde tafels of in bepaalde autos met beperkingen.
??

5.3. Augmenting Path Algoritme

s-t : Snede: een partitie van V in twee deelverzamelingen S en T met s in S en t in T
De capaciteit van een snede: som der capaciteiten van pijlen van S naar T. Dus $u[S,T] = \Sigma_{i \text{ in } S, j \text{ in } T} u_{ij}$



S = {s,2}; T = {3,t}.

De capaciteit van deze snede bedraagt:

4 + 3 + 1 = 8

De waarde van elke toelaatbare stroom is altijd kleiner of gelijk aan de capaciteit van elke snede in het netwerk. Laat x een stroom in het netwerk zijn, en (S,T) een snede. V is de stroom.

$$V = \sum_{i \text{ in } S} [\sum_{j:\,(i,j) \text{ in } A} x_{ij} - \sum_{j:\,(j,i) \text{ in } A} x_{ji}] =$$
$$= \sum_{(i,j) \text{ in } (S,T)} x_{ij} - \sum_{(i,j) \text{ in } (T,S)} x_{ij} \leq$$
$$\leq \sum_{(i,j) \text{ in } (S,T)} u_{ij} = u[S,T]$$

De maximale stroom is kleiner of gelijk aan de minimale capaciteit van de snede: zwakke dualiteit

Residu netwerk G(x):
$r_{ij} = u_{ij} - x_{ij} + x_{ji}$
"Residu-capaciteiten zijn capaciteiten TEN OPZICHTE VAN gegeven stroom"

```
Algoritme augmenting path
begin
    x:=0;
    while G(x) contains a directed path from s to t do
    begin
        identify an augmenting path P from s to t;
        d := min {r_ij: (i,j) in P};
        augment d units of flow along P;
        update G(x);
    end;
end.
```

5.4. Labeling Algoritme

```
Algoritme labeling
begin
    label node t;
    while t is labeled do
    begin
        unlabel all nodes; label node s; LIST:={s}
        while LIST is not nonempty or t is unlabeled do
        begin
            remove a node i from LIST
            for each arc (i,j) in the residual network do
            if r_ij > 0 and node j is unlabeled then
                label j; LIST:=LIST + {j};
        end;
        if t is labeled then augment;
    end;
end.
```

5.4.1. Correctheid van het algoritme

In een iteratie van het labeling algoritme wordt of sink t gelabeld, of stopt het algoritme.
Aantonen: als het algoritme stopt de resulterende flow maximaal is. Hoe doen we dat?
Door een snede te laten zien die een capaciteit heeft overeenkomend met de waarde van de huidige stroom.
Laat de verzameling nu gelabelde knopen Q zijn. We weten dat voor elke knoop i in Q, j in V\Q geldt $r_{ij} = 0$.
$r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$.

⇨ $x_{ij} = u_{ij}$, en $x_{ji} = 0$ voor alle i in Q, j in V\Q.

Kies S gelijk aan Q en vul in:

$$V = \sum_{(i,j) \text{ in } (S,T)} x_{ij} - \sum_{(i,j) \text{ in } (T,S)} x_{ij} =$$
$$= \sum_{(i,j) \text{ in } (S,T)} u_{ij} = u[S,T]$$

De waarde van een maximale stroom komt overeen met de waarde van een minimale snede.
Waar of niet waar?
Als we de capaciteit van elke pijl met een getal p ($p \geq 0$) vermenigvuldigen of verhogen, blijft de minimale snede onveranderd.
Als we het netwerk aanpassen zodanig dat voor elke pijl (i,j) in A de pijl (j,i) toegevoegd wordt aan het netwerk (met dezelfde capaciteit als de pijl (i,j)), dan blijft de maximale stroom onveranderd.

5.5. Ford Fulkerson Methode

Gegeven niet optimale stroom, hoe kunnen we dan de stroom verhogen tot optimale stroom?

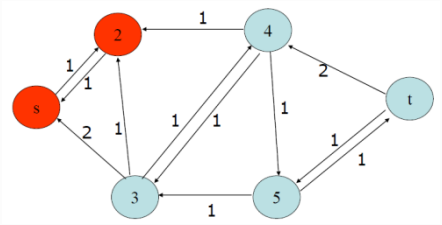
1. I : Set flow(i,j) is kleiner dan maxflow (i,j)
2. R: Set flow(I,j) kan verminderd worden: >0

Label Source

Label Knopen en Bogen op de volgende manier:
Forward arc: Indien knoop x gelabeled is, knoop y niet en(x,y) ∈ I: label (x,y) en y
Backward ard: Indien knoop x gelabeled is, knoop y niet en (y,x)∈R: label (y,x) en y
Stop indien t gelabelled is of er zijn geen andere bogen meer te labellen.

1. C bestaat uit forward arcs.
   Verhoog de flow door het netwerk met k eenheden $k = min_{(x,y)\in C} i(x,y)$
2. C bestaat uit forward and backward arcs.
   Verhoog de flow door het netwek met k=min(k_1,k_2)
   Met $k_1 = min_{(x,y)\in C\cap I} i(x,y)$ $k_2 = min_{(x,y)\in C\cap R} r(x,y)$, dus verhoog de flow door de forward arcs en verlaag de flow door de backward arcs met k

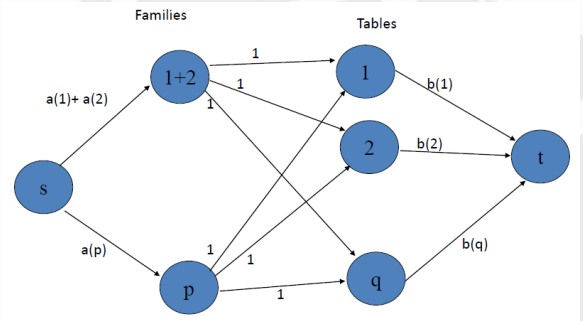
Om de optimaliteit te testen: neem een snede.

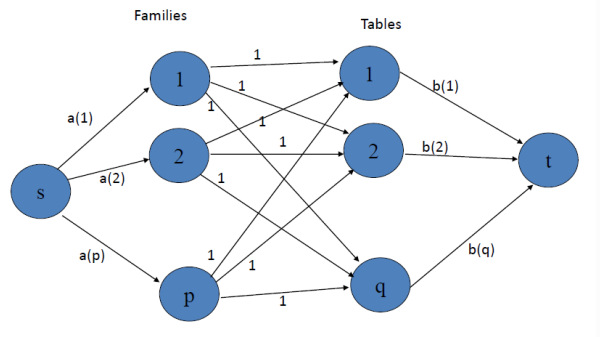


Welke punten kunnen we vanuit s bereiken?
Alleen punt 2; dat levert een snede.
S={s,2}; T={3,4,5,t}. Wat is de capaciteit van deze snede?
De capaciteit van deze snede bedraagt 3; en dat is NIET TOEVALLIG gelijk aan de waarde van de maximale stroom!

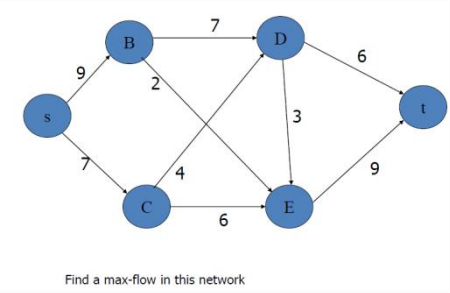
5.6. Oefeningen

Stel je organiseert een fund-raising event waarbij een aantal families worden uitgenodigd waaronder de Gotti familie, en de Soprano familie. Vanwege in het verleden onstane moeilijkheden wil je een tafelschikking vinden zodanig dat geen twee leden van dezelfde familie aan dezelfde tafel zitten. Er zijn p families, en q tafels. Neem verder aan dat familie i uit a(i) personen bestaat (i=1,…,p), en dat tafel j b(j) stoelen (j=1,…,q) heeft.
(i) Modelleer het probleem om een toelaatbare tafelschikking te vinden als een max-flow probleem
(ii) De uitkomst bevalt je niet. Vandaar dat je de volgende additionele beperking toevoegt: geen lid van de Gotti familie mag aan tafel komen met een lid van de Soprano familie. Kun je het resulterende probleem nog steeds als een max-flow probleem formuleren?
(iii) En wanneer je eist: aan elke tafel waaraan een Gotti familielid aanschuift, moet ook een Soprano familie-lid aanschuiven. Is het resulterende probleem te modelleren als een max-flow probleem? Nee



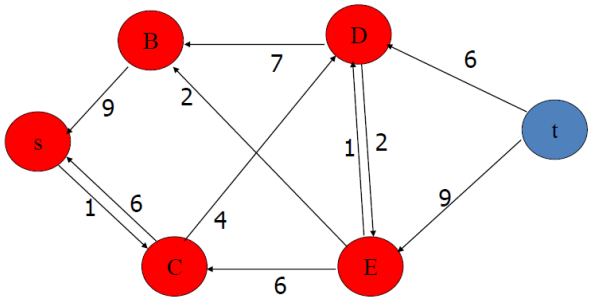


Vind een Maxflow in het volgende netwerk:



Find a max-flow in this network

Path 1: s – B – D – t, amount: 6, residual network
Path 2: s – C – E – t, amount: 6, residual network
Path 3: s – B – E – t, amount: 2, residual network
Path 4: s – B- D – E – t, amount: 1, residual network. Now what?



We have found a maximal flow of value 15. What does the minimal cut look like?

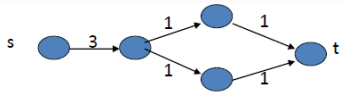
Given is a network (V, A), with a source s, and a sink t in V, and with capacities uijfor each {i,j} in A. A flow x is called an even flow when xij is an even number for all {i,j} in A. A flow x is called an odd flow when xij is an odd number for all {i,j} in A.
True or False?

(i) If all capacities uij are even numbers then there exists an even maximum flow.

TRUE (consider the algorithm, and observe "even –even = even")

(ii) If all capacities uij are odd numbers then there exists an odd maximum flow.

NOT TRUE:



If you choose "false": give a counterexample, if you choose "true": give an argument

6. Het Min Cost Flow probleem (MCNFP)

Het Min Cost Flow probleem generaliseert zowel kortste pad als max-flow.
Gegeven een netwerk N=(V,A), en gegeven getallen (capaciteit) uij≥ 0 en cij≥ 0 voor alle {i,j} in A en gegeven een vraag cq aanbod b(i) voor elke node i, vind een min cost flow die aan alle vraag en aanbod voldoet. De bogen hebben hier een kost en een capaciteit.

x_{ij} = *aantal eenheden stroom over arc {i,j}, voor alle {i,j} in A*

Minimaliseer $\sum_{(i,j) \in A} c_{ij} x_{ij}$

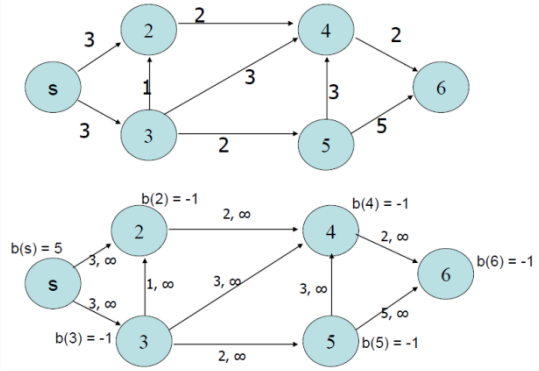
Onder de beperkingen:

$$\sum_{\{j:(i,j) \in A\}} x_{ij} - \sum_{\{j:(j,i) \in A\}} x_{ji} = b(i) \quad \text{voor alle } i$$

$$0 \le x_{ij} \le u_{ij} \quad \text{voor alle } (i,j) \text{ in } A$$

We bepalen of een min-cost flow instantie een oplossing heeft door het oplossen van een max-flow probleem.
Beschouw het min-cost flow netwerk.
1.Voeg een source s en een sink t toe.
2.Voor elke knoop i met b(i) > 0 voeg een pijl (s,i) toe met capaciteit b(i).
3.Voor elke knoop i met b(i) < 0 voeg een pijl (i,t) toe met capaciteit b(i).
4.Los nu een max-flow probleem op.
Als alle pijlen uit s `gevuld zijn" dan bestaat er een toelaatbare oplossing in het min-cost flow probleem, anders niet.

Verband met kortste pad:



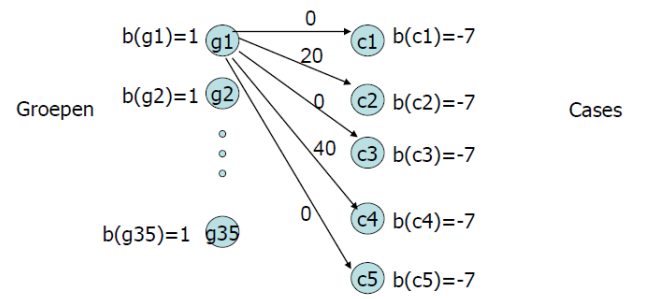
6.1. Toepassingen

Toepassing 1

Toewijzen van studentengroepen aan cases

Er zijn 35 groepen en 5 cases. Elke groep heeft aangegeven hoe graag ze een case wil doen met een coëfficiënt w_{ij}, waarbij i=1,…,35, en j=1,…,5.

Om dit probleem als min-cost flow te modelleren moeten we dus een netwerk bouwen, met punten, pijlen, kosten (of winsten), en capaciteiten. Dat is NIET hetzelfde als een IP-formulering opschrijven.



CASE	GROEPEN
1	4,10,15,25,27,29,33
2	1,12,23,24,28,31,32
3	3,6,11,20,30,34,35
4	5,9,14,18,21,22,26
5	2,7,8,13,16,17,19

De getallen stellen de winst-coëfficiënten voor;
de capaciteiten staan niet vermeld, en zijn gelijk aan 1

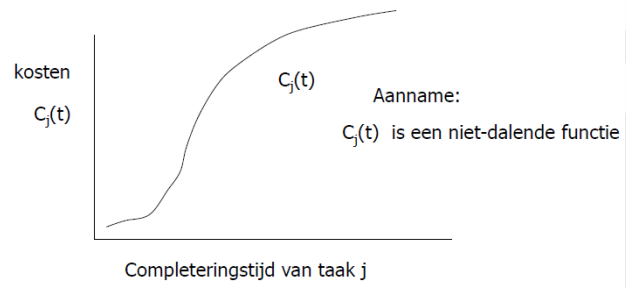
Met doelstellingswaarde: 991; 991/35 = 28,31…

Toepassing 2

scheduling with deferral costs

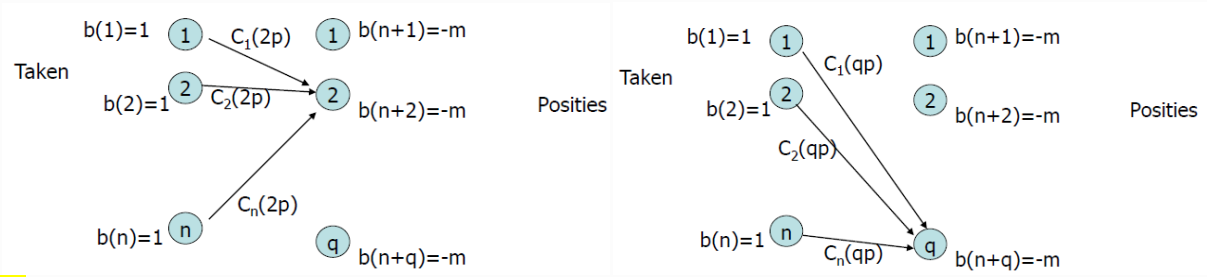
Er zijn n taken en m machines. Elke taak duurt p tijdseenheden (onafhankelijk van de machine), en moet door 1 machine bewerkt worden.

Neem (voor het gemak) aan dat n een veelvoud is van m, maw n/m=q= aantal taken per machine.



Hoe kunnen we nu de taken toewijzen aan de machines zodanig dat de totale kosten minimaal zijn?

Merk op: geen `idle time", en elke machine krijgt q taken.



??

Toepassing 3:

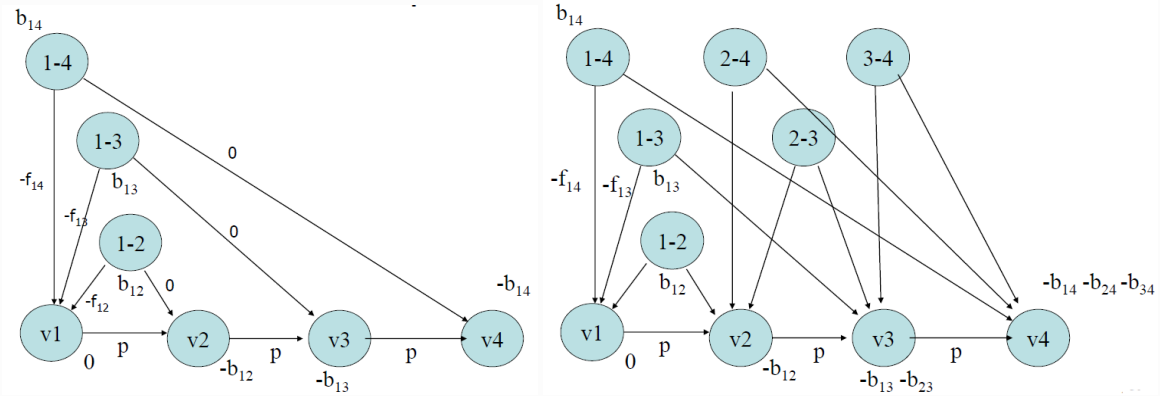
Maximaliseer de opbrengsten van uw "hopping airplane"

Beschouw 1 passagiersvliegtuig die ten hoogste p passagiers kan meenemen op een vlucht die steden 1,2,3,…,n aandoet. Op elke luchthaven kunnen passagiers zowel in-als uitstappen.

Laat b_{ij}: aantal passagiers beschikbaar op luchthaven i die naar luchthaven j willen

f_{ij}: prijs van een ticket van luchthaven i naar luchthaven j

Hoe bepaal je het aantal passagiers dat in mag stappen op elke destinatie, waarbij je zoveel mogelijk geld wil verdienen?



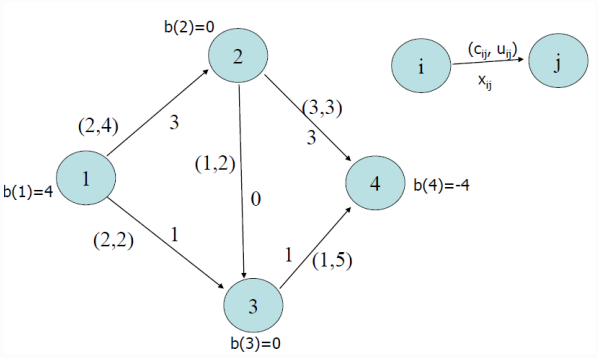
Verklaar: een strategie om passagiers in het vliegtuig te maximaliseren komt overeen met een maximale stroom in het netwerk.

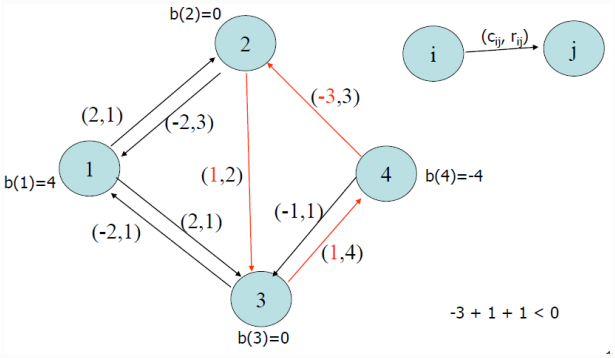
??

Extra toepassingen pg 452: Verkeerdoorgang in een bepaald gebied op een bepaald uur.

6.2. Cycle Canceling Algoritme

```
Algoritme cycle-canceling
begin
    establish a feasible flow x in the network
    while G(x) contains a negative cycle do
    begin
          use some algorithm to identify a negative cycle C;
          d:=min {r_ij: (i,j) in C};
          augment d units of flow along C;
          update G(x);
    end;
end.
```





6.3. Planning the assignment of telephone calls: a case study at Vodafone

The price of a share Mobistar / KPN
in the second half of 2000: €40./ in February2000: over €127.
Summer 2002: about €10. / September 2001: less than €3.
Current price (26/10/10): €46.84. / Current price (26/10/10): about €11.84

De doelstellingen zijn kostenreductie en verbetering van efficiëntie, door het beter realiseren schaalvoordelen van synergieen, beter beheer van kosten, (belgacom, telefonica en telekom). Het gebied van het probleem, waar we rekening mee moeten houden zijn het telecombedrijf, de internationale carriers (examples: Worldcom, Cable + Wireless), de bestemmingen en de budgetrestrictie.
De prijs die een telecombedrijf moet betalen aan een carrier hangt af van de bestemming, de duurtijd, piek of geen piektijd, mobile of vaste telefoon en de tariefstructuur.
De tariefstructuur hangt af van de prijs die de carrier zet per maand voor een minuut bellen naar alle bestemmingen, deze hangt af van de jaarlijkse volumes van die carrier voor het bepaalde telecombedrijf. We nemen aan dat de periode dezelfde is voor alle carriers

Example: We have a single period, two carriers, one of which has two intervals, and two destinations:

	KPN		C+W	Forecast
	0-1.500.000;	1.500.000-∞	0 - ∞	
Rome	0.3 ;	0.2	0.25	1.000.000
Rio	0.6 ;	0.4	0.35	1.000.000

De gegevens: de maandelijkse voorspellingen van aantal minuten voor alle bestemmingen tot en met 31 dec, de verkoop van de carrier tot nu toe en de prijzen voor elke bestemming, voor elk interval van elke carrier, beslissen welke carrier de calls krijgt toegewezen van welke bestemming.
De ingewikkelde tariefstructuur maakt dat toekomstige verkoop van de carrier invloed heeft op de huidige kosten van het telecombedrijf.

The parameters, where d is an index for destinations(5000), c is an index for carriers(5-10), m is an index for months(12), i is an index for intervals(4):
P_{dcmi}: price per minute of a call carried out by carrier c to destination d in month m, when yearly amount falls in interval i,
F_{dm}: forecast of number of call minutes in month m to destination d,
U_{ci}: Upper bound of interval i of carrier c,
L_{ci}: Lower bound of interval i of carrier c,
The decision variables:
X_{dcmi}: number of call minutes in month m to destination d to be carried out by carrier c in interval i,
Y_{ci}= 1 if total number of call minutes to be carried out by carrier c is in interval i, 0 elsewhere
(dit zijn 5000 * 10 * 4 * 12 ≥ 2.000.000 variabelen)

Minimize costs:

Minimize $\sum_{dcmi} P_{dcmi} X_{dcmi}$

Subject to

Forecast:

$\sum_{ci} X_{dcmi} = F_{dm}$ $\forall$ d,m

Carrier-interval restrictions:

$\sum_{dm} X_{dcmi} \leq Y_{ci} U_{ci}$ $\forall$ c,i

$\sum_{dm} X_{dcmi} \geq Y_{ci} L_{ci}$ $\forall$ c,i

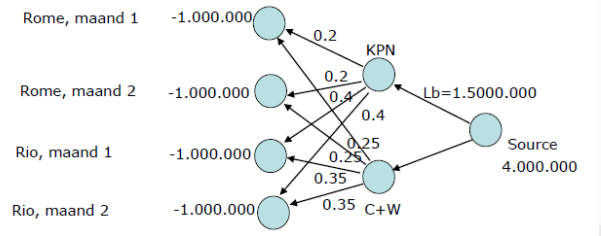
$\sum_{i} Y_{ci} = 1$ $\forall$ c

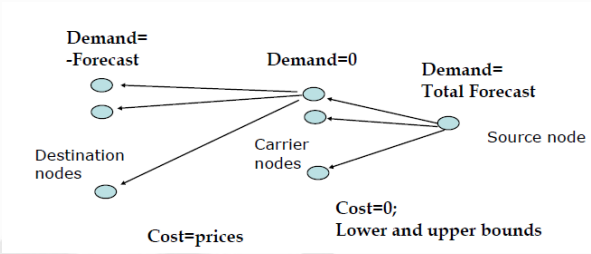
Integrality:

$Y_{ci} \in \{0, 1\}$ $\forall$ c,i

Omwille van het grote aantal variabelen, kiezen we een interval voor elke carrier, dit maakt van ons probleem eem min cost flow probleem: We hebben een knoop voor elke bestemming/maand met de voorspelde vraag: $-F_{dm}$, een knoop voor elke carrier met vraag 0, een sourcenode met supply: $sum_{dm}(F_{dm})$, een boog van de source naar iedere carrierknoop met kost 0 en de juiste upper en lower bounds, een boog van iedere carrier naar iedere bestemming/maand knoop met de kost gelijk aan de overeenkomstige prijs.

	KPN 0-1.500.000; 1.500.000-∞		C+W 0 - ∞	Forecast (maand 1 en maand 2)
Rome	0.3 ;	0.2	0.25	1.000.000
Rio	0.6 ;	0.4	0.35	1.000.000





Voor alle mogelijke manieren om intervals te selecteren een nieuw min-cost flow probleem oplossen, dit duurt ong. 3 minuten.

Other issues: Quality of Service, Brokers, Sensitivity analysis,Lower and upper bounds, both monthly and yearly, Implementation

Voordelen: Structured decision making, Getting insight into quality versus cost, Being able to get better prices

6.4. Dualiteit lineair optimaliseren

- Max $\sum_j c_j x_j$

  $\sum_j a_{ij} x_j \leq b_i$ for all i

  $x_j \geq 0$ for all j

- Min $\sum_i b_i y_i$

  $\sum_i a_{ij} y_i \geq c_j$ for all j

  $y_i \geq 0$ for all i

Het voorgaande werkt voor een lineair optimalisatieprobleem in standaard vorm
Wat gebeurt er wanneer er gelijkheden zijn? De duale variabelen moeten niet langer niet-negatief zijn! Tevoren, met ongelijkheden in de primaal, en een negatieve y-variabele, zou het teken omkeren (≤ wordt ≥), en krijgen we niet langer een toelaatbare bovengrens voor z. Nu kunnen we zonder probleem negatieve y‟s gebruiken en toch een toelaatbare bovengrens voor z bekomen.

Primaal: Max	$\Sigma^n_{j=1} c_j x_j$	
s.t.	$\Sigma^n_{j=1} a_{ij} x_j = b_i$	for i =1,2,…,m
	$x_j \geq 0$	for j=1,2,…,n
Duaal: Min	$\Sigma^m_{i=1} b_i y_i$	
s.t.	$\Sigma^m_{i=1} a_{ij} y_i \geq c_j$	for j =1,2,…,n

Wat gebeurt er wanneer er variabelen zijn zonder niet-negativiteitseis?
Wanneer de x-variabelen onbeperkt zijn, en dus ook negatief kunnen zijn, moeten we ervoor zorgen dat de coëfficiënten gevormd door onze lineaire combinatie van de beperkingen, exact overeenkomen met de coëfficiënten van de doelfunctie

Primaal: Max	$\Sigma^n_{j=1} c_j x_j$	
s.t.	$\Sigma^n_{j=1} a_{ij} x_j \leq b_i$	for i =1,2,…,m
Duaal: Min	$\Sigma^m_{i=1} b_i y_i$	
s.t.	$\Sigma^m_{i=1} a_{ij} y_i = c_j$	for j =1,2,…,n
	$y_i \geq 0$	for i =1,2,…,m

Niet-negatieve variabelen in de primaal corresponderen met ongelijkheden in de duaal,
Onbeperkte variabelen in de primaal corresponderen met gelijkheden in de duaal.
Ook:
Ongelijkheden in de primaal corresponderen met niet-negatieve variabelen in de duaal
En gelijkheden in de primaal corresponderen met onbeperkte variabelen in de dual

6.4.1. Zwakke dualiteit

Theorem: De waarde van elke toelaatbare duale oplossing vormt een bovengrens voor de waarde van elke toelaatbare primale oplossing.
Bewijs: Zorg ervoor dat x en y toelaatbare oplossingen zijn. Dan geldt:

$$\Sigma^n_{j=1} c_j x_j \leq \Sigma^n_{j=1} (\Sigma^m_{i=1} a_{ij} y_i) x_j =$$
$$= \Sigma^m_{i=1} \Sigma^n_{j=1} a_{ij} x_j y_i \leq \Sigma^m_{i=1} b_i y_i$$

6.4.2. Complementory Slackness

Beschouw een optimale oplossing voor de primaal x*en een optimale oplossing voor de duaal y*.
Dan geldt:

$y^*_i = 0$ als $\Sigma_j a_{ij} x^*_j < b_i$ for all i
$\Sigma_i a_{ij} y^*_i = c_j$ als $x^*_j > 0$ for all j

Beschouw een toelaatbare oplossing x en een toelaatbare oplossing y. Noodzakelijke en voldoende voorwaarden voor optimaliteit zijn:

$y_i = 0$ of $\Sigma_j a_{ij} x_j = b_i$ (of beiden) for all i, en
$\Sigma_i a_{ij} y_i = c_j$ of $x_j = 0$ (of beiden) for all j

6.4.3. Sterke dualiteit

De waarde van de beste duale toelaatbare oplossing is hetzelfde als de waarde van de beste primale toelaatbare oplossing dus de waarde van beide optima zijn gelijk.

6.5. Oefeningen

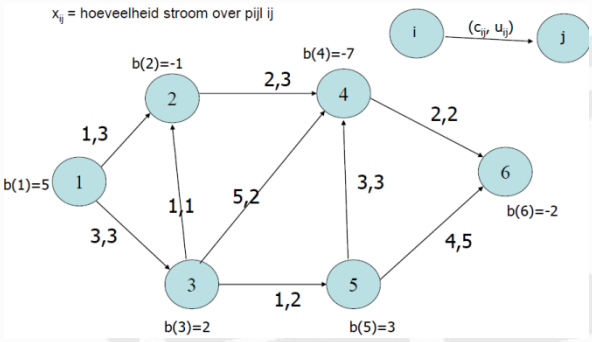
Question: is the solution x = (0, 4/3, 2/3, 5/3, 0) optimal? (Do NOT use the Simplex-method)

Max z= 7x1+ 6x2+ 5x3-2x4+ 3x5 Odb x1+ 3x2+ 5x3-2x4+ 2x5≤ 4 4x1+ 2x2-2x3+ x4+ x5≤ 3 2x1+ 4x2+ 4x3-2x4+ 5x5≤ 5 3x1+ x2+ 2x3-x4-2x5≤ 1 x1, x2, x3, x4, x5≥ 0	Max z= 4y1+ 3y2+ 5y3+y4 Odb y1+ 4y2+ 2y3+3y4≥ 7 3y1+ 2y2+4y3+ y4≥ 6 5y1-2y2+ 4y3+2y4≥ 5 -2y1+y2 -2y3-y4≥ -2 2y1+y2+ 5y3-2y4≥ 3
We vullen primale oplossing in Bep 1: 4 ≤ 4 Bep 2: 3 ≤ 3 Bep 3: 14/3 < 5. Conclusie y3 = 0. Bep 4: 1 ≤ 1	Indien xi niet gelijk is aan nul moet complementory slackness beperking voldaan zijn. x2 > 0: 3y1 + 2y2 + 4y3 + y4 = 6 x3 > 0: 5y1 - 2y2 + 4y3 + 2y4 = 5 x4 > 0: -2y1 + y2 - 2y3 - y4 = -2

Uiteindelijk: y = (1,1,0,1). Is dit een toelaatbare duale oplossing?
y1+ 4y2+ 2y3+3y4≥ 7 : 1+4+0+3 ≥ 7 oke
2y1+y2+ 5y3-2y4≥ 3 : 2+1+0-2 ≥ 3 -> niet oke : geen toelaatbare oplossing : geen optimale oplossing.

ii) What is the dual of:
Max cx
Odb Ax ≤ b

De duaal is:
Min yb
Odb yA = c
y ≥ 0



Suppose that in a "min-cost flow like" problem supply exceeds demand: Σib(i) > 0.
However, in this problem, it holds that for a vertex i with b(i) > 0 at most b(i) units of flow are sent into the network. Formulate the resulting problem as an „ordinary" min-cost flow problem.

Creer een nieuwe „demand-node" met een vraag zdd totale demand is -Σb(i) (dus som der aanbod en vraag is nu gelijk aan 0). Introduceer een arc van elke aanbod-knoop naar deze nieuwe knoop, met kost: 0 en capaciteit: M. Los het resulterende min-cost flow probleem op.

7. Column Generation

7.1. Cutting Stock Problem

Width of each jumbo-roll: W
Number of orders: n
Width of requested roll i: wi
Which *set of rolls* or pattern is used to cut a single jumbo-roll?
Available: an unlimited number of jumbo-rolls with a width (W) of 100 cm
97 rolls with width 45 cm
610 rolls with width 36 cm
395 rolls with width 31 cm
211 rolls with width 14 cm
For this example, we can distinguish j=37 different patterns. i.e.(2 0 0 0).

a_{ij}= number of rolls of order i in pattern j
bi= requested multiplicity of order I b = (97 610 395 211)
xk: number of jumbo-rolls to be produced according to pattern k, k=1, …, 37.

(P) Minimize $\sum_k x_k$

subject to $\sum_{k=1} a_{ik} x_k = b_i$ for each i

x_k integral for each k

7.1.1. Solution method

- Los de LPrelaxatie op en vorm de fractionele om tot een volledige oplossing.
- Rond af naar beneden (voor andere problemen branch and bound,zie verder)
- b – b(geproduceerd)= rest => manueel een oplossing zoeken voor de overschot maar deze past normaal gezien in 1 rol. Deze oplossing is steeds optimaal gezien de LPrelaxatie tight is, dwz, nauw aansluit bij de IPformulering: kleine gap!

7.2. Column Generation

7.2.1. Op cutting stock

Het kan zijn dat het aantal patronen al gauw heel groot wordt!
Een iteratieve procedure die in iedere stap een nieuwe variabele maakt, hierdoor worden niet alle variabelen gebruikt tegelijkertijd. Hier voegen we steeds een patroon toe en gaan we na , aan de hand van de duale variabelen of we een volgend patroon moeten toevoegen.

Min $\sum_k x_k$
$\sum_{k=1} a_{ik} x_k = b_i$ for each i
$x_k \geq 0$ for each k

The dual: Max $\sum_i b_i y_i$
$\sum_{i=1} y_i a_{ik} \leq 1$ for each k

Indien we de yi variabelen kennen, moeten we om de optimaliteit van de x-oplossing aan te tonen enkel nog nagaan of oplossing y mogelijk is. Bestaat er nog een patroon zodat:
$\sum_i w_i a_i \leq W$,
$\sum_{i=1} y_i a_i > 1$,
the a_i are non-negative integral numbers

Indien we het volgende probleem kunnen oplossen, moeten we een nieuw patroon toevoegen, de oplossing ai:

$$\text{Max } \sum_{i=1} y_i a_i$$
$$\text{st } \sum_i w_i a_i \leq W,$$
$$a_i \text{ integral for each } i$$

7.3. Algemeen

- Consider this primal formulation

$$\text{Max } \Sigma_j c_j x_j$$
$$\text{st } \Sigma_j a_{ij} x_j \leq b_i \quad \text{for all } i=1,\ldots,m$$
$$x_j \geq 0 \quad \text{for all } j=1,\ldots,n$$

And its dual:

$$\text{Min } \Sigma_i b_i y_i$$
$$\text{st } \Sigma_i a_{ij} y_i \geq c_j \quad \text{for all } j=1,\ldots,n$$
$$y_i \geq 0 \quad \text{for all } i=1,\ldots,m$$

Indien de primaire notatie veel variabelen heeft (n) dan zullen er maximaal m verschillend van nul zijn in de optimale oplossing omwille van de beperkingen. (dit resultaat komt uit de simplexmethode met dictionairs). We kennen deze m variabelen niet.
We kiezen m primaire variabelen en we lossen het LP op **(restricted masterproblem)** en we kijken of de overeenkomstige duale oplossing een oplossing is in het duale probleem. **(The pricing problem)**
Ja-> Optimale oplossing omwille van strong duality
Nee-> Een duale beperking is overtreden: deze beperking komt overeen met een primaire variabele. We passen de set van m variabelen aan zodat het deze primaire variabele bevat.

7.4. Toepassingen

Consider the following problem: given are m machines and n jobs. When processing a job on some machine, the job needs pj time-units, and you obtain a profit of €wj. Each machine has b time-units available.
The problem is to find an assignment of jobs to machines that is feasible (with respect to the time-units) and which makes you a maximum amount of money.
(i) Formulate this problem as an integer programming problem using binary variables xji= 1 if and only if job j is assigned to machine i.
(ii) Formulate this problem as an integer programming problem using binary variables z_S if job-set S is selected to be processed on some machine.

- $\text{Max } \Sigma^m_{i=1} \Sigma^n_{j=1} w_j x_{ji}$
  $\text{st } \Sigma_i x_{ji} \leq 1$ for each j=1,…,n
  $\Sigma_i p_j x_{ji} \leq b$ for each i=1,…,m
  x_{ji} in {0,1} for each i,j

$$\text{Max } \Sigma_S w_S z_S$$
$$\text{st } \Sigma_{S:\, j \text{ in } S} z_S \leq 1 \quad \text{for each } j=1,\ldots,n$$
$$\Sigma_S z_S = m$$
$$z_S \text{ in } \{0,1\} \quad \text{for each } S$$

Consider the following problem: given are K vehicles, each with capacity Q, and n locations, each with a demand di. There is a distance cij given for each pair of locations. The vehicles start their routes from a central depot which is called location 0. The problem is to find a route for each vehicle such that (i) each location is visited, (ii) capacity of vehicles is not exceeded, and (iii) total distance traveled by all vehicles is minimal.

(i) Formulate this problem as an integer programming problem using binary variables x_{ijk}= 1 if and only if vehicle k travels from location i to location j.
(ii) Formulate this problem as an integer programming problem using binary variables z_S if location-set S is visited by a single vehicle.

$$\text{Max } \sum_k \sum_i \sum_j c_{ij} x_{ijk}$$
$$\text{st } \sum_j \sum_k x_{0jk} = K$$
$$\sum_i \sum_k x_{ijk} \le 1 \quad \text{for each } j=1,\ldots,n,$$
$$\sum_j \sum_k x_{ijk} \le 1 \quad \text{for each } i=1,\ldots,n,$$
$$\sum_i \sum_j d_i x_{ijk} \le Q \quad \text{for each } k$$
subtour elimination
$$x_{ijk} \text{ in } \{0,1\} \quad \text{for each } i,j,k$$

- Consider an ordered sequence of locations S such that:

$\sum_{j \text{ in } S} d_j \le Q$. For each such set S, we define a variable z_S, and we define $c_S = \sum_{j \text{ in } S} c_{i,i+1}$

- $\text{Min } \sum_S c_S z_S$

$$\text{st } \sum_{S:\, j \text{ in } S} z_S = 1 \quad \text{for each } j=1,\ldots,n$$
$$\sum_S z_S = K$$
$$z_S \text{ in } \{0,1\} \quad \text{for each } S$$

7.5. Branch and Price: IP oplossen

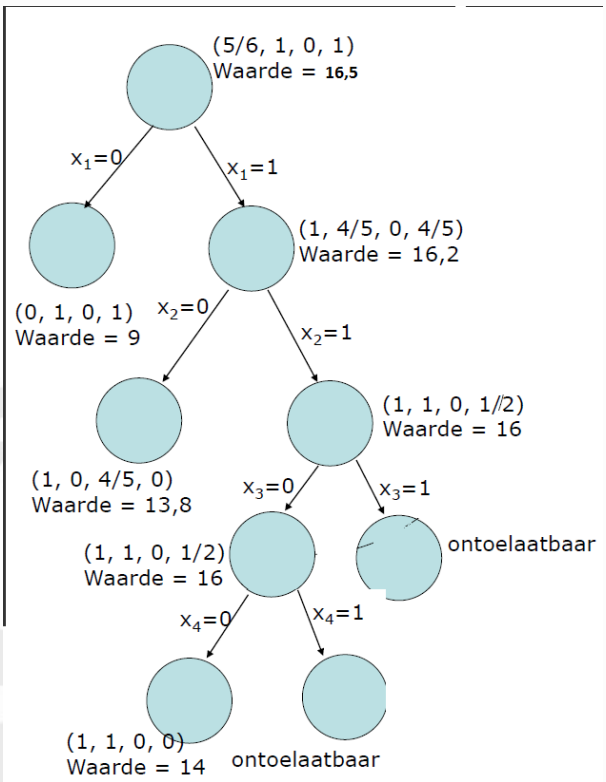
7.5.1. Branch and bound

Twee locaties om distributiecentrum en fabriek te openen, San Francisco (SF) fabriek x1, distributiecentrum x3 en Los Angeles (LA) fabriek x2 en distributiecentrumx4. Er wordt ten hoogste 1 distributie-centrum geopend, en u kunt alleen een distributie-centrum openen als er op die locatie een fabriek staat.

De doelfunctie:
max 9x1+ 5x2+ 6x3 + 4x4
De budget-beperking:
6x1+ 3x2+ 5x3 + 2x4 ≤ 10
De „ten hoogste 1 distributiecentrum"-beperking:
x3 + x4 ≤ 1
Distributie centrum alleen daar waar fabriek:
x3 –x1 ≤ 0
x4 –x2 ≤ 0
x1, x2, x3, x4 ε{0,1}

wanneer x1=1 wordt het probleem:

$$\text{Max } 9 + 5x_2 + 6x_3 + 4x_4$$
$$\text{Odb } 3x_2 + 5x_3 + 2x_4 \le 4$$
$$x_3 + x_4 \le 1$$
$$x_3 \le 1$$
$$x_4 - x_2 \le 0$$
$$x_1, x_2, x_3, x_4 \in \{0,1\}$$



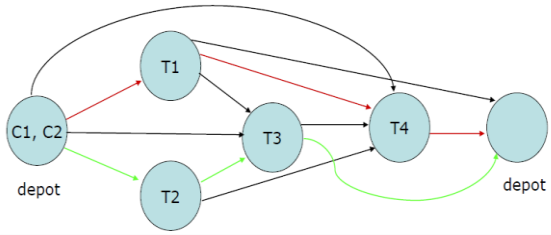
De oplossing van het LP vormt een bovengrens voor de waarde van het IP.
Toepassingen
In boek: pg 512: MIP,Knapzak, Combinatorische problemen, Machine Sceduling, TSP

7.5.2. Branch and price

De combinatie van Branch and bound en kolomgeneratie.
Crew scheduling

- Stel er zijn 4 taken en 2 crews.

	Begin	Eind	cost(C1)	cost(C2)
Taak 1	20	30	5	10
Taak 2	25	30	15	10
Taak 3	35	40	10	15
Taak 4	45	50	5	10



	cost(C1)	cost(C2)
d – 1 – d	5	10
d – 1 – 3 – d	15	25
d – 1 – 4 – d	10	20
d – 1 – 3 – 4 – d	20	35
d – 2 – d	15	10
d – 2 – 3 – d	25	25
d – 2 – 4 – d	20	20
d – 2 – 3 – 4 - d	30	35

- Laat x_{cp} =1 als crew c pad p neemt, 0 elders

Het model:

$$\min \sum_{c,p} cost(c,p)\, x_{cp}$$

Odb $\sum_c \sum_{p:\ \text{taak i in pad p}} x_{cp} = 1$ voor elke taak i

$\sum_p x_{cp} \le 1$ voor elke crew c

$x_{cp} \in \{0,1\}$ voor alle c,p

De duaal

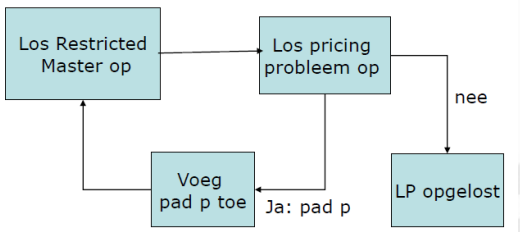
Max $\sum_i u_i + \sum_c e_c$

zdd $\sum_{i:\ \text{taak i op pad p}} u_i + e_c \le cost(c,p)$ voor alle c, p

$e_c \le 0$

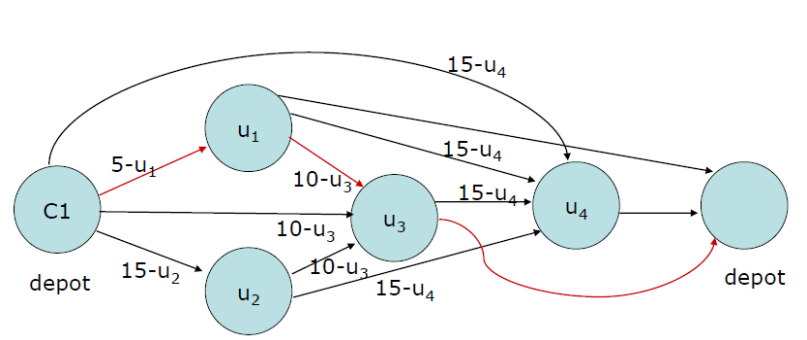
Door het grote aantal paden gebruiken we kolomgeneratie, we beginnen met een (kleine) verzameling paden in beschouwing te nemen en lossen dat LP-model op. (Restricted Master).
De gevonden x-oplossing is optimaal indien de duale variabelen een toelaatbare oplossing vormen voor de duaal , dit weten we uit de dualiteitstheorie.

7.5.3. Pricing probleem



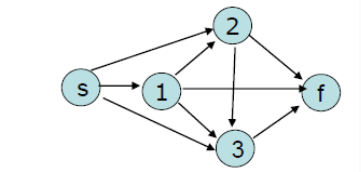
Hoe weet u nu of de gevonden duale oplossing (dus de u-tjes en de e-tjes) een toelaatbare duale oplossing vertegenwoordigen ZONDER alle beperkingen af te lopen?
Beschouw een of andere crew c, en de beperking in de duaal: voor elke crew c en pad p moet gelden:
$\sum_{i:\ \text{taak i op pad p}} u_i + e_c \le cost(c,p)$. (fixeren op 1 crew)
Oftewel, bestaat er een pad p met: $\sum_{i:\ \text{taak i op pad p}} u_i + e > cost(p)$?
Oftewel, is er een pad p met: $cost(p) - \sum_{i:\ \text{taak i op pad p}} u_i < e$?

Beschouw nu een of ander pad (zie hieronder). Wat is de lengte van dat pad?
We maken een netwerk voor crew c1: op elke pijl zetten we de kosten van die taak minus de waarde van de duale variabele van die taak.
5 –u1+ 10 –u3= 15 –u1–u3= cost(d –1 –3 –d) –u1–u3.
Indien we een korter pad vinden in dit network met een lengte die kleiner is dan e, dan moeten we dit pad toevoegen aan de restricted master, anders is de huidige x-oplossing optimaal.



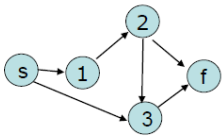
7.5.4. Branching Probleem

We bekomen een oplossing van het LP en deze is fractioneel.
We kunnen hier niet het gewone branch and bound algoritme toepassen, gezien we al dichterbij de optimale oplossing zitten dan sommige andere paden die we niet geselecteerd hebben door het oplossen van het pricing probleem.
Bestudeer het pad p van de fractionele x_{cp}, op dat pad p worden sommige taken achter elkaar uitgevoerd. Beslissingsregel om te vertakken: In een optimale oplossing worden of taak i en taak j direct na elkaar uitgevoerd, of niet.

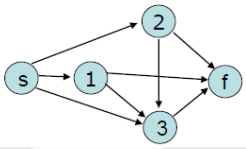


Beschouwde volgende paden: s –1 –2 –f, s –1 –3 –f, s –2 –3 –f, en overtuig uzelf dat deze oplossing toelaatbaar is : $x_{c1,s-1-2-f} = x_{c1,s-1-3-f} = x_{c2,s-2-3-f} = ½$.
Niet eisen: $x_{c1,s-1-2-f} = 1$ OF $x_{c1,s-1-2-f} = 0$
Maar:

Taak 1 en 2 direct na elkaar



Taak 1 en 2 niet direct na elkaar



8. Herhaling aantal problemen

8.1. Traveling Salesman Problem (w9.6)

Geg. n steden, vind een kortste tour die elke stad 1x aandoet, NP-compleet, veel routeringsproblemen zijn schrijfbaar als een TSP, dit betekent niet dat dit probleem dan snel kan opgelost worden.
Symmetrische notatie voor TSP:
Verzameling steden N,
Verzameling paren van steden E={{i,j}:i,j∈ $N, i \neq j$}
Lengte c_e voor elke e∈ E
Zoek de tour met minimale kost.
Niet georiënteerde beslissingsvariabelen: x_e, $\forall e \in E, = \{^{1\ indien\ e\ in\ de\ tour}_{0\ indien\ niet}$

$$\min \sum_{e \in E} c_e x_e$$

$$odb \sum_{i \neq j} x_{\{i,j\}} = 2, i \in N$$

$$\sum_{\{i,j\} \subset S} x_{\{i,j\}} \leq |S| - 1, S \subset N, 2 \leq |S| \leq |N|$$

$$x_e \in \{0,1\}, e \in E$$

Eerste beperking: elke stad maar een keer bezoeken
Tweede beperking: Voor elke mogelijke deelverzamelingen S
Stel subtour 1-2-3-1, dan moet $x_{12}+x_{23}+x_{31} \leq |\{1,2,3\}|-1$ omwille van de subtourbeperking.
Toepassing slide 55 OO.

Asymetrische TSP: MTZformulering slide 67 OO.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$odb \sum_{j=1}^{n} x_{ij} = 1; \sum_{i=1}^{n} x_{ij} = 1$$

$$u_i = 1; voori, j \neq 1, i \neq j : u_j \geq u_i + 1 - M(1 - x_{ij})$$

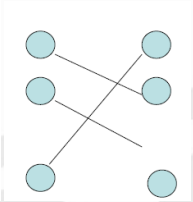
$$x_{ij} \in \{0,1\}$$

8.2. Toewijzingsprobleem

het toewijzingsprobleem: gegeven n taken en n personen, en voor elke combinatie van taak en persoon een winst wij, vind die toewijzing van taken aan personen met maximale winst.

- Laat x_{ij} = 1 als taak i uitgevoerd wordt door persoon j, 0 anders (voor elk paar i, j)

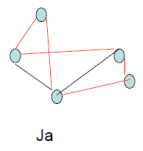
Max $\Sigma_{ij} w_{ij} x_{ij}$
St $\Sigma_i x_{ij} = 1$ voor alle j
$\Sigma_j x_{ij} = 1$ voor alle i
$x_{ij} \varepsilon \{0,1\}$ voor alle i, j



8.3. Hamiltoniaans Circuit

Bekend NP-probleem (zie complexiteit)
Gegeven is een netwerk (V, E). Bestaat er in dit netwerk een Hamiltonian Tour, dwz bestaat er een cycle in het netwerk die elke knoop precies 1 keer aandoet? Best mogelijke algoritme:

Het volgende algoritme lost het hamiltoniaans probleem op in exponentiële tijd:
Genereer alle mogelijke paden in de gerichte graaf.
Filter alle paden eruit die de gewenste beginknoop en eindknoop hebben.
Filter alle paden eruit die exact het aantal knopen bevatten als er knopen zijn in de graaf.
Filter alle paden eruit die elke knoop exact één maal passeren



8.4. Partitie

Partitie is een bekend NP-compleet probleem. A partition of a set *P* is a set of nonempty subsets of *P* such that every element *p* in *P* is in exactly one of these subsets.

Laat P={1,2,…,p}. Gegeven p integers, zeg $s_1, s_2,\ldots,s_p$, bestaat er een deelverzameling S van P zodanig dat

$$\sum_{i \text{ in } S} s_i = \sum_{i \text{ in } P-S} s_i \ ?$$

The problem is to decide whether a given multiset of integers can be partitioned into two "halves" that have the same sum. More precisely, given a multiset *S* of integers, is there a way to partition *S* into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2? The subsets S_1 and S_2 must form a partition in the sense that they are disjoint and they cover *S*. The optimization version asks for the "best" partition, and can be stated as: Find a partition into two subsets S_1,S_2 such that $\max(\text{sum}(S_1), \text{sum}(S_2))$ is minimized (sometimes with the additional constraint that the sizes of the two sets in the partition must be equal, or differ by at most 1).

8.5. Knapzak

Het knapsack probleem: u krijgt n items, en een knapsack. De knapsack heeft capaciteit C, elk item neemt ai capaciteit in. Als u item i selecteert levert u dat wi op.

items	1	2	3	4
a	10	4	7	8
w	20	9	12	18

C=15

OO: slide 10 hoofdstuk 1

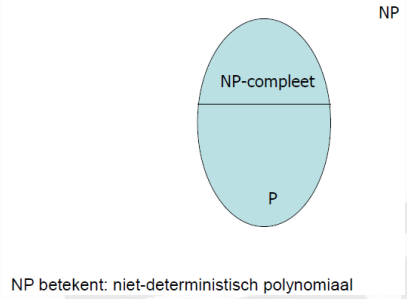
9. Complexiteit

Gaat over de relaties **tussen** problemen en de moeilijkheid **van** problemen.
We stellen de tijdscomplexiteitfunctie van een algoritme op: hiermee beoordelen we de complexiteit aan de hand van een platform onafhankelijke maat. We berekenen het maximale aantal elementaire rekenstappen: optellen, aftrekken, vermenigvuldigen, delen of vergelijken dat een algoritme moet uitvoeren om elke instantie van grootte n op te lossen. De complexiteit wordt bepaald door die instantie waarvoor het meeste aantal stappen worden overlopen.
Er zijn twee soorten complexiteitsfuncties, de polynomiale en de exponentiële, het aantal berekeningen van exponentiële complexiteitsfuncties stijgt exponentieel wanneer de instantie groter wordt. De berekeningstijd duurt dan ook veel langer (zie tabel)

Table 2. Polynomial-time algorithms take better advantage of technology

Function	Size of instance solved in one day	Size of instance solved in one day in a computer 10 times faster
n	10^{12}	10^{13}
$n \log n$	0.948×10^{11}	0.87×10^{12}
n^2	10^6	3.16×10^6
n^3	10^4	2.15×10^4
$10^8 n^4$	10	18
2^n	40	43
10^n	12	13
$n^{\log n}$	79	95
$n!$	14	15

Een bepaald probleem zit in de klasse P indien er een algoritme voor bestaat dat het probleem exact oplost met een polyniomale tijdscomplexiteitfunctie. Een probleem bevindt zich in de klasse NP (Niet deterministisch polynomiaal) indien men in polynomiaaltijd kan nagaan of een bepaalde oplossing toelaatbaar is. Dit wil niet zeggen dat men een exacte oplossing kan vinden in polynomiaaltijd.
Andere problemen in de NP klasse zijn die waarvoor er nog niemand een polynomiaal algoritme voor heeft bedacht. Deze worden NP-compleet genoemd omdat dit de moeilijkste zijn.
Er wordt gezegd dat een probleem NP-compleet is indien elk probleem in NP reduceerbaar is tot dit probleem. Dit wil zeggen dat het mogelijk is dat er een algoritme bestaat in polynomiaaltijd voor de NP-complete problemen.
Opm. Het aantal mogelijke oplossingen zegt niet onmiddellijk iets over de complexiteit van het probleem.
Indien een probleem een subprobleem is van een ander probleem, dan is de complexiteit van het probleem even groot als de complexiteit van het ander, bekend moeilijk, probleem.



NP betekent: niet-deterministisch polynomiaal

9.1. Onderverdeling problemen

x ≺ y: x is een subprobleem van y, elke x kan geschreven worden als een speciale constructie van y. Als x moeilijk is is y zeker niet makkelijker. Even alle subproblemen op een rijtje:

Combinatorische optimalisatie problemen	Complexiteit
Kortste pad	P
Kortste boom	P
Max Flow	P
Min Cost Flow	P
Knapsack	NP-compleet
Cutting Stock	NP-compleet
Crew scheduling	NP-compleet
TSP	NP-compleet
…	

Netwerk-problemen

Bewijzen mbv reductie

Algoritmes	Complexiteit			
	Exact		Approximatie	
	Poly.	Exp.	Poly.	Exp.
Dijkstra (kortste pad)	x			
Prim\Kruskal (kortste boom)	x			
Augmenting path (Max flow)	x			
Cycle cancelling (Min cost flow)	x			
Branch & Bound		x		
Kolomgeneratie + Branch & Price		x		
Probleemafh. heuristieken			x	
Local search + metaheuristieken				x

Kwaliteit? WCR

Combinatorisch probleem kan bijna altijd als een IP geformuleerd worden, ze zijn moeilijk om te lossen via klassieke methodes maar gemakkelijk te enumereren.

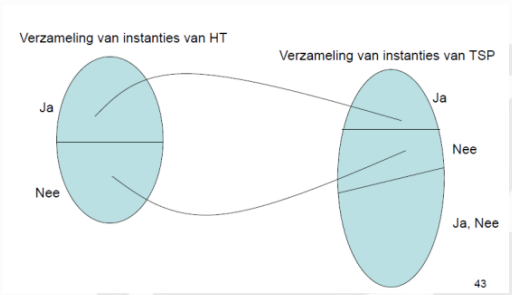
Branch and bound O(aantal knooppunten(2^n)* inspanning per knooppunt(n^2))

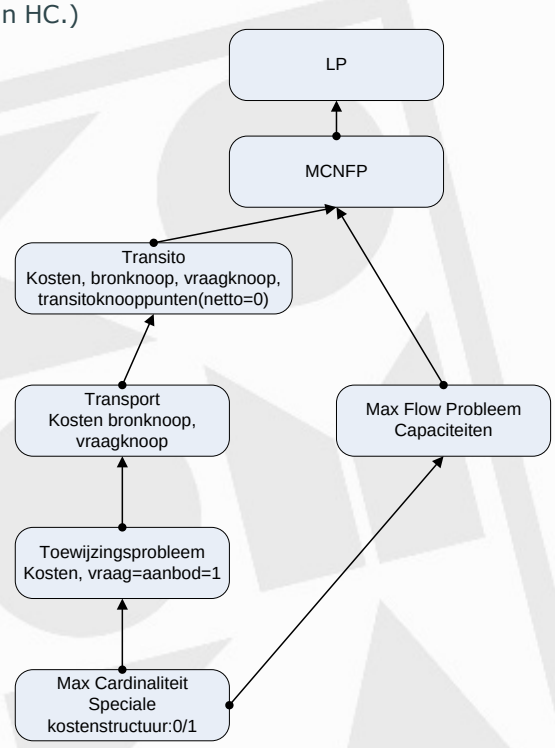
TSP, we berekenen het mogelijke aantal tours, er zijn n knopen, dus er zijn
n-1 keuzes om een arc te selecteren van node 1 naar node 2
n-2 keuzes om een arc te selecteren van node 2 naar node 3..
(n-1)(n-2)…(n-(n-2))((n-(n-1)) delen door 2: tour 1-2-3-1= tour 1-3-2-1
O((n-1)!/2

Toewijzingsprobleem: er zijn n taken en n personen, we kunnen elk van deze n taken toewijzen aan 1 van de n personen.
N keuzes om een arc te selecteren van node 1 (groep 1) naar node 1 (groep 2)
n-1 keuzes om een arc te selecteren van node 2 (groep1) naar node 2 (groep 2)
n(n-1)…(n-(n-2)((n-(n-1))=n!

Polynomiaal: Min-Cost-Flow, Max-Flow, Kortste pad, Lineair programmeren
Knapzak ≺ Langste pad (indien knapzak als kortste pad wordt geformuleerd zit de complexiteit in de knopen (aantal knopen: nC, niet polynomiaal in de input: nlog C)
Langste/kortste pad O(n^2) ≺ IP
Toewijzing ≺ MCNFP ≺ LP

NP-compleet: TSP, Bin packing, Cutting Stock, Knapsack
Partitie ≺ 0/1 Knapzak <- Partities kunnen omvormen in 0/1knapzak
0/1 Knapzak is geen subprobleem van geheeltallige knapzak!
Partitie ≺ Productie&Voorraad
Toewijzing ≺ MCNFP ≺ IP
TSP≺ IP
TSP ≺ Vehicle Routing Problem
HC ≺ TSP (HC reduceert zich tot TSP, we kunnen van iedere HC een TSP instantie maken en deze oplossing herleiden tot een antwoord op een HC.)





9.2. Beslissingsprobleem / Reductie

In een poging om aan te tonen dat het (nieuwe) probleem eigenlijk erg moeilijk is, NP-compleet, vergelijken we dit probleem met een bekend moeilijk probleem.(bekend)
We overlopen twee stappen:

1. We formuleren het nieuwe probleem als een beslissingsprobleem

   Dit is een probleem met maar twee mogelijke antwoorden (ja/nee, 1/0). De optimaliseringsvraag omzetten in een beslissingsvraag doen we door het toevoegen van een extra parameter B. Bestaat er een oplossing met waarde ≤ B?

2. We maken een reductie van bestaand een probleem naar het nieuwe probleem.

   Reductie: We gaan uit van het bestaand probleem en we maken van elke instantie van het bekende probleem een instantie van het nieuwe probleem, zodanig dat het antwoord op het beslissingsprobleem dezelfde is voor elke twee instanties.

9.2.1. Toepassing

HC Hamiltoniaans Circuit vs TSP handelsreizigersprobleem

2. We hebben een instantie van een HC met de steden, afstanden en een B van het TSP. Knoop= Stad, Geselecteerde boog in HC= afstand 1 in TSP, niet geselecteerde boog in HC= afstand 2 in TSP en we stellen B= |V|.

1. Beslissingsprobleem: Bestaat er een TSPtour met lengte ≤ B

Dit wil zeggen dat ALS er een HCoplossing bestaat, dat er een TSP is met waarde B.
Als er geen HC bestaat dan is er geen oplossing van het TSP met waarde B.

Partitie vs Knapzak

2. We hebben een instantie van een partitieprobleem met een n(aantal elementen), a(grootte), w(winsts), C(max capaciteit) en B(de beslissingsprobleemvar) uit knapzak. n:=p, a:=s, w:=s, C:= ½(Σisi), B:= ½(Σisi).
1. Beslissingsprobleem: bestaat er een knapzak met waarde ≤ B
   Als er een partitie instantie ja oplevert dan bestaat er een knapzakoplossing met waarde B.
   Als er een partitie instantie nee oplevert dan niet.

9.3. Algoritmes: overzicht

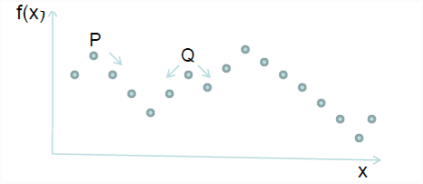
	Exact	Approximatief
Polynomiaal	P: kortste pad, max flow, min cost, LP Network flows door Ahuja (‘93) Linear programming door Chvatal(‘80)	Heuristieken: snel en probleemafh. NNB voor TSP(greedy) Approximation algorithms door Hochbaum (‘97)
Exponentieel	IP, BnB, BnC, BnP Integer and combinatorial optimization door Nemhauser (’88)	Local Search, Simulated annealing, tabu search, genetische algoritmen, artificiele neural netwerken Local search in combinatorial Optimization door Aarts(’97)

9.4. Heuristieken

Methoden die niet noodzakelijk een optimale oplossing geven maar snel een goede oplossing.
Constructie-methoden: een eerste oplossing construeren
Verbeteringsmethoden: met een startoplossing als input de huidige oplossing verbeteren.

9.4.1. Local Search

Generiek algoritme
Gegeven een oplossing o, en gegeven de definitie van een omgeving N.
•Stap 1: doorzoek de omgeving van o (N(o)). (Dat is een combinatorisch probleem op zich!)
•Stap 2: Als er een betere oplossing o‟ in N(o) zit, doe o := o‟, en ga naar Stap 1, anders stop: output o (lokaal optimum).



Nadeel van local search is dat het eindigt wanneer er een lokaal optimum wordt gevonden, dit kunnen we vermijden door ook opwaartse bewegingen toe te staan.

Simplex

Een lokale zoekmethode gezien de methode een aanliggende basisoplossing van de gegeven basisoplossing zoekt, in iedere iteratie wordt er een basisvariabele gewisseld voor een niet basisvariabele.

TSP

Goede localsearch methodes voor instanties tot 10.000 steden, 100.000 steden wordt teveel.
2opt en 3 opt leveren oplossingen een paar percent van het optimum.
Basale 2-opt: 2 kanten van een oplossing weghalen, op de enige andere mogelijke manier een tour maken.
3-opt: 3 kanten uit een tour weghalen en op een van de mogelijke manieren een nieuwe tour maken.
Restricted 3-opt: 1 stad uit de tour nemen en hem ergens anders terugplaatsen.

9.4.2. Simulated Annealing

Een weinig vereisende maar tijdrovende zoektocht door de ruimte der oplossingen:
Stap 1: pak, willekeurig, een oplossing uit de omgeving.
Stap 2: Als het een betere oplossing o‟ is, doe o := o‟, indien het een slechtere oplossing o‟ is: accept met een bepaalde kans. Ga naar Stap 1 (tot stopcriterium)
De kans hangt af van i) hoeveel slechter die oplossing is, en de iteratie: in het begin accepteer je veel, later veel minder (analogie met koelen van stoffen)

9.4.3. Tabu Search

Beschouw meerdere veranderingen tegelijkertijd en selecteer de beste (ook al is dit een verslechtering) Vervolgens zorg je ervoor middels een tabu-lijst dat je niet terugkeert naar de oplossing waar je net vandaan kwam.

Huidige oplossing

2	4	7	1	5	6	3

Waarde: 18

Top 5 kandidaten

Swap	Value
1,3	-2 **T**
2,4	-4 *
7,6	-6
4,5	-7 **T**
5,3	-9

Tabu Lijst

	2	3	4	5	6	7
1		3				
2						
3						
	4	2				
		5				
			6			

Huidige oplossing

4	2	7	1	5	6	3

Waarde: 14

Top 5 kandidaten

Swap	Value
4,5	6 **T** *
5,3	2
7,1	0
1,3	-3 **T**
2,6	-6

Tabu Lijst

	2	3	4	5	6	7
1		2				
2		3				
3						
	4	1				
		5				
			6			

Aspiratie: Indien een move de beste oplossing tot nu toe oplevert, aanvaarden ook al is die tabu

9.4.4. Genetisch algoritme

Analogie met de genetische structuur van chromosomen

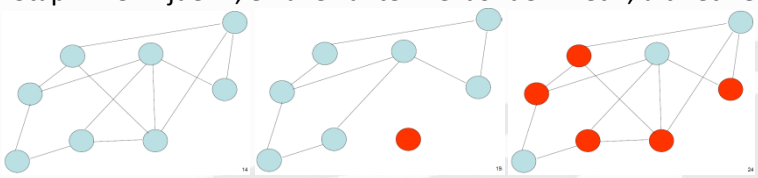
Ga uit van een populatie van oplossingen. Probeer een gemeenschappelijk kenmerk te vinden, en handhaaf dat kenmerk in de nieuw te vinden oplossingen.
Vertrek van een populatie van oplossingen (=chromosomen): mogelijke ouders met elk een “fitness value” (hoe hoger, hoe meer kans om geselecteerd te worden; gerelateerd aan de waarde van de overeenkomstige oplossing)
Selecteer 2 ouders uit de populatie, bvb:
O1: 1010010
O2: 0111001
Kies een cross-over point en creëer kinderen:
O1: 1010010 K1:1011001
O2: 0111001 K2:0110010
•Vervang een oplossing in de huidige populatie door 1 van de kinderen
•Survival of the fittest! Zwakke chromosomen weinig kans op overleven…
•Variaties in: selecteren ouders, aanpassen populatie, cross-over, mutaties, omvang populatie, fitness functie

9.5. Worst case ratio

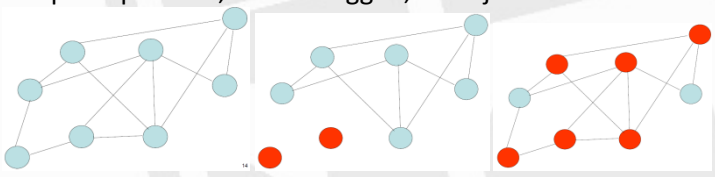
OPT(I)= de waarde van het optimum zijn voor instantie I
A(I)= de waarde van de oplossing zijn gevonden door heuristiek A.
$\max_I \{ A(I)/OPT(I) \}$ met I de instantie waarop de heuristiek het verst van de optimale oplossing zit.
We willen de worst case ratio zo klein mogelijk.

9.5.1. Node cover

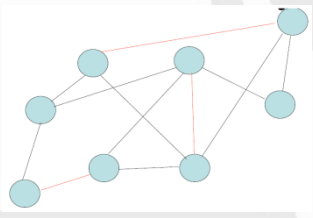
Gegeven een netwerk N=(V,E) vind een zo klein mogelijke deelverzameling W van V zodanig dat elke kant in E raakt aan een punt van W
Gegeven een netwerk G=(V,E)
Eerste algoritme
•Stap 0: W:={}.
•Stap 1: kies punt v in V met de hoogste graad, zet W:=W + {v}.
•Stap 2: verwijder v, en alle kanten verbonden met v, uit het netwerk. Ga naar Stap 1.



Geen performante heuristiek
Tweede algoritme
•Stap 0: E’:=E, W:={}
•Stap 1: Kies een kant {v,w} εE’.
•Stap 2: W:= W + {v} + {w}
•Stap 3: Update E’, dat wil zeggen, verwijder uit E’ alle kanten die raken aan v of w. Ga naar Stap 1.



Worst case ratio = 2
A(I) ≤ 2 OPT(I) voor alle instanties I

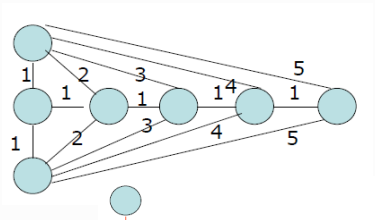


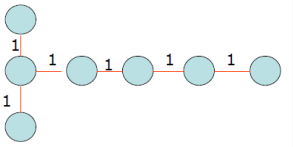
Bewijs:
We weten dat geen van deze kanten een knoop gemeenschappelijk heeft (dat vloeit voort uit de updating stap). Dus het aantal knopen dat je

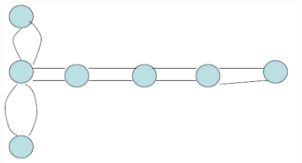
minstens nodig hebt in elke node cover (en dus ook een optimale) is gelijk aan dat aantal kanten:
OPT (I) ≥ AKANT(I).
Maar de waarde van de oplossing luidt: A(I) = 2 AKANT(I).
A(I) ≤ 2 OPT (I) voor alle I en er zijn instanties waar A(I)=2OPT(I) bvb. n=2

9.5.2. TSP

In het algemeen bestaat er geen polynomiaal algoritme voor het TSP met een eindige worst-case ratio. Voor elke worst-case ratio van elk polynomiaal algoritme vind je een instantie I zodat het algoritme een oplossing vindt die slechter is dan de |worst-case ratio|keer de lengte van de kortste tour.

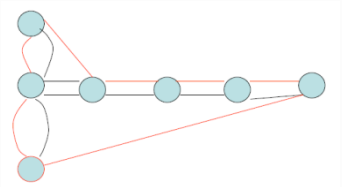






Heuristiek1

1. Vind een MSP mbv algoritme van Kruskal waarbij de lengte≤ de lengte van de optimale tour.
2. Verdubbel de MSP, we krijgen een Euler Cykel: ieder punt heeft een even graad. Dit is geen tour, er worden steden tweemaal bezocht.
3. We beginnen hieruit een tour te selecteren, indien we vastzitten maken we gebruik van de driehoeksongelijkheid: Laten we uitgaan van de zogenaamde driehoeksongelijkheid: Cij ≤ Cik + Ckj voor alle i,k,j en we gaan naar de eerstvolgende niet bezochte stad. De driehoeksongelijkheid zorgt ervoor dat dat de lengte van de rode tour ≤ lengte van de zwarte kanten.



Hieruit volgt dat de
lengte van de rode tour ≤ lengte van de zwarte kanten = 2 * lengte van MSP
en de lengte van MSP ≤ optimale tour
lengte rode tour ≤ 2 lengte van de optimale tour

Heuristiek2:
Er moet een Eulercykel onstaan of de graad van een punt moet even worden.

1. Als er een even aantal punten met een oneven graad is kunnen we deze 'matchen'
2. Op dezelfde wijze als heuristiek 1 een tour maken.

Lengte gevonden tour ≤ kortste boom + matching ≤ OPT + ½ OPT = 3/2OPT
Het is mogelijk om de optimale oplossing van moeilijke problemen te benaderen met eenvoudige probleemspecifieke oplossingsmethoden.

10. Examenvragen

Beschouw het probleem dat u in uw case aantrof. Beschrijf op liever niet meer dan 1 pagina
a. hoe een lokale zoekmethode er uit kan zien,
b. of een kolomgeneratie-aanpak toepasbaar is (zo ja, hoe dan, zo nee waarom niet).

Beschouw het volgende vehicle routing probleem. Gegeven zijn n locaties - waaronder een depot met index 1 - en een afstand cij tussen elk paar locaties i, j = 1, 2, . . . , n. In het depot staan K voertuigen. Het probleem is nu om ten hoogste K route's te vinden (eentje voor elk voertuig), zodanig dat

(i) elke locatie door precies een voertuig bezocht wordt,
(ii) elk voertuig terugkeert in het depot, en
(iii) (iii) de som der afgelegde afstanden van de K voertuigen minimaal is.

a. Geef een wiskundige formulering van dit probleem die gebruik maakt van (onder meer) de volgende binaire beslissingsvariabelen: xijk is 1 dan en slechts dan wanneer voertuig k van locatie i direct naar locatie j gaat, voor i, j = 1, . . . , n, k = 1, . . . ,K.
b. Een alternatieve formulering van het probleem maakt gebruik van een binaire variabele die correspondeert met een route. Deze variabele is 1 als een voertuig die route aflegt, en 0 als dat niet zo is. Hoe ziet een dergelijke formulering eruit?

c. Schets een oplossingsmethode voor de formulering onder b. die gebruik maakt van kolomgeneratie. Kunt u het pricing probleem identificeren? Wat zijn de voor- en nadelen van het oplossen van de laatste formulering (vergeleken met de formulering onder a.)?

Beschouw het volgende knapsack-probleem. Er zijn drie items met opbrengsten 9, 10, en 4. Er is een knapsack met capaciteit 6, en de drie items nemen deze hoeveelheid ruimte in: 3, 4, en 2 respectievelijk.
a. Geef de wiskundige formulering van deze knapsack instantie.
b. Hoe ziet een optimale oplossing van de LP-relaxatie van deze knapsack instantie eruit?
c. Kunt u een geldige ongelijkheid laten zien die deze oplossing wegsnijdt?

Gegeven:

(min, max)	Afdeling 1	Afdeling 2	Afdeling 3
Shift 1	6,8	11,12	7,12
Shift 2	4,6	11,12	7,12
Shift 3	2,4	10,12	5,7

Minima per shift: 26,24,19
Minima per afdeling: 13,32,22

Vraag: teken een netwerk met elke stroom = oplossing

Gegeven:

n items: hoogte hi, vochtigheid vi: 0 < … < 1
oven: max H = 1, max V = 1; er zijn k ovens

Vraag: minimaliseer aantal ovens

a) Wiskunde formulering: xik = 1 als item i in oven k

b) Alternatieve formulering: verzameling items = 1 als set aan een over wordt toegewezen

c) Schets een oplossing voor b via kolomgeneratie. Link leggen met pricing probleem. Voor- en nadelen formulering a & b?

d) Minimale hoogte wordt ½. Is kolomgeneratie nog toepasbaar? Schets de consequenties.

Extra examenvraag: toewijzingsprobleem
gegeven:
n taken: j = 1, …, n
m machine: i = 1, …, m
p_{ji} = tijd die machine i nodig heeft om taak j uit te voeren
b_i = tijdseenheden die machine i ter beschikking heeft
w_j = winst die het uitvoeren van taak j oplevert
Eén machine kan slechts één taak tegelijkertijd uitvoeren

Probleem: realiseer een toewijzing van taken aan machines die een zo groot mogelijke winst oplevert, zonder dat de capaciteiten van de machines worden overschreden.

a. Geef een wiskundige formulering van dit probleem die gebruik maakt van een binaire beslissingsvariabele x_{ji} die de waarde 1 heeft dan en slechts dan wanneer taak j aan machine i wordt toegewezen.

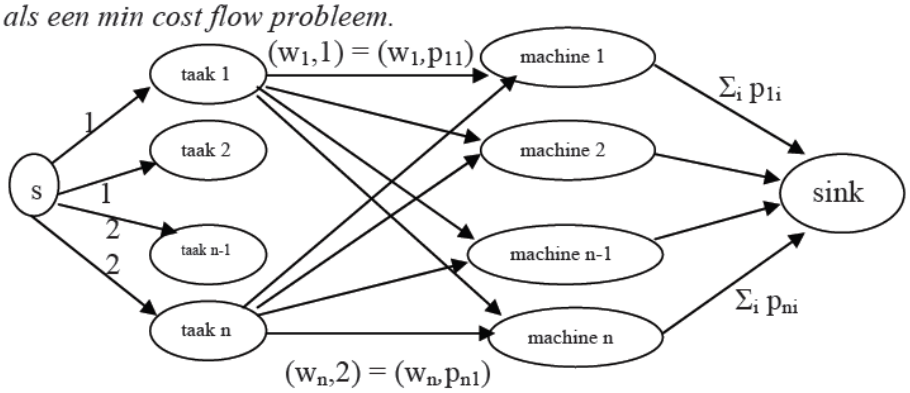
Max $\Sigma_{j,i}\, w_j\, x_{ji}$

odb $\Sigma_i\, x_{ji} \leq 1 \;\forall\, j$

$\Sigma_j\, p_{ji}\, x_{ji} \leq b_i \;\;\forall\, i$

$x_{ji} \in \{0,1\}$ voor alle i, j

b. Beschouw nu een variant hierop waarbij de lengte van elke taak 1 of 2 tijdseenheden bedraagt, maw er zijn slechts 2 soorten taken. soort 1 => $p_{ji} = 1$; soort 2 => $p_{ji} = 2$
Laat zien hoe de LP-relaxatie van de formulering onder a. voor dit speciale geval op te lossen is als een min cost flow probleem.



Min $\Sigma_{j,i}\, -w_j\, x_{ji}$

odb $\Sigma_j\, x_{sj} = \Sigma_{j,i}\, p_{ji}$ (de stroom die vertrekt in s moet volledig opgenomen worden door te taken)

$x_{sj} - \Sigma_i\, p_{ji} = 0 \;\;\forall\, j$ (de stroom die aankomt in taak j, moet ook weer vertrekken)

$\Sigma_j\, p_{ji} - x_{it} = 0 \;\;\forall\, i$ (de stroom die aankomt in machine i, moet ook weer vertrekken naar de sink)

$-\Sigma_j\, x_{it} = -\Sigma_i\, b_i$ (de stroom die aankomt in t moet gelijk zijn aan de sommatie van de capaciteiten van de machines)

$0 \leq p_{ji} \leq 2 \;\;\forall\, j,i$

$0 \leq x_{sj} \leq 2 \;\;\forall\, (s,j) \text{ in } A$

$0 \leq x_{it} \leq b_i \;\;\forall\, (i,t) \text{ in } A$

c. Een alternatieve formulering van het probleem met $p_{ji} = p_j \;\forall i$ maakt gebruik van een binaire variabele die correspondeert met de toewijzing van een verzameling taken aan een bepaalde machine. Hoe ziet een dergelijke formulering eruit?

Aangezien $p_{ji} = p_j \;\forall$ i, kunnen we verzamelingen van taken S maken die voldoen aan $\Sigma_{j:\, j \text{ in } S}\, p_j \leq b$

x_s = 1 als een verzameling taken S door één machine wordt gedaan
0 anders

$w_s = \Sigma_{j:\, j \text{ in } S}\, w_j$

Max $\Sigma_s\, w_s\, x_s$

odb $\Sigma_{j:\, j \text{ in } S}\, x_s \leq 1 \;\forall\, j$

$\Sigma_S\, x_s \leq m$

$x_s \in \{0,1\}$ voor alle S

Beschouw het volgende scheduling probleem. Gegeven zijn m machines en n taken. Elke taak heeft een bekende lengte p_j, j= 1, ..., n. Een machine kan slechts 1 taak tegelijkertijd uitvoeren. Het probleem is om elke taak toe te wijzen aan precies 1 machine, zodanig dat de voltooiingtijd (of complementeringtijd) van de laatst afgelopen taak zo vroeg mogelijk is. Een taak mag niet onderbroken worden. vb: m=2, p_1=3, p_2=4, p_3=5. Wanneer U taak 2 aan machine 2 toewijst en de overige taken aan machine 1, dan is de waarde van die oplossing 8. Laat OPT de waarde zijn van een optimale oplossing.

Eerst stellen we de wiskundige formulatie op:
Een bepaalde set van taken wordt op één machine uitgevoerd: 'j op i'
$OPT = Max\ (\Sigma_{j:\ j\ op\ i} p_j x_{j1}, \Sigma_{j:\ j\ op\ i} p_j x_{j2}, ..., \Sigma_{j:\ j\ op\ i} p_j x_{jm})$
$OPT = Minimaliseer\ \{Max_i\ (\Sigma_{j:\ j\ op\ i} p_j x_{ji})\}$
odb $\Sigma_{j:\ j\ op\ i} x_{ji} = 1\ \forall\ j$
$x_{ji}\ \varepsilon\ \{0,1\}$ voor alle (j,i)

a. Is dit waar: $OPT \geq max_j\ p_j$? Leg uw antwoord uit.

Ja, om dit aan te tonen gaan we uit van 3 situatie: j<m, j=m, j>m
- Stel 1=j<m => $OPT = Max_i\ (\Sigma_{j:\ j\ op\ i} p_j x_{ji}) = Max_i\ (p\ x_{ji}) = p$ $\geq max_j\ p_j = max\ p$ (zelfs =)
- Stel j=m => $OPT = Max_i\ (\Sigma_{j:\ j\ op\ i} p_j x_{ji}) = Max_i\ (p_j) = Max_j\ p_j$ $\geq max_j\ p_j$ (zelfs =)
- Stel j>m=1 => $OPT = Max_i\ (\Sigma_{j:\ j\ op\ i} p_j x_{ji}) = Max\ (\Sigma_j\ p_j) = \Sigma_j\ p_j$ $\geq max_j\ p_j$

b. Is dit waar: $OPT \geq \Sigma_j\ p_j/m$? Leg uw antwoord uit.
JA, dit is waar
- Stel 1=j<m => $\Sigma_j\ p_j/m = p/m$ => $OPT = p \geq p/m$
- Stel j=m => $\Sigma_j\ p_j/m = \Sigma_j\ p_j/j = gemid\ p_j$ => $OPT = Max_j\ p_j \geq gemid\ p_j$
- Stel j>m=1 => $\Sigma_j\ p_j/m = \Sigma_j\ p_j = OPT$

Heuristiek: plaats de taken in een lijst L; en zolang L niet leeg, pak de huidig eerste taak van L en plaats deze op die machine met de huidig vroegste completeringtijd.
F_{min} is de kleinste completeringtijd over de machines gevonden door dit algoritme.
c. Is dit waar: $OPT \leq 2\ F_{min}$? Leg uw antwoord uit.
NEEN, dit is NIET waar. Tegenvoorbeeld
Stel p_1= 1, p_2= 5, p_3= 5, p_4= 25 en 2 machines
machine 1: p_1 en p_3 = 1+5 = 6
machine 2: p_2 en p_4 = 5+25 = 30
F_{min} = 6 en OPT= 26
Dus $OPT \geq 2 \times 6 = 12 = 2 \times F_{min}$

d. Kunt u een bovengrens voor een worst case ratio voor de heuristiek laten zien mbhv die ongelijkheden uit a., b. en c. waarop u ja zei? Motiveer uw antwoord.
worst case = dat de taken met de grootste completeringtijden op 1 machine worden uitgevoerd.
worst case = sommatie van de (j/m) grootste p_j getallen (als j/m een breuk is, rond dan af naar boven tot het dichtstbijzijnde geheel getal).
$\Sigma_{(j/m)}\ max_j\ p_j$

e. Geef een instantie waar dit algoritme een zo slecht mogelijke oplossing geeft.
Stel p_1= 1, p_2= 6, p_3= 5, p_4= 25 en 2 machines
machine 1: p_1 en p_3 = 1+5 = 6
machine 2: p_2 en p_4 = 6+25 = 31
OPT= 26
j/m = 4/2 = 2, worst case = 6+25=31

f. Wat kunt u zeggen over de worst case ratio van dit algoritme gegeven uw antwoorden bij d. en e.
Dat deze worst case niet zo vaak zal voorkomen.