Het bachelorseminarium als toewijzingsprobleem

Mourits, J.W.

J.W. Mourits

Het bachelorseminarium als toewijzingsprobleem

Bachelorscriptie

16 juli 2019

Scriptiebegeleider: Dr. F.M. Spieksma

Universiteit Leiden
Mathematisch Instituut

Inhoudsopgave

1 Inleiding

Bij het Bachelorseminarium Analyse, Stochastiek en Besliskunde (ASB) 2019 moest elk van de 14 studenten gekoppeld worden aan één van 28 beschikbare projecten. De indeling van de projecten wordt gemaakt aan de hand van een lijst met minstens vijf voorkeuren die door elke student wordt ingeleverd. Met deze voorkeuren moet een oplossing gevonden worden waarbij zoveel mogelijk studenten hun favoriete project krijgen. In dit verslag bestuderen we verschillende aspecten van dit probleem. Hierbij wordt eerst de existentie van een oplossing, dat wil zeggen een indeling waarbij elke student een project uit zijn/haar keuzelijst krijgt, onderzocht. Ook berekenen we het aantal oplossingen en natuurlijk de optimale oplossing. Vervolgens zullen de gevolgen van extra eisen worden onderzocht en tot slot zal worden gekeken naar het benodigde aantal projecten bij verschillende aantallen studenten. Zo kan dit project hopelijk bijdragen aan een beter inzicht in dit toewijzingsprobleem en kan in de toekomst sneller een geschikte indeling gemaakt worden.

2 Het toewijzingsprobleem

Het probleem van het optimaal indelen van de studenten bij de bachelorprojecten is een voorbeeld van een toewijzingsprobleem (of "assignment problem" of "maximum-weight matching"). Bij dit toewijzingsprobleem is er sprake van een verzameling van m studenten: $S = \{1, 2, ..., m\}$, die allen een project toegewezen moeten krijgen uit een verzameling van n projecten: $P = \{1, 2, ..., n\}$, waarbij $m, n \in \mathbb{N}$ en $m \leq n$. Een student mag hiervoor een geordende voorkeurslijst van minstens vijf projecten opgeven. Er zijn vervolgens een aantal vragen die we willen onderzoeken bij dit toewijzingsprobleem. Allereerst willen we weten of er een oplossing bestaat, dat wil zeggen dat alle studenten een project uit hun keuzelijst kunnen krijgen. Als dit het geval is, is het interessant te kijken naar het aantal mogelijke oplossingen en tot slot willen we een optimale oplossing vinden. Om deze vragen te beantwoorden maken we gebruik van alternatieve formuleringen van het toewijzingsprobleem. Als voorbeeld nemen we steeds de opgegeven keuzes van het geanonimiseerde ASB-seminarium 2019.

2.1 Existentie van een toewijzing

Allereerst onderzoeken we de vraag of er een toegestane toewijzing bestaat, dat wil zeggen een toewijzing waarbij iedere student een project uit zijn/haar keuzelijst krijgt. De plaats van een project op de keuzelijst is dus nog niet van belang in deze paragraaf en we kunnen ons dus tot twee mogelijkheden beperken: een project staat in de keuzelijst of niet. Bij het onderzoeken van de existentie van een oplossing kan gebruik worden gemaakt van de Huwelijksstelling van Hall [2].

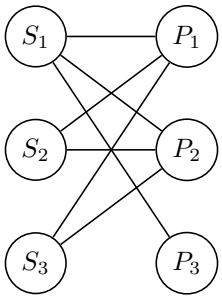
Stelling 2.2 (Huwelijksstelling van Hall) Het toewijzingsprobleem heeft een oplossing dan en slechts dan als voor ieder k-tal studenten de verzameling projecten in hun gezamenlijke voorkeurslijsten uit minstens k projecten bestaat, $k = 1, 2, ..., m$.

De Huwelijksstelling van Hall geeft dus aan wanneer het toewijzingsprobleem een oplossing heeft. Om vervolgens een oplossing te vinden kan het probleem worden geformuleerd als het vinden van een maximale koppeling in een bipartiete graaf.

2.1.1 Maximale koppeling in een bipartiete graaf

Deze paragraaf is voornamelijk gebaseerd op de hoofdstukken 16 en 17 in [4]. Een *niet-gerichte graaf* $G = (V, E)$ bestaat uit een verzameling knooppunten V en een verzameling takken E. Een graaf heet *bipartiet* als de verzameling knooppunten V bestaat uit twee deelverzamelingen $S, P \subset V$ zodanig dat $S \cup P = V$ en $S \cap P = \emptyset$, waarbij elke tak van G precies één eindpunt in S en één eindpunt in P heeft. Dit zijn in het bachelorseminarium de verzameling studenten S en de verzameling projecten P. Met andere woorden, $V = S \cup P$ en $(i, j) \in E$ als student $i \in S$ project $j \in P$ in zijn/haar voorkeurslijst heeft.

Voorbeeld 2.1
Voor $m = 3$ en $n = 3$ geldt: student 1 heeft voorkeur voor de projecten 1, 2 en 3, student 2 voor de projecten 1 en 2 en student 3 voor de projecten 1 en 2. De bijbehorende bipartiete graaf met links de studenten en rechts de projecten is nu:



Definitie Een *koppeling* (matching) $M \subset E$ is een verzameling van niet-aangrenzende takken. Een *maximale koppeling* (maximum-size matching) in G is een koppeling met een maximaal aantal takken.

Een koppeling in deze bipartiete graaf komt dus overeen met een toewijzing van projecten aan studenten. Het toewijzingsprobleem heeft een oplossing als elke student een project toegewezen kan krijgen, ofwel als er een koppeling bestaat die alle knooppunten van S bindt. Of een oplossing bestaat kan worden onderzocht met behulp van de Huwelijksstelling van Hall. Voor $W \subset S$ is $N(W)$ de verzameling knooppunten in P aangrenzend aan W, dus de verzameling projecten die gezamenlijk door de studenten in de verzameling W zijn gekozen. De Huwelijksstelling van Hall kunnen we nu hervormen in termen van een koppeling in een bipartiete graaf.

Stelling 2.1 (Huwelijksstelling van Hall) G heeft een koppeling die alle knooppunten van S bindt dan en slechts dan als voor elke $W \subset S$ geldt: $|N(W)| \geq |W|$.
Als zo'n koppeling bestaat is deze tevens een maximale koppeling.

De Huwelijksstelling van Hall geeft dus een nodige en voldoende voorwaarde voor de existentie van een oplossing. Omdat in hoofdstuk 4 een eenvoudigere methode om de existentie van een oplossing te onderzoeken wordt gegeven, passen we de Huwelijksstelling van Hall hier niet toe. Om na het bepalen van de existentie ook werkelijk een oplossing te vinden kan gebruik worden gemaakt van een efficiënt algoritme. We gebruiken een eerste algoritme van orde $O(nm)$, waarin gebruik wordt gemaakt van groeiketens. Zij M een koppeling, dan geldt het volgende.

Definitie Een *M-groeiketen* (M-augmenting path) in G is een pad van oneven lengte waarbij de eerste en laatste tak niet bevat zijn in M en de takken afwisselend niet en wel in M bevat zijn.

De invoer van dit algoritme zijn een bipartiete graaf $G = (S \cup P, E)$ en een koppeling M. Het algoritme geeft nu steeds, gebruikmakend van een groeiketen, een nieuwe koppeling M' waarvoor geldt $|M'| > |M|$.

Algoritme 2.1 Het vinden van een maximale koppeling in een bipartiete graaf $G = (S \cup P, E)$

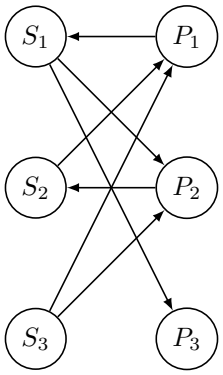
1. Bepaal een koppeling M.
2. Zij $u \in S$ en $v \in P$. Construeer een gerichte bipartiete graaf $D_M = (S \cup P, E')$ met takken $e' \in E'$, waarbij E' als volgt wordt geconstrueerd.

   Voor elke tak $e = (u, v) \in E$:

als $e \in M$: $e' = (v, u) \in E'$
als $e \notin M$: $e' = (u, v) \in E'$.

3. Zij $S_M \subset S$ en $P_M \subset P$ de verzamelingen knooppunten die geen begin- of eindpunt van een tak in M zijn.
4. *Als er geen gericht pad van S_M naar P_M in D_M*: M is een maximale koppeling. STOP.
   Anders: Bepaal een gericht pad $\mathcal{P}$ in D_M met beginpunt in S_M en eindpunt in P_M. De takken van $\mathcal{P}$ met beginpunt in S en eindpunt in P leveren een groeiketen op. Door de takken van deze groeiketen toe te voegen aan de bestaande koppeling M en de overige takken (met beginpunt in P en eindpunt in S) van $\mathcal{P}$ te verwijderen uit M ontstaat een nieuwe koppeling M' met $|M'| > |M|$.
5. $M := M'$. Ga naar stap 2.

Voorbeeld 2.1 (vervolg)
We passen de eerste iteratie van Algoritme 2.1 toe op de bipartiete graaf van voorbeeld 2.1. Een koppeling waar mee gestart kan worden bestaat uit de volgende takken: $M = \{(S_1, P_1), (S_2, P_2)\}$. De gerichte graaf D_M is nu:



Er geldt: $S_M = \{S_3\}$ en $P_M = \{P_3\}$. Een gericht pad van S_3 naar P_3 wordt nu bijvoorbeeld gegeven door: $\{S_3, P_1, S_1, P_3\}$. De takken (S_3, P_1) en (S_1, P_3) worden nu dus toegevoegd aan de koppeling en de tak (P_1, S_1) wordt verwijderd. Dit levert de nieuwe koppeling $M' = \{(S_1, P_3), (S_2, P_2), (S_3, P_1)\}$. Deze koppeling grenst aan alle studenten, dus is tevens een maximale koppeling.

Met behulp van algoritme 2.1 kan altijd een maximale koppeling gevonden worden. Als aan de Huwelijksstelling wordt voldaan, is dit tevens een koppeling die alle knooppunten van S bindt. In dat geval is hiermee dus een oplossing gevonden van het toewijzingsprobleem.

2.2 Het aantal mogelijke toewijzingen

Het aantal mogelijke oplossingen kan worden berekend met behulp van zogenaamde torenveeltermen. Deze behandelen we in deze paragraaf aan de hand van [1]. We werken hierbij met een $m \times n$ matrix B, met $b_{ij} = 1$ als student i project j op zijn/haar keuzelijst heeft en $b_{ij} = 0$ als dit niet het geval is. Als we nu de $m \times n$ matrix B als rechthoekig schaakbord van m bij n vakjes bekijken, vormen de vakjes (i, j) met $b_{ij} = 1$ een deelbord, zeg C. Dit bord kan dus elke willekeurige vorm aannemen en heeft een hoogte van maximaal m vakjes en een breedte van maximaal n vakjes, waarbij $m \leq n$ (er kunnen immers niet meer studenten dan projecten zijn). Er geldt nu dat voor $k \leq m$ de term $r_k(C)$ aangeeft op hoeveel manieren er k torens op C geplaatst kunnen plaatsen zodanig dat er geen torens zijn die elkaar slaan. De torenveelterm is de voortbrengende functie van de rij $r_k(C)$, $k = 0, ..., m$:

$$R(x, C) = r_0(C) + r_1(C)x + r_2(C)x^2 + ... + r_m(C)x^m.$$

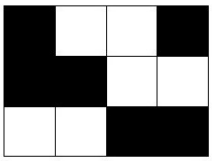
Het aantal mogelijke oplossingen bij m studenten is nu dus gelijk aan de laatste term $r_m(C)x^m$, omdat deze term precies aangeeft op hoeveel manieren er m niet-slaagse torens geplaatst kunnen worden op C. Deze term geeft dus het aantal manieren waarop elk van de m studenten een project uit zijn/haar keuzelijst toebedeeld kan krijgen, waarbij elk project maximaal één keer aan een student gekoppeld kan worden. Voor het berekenen van de torenveelterm maken we gebruik van de volgende eigenschap.

Eigenschap Zij v een vakje op bord C. Laat D het bord zijn waarbij alle vakjes uit de rij en kolom van v zijn verwijderd. E is het bord waarbij alleen vakje v is verwijderd. Voor de torenveelterm geldt nu: $R(x, C) = x \cdot R(x, D) + R(x, E)$.

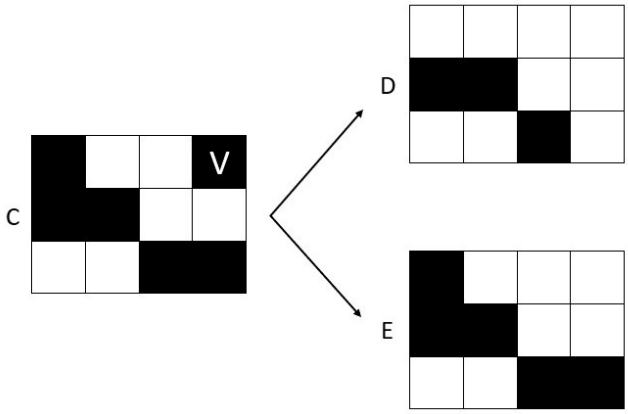
Voorbeeld 2.2 We bekijken een situatie met drie studenten en vier projecten, waarbij de volgende keuzes zijn opgegeven (volgorde is niet van belang):

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Dit geeft het volgende bord C, waarbij de zwarte vakjes de vakjes van C zijn.



Als we nu de eigenschap gebruiken, waarbij het vakje v is aangegeven, zien we dat het bord wordt opgesplitst in de volgende twee deelborden D en E:



Het gebruik van deze eigenschap kunnen we blijven herhalen tot er een bord overblijft waarvan we de torenveelterm weten (de torenveelterm van een enkel vakje is bijvoorbeeld $1 + x$, omdat er één manier is om 0 niet-slaagse torens te plaatsen en één manier om 1 niet-slaagse toren te plaatsen). In het voorbeeld is de torenveelterm van bord D gelijk aan: $R(x, D) = 1 + 3x + 2x^2$ en de torenveelterm van bord E is gelijk aan: $R(x, E) = 1 + 5x + 7x^2 + 2x^3$. Dit geeft dus dat de torenveelterm van heel bord C gelijk is aan:

$$R(x, C) = x \cdot R(x, D) + R(x, E) = x(1 + 3x + 2x^2) + (1 + 5x + 7x^2 + 2x^3)$$
$$= 1 + 6x + 10x^2 + 4x^3.$$

Omdat de laatste term gelijk is aan $4x^3$ zijn er dus vier mogelijkheden om drie niet-slaagse torens op bord C te plaatsen. Er zijn dus vier mogelijke indelingen waarbij de drie studenten allemaal een keuze van hun lijst krijgen.

Met behulp van een programma geschreven in Matlab (zie Appendix A voor de code) kan op dezelfde manier als in het voorbeeld de torenveelterm voor de voorkeuren van het ASB-seminarium 2019 berekend worden. Het aantal mogelijke indelingen is dan gelijk aan $r_{14}(C)$, omdat er 14 studenten zijn. Wegens de duur van het programma waarmee we het aantal mogelijke indelingen berekenen bekijken we alleen het aantal mogelijke indelingen waarbij elke student zijn/haar eerste, tweede of derde keus krijgt. Voor het ASB-seminarium geeft dit een totaal aantal van 5460 mogelijke indelingen waarbij elke student zijn/haar eerste, tweede of derde keus krijgt.

2.3 Een optimale toewijzing

Nu bekend is hoeveel oplossingen dit toewijzingsprobleem heeft, willen we weten wat de optimale oplossing is. De eenvoudigste manier om dit te doen is door het probleem als LP-probleem te formuleren. Dit heeft tevens als voordeel dat extra voorwaarden gesteld kunnen worden, wat met name in hoofdstuk 3 van pas komt. Voor de formulering als LP-probleem definiëren we eerst een $m \times n$ matrix V waarin de voorkeuren van de studenten worden verwerkt. Hierbij representeren de m rijen de studenten en de n kolommen staan voor de n beschikbare projecten. Als $v_{ij} = 0$ dan komt project j niet op de voorkeurslijst van student i voor. Als $v_{ij} \neq 0$ betekent dit dat project j op de voorkeurslijst van student i voorkomt. Hierbij is $v_{ij} = x$ als student i project j heeft opgegeven als x-de keuze. Voor het bachelorseminarium 2019 levert dit de volgende matrix V op:

$$V = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 1 & 2 & 5 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 3 & 6 & 5 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 1 & 2 & 4 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 1 \\
0 & 5 & 6 & 1 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 1 & 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 4 & 0 & 0 & 5 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 2 & 1 & 4 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 0 & 0 & 0 & 5 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 1 & 0 \\
0 & 0 & 0 & 1 & 4 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 1 & 0 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 2 & 1 & 5 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}.$$

Om dit LP-probleem als een maximaliseringsprobleem te formuleren moeten echter gewichten worden toegekend aan de voorkeuren. Deze gewichten geven de prioriteit van een student voor een project aan: hoe hoger het gewicht, des te gewenster het project bij deze student. Om het gewicht van een keuze aan te geven definiëren we $w_M : (V_i)_j \to \mathbb{R}_+$. Als voor een matrixelement geldt: $v_{ij} = 0$, dan geldt ook $w(v_{ij}) = 0$. Voor de overige matrixelementen worden de gewichten zo gekozen dat $v_{ij} < v_{kl} \Leftrightarrow w(v_{ij}) > w(v_{kl})$. We kiezen nu voor het eerste voorbeeld: $w(v_{ij}) = 8 - v_{ij}$ (er zijn bij het bachelorseminarium 2019 tot zevende voorkeuren opgegeven), dus de matrix met de nieuwe gewichten wordt:

$$W = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 7 & 6 & 3 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 5 & 2 & 3 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 7 & 6 & 4 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 5 & 7 \\
0 & 3 & 2 & 7 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 6 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 7 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 5 & 4 & 0 & 0 & 3 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 6 & 7 & 4 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 6 & 5 & 0 & 0 & 0 & 3 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 7 & 0 \\
0 & 0 & 0 & 7 & 4 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 7 & 0 & 0 & 0 & 5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 6 & 7 & 3 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}.$$

2.3.1 LP-formulering

Beschouw de $m \times n$ matrix $X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}$, met $x_{ij} \in \{0,1\} \forall i,j$. Deze matrix correspondeert met een toewijzing, waarbij $x_{ij} = 0$ als student i project j niet krijgt en $x_{ij} = 1$ als student i project j wel krijgt. Met behulp van deze matrix X kan een LP-formulering van het toewijzingsprobleem worden opgesteld. We willen namelijk dat een 1 in de indelingsmatrix X op de positie van een zo hoog mogelijk gewicht in de matrix W staat, maar hierbij moet wel altijd voldaan worden aan de voorwaarden dat elke student precies 1 project toebedeeld krijgt (2) en dat een project maar maximaal 1 keer gekozen kan worden (3). De LP-formulering wordt nu als volgt.
Maximaliseer:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} x_{ij} \tag{1}$$

onder de voorwaarden:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in S \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad \forall j \in P \tag{3}$$

$$x_{ij} \geq 0, \text{ geheel}, \quad \forall j \in P, \forall i \in S. \tag{4}$$

We merken hierbij op dat deze LP-formulering altijd een oplossing heeft voor $n \geq m$. Er wordt immers gemaximaliseerd aan de hand van de gewichten, maar het kan hierbij voorkomen dat een student een project krijgt dat niet op zijn/haar keuzelijst voorkomt. In dit geval is dus bij het maximaliseren een keer gewicht 0 gekozen. De voorwaarden van de LP-formulering voorkomen dit niet. Omdat altijd een indeling gemaakt moet worden bij het bachelorseminarium, is het echter toch gewenst dat er altijd een indeling gegeven wordt. Het risico dat bij een indeling niet alle studenten een van hun keuzes toebedeeld krijgen, kan dan worden geminimaliseerd met behulp van de gewichten. Door de gewichten van de gekozen projecten immers groot te kiezen, krijgt elke student zo vaak mogelijk een project uit zijn/haar keuzelijst.

Het vinden van de optimale oplossing van een LP-probleem met de eis dat x_{ij} geheel moet zijn voor alle $i \in S$ en $j \in P$, is een $\mathcal{NP}$-compleet probleem [1]. We willen daarom bekijken of deze eis wel nodig is. Als de oplossing immers altijd geheeltallig is, is de eis dat x_{ij} geheel moet zijn niet meer nodig. Dat dit inderdaad het geval is, zullen we nu bewijzen aan de hand van hoofdstuk 6 uit [2].

Bewijs
We hebben te maken met de volgende LP-formulering:

$$\max \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} x_{ij} \;\middle|\; \begin{array}{ll} \sum_{j=1}^{n} x_{ij} = 1, & \forall i \in S \\ \sum_{i=1}^{m} x_{ij} \leq 1, & \forall j \in P \\ x_{ij} \geq 0, \text{ geheel} & \end{array} \right\}. \tag{5}$$

Dit LP-probleem is te schrijven in de vorm

$$\max\{p^T x | Ax = b; x \geq 0 \text{ en geheel}\}$$

met A een geheeltallige matrix en b een geheeltallige vector. Met elke keuze van m onafhankelijke kolommen van matrix A correspondeert precies één vierkante $m \times m$ deelmatrix K. Als we nu $Kx = b$ omschrijven tot $x = K^{-1}b$, geldt dat de oplossing, dus x, geheeltallig is dan en slechts dan als K^{-1} geheeltallig is. Dankzij de regel van Cramer [3] is bekend dat $K^{-1} = \det(K)^{-1}\tilde{K}$, waarbij $\tilde{k}_{ij} = (-1)^{i+j}\det(K_{ji})$ met K_{ji} de matrix waaruit rij j en kolom i verwijderd zijn. Hieruit volgt dat K^{-1} geheeltallig is dan en slechts dan als de determinant van elke vierkante deelmatrix gelijk aan 1 of -1 is. Dit motiveert de volgende definitie.

Definitie Een $p \times q$ matrix met rang q heet *unimodulair* als alle elementen geheel zijn en de determinant van iedere vierkante $q \times q$ deelmatrix +1, -1 of 0 is. Een $m \times n$ matrix met willekeurige rang heet *totaal unimodulair* als de determinant van iedere vierkante deelmatrix +1, -1 of 0 is.

Hierbij wordt unimodulariteit geïmpliceerd door totale unimodulariteit. De relatie tussen totale unimodulariteit en de geheeltalligheid van een oplossing van een LP-probleem wordt nu gegeven door de volgende stelling, waarvan het bewijs hierboven gedeeltelijk is geschetst.

Stelling 2.3 Zij M een $p \times q$ matrix met rang p en met geheeltallige elementen. Dan is M unimodulair d.e.s.d.a. voor iedere geheeltallige vector $b \in \mathbb{R}^p$ geldt dat het polyhedron $\{x | Mx = b; x \geq 0\}$ geheeltallige hoekpunten heeft.

Om deze stelling toe te passen op de LP-formulering van het bachelorseminarium moet het LP-probleem dus alleen nog gelijkheden bevatten. We introduceren daarvoor n nieuwe variabelen: $y_1, ..., y_n$. De LP-formulering (5) kan nu geschreven worden als

$$\max \left\{ \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}x_{ij} \;\middle|\; \begin{array}{ll} \sum_{j=1}^{n} x_{ij} = 1, & \forall i \in S \\ \sum_{i=1}^{m} x_{ij} + y_j = 1, & \forall j \in P \\ x_{ij} \geq 0 & \end{array} \right\}. \tag{6}$$

Dit kunnen we compacter formuleren als:

$$\max\{w^T x | Ax = 1; x \geq 0\}, \tag{7}$$

waarbij de voorwaarden dus omgeschreven worden naar de vorm Ax. Matrix A heeft $m+n$ rijen en $m \cdot n + n$ kolommen en de kolomvectoren zijn $x = (x_{ij})_{i \in S, j \in P}$, $y = (y_1, ..., y_n)$.

$$A = \left(\begin{array}{ccccccccccccccc|c} 1 & 1 & 1 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \\ 0 & 0 & 0 & \cdots & 0 & 1 & 1 & 1 & \cdots & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & \\ & & & \vdots & & & & & \vdots & & \vdots & & & & \vdots & & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 & 1 & 1 & \cdots & 1 & \\ \hline 1 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & \\ 0 & 1 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & \\ & & & \vdots & & & & & \vdots & & \vdots & & & & \vdots & & I \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 1 & \cdots & 0 & 0 & 0 & \cdots & 1 & \end{array}\right). \tag{8}$$

De lijnen in de matrix A geven de horizontale grens aan tussen de voorwaarden 2 en 3 en de verticale grens tussen de variabelen x_{ij} en y_j.
Om nu Stelling 2.3 toe te passen moet bewezen worden dat deze matrix A unimodulair is. We gebruiken hiervoor de volgende stelling uit [2] (Stelling 6.2).

Stelling 2.4 Zij M een matrix met elementen 0, +1 met in iedere kolom hoogstens twee niet-nul elementen. M is totaal unimodulair d.e.s.d.a. de rijen van M in twee disjunctie deelverzamelingen I_1 en I_2 zijn te verdelen zodanig dat:
Voor elke twee niet-nul elementen uit dezelfde kolom behoren de bijbehorende rijen tot verschillende deelverzamelingen.

De matrix A uit (14) voldoet hieraan, als we voor I_1 de eerste m rijen kiezen en voor I_2 de laatste n rijen (dit is in (14) aangegeven met een horizontale grenslijn). Matrix A is dus totaal unimodulair en daarmee ook unimodulair. Met Stelling 2.3 volgt nu dat de LP-formulering een geheeltallige oplossing heeft en dat de voorwaarde $x_{ij} \geq 0$ voldoende is. □

2.3.2 Resultaten

Met behulp van de hiervoor gevonden LP-formulering kan een optimale oplossing gevonden worden. Hiervoor programmeren we het LP-probleem in R met de gewichtenmatrix W (zie Appendix B voor de code). Om dit probleem op te lossen is gebruik gemaakt van een ingebouwde LP-solver in R. Dit geeft de volgende optimale indeling.

Student	Project	Student	Project
1	20	8	14
2	4	9	5
3	19	10	9
4	26	11	27
5	28	12	25
6	16	13	23
7	1	14	15

Bij deze oplossing krijgen 10 studenten hun eerste keus, 3 studenten hun tweede keus en 1 student zijn/haar derde keus. Deze oplossing is niet uniek, want als we deze indeling bekijken aan de hand van de voorkeursmatrix V zien we dat bijvoorbeeld student 1 en student 3 dezelfde eerste en tweede keus hebben. In de oplossing heeft student 1 de tweede keus gekregen en student 3 de eerste keus. Dit kan echter worden verwisseld zonder dat de waarde van de doelfunctie verandert. Met behulp van de gewichten kan ook aan extra voorwaarden worden voldaan. Als men bijvoorbeeld wilt dat zo weinig mogelijk studenten hun derde keus of lager krijgen, kunnen de gewichten zo worden aangepast dat de eerste en tweede keuzes een hoger gewicht krijgen en dat daarna pas de lagere keuzes volgen. We kiezen nu bijvoorbeeld voor de gewichten:

$$\begin{aligned} w(a_{ij}) &= 10 && \text{als } a_{ij} = 1 \\ w(a_{ij}) &= 9 && \text{als } a_{ij} = 2 \\ w(a_{ij}) &= 8 - a_{ij} && \text{als } 3 \leq a_{ij} \leq 7. \end{aligned}$$

Met deze nieuwe gewichten krijgen we de volgende optimale oplossing:

Student	Project	Student	Project
1	20	8	16
2	9	9	5
3	19	10	14
4	26	11	27
5	28	12	25
6	4	13	23
7	1	14	15

Bij deze nieuwe indeling krijgen 9 studenten hun eerste keus en 5 studenten hun tweede keus. Geen enkele student krijgt zijn/haar derde keus. Door het veranderen van de gewichten kan dus aan extra prioriteiten of voorwaarden voldaan kan worden. Om een gegeven combinatie van student-project meer prioriteit te geven, moet hierbij het gewicht hoger zijn dan de overige gewichten.

2.3.3 De Hongaarse methode

Voor het vinden van de optimale toewijzing kan ook gebruik worden gemaakt van de formulering van het probleem als het vinden van een maximale koppeling met maximaal gewicht in een bipartiete graaf. Omdat rekening moet worden gehouden met de hoogte van projecten op de keuzelijst van de studenten moet echter wel een gewicht worden toegekend aan de takken van de bipartiete graaf.
Zij $w_G : E \to \mathbb{R}_+$, dan wordt voor elke $F \subset E$ het gewicht van F gedefinieerd door:

$$w_G(F) := \sum_{e \in F} w_G(e).$$

Dit gewicht hangt dan af van de hoogte van een project op de keuzelijst van een student en kan net als bij de LP-formulering gekozen worden om aan bepaalde criteria te voldoen. Als er gewichten worden toegekend aan de keuzes, kan de Hongaarse methode worden gebruikt om een maximale koppeling met maximaal gewicht te vinden. Omdat de Hongaarse methode het totale gewicht maximaliseert moeten dus ook de gewichten van de voorkeuren zó worden gekozen dat een hoge voorkeur resulteert in een hoog gewicht. De Hongaarse methode is vergelijkbaar met Algoritme 2.1, alleen zijn de invoer nu een bipartiete graaf $G = (S \cup P, E)$, met aan elke tak $e \in E(G)$ een gewicht $w(e)$ toegekend en een koppeling M.

Algoritme 2.2 Hongaarse methode

1. Bepaal een koppeling M.
2. Zij $u \in S$ en $v \in P$. Construeer een gerichte bipartiete graaf $D_M = (S \cup P, E')$ met takken $e' \in E'$, waarbij E' als volgt wordt geconstrueerd.

   Voor elke tak $e = (u, v) \in E$:
   als $e \in M$: $e' = (v, u) \in E'$ met lengte $l(e') := w(e)$
   als $e \notin M$: $e' = (u, v) \in E'$ met lengte $l(e') := -w(e)$.

3. Zij $S_M \subset S$ en $P_M \subset P$ de verzamelingen knooppunten die geen begin- of eindpunt van een tak in M zijn.

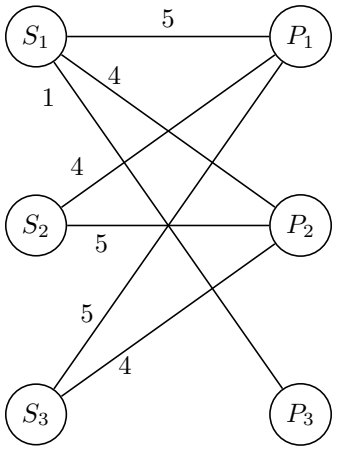
4. *Als er geen gericht pad van S_M naar P_M in is*: M is een maximale koppeling. STOP.
   Anders: Bepaal alle gerichte paden Q_i in D_M met beginpunt in S_M en eindpunt in P_M. Zij Q_j het pad met minimale lengte, ofwel $l(Q_j) \geq l(Q_i)$ voor alle $i \neq j$, waarbij $l(Q_i) = \sum_{e \in E(Q_i)} w(e)$. Als een gericht pad Q_i een ronde bevat van positieve lengte, geeft dit alleen extra paden met grotere lengte en blijft Q_j het pad met minimale lengte. Wanneer een gericht pad Q_i een pad met negatieve lengte bevat, moet het algoritme opnieuw gestart worden met een andere koppeling M. De takken van Q_j met beginpunt in S en eindpunt in P leveren een groeiketen op. Door de takken van deze groeiketen toe te voegen aan de bestaande koppeling M en de overige takken (met beginpunt in P en eindpunt in S) van Q_j te verwijderen uit M ontstaat een nieuwe koppeling M' met $|M'| > |M|$.

5. $M := M'$. Ga naar stap 2.

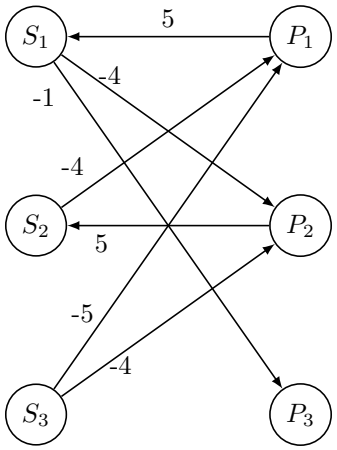
Voorbeeld 2.1 (vervolg) We bekijken weer voorbeeld 2.1, maar dit keer wordt ook rekening gehouden met de voorkeuren. De voorkeurslijsten worden gegeven door:

	Eerste keus	Tweede keus	Derde keus
Student 1	1	2	3
Student 2	2	1	-
Student 3	1	2	-

Als we de gewichten nu zó toekennen dat de eerste keus gewicht 5, de tweede keus gewicht 4 en de derde keus gewicht 1 krijgt, geeft dit de volgende graaf met gewichten.



We passen nu de Hongaarse Methode toe. Allereerst starten we met een koppeling $M = \{(S_1, P_1), (S_2, P_2)\}$. Dit geeft de gerichte graaf D_M.



Nu is $S_M = \{S_3\}$ en $P_M = \{P_3\}$. We vinden twee mogelijke gerichte paden met beginpunt in S_M en eindpunt in P_M, namelijk:

- $Q_1 = \{S_3, P_1, S_1, P_3\}$ met lengte $l(Q_1) = -1$.
- $Q_2 = \{S_3, P_2, S_2, P_1, S_1, P_3\}$ met lengte $l(Q_2) = 1$.

Verder is er nog een ronde $\{S_1, P_2, S_2, P_1, S_1\}$ met positieve lengte. Als deze ronde negatieve lengte zou hebben, moet met een andere koppeling begonnen worden.
Pad Q_1 heeft nu dus minimale lengte. We verwijderen de tak (S_1, P_1) uit M en voegen de takken (S_1, P_3) en (S_3, P_1) toe aan M. Dit geeft de nieuwe koppeling M' met takkenverzameling $\{(S_1, P_3), (S_2, P_2), (S_3, P_1)\}$.
Omdat in de volgende iteratie de verzamelingen S_M en P_M leeg zijn, bestaat er geen pad meer van S_M naar P_M in D_M, dus M' is een maximale koppeling met maximaal gewicht. In de optimale oplossing krijgt student 1 dus project 3, student 2 project 2 en student 3 project 1 toegewezen.

3 Spreiding over de docenten

Behalve een optimale oplossing zijn er nog andere interessante vragen met betrekking tot dit probleem. Bijvoorbeeld, of we ook rekening kunnen houden met de spreiding over de docenten die de projecten begeleiden. Docenten kunnen meerdere projecten opgeven voor het bachelorseminarium waar de studenten uit kunnen kiezen. Als alleen wordt gekeken naar de optimale oplossing voor de studenten, kan het voorkomen dat de ene docent meerdere studenten gaat begeleiden en een andere docent helemaal geen. Dit probleem kan op verschillende manieren opgelost worden met behulp van een nieuwe LP-formulering.

3.1 Spreiding met behulp van de gewichten

Een eenvoudige manier om voor spreiding over de docenten te zorgen is door de gewichten van de keuzes van de studenten te veranderen. Dit wordt zó gedaan dat een keuze voor een project van een docent met veel beschikbare projecten een lager gewicht krijgt dan voor die van een project van een docent met weinig projecten. Zo krijgt een student in de optimale oplossing eerder een project van een docent met weinig beschikbare projecten en vindt automatisch een spreiding plaats. Afhankelijk van de verschillen tussen de gewichten van docenten met veel en docenten met weinig projecten kan worden bepaald in hoeverre deze spreiding belangrijk is: bij kleinere verschillen is de keuze van de studenten nog steeds belangrijker dan de spreiding over de docenten. Hoe groter de verschillen, des te belangrijker de spreiding wordt.

3.2 Een maximum aantal te begeleiden studenten

Een andere, en wellicht ook interessantere, kwestie is de volgende: een docent kan zoveel projecten opgeven als hij/zij wil, met een maximum op het aantal te begeleiden studenten. Zo kan voorkomen worden dat een docent te veel studenten moet begeleiden. Als we hier rekening mee houden verandert dus de LP-formulering.
Laat $D_1, ..., D_v$ de verzamelingen projecten per docent zijn, waarbij dus v het aantal docenten is. Als bijvoorbeeld $D_1 = \{1, 4, 5\}$ betekent dit dat de projecten 1, 4 en 5 allemaal door docent 1 worden begeleid. Het maximum aantal projecten dat docent k wil begeleiden noteren we met d_k, $k = 1, ..., v$. De LP-formulering voor dit nieuwe probleem wordt nu:
Doelfunctie:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}x_{ij}. \tag{9}$$

Onder de voorwaarden:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in S \tag{10}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad \forall j \in P \tag{11}$$

$$\sum_{j\in D_k}\sum_{i=1}^{m} x_{ij} \leq d_k, \quad \forall k \in \{1, ..., v\} \tag{12}$$

$$x_{ij} \geq 0, \quad \forall j \in P, \forall i \in S. \tag{13}$$

Net als hiervoor is de voorwaarde dat de x_{ij} geheel zijn niet nodig. Het bewijs hiervan gaat deels analoog aan het bewijs in paragraaf 2.3.1.

De matrix M die hoort bij de formulering $\max\{p^T x | Mx = b; x \geq 0\}$ is dezelfde als de matrix A in 2.3.1, behalve dat er door de extra voorwaarde (12) w extra rijen zijn toegevoegd. We willen wederom bewijzen dat er geen eis gesteld hoeft te worden dat de x_{ij} geheel zijn. Om het bewijs overzichtelijk te houden, bekijken we nu eerst een voorbeeld met 3 studenten en 5 projecten, waarbij $D_1 = \{1, 2\}$, $D_2 = \{3, 4\}$ en $D_3 = \{5\}$. Hierbij is de coëfficiëntmatrix M_{vb} als volgt.

$$M_{vb} = \left(\begin{array}{ccccccccccccccc|c} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & \\ \hdashline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \\ \hdashline 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \end{array}\right) \begin{array}{l} Q \\ \\ \\ \\ \\ R \\ \\ \\ \\ \\ T. \end{array} \tag{14}$$

We willen nu voor het algemene geval, met m studenten en n projecten, wederom bewijzen dat de bijbehorende matrix M totaal unimodulair is. Hierbij gebruiken we eerst de definitie van unimodulariteit. We moeten dus laten zien dat elke vierkante deelmatrix van matrix M determinant +1, -1 of 0 heeft. Hiervoor verdelen we de rijen van matrix M in drie delen (aangegeven met een stippellijn in de matrix in het voorbeeld):

- De eerste m rijen horen bij voorwaarde (10), deze verzameling noemen we Q (in het voorbeeld zijn dit de eerste drie rijen).
- De n rijen daarna horen bij voorwaarde (11), deze verzameling noemen we R.
- De overige rijen horen bij voorwaarde (12), deze verzameling noemen we T.

Merk op dat in T elke kolom maximaal één 1 bevat, omdat een project niet door meerdere docenten begeleid wordt. Zij H nu een willekeurige vierkante deelmatrix van M. We onderscheiden vier gevallen:

- H heeft alleen rijen uit Q en R. In dit geval merken we op dat de matrix met de rijen van Q en R precies de matrix A uit paragraaf 2.3.1 is. We hebben al bewezen dat deze matrix totaal unimodulair is en dat dus elke deelmatrix determinant gelijk aan +1, -1 of 0 heeft. Hieruit volgt direct dat de determinant van H ook +1, -1 of 0 is.
- H heeft alleen rijen uit Q en T of uit R en T. In dit geval bevat elke kolom maximaal twee enen en kunnen we Stelling 2.4 toepassen met $Q = I_1$ en $T = I_2$, respectievelijk $R = I_1$ en $T = I_2$. Hieruit volgt dat H totaal unimodulair is en dat dus de determinant van elke deelmatrix van H gelijk is aan +1, -1 of 0, dus ook de determinant van H zelf.
- H heeft alleen rijen uit Q, R en T. Bekijk dan de rijen uit R en T die in H bevat zijn. Als een rij a van R en een rij b van T een 1 in een gemeenschappelijke kolom hebben, dan staat in elke kolom met een 1 in rij a ook een 1 in rij b. Dit komt door het terugkerende patroon dat de rijen van R en T volgen (modulo het totale aantal projecten n). Trek in dit geval rij a van rij b af en noem de nieuwe matrix die ontstaan is U. Er volgt dat elke

kolom van U maximaal twee enen heeft. Kies voor de matrix U, $Q = I_1$ en $R \cup T = I_2$ en pas Stelling 2.4 weer toe. Dan volgt dat U totaal unimodulair is en de determinant van U is dus gelijk aan +1, -1 of 0. Omdat de determinant van U gelijk is aan de determinant van H, volgt dat de determinant van H gelijk aan +1, -1 of 0.

We zien dus dat altijd geldt dat de determinant van de deelmatrix H gelijk is aan +1, -1 of 0. Hieruit volgt dat M totaal unimodulair is. Met Stelling 2.3 volgt nu dat de LP-formulering een geheeltallige oplossing heeft en dat de voorwaarde $x_{ij} \geq 0, i \in S, j \in P$ voldoende is.

3.3 Resultaten

Met een maximum aantal te begeleiden studenten voor elke docent, zoals beschreven in paragraaf 3.2, kunnen we onderzoeken welk effect dit heeft op de resultaten van het bachelorseminarium 2019. Bij het seminarium zijn drie docenten die meerdere projecten hebben ingediend, waarbij $D_1 = \{6, 7\}$, $D_2 = \{15, 16, 17\}$ en $D_3 = \{26, 27\}$. Voor het bachelorseminarium kiezen we ervoor dat alle docenten maximaal één student begeleiden, dus $d_1 = d_2 = d_3 = 1$. Dit geeft de volgende indeling (zie Appendix C voor de code):

Student	Project	Student	Project
1	20	8	14
2	4	9	5
3	19	10	15
4	26	11	7
5	28	12	25
6	8	13	23
7	1	14	9

Hierbij krijgen 9 studenten hun eerste keus, 3 studenten hun tweede keus, 1 student zijn derde keus en 1 student zijn vierde keus. We zien dus dat als de docenten een maximum aantal studenten willen begeleiden, dit mogelijk tot gevolg heeft dat de studenten gezamenlijk minder hoge projecten op hun keuzelijst krijgen.

4 Benodigde aantal projecten

Voor de indeling van het bachelorseminarium is het voor de organisatoren handig om te weten hoeveel projecten er nodig zijn om te voldoen aan de voorkeuren van de studenten. We onderzoeken daarom hoeveel projecten er nodig zijn bij verschillende aantallen studenten om er voor te zorgen dat in 95 procent van de gevallen alle studenten hun eerste, tweede of derde keus kunnen krijgen. Dit hebben we als volgt gedaan met behulp van een Monte-Carlosimulatie.
Voor m studenten en n projecten wordt k keer een $m \times n$ matrix E met de voorkeuren van de studenten geproduceerd. We willen alleen de eerste, tweede en derde keus van de studenten bekijken. Dit wordt gesimuleerd door in elke rij van E volgens een vooraf gekozen kansverdeling drie enen te plaatsen.
Voor elke gesimuleerde matrix bepalen we vervolgens of er een toewijzing bestaat die aan elke student één van zijn voorkeuren toekent. Een eenvoudige manier om dit te onderzoeken is met behulp van de LP-formulering. Omdat de te onderzoeken matrix E alleen nullen en enen bevat, is de waarde van de in paragraaf 2.3.1 bepaalde doelfunctie (1) gelijk aan het aantal studenten dat een project uit zijn voorkeurslijst heeft gekregen. Als er een toewijzing bestaat hebben alle studenten een project uit hun keuzelijst gekregen, dus is de waarde van de doelfunctie gelijk aan het totale aantal studenten m.

4.1 Resultaten bij een uniforme verdeling

Als we de posities van de drie enen in elke rij van matrix E bepalen met behulp van een uniforme verdeling heeft elk project evenveel kans om gekozen te worden door een student. In de volgende tabel zijn voor 13 tot en met 20 studenten (rijen) de kansen weergegeven dat er een toewijzing bestaat bij 13 tot en met 24 projecten (kolommen). Deze resultaten zijn gevonden op basis van een simulatie van 1000 matrices (zie Appendix D voor de code).

	13	14	15	16	17	18	19	20	21	22	23	24
13	0.573	0.852	0.959	0.989	0.995	0.997	0.999	1.000	1.000	1.000	1.000	1.000
14	-	0.518	0.843	0.929	0.984	0.998	0.995	0.999	0.999	1.000	0.999	1.000
15	-	-	0.497	0.798	0.925	0.974	0.991	0.998	0.999	1.000	1.000	0.998
16	-	-	-	0.443	0.777	0.911	0.973	0.992	0.998	0.999	1.000	1.000
17	-	-	-	-	0.442	0.773	0.895	0.969	0.984	0.996	0.997	0.999
18	-	-	-	-	-	0.418	0.717	0.874	0.950	0.982	0.997	0.999
19	-	-	-	-	-	-	0.386	0.674	0.860	0.944	0.979	0.993
20	-	-	-	-	-	-	-	0.355	0.681	0.839	0.933	0.965

We zien in deze tabel allereerst dat naarmate het aantal studenten toeneemt de kans dat er een toewijzing is, in het geval van een gelijk aantal studenten en projecten, afneemt. Om de onderzoeksvraag te beantwoorden zijn we steeds op zoek naar het minimale aantal projecten dat nodig is om in 95% van de gevallen een toewijzing te hebben. Uit deze resultaten volgt de volgende tabel, waarin voor elk aantal studenten te zien is hoeveel projecten nodig zijn om aan deze 95%-eis te voldoen. De getallen in de derde kolom geven aan hoeveel meer projecten dan studenten er nodig zijn, dus het verschil tussen de kolommen 1 en 2.

Aantal studenten	Aantal projecten	Verschil
13	15	2
14	17	3
15	18	3
16	19	3
17	20	3
18	21	3
19	23	4
20	24	4

Uit deze simulatie blijkt dat, om aan de 95%-eis te voldoen, ongeveer 20% meer projecten dan studenten nodig zijn. Vanaf 13 studenten blijkt dit genoeg te zijn en ook voor grotere aantallen blijkt uit simulaties dat 20% meer projecten voldoende is. De kans op een oplossing bij 20% meer projecten lijkt in simulaties zelfs naar 1 te gaan wanneer het aantal studenten groter wordt. Een logische verklaring hiervoor is dat bij een groter aantal studenten en daarmee een groter aantal projecten de kans op een oplossing toeneemt doordat er meer mogelijkheden zijn voor elke student om drie projecten te kiezen. Een wiskundige verklaring blijkt echter zeer lastig te vinden en hier zijn we dan ook niet aan toe gekomen. Wel is de kans op een oplossing eenvoudiger te onderzoeken wanneer elke student maar één keuze opgeeft.

4.1.1 Kans op een oplossing bij één opgegeven keuze

We nemen aan dat de m studenten elk uniform één project kiezen uit een verzameling van αm projecten met $\alpha \geq 1$. De onderzoeksvraag hierbij is voor welke waardes van α de kans op een oplossing naar 1 convergeert, in de limiet voor m naar oneindig.

Allereerst merken we op dat het totale aantal mogelijkheden van keuzecombinaties (dus het aantal manieren waarop door alle studenten een keuze opgegeven kan worden) van de m studenten uit de αm projecten gelijk is aan

$$(\alpha m)^m.$$

Elke student kiest immers één van de αm projecten en er zijn m studenten. Om een mogelijke indeling te kunnen maken waarbij elk project maximaal één keer wordt gekozen en elke student een project toegewezen krijgt, moeten alle studenten een verschillend project kiezen. Zodra twee studenten immers hetzelfde project kiezen is er, vanwege het feit dat de studenten maar één project opgeven, geen oplossing meer. Het aantal mogelijke keuzecombinaties waarbij er een oplossing is, is dus gelijk aan:

$$\binom{\alpha m}{m} \cdot m! = \frac{(\alpha m)!}{(\alpha m - m)!}.$$

Dit resulteert in de volgende kans op een oplossing:

$$\mathbb{P}(\text{er bestaat een oplossing}) = \frac{\frac{(\alpha m)!}{(\alpha m - m)!}}{(\alpha m)^m} = \frac{(\alpha m)!}{(\alpha m)^m (m(\alpha - 1))!}.$$

De formule van Stirling geeft als benadering:

$$x! \approx \sqrt{2\pi x} \left(\frac{x}{e}\right)^x.$$

Hiermee vinden we de volgende benadering van de kans op een oplossing:

$$\begin{aligned}
\frac{(\alpha m)!}{(\alpha m)^m (m(\alpha-1))!} &\approx \frac{\sqrt{2\pi\alpha m}\left(\frac{\alpha m}{e}\right)^{\alpha m}}{(\alpha m)^m \sqrt{2\pi m(\alpha-1)}\left(\frac{m(\alpha-1)}{e}\right)^{m(\alpha-1)}} \\
&= \sqrt{\frac{\alpha}{\alpha-1}} \cdot \frac{e^{m(\alpha-1)}}{e^{\alpha m}} \cdot \frac{(\alpha m)^{\alpha m}}{(\alpha m)^m (m(\alpha-1))^{m(\alpha-1)}} \\
&= \sqrt{\frac{\alpha}{\alpha-1}} \cdot \frac{1}{e^m} \cdot \frac{\alpha^{\alpha m} m^{\alpha m} \alpha^{-m}}{m^m (\alpha-1)^{m(\alpha-1)} m^{\alpha m - m}} \\
&= \sqrt{\frac{\alpha}{\alpha-1}} \cdot \frac{1}{e^m} \cdot \frac{\alpha^{\alpha m - m}}{(\alpha-1)^{m(\alpha-1)}} \\
&= \sqrt{\frac{\alpha}{\alpha-1}} \cdot \frac{1}{e^m} \cdot \left(\frac{\alpha}{\alpha-1}\right)^{m(\alpha-1)} \\
&= \sqrt{\frac{\alpha}{\alpha-1}} \cdot \left(\frac{\left(\frac{\alpha}{\alpha-1}\right)^{\alpha-1}}{e}\right)^m .
\end{aligned}$$

Omdat we de kans voor grote waarden van m willen onderzoeken, moeten we de limiet voor $m \to \infty$ berekenen.

$$\lim_{m\to\infty} \sqrt{\frac{\alpha}{\alpha-1}} \cdot \left(\frac{\left(\frac{\alpha}{\alpha-1}\right)^{\alpha-1}}{e}\right)^m .$$

We gebruiken hierbij de bekende limiet

$$\lim_{x\to\infty} \left(1+\frac{1}{x}\right)^x = e.$$

Uit deze limiet en het feit dat de functie $\left(\frac{\alpha}{\alpha-1}\right)^{\alpha-1}$ monotoon stijgend is[1], volgt voor $\alpha \geq 1$:

$$\left(\frac{\alpha}{\alpha-1}\right)^{\alpha-1} = \left(1+\frac{1}{\alpha-1}\right)^{\alpha-1} < e.$$

Dus:

$$\frac{\left(\frac{\alpha}{\alpha-1}\right)^{\alpha-1}}{e} < 1.$$

We kunnen nu concluderen dat voor alle $\alpha \geq 1$ geldt:

$$\lim_{m\to\infty} \sqrt{\frac{\alpha}{\alpha-1}} \cdot \left(\frac{\left(\frac{\alpha}{\alpha-1}\right)^{\alpha-1}}{e}\right)^m = 0.$$

Dit betekent dus dat, ongeacht de grootte van α, de kans op een oplossing bij m studenten en αm projecten en één opgegeven keuze per student naar 0 convergeert als het aantal studenten

[1] https://math.stackexchange.com/questions/83035/how-to-prove-11-xx-is-increasing-when-x0

naar oneindig gaat.

Het geval waarbij meer dan één keuze door de studenten mag worden opgegeven, is helaas niet op een soortgelijke manier te onderzoeken. Dit komt doordat de vraag of er een oplossing is bij meerdere keuzes een stuk ingewikkelder wordt. Het gaat dan immers niet meer direct fout als twee studenten dezelfde keuze opgeven. Een wiskundige verklaring voor deze gevallen moet dus helaas achterwege gelaten worden. Wel geven de simulaties een indicatie voor het aantal projecten dat nodig is bij kleine aantallen studenten, wat bij het ASB-seminarium de situatie is.

4.2 Resultaten bij een niet-uniforme verdeling

In werkelijkheid worden projecten meestal niet allemaal ongeveer even vaak gekozen, er is een verschil in populariteit. Om realistischere resultaten te krijgen, kiezen we daarom in een nieuwe simulatie voor een andere verdeling waarmee de posities van de drie enen in elke rij worden gekozen. Een redelijke aanname op basis van de bekende keuzeverdeling van het bachelorseminarium 2019 is de volgende verdeling:
Van de n projecten is 10% (naar boven afgerond) populairder dan de andere projecten. We stellen nu: $p = \lceil 0{,}1n \rceil$ is het aantal populaire projecten en $q = n - p = n - \lceil 0{,}1n \rceil$ is het aantal overige projecten. Nu nemen we aan dat in 20% van de gevallen uniform één van de p populaire projecten door een student gekozen wordt en in 80% van de gevallen uniform één van de q overige projecten (met gelijke kansen). Elk populair project heeft dus een kans van $\frac{0{,}2}{p}$ om gekozen te worden en elk niet-populair project heeft een kans van $\frac{0{,}8}{q}$ om gekozen te worden.
Als we met deze verdeling de drie enen in elke rij van matrix E kiezen en wederom een simulatie uitvoeren op basis van 1000 matrices, geeft dit het volgende resultaat (zie Appendix E voor de code):

	13	14	15	16	17	18	19	20	21	22	23	24
13	0.401	0.704	0.879	0.928	0.958	0.985	0.983	0.992	0.993	1.000	0.999	0.999
14	-	0.342	0.650	0.808	0.908	0.936	0.966	0.981	0.995	0.999	0.999	0.999
15	-	-	0.309	0.565	0.748	0.857	0.922	0.957	0.989	0.996	0.996	1.000
16	-	-	-	0.255	0.550	0.704	0.838	0.870	0.970	0.988	0.995	0.997
17	-	-	-	-	0.229	0.483	0.672	0.803	0.923	0.965	0.982	0.987
18	-	-	-	-	-	0.179	0.437	0.603	0.840	0.912	0.939	0.965
19	-	-	-	-	-	-	0.130	0.372	0.677	0.798	0.886	0.933
20	-	-	-	-	-	-	-	0.143	0.425	0.625	0.735	0.832

Het aantal projecten dat nodig is om ervoor te zorgen dat in 95% van de gevallen een oplossing bestaat is weer aangegeven in de volgende tabel:

Aantal studenten	Aantal projecten	Verschil
13	17	4
14	19	5
15	20	5
16	21	5
17	22	5
18	24	6
19	25	6
20	26	6

We zien dat het aantal projecten dat hier nodig is beduidend hoger ligt dan bij de uniforme verdeling. Dit was te verwachten aangezien een aantal populaire projecten ervoor zorgt dat de kans op een oplossing kleiner wordt. Elk project, dus ook een populair project, kan immers maar één keer aan een student worden toebedeeld en er blijven dan minder opties over bij de overige projecten. Bij deze verdeling zijn tussen de 30% en 40% meer projecten nodig dan het aantal studenten om te voldoen aan de 95%-eis.

5 Aanbeveling

Voor aanvang van het ASB-seminarium moeten projecten beschikbaar zijn. De niet-uniforme verdeling in paragraaf 4.2 is gebaseerd op de bekende keuzeverdeling van de eerste, tweede en derde keus van studenten bij het ASB-seminarium 2019. Bij het ASB-seminarium moeten de studenten minstens vijf projecten kiezen, dus dit vergroot de kans op een oplossing ten opzichte van paragraaf 4.2. Aangezien tussen de 30% en 40% meer projecten dan studenten bij drie keuzes al een kans van 95% op een oplossing geeft, lijkt deze marge bij vijf keuzes voldoende.
Om genoeg projecten beschikbaar te kunnen stellen kan gebruik worden gemaakt van een maximum aantal te begeleiden studenten per docent zoals beschreven in paragraaf 3.2. Hierdoor kunnen docenten zoveel projecten opgeven als ze beschikbaar hebben. Het maximum zorgt er dan voor dat de docenten niet te veel studenten moeten begeleiden.

Referenties

[1] F.M. Spieksma L.C.M. Kallenberg. *Optimalisering*. Universiteit Leiden, 2018.

[2] M.J.H. van den Bergh L.C.M. Kallenberg F.M. Spieksma. *Discrete Besliskunde*. Universiteit Leiden, 2018.

[3] R.M van Luijk. *Lineaire Algebra I*. Universiteit Leiden, 2015.

[4] Alexander Schrijver. *Combinatorial Optimization*. Springer-Verlag, Heidelberg, 2004.

Appendix A

Code van het programma voor de torenveelterm

```
---
title: "Torenveelterm scriptie"
author: "Judith"
output: pdf_document
---

# ```{r setup, include=FALSE}
# knitr::opts_chunk$set(echo = TRUE)
# ```

```{r}
#setwd("D:\\gezin\\judith\\studie\\Scriptie")
setwd("C:\\Users\\judit\\Documents\\Studie\\Bachelorscriptie")
tabel <- read.table(file = "Keuzeverdeling10.txt")
```

```{r}
N = 28 #aantal projecten
M = 14 #aantal studenten
R <- c(rep(0,N+1))
rD <- c(rep(0,N+1))
rE <- c(rep(0,N+1))

Torenveelterm <- function(A) #functie die D en E maakt uit een matrix A
{
  x <- which(A != 1, arr.ind = T) #selecteer elementen die 0 zijn
  if (nrow(x)==1){ #check of er nog 1 element 0 is
    R[1] = R[1]+1 #torenveelterm van 1 vakje is 1+x
    R[2] = R[2]+1
    return(R)
  }
  if (length(x)==0){ #check of al je elementen 1 zijn
    R[1] = R[1]+1 #torenveelterm van leeg is 1
    return(R)
  }
  i <- x[1,1] #rij van eerste element dat 0 is (v)
  j <- x[1,2] #kolom van v
  E <- A #E op blz 21
  E[i,j] <- 1 #maak v 1
  D <- A #D op blz 21
  for (k in 1:N){ #maak rij van element v 1
    D[i,k] <- 1
  }
  for (l in 1:M){ #maak kolom van element v 1
    D[l,j] <- 1
  }
  rD <- Torenveelterm(D)
  rD <- append(0,rD) #Bij D moet de torenveelterm keer x
  rE <- Torenveelterm(E)
```
```

```
  lrD <- length(rD)
  lrE <- length(rE)

  if(lrD<lrE){ #maak vectoren rD en rE even lang door nullen toe te voegen
    rD <- append(rD,rep(0,lrE-lrD))
  }
  if(lrE<lrD){ #maak vectoren rD en rE even lang door nullen toe te voegen
    rE <- append(rE,rep(0,lrD-lrE))
  }
  R <- rD + rE #de even lange vectoren kunnen nu bij elkaar opgeteld worden
  return(R)
}
Tvt <- Torenveelterm(tabel)
Tvt
```

Appendix B

Code van het programma voor de LP-formulering

```
---
title: "LP"
author: "Judith"
output: pdf_document
---

# ```{r setup, include=FALSE}
# knitr::opts_chunk$set(echo = TRUE)
# ```

```{r, echo=FALSE}
library(lpSolve)
```

```{r}
#setwd("D:\\gezin\\judith\\studie\\Bachelorscriptie")
setwd("C:\\Users\\judit\\Documents\\Studie\\Bachelorscriptie")
tabel <- read.table(file = "Keuzeverdeling.txt")
```

```{r}
A <- as.matrix(tabel) #maak matrix van tabel
m <- nrow(A) #aantal studenten
n <- ncol(A) #aantal projecten
B <- matrix(data=0, nrow=m, ncol=n) #0-matrix die gevuld wordt met
    gewichten
gewichten <- c(10,9,5,4,3,2,1) #gewichten van elke keuze
for(x in 1:7){ #vult B met de gewichten
  for(i in 1:m){
    for(j in 1:n){
      if(A[i,j]==x){
```
```

```
        B[i,j]<-gewichten[x]
      }
    }
  }
}
doelfunctie <- as.vector(t(B)) #maak van de rijen van de matrix een vector
```

```{r}
restr <- matrix(data=0, nrow=m+n, ncol=n*m) #matrix voor restricties, er zijn m+n restricties bestaande uit een vector van lengte nm
for(i in 1:m){ #restrictie 2
    p <- (i-1)*n+1
    q <- i*n
    for(j in p:q){
      restr[i,j] = 1
    }
}
for(i in 1:n){ #restrictie 3
  for(j in 1:m){
    p <- (j-1)*n+i
    restr[m+i,p] = 1
  }
}
teken <- c(rep("=",m), rep("<=",n)) #er zijn 2m restricties, m met "=" en m met "<="
een <- c(rep(1,m+n)) #de 2m restricties zijn kleiner dan of gelijk aan 1
```

```{r}
oplossing <- lp("max",doelfunctie,restr,teken,een,compute.sens = TRUE) #oplossing van het LP-probleem
indelingsmatrix <- matrix(oplossing$solution, nrow=m, ncol=n, byrow=TRUE)
indelingsmatrix #indelingsmatrix
indeling <- which(indelingsmatrix != 0, arr.ind = T)
indeling #combinaties student-project
```

Appendix C

Code van het programma voor de LP-formulering bij spreiding over docenten

```
---
  title: "LP"
author: "Judith"
output: pdf_document
---

  # ```{r setup, include=FALSE}
  # knitr::opts_chunk$set(echo = TRUE)
  # ```
```

```
10
11 ```{r, echo=FALSE}
12 library(lpSolve)
13 ```
14
15 ```{r}
16 #setwd("D:\\gezin\\judith\\studie\\Bachelorscriptie")
17 setwd("C:\\Users\\judit\\Documents\\Studie\\Bachelorscriptie")
18 tabel <- read.table(file = "Keuzeverdeling.txt")
19 testtabel <- read.table(file = "Testmatrix.txt")
20 Doceisen <- read.table(file = "Doceisen.txt")
21 ```
22
23 ```{r}
24 A <- as.matrix(tabel) #maak matrix van tabel
25 m <- nrow(A) #aantal studenten
26 n <- ncol(A) #aantal projecten
27 B <- matrix(data=0, nrow=m, ncol=n) #0-matrix die gevuld wordt met
       gewichten
28 gewichten <- c(20,18,14,12,11,10,9) #gewichten van elke keuze
29 for(x in 1:7){ #vult B met de gewichten
30   for(i in 1:m){
31     for(j in 1:n){
32       if(A[i,j]==x){
33         B[i,j]<-gewichten[x]
34       }
35     }
36   }
37 }
38 doelfunctie <- as.vector(t(B)) #maak van de rijen van de matrix een vector
39 ```
40
41 ```{r}
42 Doc <- as.matrix(Doceisen) #restrictie 4
43 rijdoc <- nrow(Doc) #aantal docenten dat eisen heeft
44 restr <- matrix(data=0, nrow=m+n+rijdoc, ncol=n*m) #matrix voor restricties
       , er zijn m+n restricties bestaande uit een vector van lengte nm
45 for(i in 1:m){ #restrictie 2
46   p <- (i-1)*n+1
47   q <- i*n
48   for(j in p:q){
49     restr[i,j] = 1
50   }
51 }
52 for(i in 1:n){ #restrictie 3
53   for(j in 1:m){
54     p <- (j-1)*n+i
55     restr[m+i,p] = 1
56   }
57 }
58 for(i in 1:rijdoc){
59   restrrij <- m+n+i #maak nieuwe restrictierij aan
60   for(j in 1:n){
```

```
61      if(Doc[i,j] == 1){ #als een project bij de eisende docent hoort
62        restrkol <- j
63        print(restrkol)
64        for(k in 1:m){
65          p <- (k-1)*n+restrkol
66          restr[restrrij,p] = 1
67        }
68      }
69    }
70  }
71  leq <- n+rijdoc
72  teken <- c(rep("=",m), rep("<=",leq)) #er zijn n+m+Doc restricties, m met
        "=" en n+Doc met "<="
73  een <- c(rep(1,m+n), rep(1,rijdoc)) #de restricties zijn kleiner dan of
        gelijk aan 1 (bij de docenten: kies maximum)
74  ```
75
76  ```{r}
77  oplossing <- lp("max",doelfunctie,restr,teken,een,compute.sens = TRUE) #
        oplossing van het LP-probleem
78  indelingsmatrix <- matrix(oplossing$solution, nrow=m, ncol=n, byrow=TRUE)
79  indelingsmatrix #indelingsmatrix
80  indeling <- which(indelingsmatrix != 0, arr.ind = T)
81  indeling #combinaties student-project
82  ```
```

Appendix D

Code van de simulatie bij uniforme verdeling

```
1   ---
2     title: "Existentie scriptie"
3   author: "Judith"
4   output: pdf_document
5   ---
6
7     # ```{r}
8     # #setwd("D:\\gezin\\judith\\studie\\Scriptie")
9     # setwd("C:\\Users\\judit\\Documents\\Studie\\Bachelorscriptie")
10    # testtabel <- read.table(file = "Testmatrix01.txt")
11    # tabel <- read.table(file = "Keuzeverdeling01.txt")
12    # ```
13  ```{r, echo=FALSE}
14  library(lpSolve)
15  ```
16
17
18  ```{r}
19  m <- 15 #aantal studenten
20  n <- 17 #aantal projecten
21  aantal <- 1000 #aantal onderzochte matrices
```

```
totex <- 0 #totale existentie
#maak restrictiematrix
restr <- matrix(data=0, nrow=m+n, ncol=n*m) #matrix voor restricties, er zijn m+n restricties bestaande uit een vector van lengte nm
for(i in 1:m){ #restrictie 1
    p <- (i-1)*n+1
    q <- i*n
    for(j in p:q){
      restr[i,j] = 1
    }
}
for(i in 1:n){ #restrictie 2
  for(j in 1:m){
    p <- (j-1)*n+i
    restr[m+i,p] = 1
  }
}
teken <- c(rep("=",m), rep("<=",n)) #er zijn m+n restricties, m met "=" en n met "<="
een <- c(rep(1,m+n)) #de 2m restricties zijn kleiner dan of gelijk aan 1
#produceer matrices en bereken LP-oplossing
for(p in 1:aantal){
  Keuzeverdeling <- matrix(0,m,n) #0-matrix
  for(i in 1:m){ #vul matrix random met 3 enen in elke rij (eerste, tweede en derde keus)
    b<- sample(1:n, size=3, replace=FALSE)
    for(j in 1:3){
      k <- b[j]
      Keuzeverdeling[i,k] <- 1
    }
  }
  doelfunctie <- as.vector(t(Keuzeverdeling))
  oplossing <- lp("max",doelfunctie,restr,teken,een,compute.sens=TRUE) #oplossing van het LP-probleem
  waarde <- oplossing$objval
  if(waarde == m){ #er is een oplossing als de waarde gelijk is aan het aantal studenten (gewichten zijn 1)
    totex <- totex + 1
  }
}
gemex <- totex/aantal #kans op een oplossing
gemex
```

Appendix E

Code van de simulatie bij niet-uniforme verdeling

```
---
  title: "Existentie scriptie"
author: "Judith"
```

```
output: pdf_document
---

  # ```{r}
  # #setwd("D:\\gezin\\judith\\studie\\Scriptie")
  # setwd("C:\\Users\\judit\\Documents\\Studie\\Bachelorscriptie")
  # testtabel <- read.table(file = "Testmatrix01.txt")
  # tabel <- read.table(file = "Keuzeverdeling01.txt")
  # ```
```{r, echo=FALSE}
library(lpSolve)
```


```{r}
m <- 19 #aantal studenten
n <- 25 #aantal projecten
aantal <- 1000 #aantal onderzochte matrices
totex <- 0 #totale existentie
#maak restrictiematrix
restr <- matrix(data=0, nrow=m+n, ncol=n*m) #matrix voor restricties, er zijn m+n restricties bestaande uit een vector van lengte nm
for(i in 1:m){ #restrictie 1
  p <- (i-1)*n+1
  q <- i*n
  for(j in p:q){
    restr[i,j] = 1
  }
}
for(i in 1:n){ #restrictie 2
  for(j in 1:m){
    p <- (j-1)*n+i
    restr[m+i,p] = 1
  }
}
teken <- c(rep("=",m), rep("<=",n)) #er zijn m+n restricties, m met "=" en n met "<="
een <- c(rep(1,m+n)) #de 2m restricties zijn kleiner dan of gelijk aan 1
#produceer matrices en bereken LP-oplossing
p <- ceiling(0.1*n)
q <- n-p
kansvector <- c(rep(0,n))
for(i in 1:p){
  kansvector[i] <- 0.20/p
}
grens <- p+1
for(j in grens:n){
  kansvector[j] <- 0.80/q
}
for(p in 1:aantal){
  Keuzeverdeling <- matrix(0,m,n) #0-matrix
  for(i in 1:m){ #vul matrix random met 3 enen in elke rij (eerste, tweede en derde keus)
```
```

```
    b <- sample(1:n, size = 3, replace = FALSE, prob = kansvector)
    for(j in 1:3){
      k <- b[j]
      Keuzeverdeling[i,k] <- 1
    }
  }
  doelfunctie <- as.vector(t(Keuzeverdeling))
  oplossing <- lp("max",doelfunctie,restr,teken,een,compute.sens=TRUE) #
      oplossing van het LP-probleem
  waarde <- oplossing$objval
  if(waarde == m){ #er is een oplossing als de waarde gelijk is aan het
      aantal studenten (gewichten zijn 1)
    totex <- totex + 1
  }
}
gemex <- totex/aantal #kans op een oplossing
gemex
```