Lectures en Lesroosters

De zoektocht naar het juiste iteratief algoritme

Class '95:

Yamie van Wijnbergen, Melissa Wijngaarden en Rob Hesselink

1 juni 2017

Abstract

Het doel van het course timetabling problem is het maken van een geldig lesrooster waarbij rekening wordt gehouden met de bijbehorde hard- en soft constraints. Om dit rooster in elkaar te puzzelen wordt gebruik gemaakt van een aangeleverde lijst met studenten, vakinschrijvingen en vakdetails. Om dit mogelijk te maken wordt er gebruik gemaakt van iteratieve algoritmes. De gebruikte algoritmes zijn Hill Climber en Simulated Annealing. Beide algoritmes beginnen met een random beginsituatie. Met behulp van verplaatsingen tussen studenten, activiteiten en lokalen wordt er een steeds beter rooster gemaakt. Hill Climber maakt gebruik van verschillende volgordes van de verplaatsingen evenals een exhaustive runner. Simulated Annealing gebruikt verschillende cooling schemes evenals verschillende volgordes van de verplaatsingen. De algoritmes draaien beiden meerdere runs van 100.000 iteraties. Hiervan worden de resultaten met elkaar vergeleken en wordt geconcludeerd dat bij deze implementatie van het probleem de Hill Climber betere roosters maakt.

1 Introductie

Voor elke student is een lesrooster van uiterst belang; om te weten welke vakken wanneer gegeven worden, hoe laat elk college plaatsvindt en in welk lokaal de student moet wezen. Een lesrooster samenstellen is alleen zo simpel nog niet. Bij het samenstellen van roosters moet er namelijk met meerdere elementen tegelijkertijd rekening worden gehouden. Carter en Laporte noemen het samenstellen van roosters het course timetabling problem. Ze beschrijven dit probleem als volgt:

"a multi-dimensional assignment problem in which students, teachers (or faculty members) are assigned to courses, course sections or classes; events (individual meetings between students and teachers) are assigned to classrooms and times" [1].

Het gaat dus om het indelen van vakken (met bijbehorende activiteiten, zoals werkgroepen, hoorcolleges en practica) en bijbehorende geregistreerde studenten in een beperkt aantal lokalen en tijdsloten tot een geheel rooster die tegelijkertijd ook voldoet aan de bijbehorende constraints. Constraints maken namelijk onder andere het verschil tussen goede en slechte roosters zodra deze als geldig verklaart wordt.

1.1 Specificaties

In deze case zijn er 29 vakken die ingedeeld moeten worden, 609 studenten die een geldig rooster nodig hebben en zeven lokalen die per tijdslot gebruikt kunnen worden. Een tijdslot is een periode van twee uur waarin een activiteit wordt gegeven. Het eerste tijdslot is van 9:00 tot 11:00 en het laatste tijdslot van 17:00 tot 19:00. Er zijn dus vijf tijdsloten per dag die gebruikt kunnen worden. Met activiteit worden de hoorcolleges, werkgroepen of practica van een vak bedoeld.

1.2 Constraints

Er kan een onderscheid gemaakt worden tussen hard constraints en soft constraints. Hard constraints zijn eisen die niet overtreden mogen worden; zodra een van de hard constraints overtreden wordt, wordt het rooster ongeldig verklaard. Hard constraints die worden gehanteerd zijn:

- Alle studenten moeten ingeroosterd worden.
- Per lokaal per tijdslot kan er maar één vak ingeroosterd zijn.
- Er mogen niet meer studenten aan een activiteit deelnemen dan de maximumcapaciteit van de activiteit.

Als soft constraints worden overtreden, is dit minder erg. Voor het overtreden van een soft constraint worden maluspunten toegekend. Deze punten worden besproken in sectie 2.2.

Het doel is dus om een geldig weekrooster te creeren met een zo hoog mogelijke puntenscore en tegelijkertijd met zo min mogelijk werkgroepen per vak.

1.3 Toestandsruimte

De toestandsruimte van het lesrooster bestaat uit drie onderdelen. Dit zijn het indelen van studenten in werkgroepen, het plaatsen van activiteiten in het rooster en het verdelen van activiteiten over lokalen.

De combinatoriek van het plaatsen van studenten in werkgroepen is te berekenen via de Stirling getallen van de tweede soort [2]. Deze getallen beschrijven het aantal mogelijkheden wanneer de afzonderlijke elementen (n) uniek zijn, maar de containers (k) niet. Ook zijn lege containers niet toegestaan. De relevante formule is in dit geval:

$$\left\{ n \atop k \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n. \tag{1}$$

Bij activiteiten en lokalen zijn de containers wel te onderscheiden, waardoor dit beschreven wordt door k^n. De resultaten van deze berekeningen staan in tabel 1.

Table 1: De grootte van de toestandsruimte

Indeling	Aantal mogelijkheden
Werkgroepen	5 x 10^{82}
Activiteiten	2 x 10^{180}
Lokalen	1 x 10^{109}
Totaal	1 x 10^{372}

Hoewel een groot deel van deze mogelijke roosters ongeldig is, blijkt uit de bovenstaande getallen dat de toestandsruimte zeer groot is. Hierdoor is zelfs na het wegnemen van ongeldige roosters een constructieve aanpak niet wenselijk, omdat dit te veel rekentijd of geheugen kost, afhankelijk van het gekozen algoritme.

2 Materiaal en Methode

2.1 Materiaal

Alle code is in Python 2.7 geschreven. Voor de grafische weergave van het rooster is een html tabel met CSS gebruikt. De gebruikte libraries zijn: Pickle, Random, Datetime, Matplotlib.pyplot, Seaborn, Os, Progressbar, Math, Copy, Operator, Networkx, Counter, Numpy, Prettytable en Webbrowser.

Als hardware hebben we drie laptops gebruikt:

- MacBook Pro (Retina, 13-inch, Mid 2014), Processor: 2,60 GHz Intel Core i5, Memory: 8 GB 1600 MHz DDR3
- HP Pavilion G6 Notebook PC (13-inch, Mid 2013), Processor: 2,10 GHz, Intel Core i7 Memory: 8 GB
- Samsung, Processor: 1,70 GHz, Intel Core i5, Memory: 4 GB

2.2 Methode

Zoals in sectie 1.3 is aangegeven, is een constructieve aanpak van dit probleem niet wenselijk. Er is derhalve gekozen voor een iteratieve aanpak. De twee gebruikte algoritmes zijn een Hill Climber (HC) en Simulated Annealing (SA).

De algoritmes beginnen altijd op een willekeurig gekozen maar geldig beginrooster. Hierbij worden activiteiten willekeurig over het rooster verdeeld, rekening houdende met een limiet van zeven activiteiten per tijdslot vanwege het aantal lokalen. De lokalen worden willekeurig toegewezen aan activiteiten. Als er meer dan één werkgroep gevormd moet worden, dan zullen de studenten gelijkmatig verdeeld worden over deze werkgroepen. Dit betekent dat bij een beginrooster het aantal studenten tussen werkgroepen maximaal met één kan verschillen. Daarnaast is ervoor gekozen om vaste groepen toe te kennen bij de werkgroepen en practica; groep 1 is dus dezelfde groep studenten bij deze twee activiteiten. Deze keuze is gemaakt omdat wanneer er groepen gemaakt worden voor een opdracht, mensen op dezelfde dag beide activiteiten dienen te volgen om te kunnen samenwerken.

Een beginrooster zal door de HC en de SA veranderen door drie soorten verplaatsingen:

1. Het verplaatsen van activiteiten naar een random ander tijdslot dat nog niet vol zit. Hierbij worden de participanten meegenomen, maar wordt het lokaal willekeurig gekozen uit de vrije lokalen in het tijdslot.
2. Het verplaatsen van studenten van één groep van een activiteit naar een (random) andere groep binnen de betreffende activiteit. Een verplaatsing van een student heeft daarmee effect op zowel de werkgroep als op de practicumgroep, omdat de student in beiden verplaatst zal worden.
3. Het verplaatsen van een activiteit naar een random ander lokaal. Een activiteit kan naar elk van de zes andere lokalen verplaatsen. Als het gekozen lokaal al in gebruik is, zullen de twee activiteiten van lokaal wisselen.

Alleen verplaatsingen die een geldig rooster produceren worden in beschouwing genomen.

Bij de Hill Climber wordt na elke verplaatsing een score berekent voor het nieuwe rooster. Als de score gelijk is gebleven of is gestegen, dan wordt het nieuwe rooster geaccepteerd. Zo niet, dan wordt de verplaatsing teruggedraaid en wordt een nieuwe verplaatsing gekozen. Het Simulated Annealing algoritme lijkt sterk op een Hill Climber, al heeft deze de mogelijkheid om een rooster met een lagere score te accepteren. Er wordt een acceptatiekans berekend, die wordt vergeleken met een willekeurig getal tussen 0 en 1. Wanneer de acceptatiekans groter is dan het willekeurig gekozen getal, wordt de verplaatsing geaccepteerd. De acceptatiekans is als volgt gedefiniëerd:

$$\text{Acceptatiekans } = e^{-(S_{i-1}-S_i)/T_i}. \tag{2}$$

Hierbij is S_{i-1} de score voor de verplaatsing en S_i de score na de verplaatsing. De acceptatiekans hangt af van de temperatuur (T_i) op iteratie i. Bij een hoge temperatuur is de kans op acceptatie van een slechter rooster hoger dan bij een lage temperatuur. Het verloop van deze temperatuur wordt bepaald door het gebruikte "cooling scheme". De gebruikte cooling schemes worden vermeld onder de relevante figuren en zijn terug te vinden in de appendix.

2.2.1 Scorefunctie

Een rooster waarbij alle vakken zijn ingeroosterd verdient 1000 punten. Vervolgens worden aan dit rooster bonus- en maluspunten toegekend.

De maluspunten worden als volgt berekend:

- Capaciteitsconflict: Als meer studenten in een lokaal zijn ingeroosterd dan de capaciteit van het lokaal toelaat, gaat er één punt af per student die de capaciteit overschrijdt.
- Roosterconflict: Als een student voor meerdere activiteiten ingeroosterd staat binnen één tijdslot, dan gaat er één punt af per conflict.
- Verspreiding: Voor elk vak gelden er 10 maluspunten als de x aantal activiteiten (hoorcolleges, werkgroepen en practica) x-1 dagen geroosterd worden, 20 maluspunten als de activiteiten x-2 ingeroosterd worden, en 30 maluspunten bij x-3.
- Avondslot: Bij het gebruik van het avondslot (17:00-19:00) worden er 50 punten afgetrokken per gebruik. Dit kan één keer per dag.

Vervolgens wordt voor elke werkgroep apart naar de verspreiding en bonuspunten gekeken, waarbij de punten genormeerd worden op het aantal werkgroepen.

Bonuspunten worden behaald door activiteiten binnen vakken zo optimaal mogelijk per student te verspreiden. Voor vakken met twee tot vier activiteiten wordt dit (ma, vr), (ma, do), (di, vr), (ma, wo, vr) of (ma, di, do, vr). Als de verdeling van activiteiten optimaal is, dan levert dit twintig bonuspunten op. Dit betekent dat er 29 keer 20 bonuspunten gehaald kunnen worden. De maximale score voor een geldig weekrooster is dus $1000 + (29 \cdot 20) = 1580$.

3 Resultaten

3.1 Random rooster

Om de puntenspreiding van random roosters weer te geven, is een histogram opgesteld met 10.000 random roosters. Deze roosters hadden een gemiddelde score van -222.4 en een standaarddeviatie van 99.95.

3.2 Hill Climber

Met het Hill Climber algoritme zijn er verschillende runs gedaan om een zo goed mogelijk rooster te krijgen met een zo hoog mogelijke score. Deze runs zijn op een aantal manieren gedaan.

Ten eerste werd de keuze gemaakt om de drie verplaatsingen besproken in sectie 2.2 achter elkaar te laten itereren, met per verplaatsing i aantal iteraties voordat de volgende verplaatsing aan de beurt was. Met runs van 50.000 iteraties, met een wisseling van soort verplaatsing na elke 1000 iteraties kwamen er scores met een gemiddeld rond de 1200.

Figure 1: Een histogram van de puntenverdeling van 10.000 random roosters. Het gemiddelde is -222.4 met $\sigma = 99.95$.

Vervolgens besloten we om in plaats van een vast aantal iteraties per verplaatsing te runnen, de verplaatsing tot zoverre te laten runnen tot er geen verbetering meer was en dan pas over te gaan op de volgende verplaatsing; de zogenoemde exhaustive runner. Met 300.000 iteraties is deze exhaustive runner toegepast met verschillende beginverplaatsingen en volgordes van de verplaatsingen. Zo beginnen we bijvoorbeeld met als eerste verplaatsing activities, als tweede studenten en als derde lokalen. Met een andere volgorde zou het dan ook kunnen zijn als eerste verplaatsing activities, als tweede lokalen en als derde studenten. Hetzelfde geldt dus ook als de twee andere verplaatsingen als beginverplaatsing worden genomen.

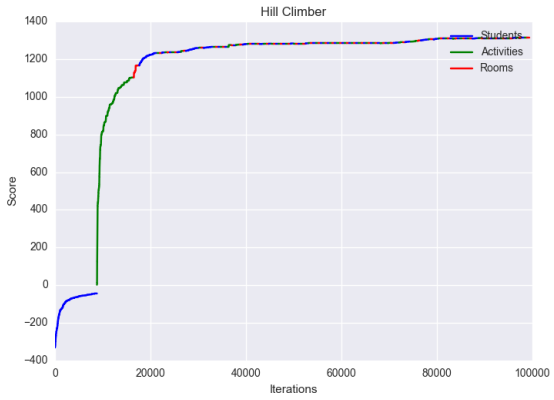

Figure 2: De beste score uit een Hill Climber. Begint verplaatsing met studenten, vervolgens activiteiten en tot slot lokalen. Score van 1315

Het optimale rooster dat uit de Hill Climber test naar voren komt is met een exhaustive runner waarbij de eerste verplaatsing die van studenten is, als tweede activiteiten en als derde lokalen. Dit resulteerde in een score van 1315 punten.

3.3 Simulated Annealing

Om een optimaal rooster te vinden zijn er met een Simulated Annealing algoritme runs gedaan met vier verschillende coolingschemes (zie appendix) van 100.000 iteraties vanuit een random rooster. Daarnaast zijn er runs gedaan waarin de volgorde van verplaatsingen varieert en ook zijn er runs gedaan met verschillende begintemperaturen. Als laatste is er ook een combinatie gedaan van Simulated Annealing en Hill Climbers, door eerst een Hill Climber te runnen en op het hoogst scorende rooster wat daaruit voortkomt een Simulated Annealing te runnen.

Uit de runs met verschillende begintemperaturen leek het aannemelijk dat een lagere begintemperatuur betere resultaten oplevert. Daarom is gebruik gemaakt van een begintemperatuur van T0 = 50 of lager.

Uit de runs met variërende volgordes van verplaatsingen bleek dat de verplaatsing van activiteiten over het algemeen het grootste en meest positieve effect heeft op de score van een rooster. Echter werd in de gedane tests met de Simulated Annealing op een random rooster de hoogste eindscore behaald met studenten als eerste verplaatsing, gevolgd door activiteiten en als laatste lokalen. Hierbij werd een score behaald van 1236 met een exponentieel coolingscheme. Deze score werd echter ruim verslagen door de Simulated Annealing die vanuit een Hill Climber rooster vertrok. Daar werd de hoogste eindscore behaald door te starten met een rooster van 1299 punten, waarna eerst activiteiten werden verplaatst, gevolgd door studenten en daarna lokalen. Hierbij werd een eindscore behaald van 1292 punten met een logaritmisch coolingscheme.

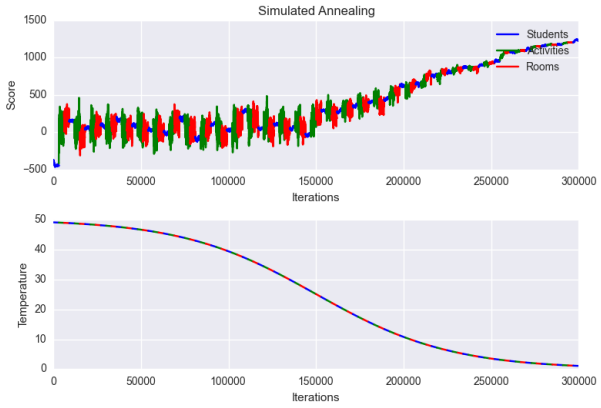

Figure 3: De beste score uit een Simulated Annealing vanuit een Random Rooster

Een opvallende bevinding is dat pas bij lagere temperaturen de score toe neemt, waar er daarvoor vooral binen een bepaalde band vrijwel horizontaal gefluctueerd werd. Om dit beter te analyseren is het aantal iteraties verhoogd naar 300.000. Echter had dit geen sterke invloed op het eerder gevonden bewegingspatroon van de grafiek.

Deze bevinding suggereert dat bij deze implementatie van het course timetabling problem een Hill Climber tot een optimaler resultaat leidt vergeleken met een simulated Annealing algoritme. Onze gevonden resultaten onderbouwen dit, doordat het hoogst gevonden re-

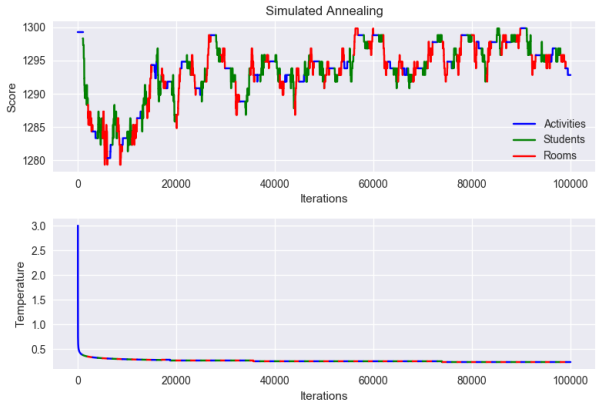

Figure 4: De beste score uit een Simulated Annealing vanuit een Hill Climber

sultaat in dit onderzoek is gevonden met behulp van een Hill Climber algoritme, namelijk een score van 1315. Een populatie van 17 Hill Climbers van 100 000 iteraties heeft een gemiddelde score van 1275 met $\sigma = 33.1$.

4 Discussie

Uit onze resultaten komt een voorkeur voor een Hill Climber algoritme ten opzichte van een Simulated Annealing algoritme naar voren. Hoewel wij veelvuldig getest hebben, kunnen we hier nog geen verdere conclusies uit trekken. Er zal meer onderzoek moeten komen, waarbij andere coolingschemes en begintemperaturen voor Simulated Annealing gebruikt moeten worden. Daarnaast hebben we vele vergelijkingen gemaakt met de volgordes van de verplaatsingen en beginverplaatsingen. Echter waren al deze scores alsnog redelijk dicht bij elkaar, waaruit blijkt dat het toch niet zo'n groot verschil maakt wanneer wat verplaatst wordt.

Dankwoord

Wij willen bedanken: Wietze Slagman voor zijn wekelijks advies, Daan van den Berg voor de kritische blikken en vragen tijdens onze presentaties en tot slot werkgroep B voor feedback tijdens de presentaties en het helpen meedenken voor mogelijke oplossingen.

Bibliografie

[1] Michael Carter and Gilbert Laporte. Recent developments in practical course timetabling. *Practice and Theory of Automated Timetabling II*, pages 3–19, 1998.

[2] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation, 1964.

Appendix

Hieronder volgen de gebruikte cooling schemes. Hierbij is T_i de temperatuur op iteratie i, T_N de eindtemperatuur en N het aantal iteraties.

Lineaire koeling:

$$T_i = T_N + (T_0 - T_N) \cdot \frac{N - i}{N} \tag{3}$$

Exponentiële koeling:

$$T_i = T_N + \frac{T_0 + T_N}{1 + e^{2\ln(T_0 - T_N)\frac{i - 0.5N}{N}}} \tag{4}$$

Logaritmische koeling (voor $\alpha \geq 1$):

$$T_i = \frac{T_0}{1 + \alpha \ln(1 + 1)} \tag{5}$$