Minor Programmeren

Lectures & Lesroosters

May 27, 2016

Een korte studie over verschillende algoritmes voor het oplossen van een lesrooster probleem

B.D.L. Chatel[a], J. Huisman[b], B.D. Sloots[c]

[a]*bastiaan.chatel@gmail.com, Universiteit van Amsterdam*
[b]*jobhuisman47@gmail.com, Universiteit van Amsterdam*
[c]*bd.sloots@gmail.com, Universiteit van Amsterdam*

Abstract

In dit onderzoek wordt het vraagstuk van "Lectures en Lesroosters" behandeld waar een lesrooster gemaakt wordt met behulp van een viertal aan algoritmes die allen het probleem op een andere manier aan pakken. Dit onderzoek behandelt behalve een random functie, ook een Hill Climber, Simulated Annealing en tot slot een Genetic Algorithm die uiteindelijk het beste resultaat opleverde.

1 Introductie

De 'Lectures & Lesroosters' case is een roostering probleem, welke als eerste werd uitgegeven in 2015 door J. Oud, w. Bohlken en R. Mokveld. 'Lectures & Lesroosters' betreft een case waarin een weekrooster gemaakt moet worden. Op het eerste gezicht lijkt het maken van roosters, dienstregelingen en vluchtschemas een redelijk logische taak. Echter wordt het maken van een rooster snel een complexe taak. Bijvoorbeeld in het geval van een lesrooster zijn er meerdere variabelen waarmee rekening gehouden dient te worden. Denk hierbij aan individuele studieprogramma's, beschikbaarheid van lokalen, capaciteit van lokalen, werkgroepindelingen en verdeling van colleges over de week. Doordat er rekening gehouden dient te worden met meerdere variabelen is het vinden van een goede oplossing complex, en is de kans klein om een goede oplossing te vinden.

1.1 Case

In de case 'Lectures & Lesroosters' is het de opdracht een zo goed mogelijk weekrooster te maken. Een weekrooster bestaat uit 5 dagen, met ieder 4 tijdsslots. In combinatie met 7 beschikbare lokale zijn de colleges, werkcolleges en practica te verdelen over 140 momenten, genaamd zaalslots. In totaal zijn er 29 in te roosteren vakken, gevolgd door 610 studenten. Een weekrooster is geldig wanneer iedere hoorcollege, werkcollege en practicum ingedeeld is in een eigen zaalslot. Een geldig weekrooster levert 1000 punten op. Naast de 1000 punten voor een geldig weekrooster kunnen er ook bonus- en maluspunten behaald worden volgens de volgende scorecriteria:

Maluspunten

Dubbelroostering: wanneer een student meerdere activiteiten per tijdsslot heeft wordt er 1 minpunt per student gerekend.
Meerdere activiteiten per dag: idealiter wordt er per vak 1 activiteit per dag ingeroosterd. Zodra dit overschreden wordt worden er 10 minpunten per keer berekend.
Capaciteitsoverschrijding: wanneer de zaalgrootte niet toereikend is voor de ingeroosterde groep studenten geldt er 1 minpunt per leerling die niet in het lokaal past.

Bonuspunten

Verdeling activiteiten over de week. Idealiter zijn de activiteiten per vak over de week verdeeld. Een vak van 2 colleges is idealiter op ma-do of di-vr ingedeeld, een vak van 3 activeiten op ma-wo-vr en een vak van 4 activiteiten op ma-di-do-vr. Een vak van 5 activiteiten heeft idealiter iedere werkdag 1 activiteit. Voor vakken met meer dan 1 activiteit geldt dat er 20 bonuspunten gerekend worden wanneer een ideale verdeling behaald is.

Toevoeging op scorecriteria

Wanneer er binnen een vak sprake is van meerdere werkgroepen, bijvoorbeeld een groep a, en b, dan worden regelingen omtrent 'meerdere activiteiten per dag' alsook de 'verdeling activiteiten over de week' per werkgroep gerekend. Het kan dus voorkomen binnen een vak dat er vanwege de verdeling over de week 20 pluspunten toegekend worden voor groep a en 20 minpunten bij groep b vanwege meerdere activiteiten op een dag.

In dit onderzoek hebben we de 'Lectures & Lesroosters' case geprobeerd op te lossen middels 4 soorten benaderingen: random generation, Hill Climber, Simulated Annealing en een Genetic Algorithm. Hierin trachten we de verschillende methoden te vergelijken, om er achter te komen welk algoritme het beste resultaat geeft voor deze case. Onze verwachting hierbij zal zijn dat het Genetic Algorithm het beste passend is voor de case.

2 Theorie

De case 'Lectures & Lesroosters' kan gerekend worden als een constrained optimization problem. De 'hard constraints' worden gevormd door de regels in het creëren van een geldig rooster. De 'soft constraints' worden gevormd door de verschillende scorecriteria. Het doel van dit onderzoek is dan ook om te onderzoeken wat voor algoritme binnen deze constraints het beste rooster kan genereren.

2.1 Heuristieken

In dit onderzoek is er gekozen om een heuristiek toe te passen op de werkgroepindeling. We willen vooraf aan het roosteren een vaste werkgroepindeling maken. Door leerlingen in vaste groepen te zetten wordt de state-space van ons probleem aanzienlijk kleiner omdat we nu alleen nog de colleges en groepen hoeven te verdelen over de beschikbare zaalslots. Ook kunnen we op deze manier voor een aantal score-evaluaties de scores per vak of groep evalueren in plaats van per leerling omdat dubbele roostering binnen een vak dan uitgesloten wordt. Hierdoor wordt het programma sneller.
Voor de grootte van de werkgroepen moeten we een afweging maken tussen de grootte van de groep en het aantal groepen. Een grote groep levert eerder minpunten op door de beperkte capaciteit van de lokalen. Veel werkgroepen maken het probleem complexer doordat er meer lessen ingeroosterd moeten worden. We baseren het aantal werkgroepen op het percentage studenten dat overblijft als alle werkgroepen maximaal gevuld zijn. We redeneren dat we liever te weinig dan teveel studenten in een werkgroep hebben, zodat werkgroepen in veel verschillende zalen kunnen worden ingedeeld, zonder dat dit minpunten oplevert.
Het blijkt dat als het aantal leerlingen dat niet wordt ingedeeld niet groter is dan 21% van de maximale werkgroepgrootte er een aanzienlijk deel van de werkgroepen kleiner zijn dan 22 leerlingen. Al deze werkgroepen kunnen in alle lokalen worden ingedeeld, terwijl het aantal minpunten beperkt blijft. We verwachten dat door deze opties de verdeling over de week eerder geoptimaliseerd wordt.

2.2 Toestandsruimte

Na toevoegen van de heuristieken blijven er in totaal 124 unieke activiteiten over die geroosterd moeten worden over 140 zaalslots. Dit resulteert in een state-space van 140 boven 124. Er zijn dus $4.27 \cdot 10^{20}$ unieke lesroosters te maken.

$$\binom{140}{124} = \frac{140!}{124! \cdot 16!} = 4.27 \cdot 10^{20} \tag{1}$$

Het theoretisch maximum wordt binnen deze toestandsruimte bepaald door het aantal werkgroepen, wat het maximaal aantal bonuspunten bepaald. Na toevoegen van bovenstaande heuristiek zijn er 45 werkgroepen. Daarmee ligt het theoretisch maximum op een score van 1900 punten bij een puntentelling van 20 punten per correct geroosterde activiteit.

2.3 Datastructuur

We maken de roosters in een dict van dicts. De informatie voor de 140 zaalslots zijn de keys die leiden naar de verschillende zaalslots. In het zaalslot wordt een dict gedaan met de unieke vaknaam. De value van deze dict is een list met unieke studentnummers van alle studenten die het vak volgen.

3 Methoden

3.1 Assumpties

In het onderzoek naar deze case worden een aantal aannames gemaakt. In de eerste plaats zijn de algoritmes geschreven op een constructieve manier, zodat er geen ongeldige roosters gegenereerd worden. Dit wordt na toepassen van algoritmes dus ook niet gecontroleerd. De functie die scores toekent geeft elk rooster dus 1000 punten en past vervolgens de overige scorecriteria toe.
Daarnaast ligt de nadruk binnen dit onderzoek niet op het optimaliseren van de code. Het vergelijken van de algoritmes op basis van de runtime zou in dat geval niet correct zijn. In plaats van de runtime worden de algoritmes vergeleken aan de hand van het aantal evaluaties als maat voor de tijd. Dit omdat we er vanuit gaan dat het evalueren van een rooster de meeste tijd zal kosten, ook als de code geoptimaliseerd is.
Voor ons probleem hebben we uiteindelijk vier verschillende algoritmes gebruikt om roosters te genereren. Een random functie, Hill Climber, Simulated Annealing en een Genetic Algorithm. Om een vergelijking te kunnen maken wordt ieder algoritme 26 keer uitgevoerd, met 5000 evaluatierondes per algoritme.

3.2 Random

De random functie maakt volledig random roosters. Iedere activiteit wordt in een random beschikbaar zaalslot geroosterd. Als alle vakken geroosterd zijn wordt het rooster gescoord. De beste 26 roosters worden bewaard.

3.3 Hill Climber

Per evaluatieronde vindt er een mutatie plaats. Een mutatie bestaat uit het verwijderen van een random vak, dat vervolgens weer random wordt ingeroosterd. Het is in theorie dus mogelijk dat een mutatie niet tot een verandering leidt.

3.4 Simulated Annealing

Dit algoritme is in principe gelijk aan de Hill Climber waarbij er een kans bestaat dat mutaties, die een mindere score tot gevolg hebben, ook aangenomen kunnen worden. Dit om een algoritme de kans te geven uit een lokaal maximum te komen. Dit wordt bewerkstelligd door een exponentiële temperatuursfunctie met de formule:

$$T = \alpha^i \tag{2}$$

Hierbij is T de temperatuur, alpha heeft een waarde van 0.99951 en i is het aantal iteraties. Als de temperatuur hoger is dan een random getal tussen de 0 en de 1 wordt de negatieve verandering aangenomen. Omdat de temperatuur per iteratie afneemt, neemt de kans op het aannemen van een slechtere score ook af.

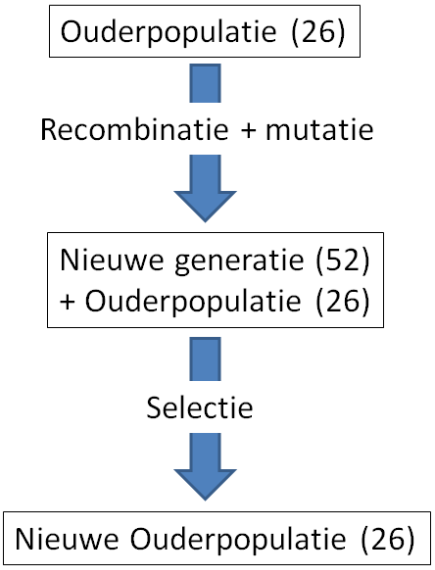


Figure 1: Schematische weergave van een cylclus van het Genetic Algorithm

3.5 Genetic Algorithm

Het Gentic Algorithm start met een ouderpopulatie van 26 random roosters. Hieruit worden 52 recombinaties gemaakt. Een recombinatie houdt in deze situatie in dat er twee of drie random dagen van de ene ouder met de andere twee dagen van de andere ouder worden gecombineerd. Nadien word nagegaan

of iedere unieke activiteit eenmaal in het rooster aanwezig is, zo niet worden de conflicten gerepareerd middels verwijdering of bijvoegen van de activiteit. Daarnaast past de functie elke recombinatieronde een enkele mutatie toe, op dezelfde manier als de Hill Climber. Zoals te zien is in figuur 1 wordt iedere recombinatie de populatie verdrievoudigd. Uit deze populatie wordt op basis van een onderstaande selectiefunctie een nieuwe oudergeneratie geselecteerd. Iedere keer dat er een nieuwe oudergeneratie gevormd is, zijn er dus twee evaluatierondes gepasseerd.

$$P = 0.6 + \frac{0.6}{i + 1} \tag{3}$$

Deze formule houdt in dat het beste rooster een kans van 120% heeft om mee genomen te worden naar de volgende generatie en de slechtste een kans van 61%. Dit zodat het beste roosters niet verloren gaat. Het is nodig voor het tegen gaan van een uiteindelijke uniforme populatie dat slechte roosters ook meegenomen kunnen worden, zodat er diversiteit en ruimte voor groei blijft.

4 Resultaten

Om de verschillende algoritmen te vergelijken laten we elk algoritme 26 roosters ontwikkelen. Door ook random roosters te maken kunnen we zien hoeveel beter het algoritme is ten opzichten van een willekeurige oplossing.

4.1 Random

De 5000 willekeurig gemaakte roosters plotten we in een histogram om de verdeling van de scores te visualiseren, te zien in figuur 2. De beste 26 roosters zijn, tienmaal vergroot, in rode bars weergegeven, ten behoeve van de zichtbaarheid. Het resultaat lijkt, zoals verwacht, normaal verdeeld te zijn.

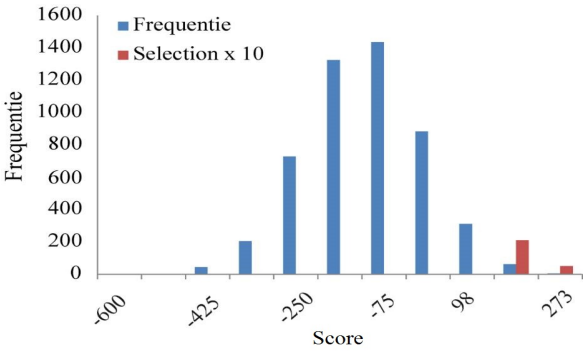


Figure 2: Histogram van 5000 willekeurige roosters, gegenereerd met de random functie. De beste 26 zijn tienmaal vergroot weergegeven.

4.2 Hill Climber

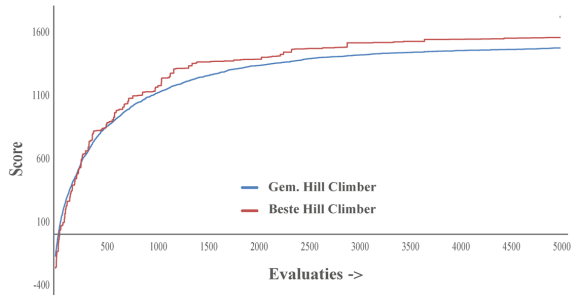
Het verloop van de Hill Climber is te zien in figuur 3a. De gemiddelde waarde van de 26 roosters is geplot tegen het aantal iteraties. Ook wordt het beste eindresultaat geplot. Dit geeft een indruk van de ontwikkeling van een enkel rooster. Hierdoor is het zichtbaar hoe lang het kan duren voordat er een verbetering gevonden wordt. De functie van de gemiddelde heeft de vorm van een logaritmische functie.

4.3 Simulated Annealing

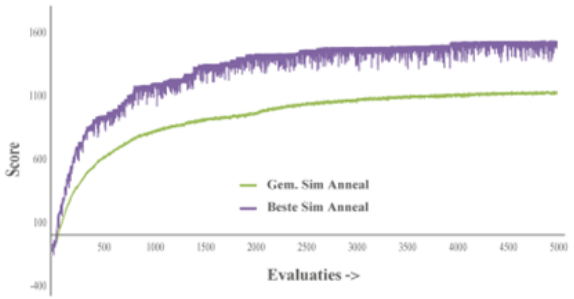
Voor de Simulated Annealing hebben we ook de gemiddelde waarde tegen het aantal evaluaties geplot. Dit is te zien in figuur 3b.
Ook de maximale eindscore is geplot. Hierbij is duidelijk te zien dat ook mindere resultaten worden geaccepteerd, wat resulteerd in een kartelig reliëf van de plot.

4.4 Genetic Algorithm

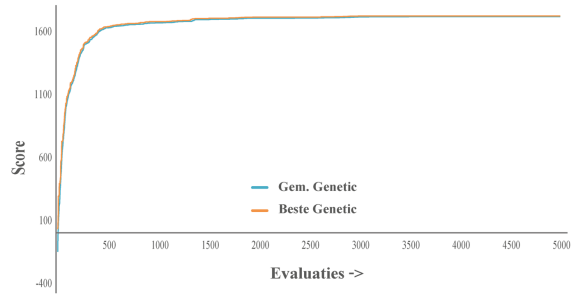
De ontwikkeling van het Genetic Algorithm kent maar 2500 cycli van recombinatie en selectie. In de data is elke cyclus als 2 evaluaties opgeslagen zodat de data wel vergelijkbaar is. De plot van de waarden tegen het aantal evaluaties is weergegeven in figuur 3c. Na een scherpe steiging vlakt de groei sterk af na 500 iteraties.



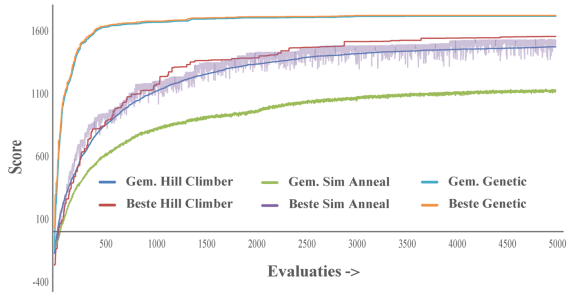
(a) Plot voor het 'Hill Climber' algoritme



(b) Plot voor het 'Simulated Annealing' algoritme



(c) Plot voor het Genetic Algorithm



(d) Alle drie de algoritmen

Figure 3: Ontwikkeling van gemiddelde en maximale score tegen het aantal evaluaties voor 26 willekeurige roosters

4.5 Vergelijken van resultaten

Als alle drie de algoritmen in een grafiek worden geplot (te zien in figuur 3d), is te zien dat het Genetic Algorithm de snelste steiging vertoont. De beste resultaten voor de Hill Climber en de Simulated Annealing zijn vergelijkbaar, al is het verloop erg verschillend.
In figuur 4 zijn de resultaten en gemiddelden van de verschillende algoritmen te zien. Ook is er een Historgram geplot van het eindresultaat van de verschillende algoritmen. Er is duidelijk te zien dat het Genetic Algorithm algoritme een kleine spreiding heeft wat betreft de resultaten. Ook is het opvallend dat een aantal resultaten van de 'Simulated Annealing' bijzonder laag zijn uitgevallen. Van de willekeurige roosters zijn ook de beste 26 weergegeven.

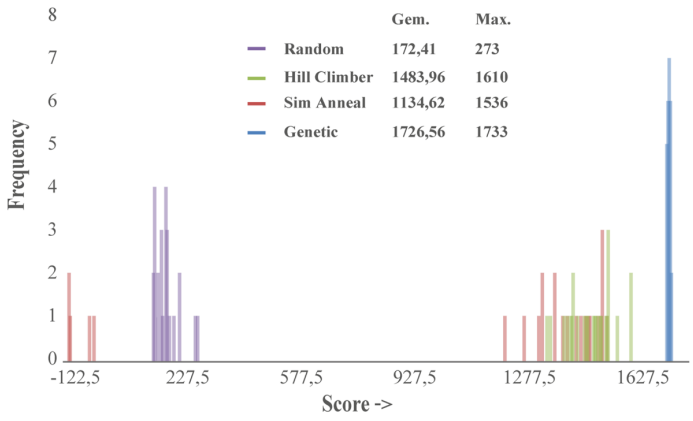


Figure 4: Histogram van de 26 beste eindscores van de vier verschillende benaderingen van het probleem

5 Conclusie

5.1 Discussie

In dit onderzoek is de toepassing van een viertal algoritmes op de case 'Lectures Lesroosters' getest. De random functie genereert zoals verwacht een diverse populatie die normaal verdeeld lijkt. Het Hill Climber algoritme functioneert en bereikt verschillende maxima. Ook het Simulated Annealing algoritme bereikt verschillende maxima en vertoont daarbij ook een grotere spreiding. Het Genetic Algorithm bereikt het hoogste maximum en vertoont ook de steilste ontwikkeling in score over de evaluaties. Op basis van de behaalde scores en het scoreverloop lijkt het Genetic Algorithm dus het meest geschikt voor de case 'Lectures Lesroosters'.
Echter kunnen er op deze studie enkele aanmerkingen gemaakt worden.
Ten eerste kan er voor ieder algoritme nog vooruitgang geboekt worden op het afstellen van de parameters. Hier hebben we bij ons onderzoek weinig tijd aan besteed. Bijvoorbeeld in het gebruik van het Simulated Annealing algoritme, speelt het temperatuurschema een grote rol. Hierdoor zou een lokaal optimum ontweken kunnen worden door het aan nemen van negatieve veranderingen. Daarom is het merkwaardig dat de Simulated Annealing niet hoger eindigt dan de Hill Climber. Daarnaast kan er gekeken worden of er in zowel de Hill Climber als de Simulated Annealing of er sneller een maximum wordt bereikt wanneer er de mutatiestap groter is.
Ten tweede leidt de keuze om de heuristiek op de werkgroepindeling toe te voegen tot een verkleining van de state space en gemakkelijker wisselen van activiteiten. Echter zou het kunnen dat daardoor het behalen van het theoretisch maximum lastiger is omdat wisselen tussen werkgroepen ten faveure van de bonusregeling niet mogelijk is.

5.2 Conclusie

In dit onderzoek werd er onderzocht welk algoritme het best passend was bij het 'Lectures & Lesroosters' dilemma. Naar aanleiding van de resultaten valt te stellen dat het Genetic Algorithm het beste resultaat oplevert, zowel met maximale score als snelste stijging van score, aangezien de groei al na ± 500 evaluaties stagneert.
***De Hill Climber is het een na beste algoritme, gevolgd door de Simulated Annealing. Kanttekening hierbij is dat de afstelling van parameters van de Simulated Annealing zeer waarschijnlijk niet optimaal is.