Inhoudsopgave

Inleiding

Ter afsluiting van de studie Informatica aan de Rijksuniversiteit te Leiden dient de student een afstudeerproject te doen. Het afstudeerverslag dat voor u ligt is het resultaat van een afstudeerproject over "Algoritmen voor het maken van roosters". Het onderwerp werd mede ingegeven door het feit dat de auteur in het onderwijs werkzaam is en daardoor regelmatig geconfronteerd werd met (de gevolgen van) slechte roosters.

Na een inleesperiode waarin het gebruik van genetische algoritmen voor roosters in het algemeen bestudeerd werd, werden een tweetal schoolroosterproblemen aan een nader onderzoek onderworpen. Voor het maken van een examenrooster, dat zoveel als mogelijk aan bepaalde eisen en wensen van studenten moest voldoen, werd een genetisch algoritme ontworpen. Er werd tevens uitgebreid geëxperimenteerd met verschillende waarden voor diverse mutatie- en crossover-operatoren. Vervolgens werd het maken van een surveillancerooster voor docenten bij een gegeven examenrooster nader bestudeerd. Voor het bijzondere geval van drie, elkaar in tijd niet overlappende, examens op één dag leverde dit een aantal interessante theoretische resultaten op bij het minimaliseren van het aantal keren dat een docent voor slechts één beurt naar school dient te komen.

Gedurende het afgelopen jaar heb ik veel steun gehad aan de inspirerende opmerkingen van de begeleidende docenten mw. dr. I.G. Sprinkhuizen-Kuyper en dr. W.A. Kosters. Beiden wil ik daar hartelijk dank voor zeggen.

Samenvatting

Twee schoolroosterproblemen zijn aan een nader onderzoek onderworpen:

MESP In het "Modular Exam Scheduling Problem" (MESP) dient op grond van inschrijvingen van studenten voor examens een examenrooster gefabriceerd te worden, dat voor de studenten zo weinig mogelijk vervelende consequenties heeft. Deze vervelende consequenties kunnen bestaan uit twee of meer examens op hetzelfde tijdstip, op dezelfde dag of op twee achtereenvolgende dagen. Het MESP-probleem blijkt NP-volledig te zijn. Met behulp van een genetisch algoritme zijn echter toch goede resultaten te behalen.

TISP In het "Teacher Invigilation Scheduling Problem" (TISP) moet op grond van een gegeven examenrooster een surveillancerooster gemaakt worden voor de surveillerende docenten. Er zijn constraints waaraan voldaan moet worden, zoals:

- zo mogelijk moet bij een examen minstens één docent surveilleren, die het betreffende vak ook geeft
- de beurten dienen eerlijk over de docenten verdeeld te worden
- docenten zijn niet altijd voor surveillance beschikbaar
- het aantal keren dat een docent voor slechts één surveillancebeurt naar school moet (een zogenaamde 1-beurt) komen, dient minimaal te zijn

Als aan de eerste drie constraints niet voldaan behoeft te worden, dan bestaat er voor het geval dat er drie examenrondes per dag zijn, een inroosterstrategie die gegarandeerd een rooster met het minimale aantal 1-beurten oplevert. Wanneer aan alle beperkingen voldaan moet worden, zal de negatieve invloed hiervan op het aantal 1-beurten in het algemeen niet al te groot zijn.

1 Wat is een genetisch algoritme?

1.1 Inleiding

In zijn werk, *The Origin of Species* (1859), verdedigde Charles Darwin het principe van *evolutie door natuurlijke selectie*:

- de natuur produceert individuen met verschillende karaktereigenschappen
- elk individu neigt er toe zijn karaktereigenschappen door te geven aan zijn nakomelingen
- de beste individuen — die met de beste eigenschappen — zullen de meeste nakomelingen voortbrengen, zodat de populatie van individuen op den duur steeds meer van deze goede eigenschappen zal gaan vertonen
- over een lange periode bezien kan er verandering van eigenschappen optreden, doordat individuen zich aanpassen aan hun natuurlijke omgeving

Het doorgeven en veranderen van karaktereigenschappen vindt plaats tijdens het proces van celdeling. In hogere planten en dieren bevat elke cel een kern, die op zijn beurt een aantal chromosomen bevat. Deze chromosomen zijn de dragers van de karakterbepalende eigenschappen (genen) van individuen, welke bij het proces van celdeling aan de nieuwe cellen worden doorgegeven en uitgewisseld.

Een *genetisch algoritme* (GA) is een zoekalgoritme, dat op bovenstaand principe gebaseerd is. Essentieel is dat het zoekalgoritme niet, zoals bij meer traditionele oplossingsmethoden, kandidaatoplossingen stuk voor stuk bekijkt maar met een hele populatie van kandidaatoplossingen werkt. Deze kandidaatoplossingen zijn als strings gecodeerd en kunnen gezien worden als de chromosomen, die elk uit een aantal genen bestaan. In plaats van een gen spreekt men ook wel van een allele. De representatie van de chromosomen en de genen hangen af van het onder handen zijnde probleem. Aan elk chromosoom wordt een *fitness* toegekend, die aangeeft hoe goed deze kandidaatoplossing is. Deze toekenning geschiedt met behulp van een zogenaamde *evaluatiefunctie*. Een nieuwe generatie van kandidaatoplossingen wordt gegenereerd door het toepassen van genetische operatoren op de populatie van kandidaatoplossingen. In het algemeen worden drie operatoren onderscheiden:

1. *selectie*: selecteren van strings uit de huidige populatie op grond van hun fitness
2. *mutatie*: een of meer genen van een chromosoom worden met een bepaalde (kleine) kans veranderd
3. *crossover*: uit twee of meer chromosomen worden met een bepaalde kans nieuwe chromosomen gecreëerd

Soms wordt ook nog wel *inversie* gebruikt, waarbij voor een deelstring van een chromosoom de volgorde van de genen wordt omgedraaid.
Na creatie van een aantal nieuwe populaties convergeert het programma naar een (hopelijk) redelijk goed optimum. Ten opzichte van de meer traditionele zoekalgoritmen (depth-first search, best-first search, hill climbing etc.) zal de

fitness van de gevonden oplossing waarschijnlijk dichter bij het werkelijke optimum liggen. Dit komt omdat nu niet lokaal (één kandidaatoplossing tegelijk) gezocht wordt, maar informatie van meerdere kandidaatoplossingen uit een groter gedeelte van de zoekruimte met elkaar wordt gecombineerd.

1.2 Genetische operatoren

In deze paragraaf zullen de operatoren bij een GA nader beschouwd worden.

1.2.1 Selectie

Door de selectie-operatie vindt "natuurlijke selectie" plaats: chromosomen met een hoge fitness hebben een grotere kans in de volgende populatie terecht te komen. Chromosomen uit de populatie kunnen op diverse manieren geselecteerd worden. Enkele van de meest gebruikte methoden zijn [Mic94, Hoofdstuk 4]:

a. *Roulette wheel.* De populatie wordt afgebeeld op een roulette, waarbij elke string correspondeert met een sector waarvan de oppervlakte overeenkomt met de relatieve fitness van de string. De relatieve fitness van een string is hierbij het quotiënt van de fitness van die string en de som van fitnesses van alle strings in de populatie. Door herhaaldelijk aan de roulette te draaien worden individuele strings geselecteerd volgens het principe van een *stochastische steekproef met teruglegging.* Duidelijk is dat strings met een hoge fitness een grotere kans hebben om één of meer keer in de nieuwe populatie voor te komen.

b. *Stochastische steekproef met teruglegging (aangepaste versie).* Voor elke string S_i wordt het quotiënt f_i/f_{avg} uitgerekend, waarbij f_i de fitness van S_i is en f_{avg} de gemiddelde fitness van alle strings in de populatie. Van elke string wordt sowieso dat aantal kopieën in de nieuwe populatie geplaatst, dat overeenkomt met het gehele gedeelte van het quotiënt. Vervolgens wordt van elke string nog een (extra) kopie in de nieuwe populatie geplaatst met een kans die gelijk is aan het gedeelte achter de decimale punt van f_i/f_{avg}. Deze selectie kan met één draai aan het roulette wheel gerealiseerd worden door de sectoren die met de strings corresponderen random op het roulette wheel te leggen en een wheel met pop_size wijzertjes te gebruiken.

c. *Rangnummers.* De strings krijgen een rangnummer op grond van hun fitness. In plaats van de individuele fitnesses worden deze rangnummers gebruikt bij de selectie. Dit wordt gedaan om te voorkomen dat door een chromosoom met een uitzonderlijk hoge fitness in vergelijking met de andere chromosomen de populatie te snel convergeert naar een populatie die alleen uit kopieën van dit ene chromosoom bestaat. Hierdoor zou zeer waarschijnlijk slechts convergentie naar een lokaal optimum plaats vinden, terwijl de informatie opgeslagen in de overige chromosomen van de populatie nauwelijks gebruikt wordt. Voor het bepalen van het kleinste en grootste rangnummer binnen de populatie en het afbeelden van de fitnesses naar een lineaire schaling tussen deze twee uitersten bestaan verschillende methoden. Eén methode gaat uit van een door de gebruiker

opgegeven bovengrens MAX van het verwachte aantal nakomelingen van het beste chromosoom. De lineaire schaling wordt nu zo aangebracht dat een chromosoom met gemiddeld fitness precies één nakomeling in de nieuwe populatie mag verwachten. Een andere methode gaat uit van een door de gebruiker opgegeven parameter q en een lineaire functie

$$prob(rank) = q - (rank - 1)r$$

waarbij r een constante is en q en r zo gekozen worden dat $prob(rank) \in [0, 1]$ en $\sum_{i=1}^{n} prob(i) = 1$. Deze functie geeft de kans dat een chromosoom met ranking $rank$ in een volgende populatie terugkeert ($rank = 1$ geeft hierbij het beste chromosoom aan en $rank = n$ het slechtste chromosoom, waarbij n de populatiegrootte is). Door het kiezen van geschikte waarden voor q en r kan er meer of minder druk worden uitgeoefend op het kiezen van chromosomen met een hoge fitness.

d. *Elitisme.* Voordat een selectiemethode wordt toegepast, wordt de string met de hoogste fitness automatisch al gekopieerd naar de volgende populatie. Hierdoor wordt er voor gezorgd dat de beste oplossing in elke volgende populatie nooit slechter kan zijn dan in de huidige populatie.

e. *Veroudering.* Voor elk chromosoom c wordt een teller bijgehouden, waarwan de waarde initieel gelijk wordt gemaakt aan $f(c)/f_{\text{avg}}$. $f(c)$ is hierbij de fitness van c en f_{avg} de gemiddelde fitness van alle chromosomen in de populatie. Steeds wanneer het chromosoom wordt geselecteerd voor reproductie wordt de teller met een constante hoeveelheid afgelaagd. Zodra de teller negatief geworden is komt het chromosoom niet langer meer in aanmerking voor selectie.

1.2.2 Crossover

Crossover is een operatie om uit twee of meer chromosomen nieuwe chromosomen te creëren. Elke gen in een nieuw chromosoom is afkomstig van één van de "ouder"-chromosomen. Hierdoor wordt informatie tussen de chromosomen uitgewisseld waardoor nieuwe en mogelijk betere kandidaatoplossingen worden gegenereerd. Als bijvoorbeeld in de strings

$$(* * * * * \alpha_1 \alpha_2 \alpha_3 * * * * * * *)$$

en

$$(* * * * * * * * * * \beta_1 \beta_2 * **)$$

de α's en β's stukjes genetisch materiaal bevatten van een (bijna) optimale oplossing en de $*$'s willekeurige genen voorstellen, dan zou door een crossover-operatie als nieuw chromosoom

$$(* * * * * \alpha_1 \alpha_2 \alpha_3 * * \beta_1 \beta_2 * **)$$

kunnen ontstaan. Dit laatste chromosoom heeft meer interessant genetisch materiaal op de juiste plaats staan en daardoor een hogere fitness. Crossover wordt door velen beschouwd als de meest essentiële operatie binnen een GA.
Enkele veel gebruikte crossover-operaties zijn:

a. *één-punts-crossover*. Er wordt een willekeurig punt gekozen en vervolgens worden de deelstrings achter dit punt tussen beide ouderstrings uitgewisseld.

b. *meer-punts-crossover*. Er worden nu meerdere crossover-punten gekozen en er wordt een nieuwe string gecreëerd door deelstrings steeds om en om van beide ouders te kiezen.

c. *uniforme crossover*. Voor elk gen van het nieuw te creëren chromosoom wordt random bepaald welk ouder-chromosoom zijn gen doorgeeft aan het kind-chromosoom.

In feite zijn één-punts-crossover en meer-punts-crossover bijzondere gevallen van uniforme crossover.

1.2.3 Mutatie

Mutatie is een operatie die met kleine kans een gen van een chromosoom verandert in een andere (toelaatbare) waarde. De mutatie-operatie zorgt er voor dat in de omgeving van een kandidaatoplossing naar een betere oplossing wordt gezocht. De kans op mutatie moet niet al te groot zijn, omdat het GA dan vervalt tot random zoeken.

1.2.4 Inversie

Inversie is evenals mutatie een unaire operator: tussen twee aselect gekozen punten in een string wordt de volgorde van de genen omgedraaid, maar wel zo dat de oorspronkelijke betekenis van elk gen onthouden wordt. Inversie kan van belang zijn bij het vinden van de beste rangschikking voor zogenaamde *building blocks*. Wat building blocks zijn en waarom vorming hiervan belangrijk is, wordt in de volgende paragraaf uitgelegd.

1.3 Waarom werkt een genetisch algoritme?

De reden waarom een GA werkt zijn we (informeel) bij de beschrijving van de crossover-operatie al tegengekomen: substrings van twee chromosomen, die verantwoordelijk zijn voor een hoge fitness, kunnen door middel van de crossover-operatie in één nieuw chromosoom terecht komen, dat daardoor een nog hogere fitness krijgt. In feite worden dus twee stukjes informatie, die elk bijdragen tot een goede oplossing, gecombineerd tot een groter stuk.
Een meer formele reden waarom een GA werkt wordt onder ander in [Gol89] gegeven voor het (klassieke) geval dat een string alleen uit nullen en enen kan bestaan. Hiertoe definiëren we een *schema* als een template dat een deelverzameling van strings beschrijft met overeenkomsten op bepaalde stringposities. Een voorbeeld van een schema is $**10**1*$, dat de verzameling van alle strings van lengte 8 beschrijft met een 1 op de derde en zevende positie en een 0 op de vierde positie. De orde $o(H)$ van een schema H is gelijk aan het aantal gedefinieerde posities in het schema. De lengte $\delta(H)$ van een schema H is gelijk aan de maximale afstand tussen twee gedefinieerde posities. Als bijvoorbeeld H

gelijk is aan $**10**1*$, dan is $o(H) = 3$ en $\delta(H) = 4$. Volgens het *Schema Theorema* geldt nu dat:

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{\overline{f}} \left(1 - \frac{p_c \delta(H)}{l-1}\right) (1 - p_m)^{o(H)}$$

waarbij:

- $m(H, t)$ is het aantal voorkomens van schema H op tijdstip t
- $f(H) / \overline{f}$ is de reproduktie-faktor, de verhouding tussen de gemiddelde fitness van alle strings die schema H representeren en de gemiddelde fitness van de populatie
- $1 - \frac{p_c \delta(H)}{l-1}$ is de overlevingskans van een schema met lengte $\delta(H)$ onder éénpunts-crossover, met p_c de kans dat op een string crossover wordt uitgevoerd en l de lengte van de string
- $(1 - p_m)^{o(H)}$ is de overlevingskans van een schema van orde $o(H)$ onder mutatie, met p_m de kans dat een bit van de string wordt gemuteerd

Een bewijs voor het Schema Theorema is onder andere te vinden in [DanNig93]. Uit het Schema Theorema volgt dat korte, lage orde en boven gemiddelde schema's exponentieel toenemende sporen in volgende generaties nalaten. Dergelijke schema's worden ook wel *building blocks* genoemd.

1.4 Roosterproblemen en GA's

1.4.1 Inleiding

Roosterproblemen behoren tot een categorie van optimaliseringsproblemen waar nog veel aan te onderzoeken valt. Enkele bekende roosterproblemen zijn:

- *schoolroosters*:
  - *lesrooster*: voor gegeven verzamelingen van uren, docenten, klassen, vakken en klaslokalen dient een rooster gemaakt te worden zo dat:
    - elke klas het juiste aantal lessen in elk vak krijgt van een docent die dat vak geeft
    - elke docent voor het juiste aantal lessen wordt ingeroosterd en alleen op voor hem/haar beschikbare uren
    - voor elke combinatie van uur en klaslokaal hoogstens één combinatie van klas en docent is ingeroosterd voor een vak dat in dat klaslokaal gegeven kan worden
  - *examenrooster*: op grond van inschrijvingen van studenten voor examens dient een examenrooster gemaakt te worden, waarin het aantal maal dat twee examens voor een student op hetzelfde tijdstip, op dezelfde dag of op twee opeenvolgende dagen valt, zo klein mogelijk is

  - *surveillancerooster*: op grond van een examenrooster moet een surveillancerooster voor de docenten gemaakt worden. Belangrijk hierbij is dat de surveillancebeurten eerlijk over de docenten verdeeld worden en dat het aantal keren dat een docent voor slechts één beurt op een dag naar school moet, zo klein mogelijk is

- *job-shop scheduling*: er zijn j jobs en m machines; elke job bestaat uit een aantal taken, die elk op een andere machine en in een bepaalde volgorde moeten worden uitgevoerd en elk een taakafhankelijke tijdsduur duren. Het probleem is nu om de totale doorlooptijd van het begin van de eerste taak tot het beëindigen van de laatste taak te minimaliseren.

- *dienstregeling voor treinen*: treinen dienen zo ingeroosterd te worden dat aan alle constraints betreffende frequentie, overstapmogelijkheden voor passagiers etcetera zo goed mogelijk voldaan wordt

1.4.2 NP-volledigheid van roosterproblemen

In het algemeen zijn roosterproblemen vrij lastige problemen, omdat:

- er dikwijls weinig bekend is over de optimale strategie bij het maken van een rooster; in paragraaf 3.2 zal voor een speciaal geval van het surveillanceroosterprobleem een strategie gegeven worden voor het minimaliseren van het aantal keren dat een docent voor slechts één surveillancebeurt naar school moet komen

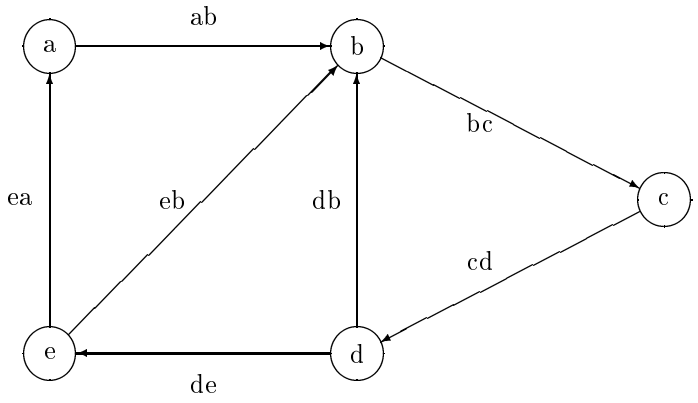
- veel problemen NP-volledig zijn, wat tot gevolg heeft dat er zeer waarschijnlijk geen efficiënt, polynomiaal begrensd, algoritme voor bestaat

- instanties van roosterproblemen dikwijls vrij groot qua omvang zijn

In [EveItaSha76] wordt aangetoond dat zelfs een zeer eenvoudig lesroosterprobleem met slechts drie lesuren, docenten die twee of drie uur les geven en klassen die van elke docent hoogstens 1 uur les krijgen, al NP-volledig is. Hiertoe wordt een polynomiale reductie gefabriceerd van het bekende "3-SAT"-probleem naar dit beperkte lesroosterprobleem. Van het examenroosterprobleem wordt in paragraaf 2.3 aangetoond dat het NP-volledig is door een bijectie aan te geven tussen het "graph coloring problem" en een beperkte versie van het examenroosterprobleem. Ook van het job-shop scheduling probleem is bekend dat het NP-volledig is (een bewijs hiervoor wordt gegeven in [GarJoh79]).

Vanwege de complexiteit van veel roosterproblemen is het vaak practisch onmogelijk om alle mogelijke kandidaatoplossingen op hun optimaliteit te testen. Om deze reden wordt er veelvuldig gebruik gemaakt van heuristieken en van genetische algoritmen. In dit afstudeerverslag worden onder andere ervaringen weergegeven met genetische algoritmen voor het examenroosterprobleem en het surveillanceroosterprobleem.

Juist omdat veel roosterproblemen NP-volledig zijn is het artikel van de hand van Kenneth A. De Jong en Willam M. Spears (zie [JonSpe89]) interessant. In dit artikel wordt een strategie gepresenteerd voor het gebruik van GA's bij het oplossen van NP-volledige problemen. Deze strategie bestaat er



Figuur 1: een graaf met een Hamilton-Circuit

uit een NP-volledig probleem, waarvoor geen goede GA representatie bestaat, te transformeren naar een ander NP-volledig probleem, dat wel een geschikte representatie bezit. Vervolgens wordt het probleem met behulp van een GA opgelost en wordt de oplossing teruggetranformeerd naar termen van het oorspronkelijke probleem. Daar de transformaties tussen beide problemen in polynomiale tijd kunnen geschieden, is dit een redelijk efficiënte strategie. De Jong en Spears nemen het "Boolean Satisfiability Problem" (SAT) als kanoniek probleem waarnaar elk ander NP-volledig probleem getransformeerd dient te worden. SAT heeft een zeer voor de hand liggende string-representatie, namelijk binaire strings van lengte N waarin het i-de bit de waarheidswaarde van de i-de boolese variabele van de N boolese variabelen in de expressie representeert. Als evaluatiefunctie voor SAT wordt:

$$val\,(\text{NOT } e) = 1 - val\,(e)$$

$$val\,(\text{OR } e_1 \cdots e_n) = \text{MAX}\,(val\,(e_1), \cdots, val\,(e_n))$$

$$val\,(\text{AND } e_1 \cdots e_n) = [\text{AVE}\,(val\,(e_1), \cdots, val\,(e_n))]^p$$

genomen, waarbij AVE voor het gemiddelde staat en $p \geq 1$. Het schijnt dat $p = 2$ een goede keuze is, als gekeken wordt naar de performance van het GA.

In het artikel willen De Jong en Spears de lezers ook laten zien hoe het Hamilton-Circuit (HC) probleem afgebeeld kan worden op SAT. Voor elke oplossing van het HC-probleem moet uiteraard gelden dat iedere knoop precies één inkomende en precies één uitgaande tak heeft. Als equivalente boolese expressie wordt daarom voor elke knoop de conjunctie genomen van de termen die geldige takcombinaties voor de ingaande en voor de uitgaande takken representeren. Zo geldt voor de graaf in figuur 1 dat deze wordt afgebeeld op:

```
( and   ab
        bc
        cd
        de
        ea
        ( or    ( and db ( not de ))
                ( and ( not db ) de ))
        ( or    ( and ea ( not eb ))
                ( and ( not ea ) eb ))
        ( or    ( and ab ( not db ) ( not eb ))
                ( and ( not ab ) db ( not eb ))
                ( and ( not ab ) ( not db ) eb ))
)
```

Helaas voor De Jong en Spears is hun redenering niet geheel sluitend, want ook voor een verzameling cykels, die alle knopen van een graaf bedekken, geldt dat iedere knoop precies één inkomende en precies één uitgaande tak heeft. En deze cykels vormen geen oplossing voor het HC-probleem.

Concluderend kunnen we zeggen dat, hoewel het idee van de Jong en Spears zeer aardig is, het vinden van een eenvoudige en correcte transformatie naar het SAT-probleem nog nader onderzoek verdient.

We zullen nu eerst het examenroosterprobleem bekijken en vervolgens het surveillanceroosterprobleem.

2 Het "Modular Exam Scheduling Problem" (MESP)

2.1 Beschrijving van het probleem

Op grond van inschrijvingen van studenten voor examens dient een examenrooster gefabriceerd te worden, dat voor de studenten zo weinig mogelijk vervelende consequenties heeft. Deze vervelende consequenties kunnen bestaan uit:

- twee of meer examens op hetzelfde tijdstip
- twee of meer examens op dezelfde dag
- twee of meer examens op twee opeenvolgende dagen

Laat voor een meer formele beschrijving:

$\{1, \ldots, t\}$	eindige verzameling van t timeslots
$E = \{1, \ldots, e\}$	eindige verzameling van e examens
$\{S_1, \ldots, S_s\}$	eindige verzameling van s deelverzamelingen van E

Hierbij stelt elke S_k de collectie van examens voor waaraan student k wil deelnemen ($k = 1, \ldots, s$). Zonder beperking van de algemeenheid kan verondersteld worden dat $E = \bigcup_k S_k$. Verder nemen we aan dat de timeslots onderling disjunct zijn. Hierdoor kunnen twee examens elkaar alleen maar overlappen als ze op exact hetzelfde timeslot worden gegeven. Van de timeslots is verder ook nog bekend op welk deel van welke dag ze vallen. Het MESP-probleem bestaat nu uit twee elementen:

1. beantwoording van de vraag of er een rooster mogelijk is zonder enige vervelende consequentie voor de studenten
2. constructie van een rooster met het minst aantal vervelende consequenties voor de studenten

We zullen het *Restricted MESP (RMESP)* definiëren als het MESP-probleem waarbij het voor elke student verboden is twee of meer examens gelijktijdig te hebben. Andere eisen (zoals geen twee examens op één dag) zullen we hierbij beschouwen als wensen waaraan we zoveel als mogelijk tegemoet willen komen. Voor RMESP wordt nu een functie (in feite het rooster)

$$f : \{1, \ldots, t\} \times \{1, \ldots, e\} \to \{0, 1\}$$

gezocht zo dat:

1. $f(i, j) = 1$ dan en slechts dan als examen j op timeslot i wordt gehouden
2. $\sum_{i=1}^{t} f(i, j) = 1 \ \forall j$ (elk examen wordt op precies 1 timeslot gegeven)
3. geen enkele student heeft twee of meer examens op hetzelfde timeslot

f geeft aan op welk timeslot een bepaald examen gehouden wordt. Door over te stappen op matrix-notatie kan de derde eis ook meer wiskundig gespecificeerd worden. Laat de $e \times s$-matrix (S_{jk}) gedefineerd zijn door:

$$S_{jk} = \begin{cases} 1 \text{ als examen } j \in S_k \\ 0 \text{ als examen } j \notin S_k \end{cases}$$

Gezocht is nu een $t \times e$-matrix (T_{ij}) zo dat

1. $T_{ij} = \begin{cases} 1 \text{ als examen } j \text{ op timeslot } i \\ 0 \text{ als examen } j \text{ niet op timeslot } i \end{cases}$
2. $\sum_{i=1}^{t} T_{ij} = 1 \ \forall j$
3. $R_{ik} = \sum_{j=1}^{e} T_{ij} \cdot S_{jk} \leq 1 \quad \forall i, k$

R_{ik} geeft het aantal examens dat op timeslot i door student k moet worden afgelegd.

Voor een gegeven instantie van MESP of RMESP zal het dikwijls niet mogelijk zijn een rooster te construeren, waarbij aan alle eisen en wensen wordt voldaan. We kunnen dan alleen proberen een rooster te maken, dat zoveel als mogelijk aan deze eisen en wensen tegemoet komt. We hebben dan te maken met een optimaliseringsprobleem. In de volgende paragraaf wordt beschreven hoe deze optimalisatie kan worden nagestreefd met een GA.

2.2 Optimalisatie van MESP met een GA

Om een GA voor een bepaald probleem toe te passen dient vooraf goed nagedacht te worden over de te gebruiken representatie, evaluatiefunctie en operatoren. In de volgende deelparagrafen wordt uitgelegd welke keuzes er wat dit betreft gemaakt zijn. Hierbij is gebruik gemaakt van enkele ideeën uit [CorFanMel93].

2.2.1 Representatie

De meest voor de hand liggende representatie voor een chromosoom is een string $[y_1, \ldots, y_t]$ waarbij $y_i = \{E_{i_1}, \ldots, E_{i_r}\}$ de verzameling van i_r examens voorstelt die op timeslot T_i worden afgenomen. Dit geeft echter problemen, omdat bij toepassing van de crossoveroperatie er gemakkelijk chromosomen kunnen ontstaan die geen geldige oplossing voor het MESP-probleem zijn. Crossover kan immers kindchromosomen produceren waarbij sommige examens meer dan één keer worden afgenomen, terwijl andere examens helemaal niet zijn ingeroosterd. Uiteraard zou op dergelijke kindchromosomen een reparatie-algoritme kunnen worden toegepast, maar behalve dat dit de performance niet ten goede komt, is het ook nog maar de vraag in hoeverre de crossover-operatie na de reparatie nog te herkennen is in de kindchromosomen.
Beter is het daarom als representatie te kiezen: $[x_1, x_2, \ldots, x_e]$, met $1 \leq x_i \leq t$. Bijvoorbeeld $[2, 4, 7, 3, 7]$ wordt geïnterpreteerd als: examen 1 vindt plaats op timeslot 2, examen 2 vindt plaats op timeslot 4, etc. Deze representatie heeft als voordeel dat elk examen altijd precies één keer is (en blijft) ingeroosterd, zodat altijd sprake is van een geldige oplossing.

2.2.2 Evaluatiefunctie

De fitness van een chromosoom c hangt af van de studentgegevens d (voor hoeveel en welke examens hebben de studenten zich ingeschreven?). Er worden boetes uitgedeeld voor elke keer dat een bepaalde ongewenste situatie zich voordoet. Als fitnessfunctie wordt vervolgens gebruikt:

$$f(c, d) = \frac{1}{1 + \sum_{i=1}^{n} w_i m_i(c, d)}$$

n	: aantal verschillende ongewenste situaties
$m_i\,(c,d)$	: aantal malen dat de ongewenste situatie i zich voordoet
w_i	: gewicht (grootte van de boete) voor het optreden van situatie i

De fitness zit nu tussen 0 en 1, waarbij 1 een "perfect" rooster betekent.

Ongewenste situaties (in dalende volgorde van ongewenstheid) zijn bijvoorbeeld:

- een student heeft meer dan één examen tegelijk
- een student heeft meer dan twee examens op een dag
- een student heeft twee examens in twee opeenvolgende timeslots op één dag
- een student heeft twee examens in twee opeenvolgende timeslots op twee aaneensluitende dagen
- een student heeft twee examens op twee opeenvolgende dagen

In [CorFanMel93] wordt ook van een fitnessfunctie als boven gesproken, maar dan met "inverse square of the punishment". Hoewel in deze publicatie deze laatste functie niet precies wordt gedefinieerd, wordt hiermee waarschijnlijk de functie bedoeld die uit f ontstaat door de noemer te kwadrateren.

2.2.3 Gebruikte operatoren

Bij het experimenteren is gebruik gemaakt van de volgende operatoren:

- selectie
  - volgens de roulettewheel methode, waarbij een tussenpopulatie van grootte pop_size wordt gegenereerd
- crossover
  - uit de tussenpopulatie werden steeds random en zonder teruglegging twee ouders uitgekozen die door middel van crossover twee kinderen in de nieuwe populatie gaven. Er werden verschillende vormen van crossover gebruikt, te weten:
    - één-punts crossover
    - twee-punts crossover
    - uniforme crossover
    - combinaties van één-punts, twee-punts en uniforme crossover; dat wil zeggen afwisselend wordt volgens een bepaalde kansverdeling een bepaald type crossover toegepast
- mutatie
  - *"gewone" mutatie*: het timeslot van een examen verandert door random een timeslot te kiezen (met een kleine kans van $1/t$ kan toepassing van deze mutatie weer hetzelfde chromosoom opleveren)

- *wisselmutatie*: twee examens wisselen onderling van timeslot
- *dagwisselmutatie*: twee hele dagen worden omgewisseld
- *inversiemutatie*: toepassing van de inversie-operator op het chromosoom
- combinaties van bovenstaande mutaties; dat wil zeggen afwisselend wordt volgens een bepaalde kansverdeling een bepaald type mutatie toegepast

Er werd geen gebruik gemaakt van enige vorm van elitisme. De crossover-operatoren zijn, als altijd, bedoeld voor het uitwisselen van stukjes informatie over goede roosters. De gewone mutatie en de inversiemutatie zorgen voor enige, resp. veel variatie in mogelijk goede oplossingen. Wisselmutatie en dagwisselmutatie zijn vormen van mutatie die afgeleid zijn van de manier hoe een mens het probleem te lijf zou gaan: wat gebeurt er als we twee examens, resp. twee dagen van examens omruilen. Het is denkbaar dat door dit omruilen een deel van de resterende problemen wordt opgelost.

2.2.4 Resultaten voor aselecte data

Om enige ervaring met het werken met een GA op te doen, werd het GA allereerst toegepast op aselect gegenereerde studentinschrijvingen voor examens met de volgende kenmerken:

aantal studenten	20
aantal examens	25
aantal timeslots per dag	3 (laatste dag 2)
totaal aantal timeslots	20
maximaal aantal examens per student	6

De gegenereerde data waren:

Studentnummer	inschrijvingen
1	7 20
2	3 2 23 8 15 11
3	10
4	12 25 18 6
5	7 19 16
6	15 2 19 21 13
7	18 21 4
8	9 7 23 1 4
9	3 10 21 2 7 6
10	13 7 2 15 23 5
11	2 6 18 13 12
12	14
13	12 3 13 11
14	23 24 21 10 8 6
15	22 14 25 19 24 8
16	23 12 16 8 6
17	9 25 11
18	10
19	2 22
20	18 1 22 11 8 3

Elk nummer in de inschrijvingen verwijst hierbij naar een bepaald examen.

Het GA gebruikte standaard één-punts-crossover en gewone mutatie. Bij mutatie werd een examennummer vervangen door een willekeurig gekozen examennummer, dat eventueel ook gelijk kon zijn aan het oorspronkelijke.

Voor verschillende kansen op crossover en mutatie werden tien keer, uitgaande van een aselect gekozen beginpopulatie van 50 chromosomen, 200 nieuwe generaties gegenereerd. Geregistreerd werden:

- het beste chromosoom in elk van de tien laatste (201^e) generaties en zijn fitness
- de gemiddelde fitness van de 10 beste chromosomen

Voor het bepalen van de fitness werden per student de volgende boetes gebruikt:

- twee examens op zelfde timeslot: boete = 30
- twee examens op zelfde dag: boete = 3
- drie examens op zelfde dag: boete = 9

De fitness werd hieruit met behulp van de eerder vermelde standaard fitnessfunctie bepaald.

In de nu volgende tabel wordt een overzicht gegeven van de behaalde resultaten. Hierbij staat PCross voor de kans op crossover en PMut voor de kans op mutatie.

PCross	PMut	beste fitness	gem. beste fitness
0.3	0.005	0.25000	0.14236
0.4	0.005	0.14286	0.09559
0.5	0.005	0.14286	0.09885
0.6	0.005	0.25000	0.10388
0.7	0.005	0.14286	0.09514
0.8	0.005	0.25000	0.11146
0.3	0.01	0.14286	0.11253
0.4	0.01	0.25000	0.16911
0.5	0.01	0.25000	0.16643
0.6	0.01	0.25000	0.15984
0.7	0.01	0.25000	0.12324
0.8	0.01	0.25000	0.17339
0.3	0.05	0.25000	0.11161
0.4	0.05	0.25000	0.15426
0.5	0.05	0.25000	0.19857
0.6	0.05	0.25000	0.19081
0.7	0.05	0.25000	0.19626
0.8	0.05	0.25000	0.20286

Uit dit overzicht kan worden geconcludeerd dat een hele kleine mutatiekans van 0.005 slechtere resultaten geeft dan een mutatiekans van 0.05. Het veranderen van de kans op crossover gaf veel minder duidelijke verschillen, al hebben crossoverkansen van minstens 0.5 de neiging om iets betere resultaten te geven. In bijna alle gevallen wordt een chromosoom gevonden met een fitness van 0.25, hetgeen overeenkomt met één student die één keer twee examens op een dag heeft.

2.2.5 Resultaten met echte data (1)

Eén belangrijk nadeel van aselecte data is dat het aantal studenten per examen redelijk uniform verdeeld zal zijn. In de praktijk zal dit vrijwel nooit het geval zijn. Verder waren in de gebruikte data uit de vorige paragraaf zowel het aantal studenten als het aantal examens redelijk beperkt. Om deze redenen zeggen de in de vorige paragraaf gevonden resultaten niet zoveel over de praktijk.

Via de avondschool van de "Hogeschool voor Economische Studies" (HES) te Rotterdam kon gebruik gemaakt worden van echte inschrijvingen van studenten voor tentamens. Het betrof hier hertentamens die in de periode van 15 augustus tot en met 26 augustus 1994 werden gegeven. Om in het tijdsbestek van 10 dagen (in het weekend werd niet getentamineerd) 95 hertentamens te kunnen afnemen werden een aantal tentamens overdag afgenomen. In het algemeen waren er 3 timeslots per dag:

1. 's morgens van 9.00 uur tot 11.00 uur
2. 's avonds van 17.30 uur tot 19.30 uur
3. 's avonds van 20.00 uur tot 22.00 uur.

Voor de diverse hertentamens hadden zich in totaal 153 studenten ingeschreven. Voor een volledig overzicht van de inschrijvingen wordt verwezen naar Appendix A. Passen we de fitnessfunctie toe op de indeling die toendertijd met de hand werd gemaakt dan vinden we een fitness van 0.025, hetgeen overeenkomt met een totale boete van 39 punten.

Het vinden van geschikte waarden voor de populatiegrootte, het aantal te genereren populaties, de kans op crossover en de kans op mutatie was aanvankelijk een groot probleem. Het lag voor de hand het eerst te proberen met een kans op één-punts crossover van 80% en een kans van 5% op gewone mutatie, daar dit bij de aselecte data redelijk goed beviel. Zoals uit de gegevens van de volgende tabel blijkt, was het resultaat niet erg bemoedigend: zelfs de fitness van 0.025 voor de met de hand gemaakte tentamenindeling werd bij lange na niet gehaald.

Pop_grootte	**runs**	**pops**	**PCross**	**PMut**	**beste fitness**
50	10	200	0.80	0.05	0.00704
50	1	500	0.80	0.05	0.00690
100	2	500	0.80	0.05	0.01847
100	1	1000	0.80	0.05	0.01176
50	1	1000	0.80	0.05	0.01099
50	1	1500	0.80	0.05	0.01429
50	1	2000	0.80	0.05	0.00719
50	1	4000	0.80	0.05	0.01724

Verassend genoeg bleken de in [CorFanMel93] opgegeven waarden veel beter te voldoen:

Pop_grootte	**runs**	**pops**	**PCross**	**PMut**	**beste fitness**
50	10	300	0.70	0.003	1

Wat misschien nog veel meer zegt is dat bij 9 van de 10 runs van 300 nieuwe generaties het beste chromosoom een fitness had van 0.25 (dus veel beter dan het met de hand bereikte resultaat) en in één geval het beste chromosoom zelfs een fitness van 1 had. Dit laatste betekent dat er een rooster werd gevonden zonder enige ongewenste situatie voor een student. Redenen waarom het bij deze waarden ineens veel beter gaat kunnen zijn:

- de data is niet langer aselect.
- de chromosoomlengte (= aantal tentamens) is nu 95 tegen 25 bij de aselecte data. Hierdoor worden bij dezelfde kans op mutatie gemiddeld bijna vier keer zoveel genen veranderd. Bij een mutatiekans van 5% worden bij een chromosoom van lengte 95 gemiddeld teveel genen in één keer veranderd. Dit verklaart dat een veel lagere mutatiekans nu veel beter werkt.

2.2.6 Resultaten met echte data (2)

Na het vinden van een optimale oplossing met behulp van één-punts crossover en gewone mutatie werd het interessant na te gaan of gebruik van andere (combinaties van) crossover-operatoren en mutatie-operatoren tot een snellere convergentie leidt. Bij het experimenteren met één-punts crossover, twee-punts crossover en uniforme crossover voor het MESP probleem werd uitgegaan van:

1. de data van de avondschool van de HES
2. een crossover-kans van 0.7
3. een kans op gewone mutatie van 0.003
4. bij uniforme crossover een kans op uitwisseling van 0.5 per allele

De kansen 0.7 en 0.003 zijn zo gekozen, omdat daar in de vorige paragraaf al goede ervaringen mee zijn opgedaan.

Er is tevens geëxperimenteerd met combinaties van verschillende crossovers. Hierbij werd, steeds wanneer met een kans van 0.7 crossover moest worden uitgevoerd, aselect één uit twee of drie mogelijke crossovers gekozen. Op deze manier onstonden 7 mogelijke combinaties:

1. alleen één-punts crossover
2. alleen twee-punts croosover
3. alleen uniforme crossover
4. één-punts en twee-punts crossover
5. één-punts en uniforme crossover
6. twee-punts en uniforme crossover
7. één-punts, twee-punts en uniforme crossover

Voor elke combinatie werden 10 keer, uitgaande van een aselect gekozen populatie van 50 chromosomen, 300 nieuwe populaties gegenereerd, zonder gebruik te maken van elitisme. De tabellen in Appendix B geven om de 50 generaties de volgende statistische gegevens:

- maximale fitness van het beste chromosoom in de 10 runs samen
- de boete voor dat beste chromosoom
- de gemiddelde maximale fitness van de 10 beste chromosomen (uit elke run het beste chromosoom)
- de gemiddelde boete van de 10 beste chromosomen

Onderlinge vergelijking van de tabellen leert dat:

1. bij elke (combinatie van) crossover(s) uiteindelijk in één of meer runs wel het optimum met boete 0 gevonden wordt

2. (alleen) twee-punts crossover sneller tot het vinden van het optimum leidt dan (alleen) één-punts crossover of uniforme crossover

3. een combinatie van twee soorten crossovers niet tot een sneller vinden van het optimum leidt (in vergelijking met twee-punts crossover)

4. een combinatie van twee soorten crossovers wel tot een betere gemiddelde fitness van het beste chromosoom leidt (met name de combinatie van twee-punts crossover en uniforme crossover springt er positief uit)

5. een combinatie van alle 3 de crossovers niet tot enige verbetering leidt ten opzichte van een combinatie van twee crossovers

Vervolgens werden ook de verschillende mutatie-operatoren aan een nader onderzoek onderworpen. De resultaten van de experimenten zijn in Appendix C te vinden. Hieruit blijkt dat de resultaten er in vergelijking met de resultaten uit Appendix B niet beter van werden. We kunnen derhalve concluderen dat voor het MESP-probleem gewone mutatie de beste mutatie-operator is.

Tenslotte werd ook geëxperimenteerd met iets andere en zwaardere boetes, te weten:

examens	**boete**
gelijktijdig	100
op zelfde dag	10
's avonds en volgende ochtend	10
2 opeenvolgende dagen	1

Door verhoging van de boetes en uitbreiding van het boetestelsel was het onmogelijk een optimaal rooster met boete = 0 te vinden. Eén student had zich voor 10 examens ingeschreven. Daar de examenperiode uit 12 dagen (inclusief een zaterdag en een zondag) bestond heeft deze student sowieso al een boete van 9. Bovendien beperkt een optimaal rooster voor deze student enorm de mogelijkheden van optimalisatie voor de andere studenten.

Uit Appendix D blijkt dat uniforme crossover in dit geval beter presteert dan één-punts of twee-punts crossover. De combinatie met gewone mutatie en dagwisselmutatie gaf het beste resultaat.

Conclusie: Voorzichtig kunnen we uit alle bovenstaande experimenten concluderen dat één-punts crossover de slechtste resultaten geeft. De keuze voor twee-punts-crossover of uniforme crossover lijkt mede af te hangen van de grootte van de boetes. Van alle gebruikte mutatie-operatoren geeft gewone mutatie eventueel gecombineerd met dagwisselmutatie het beste resultaat.

2.3 NP-volledigheid van MESP

Is MESP (of RMESP) ook NP-volledig? RMESP lijkt nog het meest op het zogeheten *integer-programming problem*: gegeven een $m \times n$-matrix A en een integer m-vector b, bestaat er een integer n-vector x met elementen uit $\{0, 1\}$

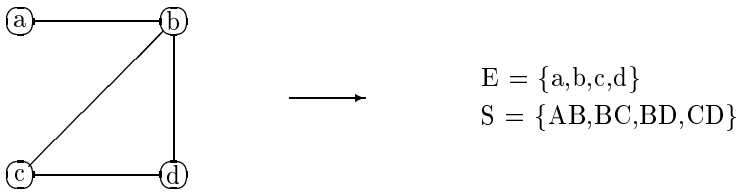
zo dat $Ax \leq b$. Van het integer-programming problem is bekend dat het NP-volledig is (zie bijvoorbeeld [CorLeiRiv92, p. 960]).
Om te bepalen of het MESP-probleem al dan niet NP-volledig is, moet het eerst als beslissingsprobleem geformuleerd worden. Dit kan als volgt: gegeven een instantie van het probleem, een boete-regeling voor allerlei vervelende situaties en een positief geheel getal l, bestaat er een rooster dat gebruikt maakt van hoogstens l timeslots, zo dat voor geen enkele student een vervelende situatie optreedt? Om op deze vraag een antwoord te vinden bekijken we het beperktere RMESP*-probleem, dat als volgt uit het MESP-probleem getransformeerd kan worden:

- alleen gelijktijdigheid van twee examens op één timeslot wordt als een vervelende situatie voor een student beschouwd
- elke student die aan één examen deelneemt, laten we in RMESP* weg, want deze maakt het probleem toch niet eenvoudiger of moeilijker
- voor elke student die binnen de instantie van MESP aan $n \geq 2$ examens deelneemt, creëren we $\frac{1}{2}n\,(n-1)$ studenten, waarbij elke student aan precies twee van de oorspronkelijke n examens deelneemt en geen enkele student aan hetzelfde tweetal examens deelneemt, en waarbij alle mogelijke combinaties van twee verschillende uit de oorspronkelijke n examens aan bod komen
- dubbele voorkomens (dat wil zeggen meer dan één student die aan hetzelfde tweetal examens deelneemt) worden verwijderd

Als RMESP* NP-volledig is, dan zeker ook MESP.
Het RMESP*-probleem behoort zeker tot de klasse NP. Voor een kandidaatoplossing (T_{ij}) kan door een matrixvermenigvuldiging en het checken van de elementen van het resultaat gecontroleerd worden of de kandidaatoplossing aan de eisen voldoet. Dit kan in polynomiale tijd.
Om te bewijzen dat RMESP* NP-volledig is zullen we een een polynomiale reductie construeren van het bekende *graph coloring problem*. Hierbij is de vraag of de knopen van een gegeven graaf G zo gekleurd kunnen worden dat geen twee aangrenzende knopen dezelfde kleur hebben.
Tussen het graph coloring problem en RMESP* is op eenvoudige wijze een bijectie te construeren. Laat hiertoe A de verzameling van alle mogelijke instanties van het graph coloring problem zijn en B de verzameling van alle mogelijke instanties van RMESP*. $A = \{(G,k)\}$ met $G = (V,T)$ een graaf, $V = \{V_1, \ldots, V_e\}$ de verzameling van knopen, T de verzameling van takken en k een niet negatief geheel getal (k is het aantal kleuren waarmee men de graaf wil kleuren). $B = \{(E,S,l)\}$ met E de verzameling van examens, S de verzameling van studenten en l een niet negatief geheel getal (l is het aantal timeslots dat gebruikt mag worden bij inroostering). De functie $f : A \to B$ wordt gedefinieerd door

$$f\,((G,k)) = (E,S,l)$$

waarbij E ontstaat door voor elke knoop in G een examen te creëren, S ontstaat door voor elke tak uit G een student te creëren die aan die examens deelneemt



Figuur 2: transformatie tussen het graph coloring problem en RMESP*

die overeenkomen met de twee knopen van de tak en $l = k$. Knopen die in een kleuring van de graaf dezelfde kleur krijgen, corresponderen met examens die op hetzelfde timeslot geëxamineerd worden.
In figuur 2 wordt deze afbeelding f voor een eenvoudig voorbeeld in beeld gebracht. Merk op dat de graaf in figuur 2 3-kleurbaar is: kleur bijvoorbeeld de knopen a en c blauw, knoop b wit en knoop d rood. Het volgende examenrooster correspondeert met deze graaf:

timeslot	**examen**	**studenten**
T_1	a, c	AB, BC, CD
T_2	b	AB, BC, BD
T_3	d	BD, CD

Eenvoudig is in te zien dat f goed-gedefinieerd, surjectief en injectief is. Verder is duidelijk dat een instantie van A k-kleurbaar is dan en slechts dan als voor de bijbehorende instantie van B geldt dat hoogstens l timeslots nodig zijn voor een probleemloos rooster. Omdat het graph coloring problem NP-volledig is (zie bijvoorbeeld [CorLeiRiv92, p. 962], is hiermee aangetoond dat RMESP eveneens NP-volledig is.

□

3 Het "Teacher Invigilation Scheduling Problem" (TISP)

3.1 Beschrijving van het probleem

Naast het MESP, het maken van een examenrooster voor de studenten, hebben we een gelijksoortig probleem voor de surveillerende docenten. Bij een gegeven examenrooster dient een surveillancerooster gemaakt te worden voor de surveillerende docenten. Dit rooster dient aan een aantal eisen te voldoen en zo veel als mogelijk aan een aantal wensen tegemoet te komen. Deze eisen en wensen worden hieronder opgesomd.

Eisen:

- geen docent mag twee keer op hetzelfde tijdstip surveilleren
- geen docent mag ingezet worden op momenten dat hij/zij absoluut niet kan
- het aantal surveillancebeurten dient evenredig te zijn met de taakomvang van de docent, dat wil zeggen de beurten moeten "eerlijk" verdeeld worden

Wensen:

- laat een docent niet voor slechts één surveillancebeurt op een dag (een zogenaamde *1-beurt*) naar school komen
- het aantal *tussenbeurten* dient minimaal te zijn. Een tussenbeurt is hierbij een vrije periode tussen twee surveillancebeurten van een docent op één dag, gedurende welke er wel examens worden afgenomen
- bij elk examen voor module (vak) X is minstens één docent aanwezig die module (vak) X ook geeft

Een tweetal aannames liggen bij TISP voor de hand:

1. de timeslots, waarop geëxamineerd wordt, zijn onderling *disjunct*. Hiermee wordt niet alleen bedoeld dat de timeslots elkaar niet overlappen, maar ook dat er voldoende ruimte tussen elk tweetal timeslots zit opdat een docent bij beide kan surveilleren
2. voor elk timeslot zijn er voldoende docenten als surveillant beschikbaar

Een preciezere formulering is:

$\{1, 2, \ldots, d\}$: verzameling van docenten

$\{1, 2, \ldots, t\}$: (stijgend gesorteerde) verzameling van timeslots

S_i $(i = 1, \ldots, t)$: aantal benodigde docenten voor timeslot i

U_j $(j = 1, \ldots, d)$: aantal beurten voor docent j (afgerond op een geheel getal)

$X_j = \{k_{j,1}, \ldots, k_{j,l_j}\}$ $(j = 1, \ldots, d)$: de timeslots waarop docent j niet beschikbaar is

$MT_i = \{M_{l_1}, \ldots, M_{l_i}\}$: de modules die op timeslot i geëxamineerd worden

$MD_j = \{M_{j,1}, \ldots, M_{j,m}\}$: de modules die door docent j gegeven worden

Gezocht wordt een functie, het surveillancerooster,

$$f : \{1, \ldots, t\} \times \{1, \ldots, d\} \to \{0, 1\}$$

zo dat

1. $f(i,j) = 1$ dan en slechts dan als docent j op timeslot i moet surveilleren
2. $i \in X_j \Rightarrow f(i,j) = 0$
3. $\sum_{j=1}^{d} f(i,j) = S_i \ \forall i$ (juiste aantal docenten per timeslot)
4. $\sum_{i=1}^{t} f(i,j) = U_j \ \forall j$ ("juiste" aantal beurten per docent)
5. $(\forall M_l \in MT_i)\,(\exists j\,[M_l \in MD_j \wedge f(i,j) = 1])\ \ (i = 1, \ldots, t)$ (bij elk examen is een "vakdocent" aanwezig)
6. $[\sum f(i,j) \mid \nexists k\,[k \neq i \wedge f(k,j) = 1 \wedge i \text{ en } k \text{ op zelfde dag}]]$ is minimaal onder de eis dat aan de overige voorwaarden voldaan wordt (zo weinig mogelijk dagen met slechts één surveillancebeurt)
7. $(\sum f(i,j) \mid [(f(i,j) = 1) \wedge (f(k,j) = 1) \wedge (i \text{ en } k \text{ op zelfde dag}) \wedge (i < k) \wedge [\exists l | i < l < k \wedge f(l,j) = 0]])$ is minimaal onder de eis dat aan de overige voorwaarden voldaan wordt (zo weinig mogelijk "tussenbeurten" voor de surveillanten, daar deze soms beschouwd worden als overeenkomend met twee 1-beurten)

Van deze zeven punten zijn de eerste drie zonder meer verplicht. De wens dat bij elk examen een vakdocent aanwezig is, is weliswaar zeer wenselijk, maar zal in de praktijk niet altijd haalbaar zijn. Het eerlijk verdelen van de beurten onder de docenten heeft een hogere prioriteit dan het minimaliseren van het aantal 1-beurten. Overigens hoeft het niet altijd mogelijk te zijn om de beurten eerlijk te verdelen. In paragraaf 3.4 wordt nader ingegegaan over het al dan niet eerlijk kunnen verdelen van de beurten.

In de volgende paragrafen zal allereerst een wiskundige afleiding gegeven worden van het minimale aantal 1-beurten en tussenbeurten voor het geval dat een dag precies 3 disjuncte timeslots bevat. Dit zal tevens richtlijnen voor de te volgen strategie opleveren. Vervolgens zullen voor een redelijk grote instantie van het probleem resultaten gepresenteerd worden van een algoritme dat deze strategie navolgt. Tenslotte zullen de resultaten van een genetisch algoritme op dezelfde instantie van het probleem besproken worden.

3.2 Optimalisatie van 1-beurten

We zullen nu onderzoeken wat bij een drietal timeslots op één dag het minimale aantal 1-beurten is dat aan de docenten moet worden gegeven. De volgende aannames worden gemaakt:

1. een dag bestaat uit 3 disjuncte timeslots waarop respectievelijk x, y en z docenten nodig zijn
2. er zijn altijd voldoende docenten beschikbaar
3. de beschikbare docenten kunnen eventueel de gehele dag worden ingezet

Zowel de opvatting dat een tussenbeurt in het geheel niet erg is, als de opvatting dat een tussenbeurt even erg is als twee 1-beurten is te verdedigen. Daarom zijn voor beide (uiterste) gevallen in de volgende paragrafen formules voor het minimale aantal 1-beurten afgeleid. Iets algemener zou zijn om te stellen dat een tussenbeurt α keer zo erg is dan een 1-beurt waarbij $0 \leq \alpha \leq 2$.

3.2.1 Het geval: één tussenbeurt komt overeen met twee 1-beurten

Voor alle $x, y, z \in \mathbf{N}$ wordt de functie $f(x, y, z)$ gedefinieerd als het aantal 1-beurten dat resteert bij optimale indeling van 2/3-beurten, waarbij één tussenbeurt overeenkomt met twee 1-beurten. Hierbij zijn x, y en z het aantal docenten dat voor het eerste, het tweede respectievelijk het derde timeslot op een dag nodig is. Deze functie heeft de volgende eigenschappen:

1. $f(x, y, z) \geq 0$
2. $f(x, y, 0) = |x - y|$
3. $f(x, 0, z) = x + z$
4. $f(x, y, z) = f(z, y, x)$
5. $\forall x, y, z > 0$:
   $f(x, y, z) = \min(f(x-1, y-1, z-1), f(x-1, y-1, z), f(x, y-1, z-1))$
6. $y = \min(x, y, z) \rightarrow f(x, y, z) = x - 2y + z$
7. $z = \min(x, y, z) \rightarrow f(x, y, z) = \begin{cases} x - y & \text{als} \quad x \geq y \\ y - x - z & \text{als} \quad x \leq y - z \\ 0 & \text{als} \quad y - z < x < y \end{cases}$
8. $x = \min(x, y, z) \rightarrow f(x, y, z) = \begin{cases} z - y & \text{als} \quad z \geq y \\ y - x - z & \text{als} \quad z \leq y - x \\ 0 & \text{als} \quad y - x < z < y \end{cases}$
9. $f(x, y, z) = \max(0, x - 2y + z, y - x - z, \max(x, z) - y)$

De eigenschappen 1 tot en met 4 zijn eenvoudig te verifiëren. Eigenschap 8 volgt direct uit de eigenschappen 4 en 7. Eigenschap 9 volgt uit de eigenschappen 6 tot en met 8 door de afzonderlijke gevallen samen te nemen. De bewijzen van

eigenschappen 5, 6 en 7 volgen hieronder. Voor het bewijs van de eigenschappen 6 en 7 maken we gebruik van volledige inductie.

Bewijs van eigenschap 5

Het geven van een tussenbeurt komt overeen met het toekennen van 2 keer een 1-beurt, zodat het toekennen van een tussenbeurt nooit tot een beter resultaat kan leiden dan het toekennen van een 1-beurt op het eerste of laatste timeslot. Het toekennen van een 1-beurt kan niet tot een beter resultaat leiden dan het toekennen van een 2-beurt of een 3-beurt (twee respectievelijk drie aaneensluitende surveillancebeurten). Immers: daar $x, y, z > 0$ is het uitdelen van alleen maar 1-beurten zeker niet optimaal. Het optimale pad bevat dus minstens één 2-beurt of 3-beurt. Omdat op het optimale pad 1-, 2- en 3-beurten onderling verwisselbaar zijn, wordt er niets verloren als met een 2-beurt of 3-beurt gestart wordt. Eigenschap 5 zegt precies dat het optimale pad hetzij een 3-beurt, hetzij een vroege 2-beurt, hetzij een late 2-beurt bevat.

□

Bewijs van eigenschap 6

Met inductie naar y.
$\underline{y = 0}$:
$f(x, y, z) = f(x, 0, z) = x + z = x + z - 2y$

$\underline{y > 0}$:
Neem aan dat eigenschap 6 geldt tot en met $y - 1$. Te bewijzen: eigenschap 6 geldt ook voor y.
Omdat $y = \min(x, y, z)$ geldt ook dat
$y - 1 = \min(x - 1, y - 1, z - 1) = \min(x - 1, y - 1, z) = \min(x, y - 1, z - 1)$
en dus (er kunnen 2- of 3-beurten uitgedeeld worden):

$$\begin{aligned} f(x, y, z) &= \min(f(x - 1, y - 1, z - 1), f(x - 1, y - 1, z), f(x, y - 1, z - 1)) \\ &= \min(x - 1 + z - 1 - 2(y - 1), x - 1 + z - 2(y - 1), x + z - 1 - 2(y - 1)) \\ &= \min(x + z - 2y, x + z - 2y + 1, x + z - 2y + 1) \\ &= x - 2y + z \end{aligned}$$

Dus eigenschap 6 geldt ook voor y.

□

Bewijs van eigenschap 7

Met inductie naar z.
$\underline{z = 0}$:

$$f(x, y, z) = f(x, y, 0) = |x - y| = \begin{cases} x - y & \text{als} \quad x \geq y \\ y - x = y - x - z & \text{als} \quad x \leq y - z \end{cases}$$

Het derde alternatief komt nu niet voor, daar dan $y < x < y$ zou zijn.

$\underline{z > 0}$:
Neem aan dat eigenschap 7 geldt tot en met $z - 1$. Te bewijzen: eigenschap 7 geldt ook voor z.
We splitsen de 3 gevallen uit:
$\underline{\text{geval a}}$: $x \geq y$ en $z = \min(x, y, z)$

1. *(3-beurt)* Omdat in dit geval $x-1 \geq y-1 \geq z-1$ geldt vanwege de inductiehypothese:
   $f(x-1, y-1, z-1) = x-1-(y-1) = x-y$

2. *(vroege 2-beurt)*

   (a) Als $y > z$, dan is $x-1 \geq y-1 \geq z$ en dus vanwege de inductiehypothese:
   $f(x-1, y-1, z) = x-1-(y-1) = x-y$

   (b) Als $y = z$, dan is $y-1 = \min(x-1, y-1, z)$, zodat volgens eigenschap 6:
   $f(x-1, y-1, z) = x-1+z-2(y-1) = x-y+1$

3. *(late 2-beurt)* Omdat $x \geq y-1 \geq z-1$ geldt vanwege de inductiehypothese:
   $f(x, y-1, z-1) = x-(y-1) = x-y+1$

Hieruit volgt dat:

$$\begin{aligned} f(x,y,z) &= \min(f(x-1,y-1,z-1), f(x-1,y-1,z), f(x,y-1,z-1)) \\ &= \min(x-y, x-y(+1), x-y+1) \\ &= x-y \end{aligned}$$

geval b: $x \leq y-z$ en $z = \min(x,y,z)$

1. *(3-beurt)* Omdat uit de voorwaarde volgt dat $z-1 \leq x-1 \leq y-1-(z-1)$ geldt vanwege de inductiehypothese:
   $f(x-1, y-1, z-1) = (y-1)-(x-1)-(z-1) = y-x-z+1$

2. *(vroege 2-beurt)*

   (a) Als $z < x$ dan geldt: $z \leq x-1 \leq (y-1)-z$, zodat vanwege de inductiehypothese:
   $f(x-1, y-1, z) = (y-1)-(x-1)-z = y-x-z$

   (b) $z = x \Rightarrow f(x-1, y-1, z) = f(x-1, y-1, x) = f(x, y-1, x-1)$
   Daar $x-1 \leq x \leq (y-1)-(x-1)$ geldt volgens de inductieveronderstelling: $f(x-1, y-1, z) = (y-1)-x-(x-1) = y-x-z$

3. *(late 2-beurt)* Uit de voorwaarde volgt dat: $z-1 \leq x \leq (y-1)-(z-1)$, zodat vanwege de inductiehypothese:
   $f(x, y-1, z-1) = (y-1)-x-(z-1) = y-x-z$

Hieruit volgt dat:

$$\begin{aligned} f(x,y,z) &= \min(f(x-1,y-1,z-1), f(x-1,y-1,z), f(x,y-1,z-1)) \\ &= \min(y-x-z+1, y-x-z, y-x-z) \\ &= y-x-z \end{aligned}$$

geval c: $y-z < x < y$ en $z = \min(x,y,z)$
Uit de voorwaarde volgt direct: $z-1 \leq x-1$ en $x-1 < y-1$.

(a) Als $y-z < x-1$, dan is ook $(y-1)-(z-1) < (x-1)$, zodat wegens de inductieveronderstelling: $f(x-1, y-1, z-1) = 0$

(b) Als $y - z = x - 1$, dan $x - 1 \leq (y - 1) - (z - 1)$, zodat volgens geval b: $f(x-1, y-1, z-1) = (y-1) - (x-1) - (z-1) = y - x - z + 1 = x - 1 - x + 1 = 0$

Wegens eigenschap 1 volgt hieruit dat:

$$\begin{aligned} f(x,y,z) &= \min(f(x-1,y-1,z-1), f(x-1,y-1,z), f(x,y-1,z-1)) \\ &= \min(0, f(x-1,y-1,z), f(x,y-1,z-1)) \\ &= 0 \end{aligned}$$

Dus eigenschap 7 geldt ook voor z.

□

Uit het bewijs volgt rechtstreeks de optimale strategie voor het verdelen van 2/3-beurten. Behalve als $z \leq x \leq y - z$ of $x \leq z \leq y - x$ dient begonnen te worden met het weggeven van een aantal 3-beurten. Dit aantal is gelijk aan:

$$\begin{cases} y & \text{als } y = \min(x,y,z) \\ z & \text{als } x \geq y \geq z \\ x - y + z & \text{als } y - z < x < y \text{ en } z \leq x \\ 0 & \text{als } z \leq x \leq y - z \\ x & \text{als } z \geq y \geq x \\ x - y + z & \text{als } y - x < z < y \text{ en } x \leq z \\ 0 & \text{als } x \leq z \leq y - x \end{cases}$$

Na het verdelen van 3-beurten dienen zoveel mogelijk 2-beurten uitgedeeld te worden. Hierbij zijn vier gevallen te onderscheiden, waarbij x, y en z steeds de oorspronkelijk benodigde aantallen surveillanten aangeven:

1. $y = \min(x, y, z)$
   Er worden nu geen 2-beurten uitgedeeld.

2. $x \geq y \geq z$ of $z \geq y \geq x$
   Voor het laatste respectievelijk eerste timeslot is nu geen docent meer nodig. De 2-beurten worden dan over de overige twee timeslots verdeeld.

3. $z \leq x \leq y - z$ of $x \leq z \leq y - x$
   Omdat $x + z \leq y$ zal x keer een docent op de eerste twee timeslots en z keer een docent op de laatste twee timeslots worden ingeroosterd.

4. ($y - z < x < y$ en $z \leq x$) of ($y - x < z < y$ en $x \leq z$)
   Als voor de drie timeslots nog xx, yy en zz docenten nodig zijn, geldt na de verdeling van de 3-beurten dat $xx + zz = yy$, zodat xx keer een docent op de eerste twee timeslots en zz keer een docent op de laatste twee timeslots zal worden ingeroosterd. Dat $xx + zz = yy$ volgt uit het feit dat $xx = x - (x - y + z) = y - z$, $yy = y - (x - y + z) = 2y - x - z$ en $zz = z - (x - y + z) = y - x$.

Als de timeslots op een dag onderling verwisseld mogen worden kan alleen in de volgende gevallen het aantal 1-beurten nog verder geminimaliseerd worden.

1. Als $z = \min(x, y, z)$ en $x \geq y$, wissel dan de eerste 2 timeslots om. Er vindt dan een besparing plaats van $\min(z, x - y)$ 1-beurten.

2. Als $x = \min(x, y, z)$ en $z \geq y$, wissel dan de laatste 2 timeslots om. Er vindt dan een besparing plaats van $\min(x, z - y)$ 1-beurten.

3. Als $y = \min(x, y, z)$, wissel het tweede timeslot dan eerst om met het laatste timeslot en ga vervolgens verder als boven. De besparing is nu

$$\begin{cases} 2 \times \min(x, z) - y & \text{als } y \leq |z - x| \\ x - 2y + z & \text{als } y > |z - x| \end{cases}$$

Opgemerkt dient te worden dat de functie f slechts een theoretische ondergrens voor het aantal 1-beurten geeft en dat deze grens alleen geldt als docenten altijd gedurende een gehele dag ingezet kunnen worden. In de praktijk zullen docenten niet altijd beschikbaar zijn voor surveillance. Bovendien houdt f geen rekening met een eerlijke verdeling van het aantal surveillancebeurten over de docenten, terwijl dit wel een eis is waaraan een surveillancerooster dient te voldoen (zie paragraaf 3.1). Zoals uit een praktijkvoorbeeld in paragraaf 3.3 blijkt kan het aantal 1-beurten door deze oorzaken in de praktijk behoorlijk boven het theoretische minimum liggen.

3.2.2 Het geval: één tussenbeurt komt overeen met nul 1-beurten

Voor alle $x, y, z \in \mathbf{N}$ wordt de functie $f(x, y, z)$ gedefinieerd als het aantal 1-beurten dat resteert bij optimale indeling van 2/3-beurten, waarbij tussenbeurten in het geheel niet meetellen bij het bepalen van het aantal 1-beurten. Hierbij zijn x, y en z het aantal docenten dat voor het eerste, het tweede respectievelijk het derde timeslot op een dag nodig is. Deze functie heeft de volgende eigenschappen:

1. $f(x, y, z) \geq 0$

2. $f(x, y, z) = f(x, z, y) = f(y, x, z) = f(y, z, x) = f(z, x, y) = f(z, y, x)$

3. $\forall x, y, z > 0$:
   $f(x, y, z) = \min(f(x - 1, y - 1, z - 1), f(x - 1, y - 1, z),$
   $f(x, y - 1, z - 1), f(x - 1, y, z - 1))$

4. als $x \geq y \geq z$ dan $f(x, y, z) = \max(0, x - y - z)$

Eigenschap 1 is eenvoudig te verifiëren.

Bewijs van eigenschap 2

(a) Uit symmetrie-overwegingen volgt direct dat $f(x, y, z) = f(z, y, x)$.

(b) Laten $A = (x, y, z)$ en $B = (y, z, x)$ twee verdelingen van benodigde docenten over drie timeslots zijn. Een 2-beurt voor de eerste twee timeslots in A correspondeert met een tussenbeurt in B. Een 2-beurt voor de laatste twee timeslots in A correspondeert met een 2-beurt voor de eerste

2 timeslots in B. Een tussenbeurt in A correspondeert met een 2-beurt voor de laatste twee timeslots in B. Een 3-beurt in A correspondeert met een 3-beurt in B. Omdat tussenbeurten en 2-beurten hetzelfde bijdragen tot het aantal 1-beurten volgt hieruit dat we x, y en z cyclisch mogen verwisselen.

(c) Uit (a) en (b) volgt eigenschap 2.

□

Bewijs van eigenschap 3

Het toekennen van een 1-beurt kan niet tot een beter resultaat leiden dan het toekennen van een 2-beurt, een tussenbeurt of een 3-beurt. Immers: daar $x, y, z > 0$ is het uitdelen van alleen maar 1-beurten zeker niet optimaal. Het optimale pad bevat dus minstens één 2-beurt, één tussenbeurt of één 3-beurt. Omdat op het optimale pad 1-, 2-, 3- en tussenbeurten onderling verwisselbaar zijn, wordt er niets verloren als met een 2-beurt, tussenbeurt of 3-beurt gestart wordt.

□

Bewijs van eigenschap 4

Met inductie naar (z, y, x) lexicografisch.
$\underline{xyz = 0}$:
Zeker geldt nu dat $z = 0$. Eenvoudig is in te zien dat het minimale aantal 1-beurten in dit geval gelijk is aan $x - y$, terwijl eveneens $f(x, y, 0) = \max(0, x - y - 0) = x - y$.

$\underline{x, y, z \geq 1}$:
Neem aan dat eigenschap 4 geldt voor alle (x_1, y_1, z_1) met $x_1 \leq x$, $y_1 \leq y$, $z_1 \leq z$, $x_1 \geq y_1 \geq z_1$ en $x_1 + y_1 + z_1 < x + y + z$.
Vanwege eigenschap 3 en de inductieveronderstelling geldt:

$$\begin{aligned} f(x, y, z) &= \min(f(x-1, y-1, z-1), f(x-1, y-1, z), \\ &\qquad f(x, y-1, z-1), f(x-1, y, z-1)) \\ &= \min(\max(0, x-y-z+1), f(x-1, y-1, z), \\ &\qquad \max(0, x-y-z), f(x-1, y, z-1) \end{aligned}$$

We onderscheiden nu twee gevallen:

1. als $x - y - z \leq 0$ dan is $\max(0, x - y - z) = 0$, zodat wegens eigenschap 1 geldt dat $f(x, y, z) = 0$

2. als $x - y - z > 0$ dan is $x > y$ zodat wegens de inductieveronderstelling $f(x-1, y, z-1) = \max(0, x - y - z) = x - y - z$
   Voor $f(x, y, z)$ geldt nu dat:
   $f(x, y, z) = \min(x - y - z, f(x-1, y-1, z))$
   Wederom twee gevallen:

   (a) als $y > z$, dan is wegens de inductieveronderstelling $f(x-1, y-1, z) = x - y - z$

(b) als $y = z$, dan is wegens eigenschap 2 en de inductieveronderstelling $f(x-1, y-1, z) = f(x-1, z, y-1) = x - y - z$

In beide gevallen geldt nu dat $f(x, y, z) = x - y - z$

Dus eigenschap 4 geldt ook voor (x, y, z).

□

Uit het bewijs van eigenschap 4 volgt rechtstreeks de optimale strategie voor het verdelen van 2/3-beurten. Het aantal te verdelen 3-beurten is gelijk aan:

$$\begin{cases} \max(0, y - |x - z|) \\ \max(0, z - |x - y|) \\ \max(0, x - |y - z|) \end{cases}$$

Na het verdelen van 3-beurten dienen zoveel als mogelijk 2- en tussenbeurten uitgedeeld te worden. Voor een minimaal aantal 1-beurten moet dan de volgende strategie gevolgd worden:

1. bepaal de twee timeslots met de grootste behoefte aan docenten; zeg dat deze behoeftes t_1 respectievelijk t_2 zijn
2. wijs aan deze twee timeslots $\min(t_1, t_2)$ docenten toe middels 2-beurten of tussenbeurten; voor minstens één van de timeslots is nu geen docent meer nodig
3. wijs, zolang er nog twee timeslots een behoefte aan docenten hebben, middels een 2-beurt of tussenbeurt een docent aan deze timeslots toe

Om te bewijzen dat deze strategie optimaal is mogen we vanwege eigenschap 2 aannemen dat $y \leq x \leq z$. We onderscheiden 2 gevallen:

(1) $y \leq |x - z|$
Er zijn nu geen 3-beurten uitgedeeld zodat we nog steeds (x, y, z) als verdeling hebben. Toepassing van de eerste twee stappen van de strategie levert de verdeling $(0, y, z - x)$. Na toepassing van stap 3 is de verdeling gelijk aan $(0, 0, z - x - y)$. Het aantal 1-beurten is nu gelijk aan $|z - x| - y$. Dit aantal is optimaal volgens eigenschap 4.

(2) $y > |x - z|$
Na het uitdelen van $y - |x - z|$ 3-beurten is de verdeling van de resterende benodigde docenten gelijk aan $(z - y, z - x, 2z - y - x)$. Het middelste timeslot heeft nu de kleinste behoefte aan docenten, zodat toepassing van stap 1 en 2 van de strategie $(0, z - x, z - x)$ als verdeling oplevert. Na stap 3 is de overblijvende verdeling $(0, 0, 0)$. Er zijn dus 0 1-beurten, hetgeen uiteraard optimaal is.

□

Evenals in paragraaf 3.2.1 geldt ook nu weer dat de functie f slechts een theoretische ondergrens voor het aantal 1-beurten geeft en dat het werkelijke aantal 1-beurten hoger kan uitvallen als met andere eisen rekening wordt gehouden.

Tot slot kunnen we opmerken dat ongeacht of een tussenbeurt voor nul of twee 1-beurten telt, de optimale strategie bestaat uit het eerst uitdelen van een geschikt aantal 3-beurten, vervolgens een geschikt aantal 2-beurten en tenslotte de noodzakelijke 1-beurten.

3.3 Praktijkresultaten

3.3.1 Inleiding

De vraag is in hoeverre de in paragraaf 3.2.1 en 3.2.2 berekende theoretische ondergrenzen gehaald kunnen worden als het surveillancerooster ook nog de volgende beperkingen kent:

1. docenten kunnen, om wat voor reden dan ook, voor één of meer timeslots niet beschikbaar zijn voor surveillance,
2. het aantal surveillancebeurten van een docent dient evenredig te zijn met zijn/haar taakomvang,
3. zoveel als mogelijk dient bij elk examen een docent aanwezig te zijn die het te examineren vak ook geeft.

ad 1. Allereerst merken we op dat aanname 2 op pagina 22 (voor elk timeslot zijn altijd voldoende docenten beschikbaar) onverminderd van kracht blijft, daar anders sowieso geen surveillancerooster te maken is. Zoals in paragraaf 3.4 zal worden aangetoond, kan een rooster dan en slechts dan gemaakt worden als voor elke verzameling van timeslots voldoende docenten beschikbaar zijn.
Als het aantal docenten, dat gedurende de gehele dag voor surveillance beschikbaar is, kleiner is dan het grootste aantal docenten dat voor een bepaald timeslot op die dag nodig is, kan dat een negatieve invloed hebben op het aantal 1-beurten. Dit is bijvoorbeeld het geval als we op die dag alleen maar 3-beurten wensen uit te delen.

ad 2. Om het aantal surveillancebeurten voor een docent evenredig te laten zijn met zijn taakomvang, wordt het aantal surveillancebeurten voor docent j berekend als $(t_j / \sum t_i) \cdot S$, waarbij

- t_j: taakomvang van docent j, uitgedrukt als een fraktie van een volledige baan
- $\sum t_i$: totale taakomvang van alle docenten samen
- S totaal aantal te verdelen surveillancebeurten

In het algemeen zal het op deze manier berekende aantal beurten niet op een geheel getal uitkomen. Zowel $\lfloor (t_j / \sum t_i) \cdot S \rfloor$ als $\lceil (t_j / \sum t_i) \cdot S \rceil$ zullen daarom als een "eerlijk" aantal beurten voor docent j geaccepteerd worden. Dit geeft tevens een beperkte speelruimte van maximaal 1 beurt

per docent om daarmee het rooster (nog verder) te optimaliseren. Ondanks deze speelruimte kan het aantal 1-beurten drastisch toenemen als de beurten eerlijk verdeeld moeten worden, zoals het volgende (extreme) voorbeeld aantoont. Honderd docenten van een school hebben alle een volledige baan en moeten ieder precies 8.5 keer surveilleren. Op de laatste dag na is er voor alle dagen reeds een rooster gemaakt, waarin aan alle docenten 8 surveillancebeurten zijn toebedeeld. De 50 beurten van de laatste dag moeten nu elk aan een andere docent worden toegekend als we aan de eis van een eerlijke verdeling willen vasthouden. Dit betekent dat alleen deze dag reeds 50 1-beurten oplevert!

ad 3. Onmiddellijk is duidelijk dat wanneer de vakdocent op de dag waarop zijn vak geëxamineerd wordt alleen beschikbaar is op het betreffende timeslot, dit mogelijk een extra 1-beurt kan opleveren.

Interessant is nu de vraag of het überhaupt mogelijk is om een surveillancerooster te maken dat aan alle eisen voldoet en zo ja, of het mogelijk is dit te doen met niet meer dan k extra 1-beurten boven het theoretische minimum, met k een niet negatief geheel getal. Op de vraag of dit beslissingsprobleem NP-volledig is, wordt in paragraaf 3.4 nader ingegaan.

3.3.2 Beschrijving van een probleeminstantie

Geïnspireerd door de situatie op de dagschool van de "Hogeschool voor Economische Studies" te Rotterdam werd een redelijk grote instantie van het TISP-probleem gecreëerd. Deze instantie bestond uit:

- 95 modules
- 15 dagen van elk 3 timeslots
- 114 docenten

Appendix E geeft een complete beschrijving van deze instantie. Voor de gegeven instantie zijn de theoretische ondergrenzen voor het aantal 1-beurten bepaald. Het resultaat staat in onderstaande tabel. De voorlaatste en laatste kolom geven hierin het minimum aantal 1-beurten als een tussenbeurt voor twee respectievelijk nul 1-beurten geldt.

DagNr	x	y	z	Tus $\equiv 2$	Tus $\equiv 0$
1	10	17	15	0	0
2	15	20	20	0	0
3	27	5	21	38	1
4	5	9	29	20	15
5	29	15	20	19	0
6	29	9	6	20	14
7	28	9	25	35	0
8	27	25	25	2	0
9	6	15	23	8	2
10	21	23	28	5	0
11	17	28	13	0	0
12	23	6	25	36	0
13	8	13	16	3	0
14	14	8	9	7	0
15	27	25	19	2	0

Het totaal aantal 1-beurten bedraagt 195 als een tussenbeurt voor twee 1-beurten telt en 32 als een tussenbeurt equivalent is met nul 1-beurten.

3.3.3 Resultaten van een directe implementatie

Voor de eerder beschreven instantie van het TISP-probleem is een algoritme geïmplementeerd dat van de volgende principes uitgaat:

1. de gebruiker kan allereerst kiezen of een tussenbeurt overeen dient te komen met nul of twee 1-beurten
2. geen enkele docent mag te vaak surveilleren; met andere woorden als het aantal surveillancebeurten van een docent uiteindelijk niet evenredig blijkt te zijn met zijn/haar taakomvang, dan kan dit alleen maar komen doordat deze docent een te klein aantal malen moet surveilleren, terwijl andere docenten $\lceil (t_j / \sum t_i) \cdot S \rceil$ beurten hebben om dit te compenseren
3. de docenten worden dag voor dag ingeroosterd voor surveillance, waarbij de dagen in willekeurige volgorde werden gekozen
4. per dag worden eerst die docenten ingeroosterd die een module geven die op deze dag geëxamineerd wordt. Als voor een module zo'n docent niet gevonden kan worden, wordt een docent ingeroosterd die hetzelfde vak geeft. Als dit ook niet lukt wordt een willekeurige docent genomen. Bij het inroosteren wordt voor deze docenten, afhankelijk van de waardering van een tussenbeurt, steeds zoveel als mogelijk één van de twee in paragraaf 3.2 vermelde strategiën gevolgd die moet leiden tot een minimaal aantal 1-beurten
5. vervolgens worden de overige te verdelen beurten voor een dag onder de andere docenten verdeeld, waarbij ook nu steeds de eerder geformuleerde strategie wordt gevolgd
6. als er keuze is tussen meerdere docenten om in te roosteren krijgen de volgende docenten de voorkeur:

(a) docenten die op de nog niet ingeroosterde dagen vaak verhinderd zijn

(b) docenten die na het inroosteren op deze dag of helemaal niet meer of voor minstens twee beurten ingeroosterd kunnen worden

Met de eerste voorkeursmethode wordt geprobeerd te voorkomen dat het algoritme vastloopt, omdat er op een gegeven moment te weinig docenten beschikbaar zijn. Met de tweede voorkeursmethode wordt geprobeerd te voorkomen dat een nog toe te kennen 2- of 3-beurt moet worden opgesplitst in een aantal 1-beurten, omdat een docent anders te vaak moet surveilleren

7. als er geen docent is die voor een (volgens de optimale strategie wenselijke) 2-, tussen of 3-beurt kan worden ingezet, dan wordt deze beurt opgesplitst in een aantal kleinere beurten. Bij een 3-beurt wordt dan eerst nog gekeken of opsplitsing in een 2- of tussenbeurt en een 1-beurt mogelijk is

8. in het geval dat een tussenbeurt correspondeert met twee 1-beurten, is, wanneer er alleen nog maar docenten voor het eerste en derde timeslot nodig zijn, zo veel als mogelijk één docent op deze timeslots ingeroosterd (en dus een tussenbeurt gecreëerd). Dit is gedaan omdat ver weg wonende docenten vermoedelijk de voorkeur zullen geven aan een dag met een tussenbeurt boven een extra dag waarop gesurveilleerd moet worden

Problemen met een tekort aan beschikbare docenten die nog voor één of meer keren kunnen worden ingeroosterd, zijn vooral bij het inroosteren van de laatste (paar) dag(en) te verwachten. Bovendien zullen er dan relatief veel docenten zijn die wel nog voor een 1-beurt, maar niet meer voor een 2-, 3- of tussenbeurt ingeroosterd kunnen worden. Om deze reden werd het programma 70 keer gerund, waarbij de volgorde van inroosteren van de dagen steeds random werd gekozen. In de helft van de gevallen werd een tussenbeurt gelijk gesteld aan nul tussenbeurten, terwijl in de andere helft een tussenbeurt voor twee 1-beurten telde. Tien keer eindigde een run voortijdig, omdat er op de laatste dag een tekort aan beschikbare, en nog inzetbare, docenten werd geconstateerd. De resultaten van de overige 60 runs staan in Appendix F. Uit deze resultaten blijkt dat een geheel beëindigde run nooit een erg slecht resultaat te zien gaf: het aantal extra 1-beurten ten opzichte van het theoretische minimum was altijd erg klein, terwijl slechts een klein gedeelte van de docenten een aantal beurten had dat niet in overeenstemming was met hun taakomvang. In één geval werd zelfs het minimale aantal 1-beurten bereikt, terwijl de beurten toch eerlijk over de docenten verdeeld werden.

3.3.4 Resultaten met een GA

Als representatie van een chromosoom C is gekozen voor een $t \times d$-matrix (A_{ij}), met t: het aantal timeslots en d: het aantal docenten.

$$A_{ij} = \begin{cases} 1 & \text{als docent } j \text{ op timeslot } i \text{ surveilleert} \\ 0 & \text{als docent } j \text{ op timeslot } i \text{ niet surveilleert} \end{cases}$$

Elk chromosoom heeft t genen; elk gen correspondeert met een rij (timeslot) uit de matrix (A_{ij}).

De beginpopulatie bestaat uit 50 surveillanceroosters, waarbij op een random wijze docenten aan timeslots zijn gekoppeld, maar wel zo dat voor elk timeslot het juiste aantal docenten is ingeroosterd. Een rooster kan dan alleen niet optimaal zijn vanwege het niet eerlijk verdeeld zijn van de beurten over de docenten en/of een te groot aantal 1-beurten. Deze niet-optimaliteit komt tot uitdrukking in de evaluatiefunctie:

$$f(C) = \frac{A_0}{A_1 + W \cdot A_2}$$

met

A_0	:	theoretisch minimum aantal 1-beurten; dit zorgt er voor dat voor een optimaal rooster C geldt dat $f(C) = 1$
A_1	:	totaal aantal 1-beurten; afhankelijk van de wens van de gebruiker telt een tussenbeurt voor nul of twee 1-beurten
A_2	:	$\sum (\min \mid (\lfloor (t_j / \sum t_i) \cdot S \rfloor - Q_j \mid, \mid \lceil (t_j / \sum t_i) \cdot S \rceil - Q_j \mid))$, waarbij Q_j het werkelijke aantal beurten van docent j in het rooster is en er gesommeerd wordt over alle docenten. A_2 geeft aan in hoeverre het rooster afwijkt van een eerlijke verdeling van de beurten
$W > 0$	:	wegingsfactor, die aangeeft hoe ernstig een verschil tussen werkelijk en gewenst aantal beurten is ten opzichte van de 1-beurten

Voor de eerder beschreven instantie van het TISP-probleem is deze functie goed gedefinieerd, omdat de noemer nooit 0 kan worden daar er minstens 32 1-beurten zijn.

De gebruikte operatoren zijn:

1. *selectie* via de roulettewheel methode, waarbij een tussenpopulatie van grootte pop_size werd gegenereerd

2. *crossover*: als twee chromosomen overeenkomstige genen met elkaar uitwisselen, komt dit neer op het uitwisselen van twee roosters voor twee overeenkomstige timeslots. Dit heeft tot gevolg dat het aantal ingeroosterde docenten per timeslot niet verandert. Crossover tussen twee roosters, die elk een juist aantal docenten per timeslot hebben, kan de correctheid wat betreft dit punt dus nooit tenietdoen. Er kunnen twee soorten crossover gebruikt worden:

   (a) *rijcrossover*: twee roosters wisselen een rij met elkaar uit

   (b) *uniforme crossover*: voor elke rij van de twee nieuw te creëren roosters wordt random bepaald welk ouder-rooster zijn rij doorgeeft aan welk kind-rooster

   In beide gevallen geven de twee aselect gekozen ouders uit de tussenpopulatie twee kinderen in de nieuwe populatie.

3. *permutatie*: binnen één rooster worden er beurten voor docenten veranderd. We onderscheiden:

   (a) *verplaatsmutatie*: voor elk timeslot (rij) van het rooster worden de volgende stappen ondernomen:

i. kies uit alle op dit timeslot ingeroosterde docenten, die in totaal te vaak zijn ingeroosterd, aselect een docent

ii. kies uit alle op dit timeslot niet ingeroosterde, maar wel beschikbare docenten, die een tekort aan surveillancebeurten hebben, aselect een docent

iii. wanneer voor beide voorgaande stappen een docent wordt gevonden, krijgt de tweede docent de beurt van de eerste docent toegewezen

Verplaatsmutatie is een operatie waarbij doelbewust wordt gewerkt aan een eerlijkere verdeling van de surveillancebeurten over de docenten

(b) *wisselmutatie*: twee aselect gekozen docenten ruilen zo mogelijk twee (aselect gekozen) beurten om. Deze operatie is toegevoegd omdat dit ruilen in de praktijk vaak gebeurt en een positieve invloed kan hebben op het aantal 1-beurten van een rooster

De eerste resultaten met het hierboven beschreven GA waren niet erg gunstig. Met bijvoorbeeld een kans op uniforme crossover van 40%, een kans op verplaatsmutatie van 5% en een kans op wisselmutatie van 1% werd er in het geval dat een tussenbeurt gelijk stond aan nul 1-beurten na 3000 generaties als beste chromosoom een rooster gevonden met 472 1-beurten en een totaal absoluut verschil van 129 tussen het gewenste aantal beurten per docent en het werkelijke aantal beurten. Andere runs met andere waarden voor de parameters leverden geen noemenswaardig beter resultaat op. Nog meer veelzeggend is dat de gemiddelde fitness van de chromosomen in de laatst gegenereerde populatie nauwelijks beter was dan dat van de beginpopulatie. Wat kan van deze slechte performance van het GA de oorzaak zijn? Naar de mening van schrijver dezes ligt de oorzaak hiervan in de crossoveroperatie(s). Veronderstel dat een docent tien surveillancebeurten moet doen. Als een docent in de twee ouder-roosters inderdaad deze tien beurten heeft, maar wel op andere timeslots, dan is de kans zeer groot dat in elk van de twee kind-roosters het aantal beurten ongelijk is aan tien (bijvoorbeeld 13 in het eerste kind-rooster en 7 in het tweede). De crossover-operatie verhindert daardoor dat er convergentie naar een goed rooster optreedt.
Eén mogelijkheid om het algoritme te verbeteren is om de crossover-operatie niet of slechts met een zeer kleine kans uit te voeren. Dit heeft echter als nadeel dat het algoritme vanwege de verplaatsmutatie vervalt tot een gewoon "hill-climbing" algoritme. Een andere mogelijkheid zou zijn om een andere representatie te kiezen, waarin wel een geschikte crossover-operatie toepasbaar is. Het vinden van een dergelijke representatie is nog onderwerp van studie.
Tot slot kan nog opgemerkt worden dat het "falen" van het GA niet een zeer grote ramp is, daar het nauwelijks te verwachten is dat een GA tot betere resultaten leidt dan de directe implementatie in paragraaf 3.3.3. Dit in tegenstelling tot het MESP-probleem, waarvoor geen directe implementatie bekend is.

3.4 Is TISP NP-volledig?

Is het mogelijk om voor een gegeven instantie van het TISP-probleem een surveillancerooster voor docenten te maken dat voldoet aan de drie op pagina 22 geformuleerde eisen? Wanneer deze vraag positief beantwoord kan worden, is het vervolgens interessant om na te gaan in hoeverre de beperkingen van de docenten en de eis dat de beurten "eerlijk" verdeeld moeten zijn, een negatieve invloed hebben op het minimale aantal 1-beurten. Met andere woorden: is het mogelijk een rooster te maken met niet meer dan k extra 1-beurten boven het theoretische minimum? Hierbij is k een niet negatief geheel getal.
In deze paragraaf wordt een eerste aanzet gegeven tot de beantwoording van de vraag of deze twee beslissingsproblemen al dan niet NP-volledig zijn. We beperken ons hierbij voorlopig tot het geval dat het aantal surveillancebeurten van een docent geheeltallig is en dat het verboden is om een docent meer of minder vaak te laten surveillen dan dit opgegeven aantal. Dit probleem zullen we TISP* noemen.

Laat voor TISP* de $t \times d$-matrix (A_{ij}) gedefinieerd zijn door

$$A_{ij} = \begin{cases} 0 & \text{als docent } j \text{ verhinderd is op timeslot } i \\ 1 & \text{als docent } j \text{ beschikbaar is op timeslot } i \end{cases}$$

waarbij t het aantal timeslots is en d het aantal docenten. Laat verder a_i het aantal benodigde docenten op timeslot i zijn, b_j het aantal surveillancebeurten van docent j en t het totaal aantal surveillancebeurten van alle docenten samen. Zonder beperking van de algemeenheid mogen we aannemen dat elke $a_i > 0$ en elke $b_j > 0$. Er moet nu gelden dat $t = \sum a_i = \sum b_j$. Uit de matrix (A_{ij}) is een vierkante $(0,1)$-matrix (B_{ij}) te construeren, waarbij elke docent precies één keer moet surveilleren en elk timeslot precies één docent nodig heeft. (B_{ij}) ontstaat uit (A_{ij}) door iedere rij i a_i keer te kopiëren en vervolgens elke kolom j b_j keer. Het TISP*-probleem is nu op te vatten als een koppelingsprobleem: koppel elk timeslot aan precies één docent. Volgens de huwelijksstelling van Hall is dit dan en slechts dan mogelijk als voor elke verzameling van k timeslots minstens k docenten beschikbaar zijn ($1 \leq k \leq t$).

Een aantal elementen in een matrix heet *onafhankelijk* als er geen twee of meer in dezelfde rij of kolom liggen. Om te controleren of een surveillancerooster mogelijk is, kan ook worden nagegaan of het maximale aantal onafhankelijke énen in B gelijk is aan t. Toepassing van de max-min stelling van König-Egerváry geeft vervolgens dat een surveillancerooster mogelijk is als het minimale aantal rijen en kolommen, die tezamen alle énen bevatten, gelijk is aan t.

De matrix B correspondeert als volgt met een bipartiete graaf $G = (V, E)$: laat V_1 een verzameling van knopen zijn door voor elk timeslot een knoop te nemen en V_2 een verzameling van knopen door voor elke docent een knoop te nemen. Laat verder $V = V_1 \cup V_2$ en $(v, w) \in E$ als de docent, die correspondeert met w, beschikbaar is op het timeslot, dat correspondeert met v. Omgekeerd hoort bij iedere bipartiete graaf $G = (V_1 \cup V_2, E)$ met $E \subset V_1 \times V_2$ en $|V_1| = |V_2|$ een instantie van het TISP*-probleem: neem voor elke knoop $v \in V_1$ een timeslot, voor elke knoop $w \in V_2$ een docent en laat een docent beschikbaar zijn op een bepaald timeslot als tussen de hiermee corresponderende knopen in V_1 en V_2

een tak bestaat. Een verzameling $W \subset V$ heet een *knooppuntenbedekking* als iedere $e \in E$ incident is met minstens één $w \in W$.
Claim: Daar G bipartiet is, is het maximum aantal takken in een koppeling gelijk aan het minimaal aantal knooppunten in een knooppuntenbedekking.
Bewijs: Toepassing van de stelling van König-Egerváry op de bijbehorende $(0,1)$-matrix van G geeft dat het maximum aantal onafhankelijke énen gelijk is aan het minimum aantal rijen en kolommen waar alle énen in zitten. Onafhankelijke énen corresponderen met een koppeling. De rijen en kolommen waar alle énen inzitten vormen een knooppuntenbedekking.
Gevolg: Voor een instantie van het TISP*-probleem bestaat een surveillancerooster dan en slechts dan als een knooppuntenbedekking voor de bijbehorende bipartiete graaf minimaal t knopen bevat.

Het "vertex cover" probleem vraagt of er voor een gegeven graaf en een gegeven positief geheel getal k een knooppuntenbedekking bestaat die k knopen bevat. Van het "vertex cover" probleem is bekend dat het NP-volledig is (zie voor een bewijs [CorLeiRiv92, p. 950]). Bij beperking tot bipartiete grafen $G = (V_1 \cup V_2, E)$ met $E \subset V_1 \times V_2$ en $|V_1| = |V_2|$ wordt het vinden van een knooppuntenbedekking met k knopen triviaal zodra $k \geq |V_1|$. Elke verzameling W die bijvoorbeeld V_1 bevat, voldoet dan immers. De grote vraag die beantwoord moet worden, is of er ook een knooppuntenbedekking bestaat met $V_1 - 1$ knopen. De beperking dat de graaf bipartiet is, maakt het beantwoorden van de vraag ongetwijfeld gemakkelijker. Of daarmee dit beslissingsprobleem dan niet meer NP-volledig wordt, is bij de schrijver van dit afstudeerverslag onbekend. Het zou een onderwerp van studie kunnen zijn.

Zelfs als het TISP*-probleem in polynomiale tijd opgelost kan worden, wil dat nog niet zeggen dat het voor het TISP-probleem ook het geval is. Voor elke docent kan immers $(t_j / \sum t_i) \cdot S$ zowel naar boven als naar beneden worden afgerond, zodat het simpelweg controleren van de 2^d mogelijke TISP*-problemen niet tot een polynomiaal algoritme voor het TISP-probleem leidt.

4 Conclusies en aanbevelingen

In dit afstudeerverslag zijn een tweetal schoolroosterproblemen (MESP, het examenroosterprobleem en TISP, het surveillanceroosterprobleem) beschreven en aan een onderzoek onderworpen. Er is aangetoond dat het MESP-probleem, evenals RMESP en RMESP*, tot de klasse van NP-volledige problemen behoort. RMESP* is zelfs volkomen equivalent met het bekende graph coloring problem. Verder is aangetoond dat een GA in korte tijd een redelijk goed (maar niet noodzakelijk optimaal) examenrooster op kan leveren. Bij gebruik van dit GA leiden 2-punts crossover en uniforme crossover tot duidelijk betere resultaten dan 1-punts crossover.
Van het TISP-probleem kon niet worden aangetoond of het al dan niet NP-volledig is. Wel kon worden aangetoond dat TISP* equivalent is met het vertex cover problem voor bipartiete grafen, maar of de beperking tot bipartiete grafen de NP-volledigheid van het vertex cover problem verstoord is voor ons nog steeds een open vraag. Voor het TISP-probleem werd in het geval dat

- er drie disjuncte timeslots per dag zijn
- er altijd voldoende docenten beschikbaar zijn
- beschikbare docenten de gehele dag kunnen worden ingezet
- een tussenbeurt gelijkgesteld wordt aan nul of twee 1-beurten

een optimale strategie voor het verdelen van 2/3-beurten ontwikkeld ter minimalisatie van het aantal 1-beurten. Wanneer docenten niet altijd de gehele dag beschikbaar zijn en de beurten "eerlijk" over de docenten verdeeld dienen te worden, laat de praktijk zien dat deze strategie tot een rooster leidt waarvoor het aantal 1-beurten in het algemeen niet ver boven het theoretische minimum ligt.

Er is op het gebied van (school)roosters nog veel werk dat gedaan kan worden. In dit afstudeerproject is, om maar iets te noemen, geen aandacht geweest voor het lesroosterprobleem. Maar ook als we ons beperken tot MESP en TISP verdienen vele zaken nog nadere aandacht:

- omdat MESP en het graph coloring problem veel met elkaar te maken hebben, mag verwacht worden dat een GA dat voor één van beide problemen ontwikkeld is, na de noodzakelijke aanpassingen even goede resultaten voor het andere probleem zal opleveren. Nagegaan kan worden of dit inderdaad het geval is.
- hetzelfde kan gedaan worden voor TISP en het vertex cover problem voor bipartiete grafen
- generalisatie van MESP door ook rekening te houden met de beschikbare ruimte en het aantal docenten dat per timeslot voor surveillance beschikbaar is
- vinden van een goede representatie en bijbehorende crossover- en mutatie-operatoren voor een GA voor TISP

- wat stelt minimalisatie van het aantal 1-beurten in TISP voor in het equivalente vertex cover probleem en hoe vertaalt zich het algoritme, dat het aantal 1-beurten minimaliseert, in termen van de bipartiete grafen?
- generalisatie van het TISP-probleem door:
  - een tussenbeurt gelijk te stellen aan $\alpha \cdot$ 1-beurt met $0 \leq \alpha \leq 2$
  - meer dan drie timeslots op een dag
- onderzoek naar de eventuele NP-volledigheid van het TISP-probleem
- combinatie van MESP en TISP: gezocht wordt een algoritme dat een zo optimaal mogelijk examenrooster voor studenten produceert, maar dat tegelijk in meerdere of mindere mate rekening houdt met de wensen van de docenten. Het algoritme dient er in ieder geval voor te zorgen dat het aantal benodigde docenten voor een bepaald timeslot het aantal beschikbare docenten niet te boven gaat. Daarnaast zou het voorwaarden kunnen scheppen voor een surveillancerooster, waarbij het aantal 1-beurten zo klein mogelijk is

Referenties

[Baa91] Baase, S., *Computer Algorithms, Introduction to Design and Analysis*, second edition, Addison-Wesley, 1991.

[ColDorMan91] Colorni, A., Dorigo, M., en Maniezzo, V., *Genetic Algorithms and Highly Constrained Problems: The Time-Table Case*, in Schwefel, H.-P., and Männer, R. (Editors), Proceedings of the First International Conference on Parallel Problem Solving from Nature (PPSN), Springer Verlag, Lecture Notes in Computer Science, Vol. 496, 1991.

[CorFanMel93] Corne, D., Fang, H.-L., en Mellish, C., *Solving the Modular Exam Scheduling Problem with Genetic Algorithms*, 1993, DAI Research Paper No. 622, gepubliceerd in: Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Proceedings of the 6th International Conference, Edinburgh, 1992.

[CorLeiRiv92] Cormen, T.H., Leiserson, C.E., Rivest, R.L., *Introduction to Algorithms*, MIT Press, 1992

[DanNig93] Dannis B., en Nigten R., *Genetische algoritmen* in Seminarium Geavanceerde Algoritmen voorjaar 1993, Vakgroep Informatica RUL, 1993, Jeannette de Graaf, Walter Kosters (Editors). (via WWW verkrijgbaar op URL http://www.wi.leidenuniv.nl/home/kosters/boek93.ps.gz)

[EveItaSha76] Even, S., Itai, A., en Shamir, A., *On the Complexity of Timetable and Multicommodity Flow Problems*, SIAM Journal on Computing **5**, (1976) 691–703.

[GarJoh79] Gary, M.R., en Johnson, D.S., *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, 1979.

[Gol89] Goldberg, D.E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, 1989.

[JonSpe89] Kenneth A. De Jong, William M. Spears, *Using Genetic Algorithms to Solve NP-Complete Problems*, 124–133, in Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, 1989.

[Mic94] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, second edition, Springer Verlag, 1994.

[PaeLucPet94] Paechter, B., Luchian, H., en Petruic, M., *Two solutions to the General Timetable Problem Using Evolutionary Methods*, in Proceedings of the Evolutionary Computation Conference, part of the IEEE World Congress on Computational Intelligence, Orlando, 26-29 June 1994, IEEE Press, 1994, Z. Michalewicz, H. Kitano, D. Schaffer, H.-P. Schwefel, D. Fogel (Editors).

Appendix A: inschrijvingen voor examens

In Appendix A worden de inschrijvingen gepresenteerd van studenten voor de hertentamens aan de Hogeschool voor Economische Studies te Rotterdam. Van elke student wordt het studentnummer gegeven en de tentamens waarvoor hij zich heeft ingeschreven. De unieke modulecodes van de tentamens zijn omgezet naar tentamennummers tussen 1 en 95. Deze data vormden de invoer voor het MESP–probleem. Enkele statistieken:

het aantal studenten bedraagt	: 153
het aantal tentamens bedraagt	: 95
het aantal timeslots per dag	: 3
het totaal aantal timeslots	: 30

6467 13;
6521 65 32 33 64;
6564 3 54;
6645 45 26 40;
6696 37;
6807 28 66 4;
6882 94 95 87;
6904 4 45 32 94;
6920 94;
6998 66 38 67 23;
7102 26;
7110 54 55;
7234 26;
7250 85 36 5;
7269 95 13 12;
7277 95 26;
7315 59;
7358 7;
7382 55 34 51 54;
7390 70 6;
7528 3;
7552 32 48 8 41 3 55 35 54 34 33;
7609 45 94;
7641 73;
7684 13 53;
7765 95 49 51 41;
7773 4 3;
7811 32 36 15 84;
7889 87;
7897 27 3 4;
7900 40 23 3 27 22;
7927 28 67;
7943 21 11 8 10;
7951 35 13;

7986 27;
8060 27 4 26 86 23 20;
8095 14;
8109 76;
8133 21;
8176 27 3 4 23;
8214 4;
8222 52;
8230 2 74 24;
8265 38 54;
8281 66;
8303 21 1 22;
8311 68;
8362 39 23;
8370 4;
8389 91;
8427 29 7 20 21;
8478 8 37;
8532 4;
8540 46 67 83;
8559 66 23;
8575 35;
8583 37;
8591 23;
8613 40;
8621 10 94;
8664 94 89;
8680 36;
8699 48 50 49 12 13;
8729 14;
8788 18;
8796 89 94;
8818 12 48;
8826 50;
8834 58 5;
8869 21 10;
8877 25 26;
8907 73;
8923 75;
8974 25 26;
8982 5 6;
9024 94;
9032 51 55;
9040 85 60;
9059 26 21;
9075 33 48 8 94;
9083 8 89 94 7;
9113 46 47 68 34;

9121 81 90 41 12;
9156 94 8;
9164 57;
9202 25 37;
9210 21 26 29;
9253 4 13 35;
9261 8 10;
9296 21;
9326 75;
9334 94;
9350 83;
9385 94 10;
9423 30 56 8;
9466 32;
9474 21;
9504 8 94;
9512 72;
9520 25;
9539 20 95 26 94 21;
9547 3;
9555 46 7 8 47;
9644 30 93 7;
9652 93 30;
9733 18 46 12 61;
9849 19 91 12 18 30 31;
9865 16 78 9;
9911 31;
9946 30;
9954 30 94;
10049 8 30 31;
10073 31 91 90 30;
10138 8;
10197 12;
10340 47;
10359 46;
10472 88 8;
10502 41;
10545 61 12;
10596 31 19 18 12;
10626 92;
10650 81 91;
10669 61;
10707 31 8 93 30;
10723 81;
10731 31 12 91;
10766 8;
10782 91 8;
10812 8 12 7 17 31;

10820 32 34 63 43 69;
10839 12 91 90 61 62 17;
10847 39 22 23;
10863 7;
10898 12 19 90;
10952 30 16;
10995 42;
11002 7 31;
11037 5;
11053 10 94;
11118 90 30 12 61 91 31 81 19;
11134 41 12;
11223 82;
11363 27 26 25;
11479 11 7 26 8 10;
11517 31 91 19 18 12;
11614 47;
11622 46 90 12;
11630 5 84 6 33 14;
11657 8;
53619 12 91 46 41;
60232 80 71 77 78 79;
65471 44;

Appendix B: resultaten voor MESP

Appendix B geeft een overzicht van de resultaten van een GA met echte data voor het MESP-probleem. Hierbij werd steeds gebruikt gemaakt van gewone mutatie en werd er geëxperimenteerd met verschillende vormen van crossover.

alleen 1-punts crossover

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00719	138	0.00467	223.80
51	0.03571	27	0.02285	48.30
101	0.07692	12	0.04374	24.00
151	0.10000	9	0.07151	14.10
201	0.25000	3	0.16626	6.30
251	1.00000	0	0.27786	4.20
301	1.00000	0	0.29286	3.60

alleen 2-punts crossover

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00571	174	0.00473	214.50
51	0.06250	15	0.03685	29.10
101	1.00000	0	0.15328	17.70
151	1.00000	0	0.19948	8.70
201	1.00000	0	0.22841	6.00
251	1.00000	0	0.27143	4.20
301	1.00000	0	0.50714	3.00

alleen uniforme crossover

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00625	159	0.00486	209.10
51	0.03571	27	0.02535	42.30
101	0.10000	9	0.05921	19.20
151	0.25000	3	0.11374	11.40
201	1.00000	0	0.23581	7.20
251	1.00000	0	0.27769	4.80
301	1.00000	0	0.37857	3.00

1-punts crossover en 2-punts crossover

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00690	144	0.00472	223.50
51	0.03226	30	0.02159	48.00
101	0.10000	9	0.04845	24.90
151	0.25000	3	0.09142	14.70
201	1.00000	0	0.25804	6.60
251	1.00000	0	0.43198	3.90
301	1.00000	0	0.46429	2.40

1-punts crossover en uniforme crossover

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00719	138	0.00489	210.30
51	0.06250	15	0.03386	32.70
101	1.00000	0	0.17420	15.00
151	1.00000	0	0.22021	7.50
201	1.00000	0	0.35714	3.60
251	1.00000	0	0.52857	2.40
301	1.00000	0	0.67857	1.80

2-punts crossover en uniforme crossover

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00637	156	0.00477	214.50
51	0.14286	6	0.04597	29.10
101	0.25000	3	0.13348	12.30
151	1.00000	0	0.41055	4.50
201	1.00000	0	0.52857	2.40
251	1.00000	0	0.62500	1.50
301	1.00000	0	0.62500	1.50

1-punts, 2-punts en uniforme crossover

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00806	123	0.00488	215.70
51	0.05263	18	0.03051	36.90
101	0.14286	6	0.06668	20.40
151	0.25000	3	0.12501	9.30
201	1.00000	0	0.41698	4.50
251	1.00000	0	0.50286	3.30
301	1.00000	0	0.50714	3.00

Appendix C: resultaten voor MESP

In deze Appendix wordt een overzicht gegeven van de resultaten van een GA met echte data voor het MESP-probleem. Hierbij werd er geëxperimenteerd met verschillende vormen van mutatie.

1-punts crossover met kans : 7.0000000000E-01
wisselmutatie met kans : 3.0000000000E-03

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00552	180	0.00447	225.00
51	0.01818	54	0.01274	85.50
101	0.01923	51	0.01334	81.60
151	0.02041	48	0.01368	80.10
201	0.01923	51	0.01358	80.10
251	0.01923	51	0.01416	76.50
301	0.01923	51	0.01401	77.40

2-punts crossover met kans : 7.0000000000E-01
wisselmutatie met kans : 3.0000000000E-03

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00826	120	0.00502	212.40
51	0.01923	51	0.01406	74.10
101	0.02174	45	0.01417	73.80
151	0.02174	45	0.01411	74.10
201	0.02174	45	0.01417	74.10
251	0.02174	45	0.01427	73.80
301	0.02174	45	0.01427	73.80

uniforme crossover met kans : 7.0000000000E-01
wisselmutatie met kans : 3.0000000000E-03

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00602	165	0.00445	230.40
51	0.05263	18	0.02800	38.70
101	0.05263	18	0.03041	34.80
151	0.05263	18	0.03061	34.50
201	0.05263	18	0.03385	31.50
251	0.05263	18	0.03385	31.50
301	0.06250	15	0.03610	30.30

1-punts crossover met kans : 7.0000000000E-01
dagwisselmutatie met kans : 3.0000000000E-03

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00704	141	0.00523	196.80
51	0.02041	48	0.01449	74.10
101	0.02041	48	0.01471	72.90
151	0.02703	36	0.01648	65.70
201	0.05263	18	0.02097	55.50
251	0.05263	18	0.02271	52.80
301	0.05263	18	0.02325	51.30

2-punts crossover met kans : 7.0000000000E-01
dagwisselmutatie met kans : 3.0000000000E-03

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00649	153	0.00498	207.30
51	0.02941	33	0.01818	58.50
101	0.03226	30	0.02025	53.40
151	0.03226	30	0.02091	52.20
201	0.04000	24	0.02217	50.70
251	0.10000	9	0.03317	42.60
301	0.10000	9	0.03479	41.40

uniforme crossover met kans : 7.0000000000E-01
dagwisselmutatie met kans : 3.0000000000E-03

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00592	168	0.00441	229.80
51	0.06250	15	0.03494	30.90
101	0.07692	12	0.04281	25.80
151	0.07692	12	0.04534	24.30
201	0.10000	9	0.05123	22.50
251	0.10000	9	0.05954	18.60
301	0.10000	9	0.05900	18.90

1-punts crossover met kans : 7.0000000000E-01
inversiemutatie met kans : 3.0000000000E-03

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00602	165	0.00454	224.70
51	0.02174	45	0.01402	78.60
101	0.02174	45	0.01492	73.80
151	0.02703	36	0.01638	69.30
201	0.02941	33	0.01649	69.60
251	0.02941	33	0.01649	69.60
301	0.03571	27	0.01771	65.10

2-punts crossover met kans : 7.0000000000E-01
inversiemutatie met kans : 3.0000000000E-03

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00806	123	0.00502	213.00
51	0.02326	42	0.01652	63.60
101	0.02326	42	0.01755	58.80
151	0.02326	42	0.01750	59.10
201	0.02326	42	0.01738	59.40
251	0.02326	42	0.01738	59.40
301	0.02326	42	0.01754	58.80

uniforme crossover met kans : 7.0000000000E-01
inversiemutatie met kans : 3.0000000000E-03

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.01563	63	0.00596	199.80
51	0.07692	12	0.03002	41.10
101	0.07692	12	0.03117	39.60
151	0.07692	12	0.03156	39.00
201	0.07692	12	0.03149	39.00
251	0.07692	12	0.03477	34.50
301	0.07692	12	0.03477	34.50

2-punts crossover met kans : 7.0000000000E-01
gewone mutatie met kans : 1.5000000000E-03
inversiemutatie met kans : 1.5000000000E-03
Slechts 1 van beide mutaties werd met een totale kans van 0.003 gekozen.

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00826	120	0.00467	224.10
51	0.04545	21	0.02359	51.00
101	0.10000	9	0.04419	31.80
151	0.14286	6	0.05976	22.50
201	1.00000	0	0.16859	15.30
251	1.00000	0	0.19677	11.40
301	1.00000	0	0.37449	6.60

2-punts crossover met kans : 7.0000000000E-01
gewone mutatie met kans : 1.5000000000E-03
wisselmutatie met kans : 1.5000000000E-03
Slechts 1 van beide mutaties werd met een totale kans van 0.003 gekozen.

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00787	126	0.00523	199.20
51	0.04000	24	0.02370	49.20
101	0.14286	6	0.05224	30.30
151	0.25000	3	0.09984	18.90
201	1.00000	0	0.27427	12.30
251	1.00000	0	0.31169	7.80
301	1.00000	0	0.41883	4.80

Appendix D: resultaten voor MESP

In deze Appendix wordt een overzicht gegeven van de resultaten van een GA met echte data voor het MESP-probleem. Hierbij werd werd er geëxperimenteerd met andere en zwaardere boetes.

1-punts crossover met kans: 0.7
gewone mutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00147	681	0.00106	964.20
51	0.00325	307	0.00281	364.40
101	0.00524	190	0.00372	275.90
151	0.00746	133	0.00472	223.50
201	0.00980	101	0.00572	189.00
251	0.00971	102	0.00688	154.60
301	0.01042	95	0.00764	138.60

2-punts crossover met kans: 0.7
gewone mutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00112	891	0.00100	1000.10
51	0.00417	239	0.00300	344.10
101	0.00493	202	0.00389	266.50
151	0.00629	158	0.00458	225.60
201	0.00654	152	0.00535	189.80
251	0.00926	107	0.00628	166.00
301	0.01190	83	0.00717	147.00

uniforme crossover met kans: 0.7
en met een PChange van: 0.5
gewone mutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00149	672	0.00113	899.40
51	0.00474	210	0.00370	278.80
101	0.00613	162	0.00469	220.40
151	0.00813	122	0.00570	180.90
201	0.00800	124	0.00649	158.30
251	0.01370	72	0.00812	128.50
301	0.01587	62	0.00909	116.60

1-punts crossover met kans: 0.7
gewone mutatie met kans : 0.003
wisselmutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00129	776	0.00107	949.50
51	0.00360	277	0.00250	426.70
101	0.00403	247	0.00329	314.20
151	0.00483	206	0.00378	269.00
201	0.00654	152	0.00455	226.70
251	0.00763	130	0.00531	192.70
301	0.00971	102	0.00610	171.30

1-punts crossover met kans: 0.7
gewone mutatie met kans : 0.003
dagwisselmutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00139	719	0.00110	935.90
51	0.00313	319	0.00254	399.30
101	0.00448	222	0.00330	312.80
151	0.00578	172	0.00427	238.60
201	0.00746	133	0.00552	184.70
251	0.01266	78	0.00673	159.30
301	0.01389	71	0.00834	129.20

1-punts crossover met kans: 0.7
gewone mutatie met kans : 0.003
inversiemutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00127	787	0.00096	1060.90
51	0.00303	329	0.00239	421.30
101	0.00333	299	0.00288	351.20
151	0.00397	251	0.00327	309.30
201	0.00490	203	0.00360	285.70
251	0.00556	179	0.00413	252.40
301	0.00800	124	0.00479	229.00

2-punts crossover met kans: 0.7
gewone mutatie met kans : 0.003
wisselmutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00142	704	0.00107	954.40
51	0.00376	265	0.00289	356.50
101	0.00463	215	0.00356	284.10
151	0.00613	162	0.00414	246.70
201	0.00685	145	0.00460	226.20
251	0.00806	123	0.00507	205.50
301	0.01042	95	0.00573	184.80

2-punts crossover met kans: 0.7
gewone mutatie met kans : 0.003
dagwisselmutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00167	598	0.00108	975.70
51	0.00373	267	0.00310	330.70
101	0.00690	144	0.00423	249.60
151	0.00833	119	0.00536	201.30
201	0.00917	108	0.00636	163.40
251	0.01389	71	0.00752	145.50
301	0.01449	68	0.00859	124.80

2-punts crossover met kans: 0.7
gewone mutatie met kans : 0.003
inversiemutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00155	644	0.00116	903.70
51	0.00450	221	0.00281	372.30
101	0.00488	204	0.00363	285.00
151	0.00568	175	0.00434	239.80
201	0.00690	144	0.00510	206.30
251	0.00862	115	0.00550	196.50
301	0.00901	110	0.00589	181.40

uniforme crossover met kans: 0.7
en met een PChange van: 0.5
gewone mutatie met kans : 0.003
wisselmutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00139	717	0.00106	958.70
51	0.00463	215	0.00352	298.30
101	0.00704	141	0.00485	214.80
151	0.00962	103	0.00600	176.50
201	0.01042	95	0.00688	155.20
251	0.01111	89	0.00738	143.10
301	0.01111	89	0.00801	132.50

uniforme crossover met kans: 0.7
en met een PChange van: 0.5
gewone mutatie met kans : 0.003
dagwisselmutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00130	770	0.00108	934.60
51	0.00658	151	0.00375	290.50
101	0.01099	90	0.00523	213.90
151	0.01639	60	0.00715	166.60
201	0.01695	58	0.00815	143.40
251	0.01724	57	0.00899	126.40
301	0.02381	41	0.01122	105.60

uniforme crossover met kans: 0.7
en met een PChange van: 0.5
gewone mutatie met kans : 0.003
inversiemutatie met kans : 0.003

Nr	Max	Boete	Gem. Max	Gem. Boete
1	0.00126	791	0.00107	943.40
51	0.00372	268	0.00318	321.30
101	0.00518	192	0.00391	262.90
151	0.00559	178	0.00440	233.70
201	0.00685	145	0.00515	202.00
251	0.00806	123	0.00560	186.70
301	0.00806	123	0.00598	175.10

Appendix E: een instantie voor het TISP-probleem

Appendix E geeft een overzicht van de gebruikte data bij het TISP-probleem. De eerste tabel geeft voor elke docent aan wat zijn taakomvang is (waarbij 1.000 correspondeert met een volledige baan) en voor welke timeslots hij/zij niet beschikbaar is voor surveillance.

doc	omvang taak	afwezig op timeslot(s)	doc	omvang taak	afwezig op timeslot(s)
D01	1.000		D41	1.000	1 8 14 19 23 36
D02	1.000	3 27 30 40	D42	1.000	19 34 40
D03	1.000	15 17	D43	1.000	17 19 31 37
D04	1.000	23 28 42 45	D44	1.000	4 17 29 45
D05	0.822	22	D45	1.000	1 7 29 44
D06	1.000	40	D46	0.307	1 4 16 24 30 41
D07	1.000	8 14 23 35 36 39 40	D47	1.000	37
D08	1.000		D48	1.000	
D09	1.000	3 4 9 18 28 44	D49	1.000	2 8 24 27
D10	1.000	16 30	D50	0.586	13 20 37
D11	1.000	24	D51	1.000	3 11 41
D12	1.000	16 18 39 41 42 45	D52	1.000	15 29 36 38 39
D13	1.000	17 21 25 27 37	D53	1.000	22 23
D14	1.000	9 13 15 19 25 32	D54	1.000	2 10 14 22 23 44
D15	1.000		D55	1.000	
D16	1.000	6 13 25 27 33 37	D56	1.000	
D17	1.000	3 41 42	D57	1.000	20 23
D18	1.000	13 25 27 36 41 42	D58	1.000	30 34
D19	0.689	7 19 28	D59	1.000	
D20	1.000	16 22 24	D60	0.550	
D21	0.560	1 33	D61	0.399	27 29 43
D22	1.000	22	D62	1.000	4 9 10 16 17
D23	1.000	7 12 13 32 34 44	D63	0.226	4 11 18 26 31 45
D24	0.929	4 20 24 26 43 45	D64	1.000	2 16 21 28 30 34 39
D25	1.000	11 15 16 25 31	D65	1.000	3 44
D26	1.000	14 19 32	D66	1.000	16 22 23
D27	1.000	20	D67	0.732	5 9 14 24 28
D28	1.000		D68	1.000	11 12 15 30 35 40
D29	0.183	1 14 19 21 25 34 37	D69	1.000	12
D30	1.000		D70	1.000	1 10 21 41
D31	1.000		D71	0.916	6 14
D32	1.000		D72	1.000	11 28 32 45
D33	1.000	16 18 22 23 39	D73	1.000	9 20
D34	0.595	40	D74	1.000	30
D35	1.000	2 25 28 30 32	D75	1.000	1 32 37 41
D36	1.000	6 16 25 29 33 41	D76	1.000	5 25 31 32 40
D37	1.000	11 14 25 35 45	D77	1.000	
D38	0.170	5 8 19 22	D78	1.000	8 22 26 27 39 42 43
D39	1.000	4 20 37 42	D79	1.000	9 13 21 24 25 26 31
D40	1.000	16 17 36	D80	0.803	8 25 31 32

Uit de tweede tabel kan worden afgelezen welke docent(en) een bepaalde module of vak geeft.

module	vakdocent
AEC001	D01 D04
AEC002	D01
AEC003	D02
AEC004	D05
AEC005	D05
AEC006	D02
AEC007	D02 D03 D04 D06
AEC008	D01 D03 D06
AEC009	D01 D03 D06
AEC010	D01 D05 D06
AEC011	D03
AEC012	D03
BCA001	D07 D09 D11
BCA002	D07 D08 D10
BCA003	D09 D10
BCA004	D07 D12
BCA005	D07 D09
BCA006	D11
BCA007	D08 D09 D10
BCA008	D07 D09 D10 D11 D12
BCA009	D07 D08 D09 D10
BEC001	D13 D14 D15 D18
BEC002	D14 D17
BEC003	D15
BEC004	D13 D15
BEC005	D13 D14 D17
BEC006	D16 D17 D18
BEC007	D13 D14 D15 D16
BEC008	D18
BEC009	D13 D14
BEC010	D13 D15 D16 D18
BEC011	D13 D14 D16 D17
CEC001	D19 D20 D21
CEC002	D19 D21 D22
CEC003	D19 D21 D23
CEC004	D24
CEC005	D22 D23
CEC006	D20 D22 D23
CEC007	D19 D23 D24
CEC008	D19 D21 D23 D24
CEC009	D19 D21 D23
CEC010	D19 D20 D21 D23 D24

module	vakdocent
DUI001	D25 D27 D30
DUI002	D27 D29 D30
DUI003	D25 D26 D27 D30
DUI004	D25 D26 D29
DUI005	D25 D27 D28 D29
DUI006	D28 D29
DUI007	D25 D26 D28 D29
DUI008	D25 D27 D28 D29
DUI009	D28 D29 D30
DUI010	D25 D26 D27 D30
DUI011	D25 D29
DUI012	D25
ENG001	D31 D32 D35
ENG002	D31 D32 D35
ENG003	D31 D32 D34 D35
ENG004	D32 D33 D35 D36
ENG005	D31 D33 D36
ENG006	D31 D36
ENG007	D35
ENG008	D33 D34 D36
ENG009	D33
FRA001	D39
FRA002	D40 D41 D42
FRA003	D42
FRA004	D37 D38 D39 D42
FRA005	D37 D38 D40 D41 D42
FRA006	D40 D41 D42
FRA007	D38 D42
FRA008	D37 D38 D39
INF001	D43 D45 D46 D47 D48
INF002	D44
INF003	D43 D44 D46 D48
INF004	D43 D46
INF005	D43 D44 D45 D46 D48
INF006	D46 D47
INF007	D45
INF008	D44 D45 D47
MAN001	D52 D53 D54
MAN002	D49 D52 D53
MAN003	D52
MAN004	D52
MAN005	D49 D52 D53

module	vakdocent
MAN006	D50 D51 D52 D53
MAN007	D51 D54
MAN008	D50 D51 D53
MAN009	D49 D50 D52 D53 D54
MAN010	D54
REC001	D55 D57 D60
REC002	D55 D60
REC003	D58 D59 D60
REC004	D58
REC005	D55 D57 D60
REC006	D55 D58 D59 D60
REC007	D56 D58
REC008	D60
SPA001	D62
SPA002	D61 D62 D64 D65

module	vakdocent
SPA003	D62 D63
SPA004	D63 D65
SPA005	D62 D64 D66
SPA006	D63 D64 D66
SPA007	D62 D64 D65 D66
SPA008	D61 D63
SPA009	D61 D62 D66
WST001	D68
WST002	D69 D70
WST003	D68 D70
WST004	D67 D68
WST005	D67 D70 D71 D72
WST006	D67 D68 D72
WST007	D67 D68 D69 D71
WST008	D68 D70

De laatste tabel geeft per timeslot de te examineren modules en het aantal benodigde surveillerende docenten.

timeslot	aantal docenten nodig	modules
1	10	AEC001 BEC003
2	17	DUI005 FRA007
3	15	REC004 AEC002 BEC004
4	15	DUI006 FRA008 REC005
5	20	AEC003 BEC005
6	20	DUI007 INF001 REC006
7	27	AEC004 BEC006
8	5	DUI008 INF002 REC007
9	21	AEC005 BEC007
10	5	DUI009 INF003 REC008
11	9	AEC006 BEC008 DUI010
12	29	INF004 SPA001 AEC007
13	29	BEC009 DUI011 INF005
14	15	SPA002 AEC008 BEC010
15	20	DUI012 INF006
16	29	SPA003 AEC009
17	9	BEC011 ENG001 INF007
18	6	SPA004 AEC010 CEC001
19	28	ENG002 INF008
20	9	SPA005 AEC011 CEC002
21	25	ENG003 MAN001 SPA006
22	27	AEC012 CEC003 ENG004
23	25	MAN002 SPA007 BCA001
24	25	CEC004 ENG005 MAN003
25	6	SPA008 BCA002 CEC005
26	15	ENG006 MAN004
27	23	SPA009 BCA003
28	21	CEC006 ENG007 MAN005
29	23	WST001 BCA004 CEC007
30	28	ENG008 MAN006 WST002
31	17	BCA005 CEC008 ENG009
32	28	MAN007 WST003
33	13	BCA006 CEC009
34	23	FRA001 MAN008 WST004
35	6	BCA007 CEC010
36	25	FRA002 MAN009
37	8	WST005 BCA008
38	13	DUI001 FRA003
39	16	MAN010 WST006 BCA009
40	14	DUI002 FRA004 REC001
41	8	WST007 BEC001
42	9	DUI003 FRA005
43	27	REC002 WST008
44	25	BEC002 DUI004
45	19	FRA006 REC003

Appendix F: resultaten van een directe implementatie

Appendix F geeft een overzicht van de resultaten van een in paragraaf 3.3.3 beschreven algoritme. De volgorde van de dagen werd steeds aselect gekozen. Geen enkele docent kan te vaak surveilleren. Daarom kan V-B alleen positief zijn en geeft het totaal aantal beurten die sommige docenten te weinig surveilleren voor wat betreft hun taakomvang. Verder geeft 1-B het aantal 1-beurten aan en T-B het aantal tussenbeurten.

1-beurt ≡ 2 tussenbeurten			
run	**1-B**	**T-B**	**V-B**
1	205	64	1
2	201	60	3
3	206	57	1
4	209	57	1
5	202	58	5
6	203	59	2
7	198	57	2
8	202	57	5
9	198	57	3
10	199	58	2
11	205	64	2
12	198	57	3
13	198	57	4
14	202	59	4
15	210	59	3
16	206	59	2
17	201	59	2
18	198	57	4
19	198	57	4
20	216	61	3
21	198	57	1
22	196	55	3
23	198	57	2
24	200	57	4
25	195	55	0
26	204	57	4
27	199	58	1
28	198	57	5
29	199	58	4
30	205	64	2

1-beurt ≡ 0 tussenbeurten			
run	**1-B**	**T-B**	**V-B**
1	33	108	2
2	35	108	3
3	32	105	1
4	33	106	0
5	33	106	1
6	35	106	0
7	32	107	2
8	34	102	1
9	34	106	1
10	38	103	1
11	32	106	4
12	37	102	2
13	36	106	1
14	32	106	3
15	33	108	2
16	33	108	1
17	42	106	0
18	34	107	0
19	35	106	0
20	32	107	3
21	34	102	1
22	36	110	2
23	34	106	1
24	37	103	0
25	34	105	1
26	38	104	3
27	32	106	1
28	35	99	1
29	35	102	3
30	37	103	0

De theoretische minima voor het aantal 1-beurten bedragen 195 respectievelijk 32. Alle runs geven een aantal 1-beurten te zien die nooit ver boven de theoretische minima uitstijgen. Daar V-B altijd klein is zijn bovendien de beurten

ook redelijk eerlijk verdeeld. Het beste resultaat geeft run 25 in de eerste kolom: alle beurten zijn eerlijk verdeeld, zonder dat daarvoor extra 1-beurten moesten worden uitgedeeld. Dit resultaat is daarom optimaal.