**KAARTKLEUREN: Een minimaal aantal kleuren**

Casimir Boot

Universiteit van Amsterdam

emailadres@email.uva.nl

Lonneke Lammers

Universiteit van Amsterdam

emailadres@email.uva.nl

Alwin Lijdsman

Universiteit van Amsterdam

alwinlijdsman@outlook.com

**1. Inleiding**

Het minimaliseren van het aantal kleuren wanneer een kaart wordt ingekleurd is een bekend probleem. In 1852 vermoedde Guthrie dat, wanneer elk land wordt ingekleurd waarbij aangrenzende landen niet dezelfde kleur mogen hebben, er maximaal vier kleuren nodig zijn [2]. Dit wordt het Guthrie’s Problem genoemd [2].

Het eerste wetenschappelijke artikel over dit probleem werd geschreven in 1878. In 1976 werd Gurthrie’s probleem opgelost door Kenneth Appel en Wolfgang Haken [1], hiermee werdt de Four Color Theorem ontwikkeld [1]. Het was de eerste grote stelling die bewezen werd met behulp van een computer[1].

In dit verslag is de Four Color Theorem, getoetst door drie verschillende kaarten, een kaart van Pennsylvania, een kaart van een Indiasche provincie en een kaart van een sociaal netwerk met zo min mogelijk kleuren in te kleuren. Daarnaast zijn de twee heurisiteken die gebruikt zijn om de Four Color Theorem te bewijzen met elkaar vergelijkt om te onderzoeken welke heuristiek het beste werkt.

De bruto toestandsruimte grote in dit verslag wordt geschat op , waar k gelijk is aan het aantal kleuren en n gelijk is aan het aantal landen op de kaart. Als ieder land op de kaart een andere kleur toegewezen krijgt, is er altijd een oplossing. De restrictie op de transitie is, zoals eerder gesteld, dat aangrenzende landen nooit dezelfde kleur mogen kennen en dat het minimaal aantal kleuren moet worden gebruikt.

**2. Methodes**

Om het minimaal aantal kleuren voor het kleuren van een kaart te benaderen is er in dit verslag gewerkt met een random sampling algoritme en een zelf ontwikkeld gericht algoritme.

**2.1** **Random sampling algoritme**

herhaalbaar

**2.2 Gericht algoritme: land met meeste buurlanden eerst kleuren**

**3. Resultaten**

In excel histogram nog maken

*Hier bespreek je heel droog je resultaten. Als je statistieken hebt: toevoegen. Als je vergelijkingen hebt met randomposities: toevoegen. Alles is woord en getal, alle details en het liefst ook het één en ander in grafieken, plaatjes of anderzins.*

Fig. 2. Aantal misfits per algoritme, gemiddeld over 5 trials. Ondanks dat alle methodes de puzzel relatief snel oplosten, is het BIDIBENCH algoritme zowel qua eindresultaat als qua algehele trajectperfomance te prefereren over beide andere methodes. *Een goede grafiek is, net als een goed plaatje, in één oogopslag duidelijk en vereist weinig arbeid van de lezer. Voorzie de assen van labels, de grafiek van een titel en kies je kleuren zo dat ze maximaal contrasteren en dus gemakkelijk leesbaar zijn. De legenda rechts van deze grafiek had rechtsbovenin het plaatje gemogen, dat had ruimte bespaard, had de grafiek iets groter kunnen zijn en dus beter leesbaar.*

**4. Conclusies**

*Hier schrijf je je conclusies, eventuele overdenkingen (hoe zou het nog beter kunnen, is het algoritme ook in andere gebieden toepasbaar).*

**5. Referenties**

*Als je literatuur hebt gebruikt, hier toevoegen. Als je eraan refereert in de tekst, zet je op die plek alleen [1], zodat mensen achterin de details kunnen vinden. Als je geen literatuur gebruikt, weglaten.*

*Wat ook nog kan is een dankwoord, bijvoorbeeld voor mensen die wel geholpen hebben maar geen auteur zijn, mensen die je een inzicht hebben gegeven, of administrators die je even hun supercomputer hebben laten gebruiken. Altijd naam en bedrijf noemen en zorgen dat de bedankte persoon zich er goed over voelt.*

*Als je zowel een dankwoord als een referentiesectie hebt: de referentiesectie is \*altijd\* het laatste onderdeel van je verslag.*

[1] Artificial Intelligence, a modern approach, Russel & Norvig,3rd Edition, Addison-Wesly, pg 287-387.

[2] “A packing problem with applications to lettering of maps” Michael Formann and Frank Wagner (1991) SCG '91 Proceedings of the seventh annual symposium on Computational geometry

1. *http://mathworld.wolfram.com/Four-ColorTheorem.html*
2. [*http://mathworld.wolfram.com/GuthriesProblem.html*](http://mathworld.wolfram.com/GuthriesProblem.html)

***DONE maar goed om te checken:***

1. ***Inleiding***

*Hier schrijf je een inleiding die in elk geval kort, bondig en compleet de hele vraagstelling bevat. Ook moet er een inschatting komen van de toestandsruimtegrootte (belangrijk, daar deden we het voor), en eventuele restricties op transities in beschreven worden, maar nog niet de methodes die je gebruikt om de toestandsruimte te doorzoeken.*

Eternity II is een edge-matching puzzle uitgegeven door Tomy in november 2012. Het doel is bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla $1.000.000 te winnen.

Fig. 1: een kleine-schaal versie van Eternity II met centrumstukken, randstukken en hoekstukken. Stukken mogen 90o geroteerd worden in beide reichtingen, en twee aangrenzende stukjes ‘passen’ als de rakende kanten hetzelfde symbool hebben. De puzzel is opgelost als alle stukjes passen.

*Figuren moeten altijd genummerd zijn, en \_eigenlijk\_ moet er ook altijd naar verwezen worden in de tekst. Een goede figuur maken is een kunst. Het kan enorm bijdragen aan de toegankelijkheid van je artikel, maar een slechte figuur doet afbreuk. Iedere figuur heeft een bijschrift (of ‘caption”). Bijschriften zijn zo kort mogelijk, maar niet korter dan dat (moeilijke nuance). Als richtlijn kun je voor een eenvoudige figuur als deze met twee tot drie korte duidelijke zinnen klaar zijn.*

Ieder stukje kan op 4 manieren geplaatst worden bla bla bla bla bla bla bla bla bla bla bla. De bruto toestandsruimte van Eternity bevat daarom 81226 \* 1027 elementen. Stukjes 13 en 71 zijn identiek, net als bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. De netto toestandsruimte bevat daarom 2341 \* 1023 elementen, dit aantal is te groot om met een exhaustive method binnen redelijke tijd tot een goed einde te komen.