TI2736-A Assignment 3: Path finding & TSP

David Akkerman - 4220390
Jan Pieter Waagmeester - 1222848

29 oktober 2014

3.1: Path finding through ACO

1. - Lange doodlopende straten
   - Straten met heel veel korte doodlopende zijstraten
   - Grote open gebieden

2. We, of de mieren, laten feromoon vallen om de voordelige paden aan te duiden. Ze laten het feromoon vallen 'op de trugweg', afhankelijk van hoe lang het pad was. Er is een waarde Q die ongeveer de geschatte routelengte is.

$$\Delta\tau^k = Q \cdot \frac{1}{L^k}$$

   In deze formule staat $\Delta\tau^k$ voor de toe te voegen feromoon afkomstig van mier k, L^k de lengte van het pad. Als een pad korter is zullen de afzonderlijke verbindingen van dat pad meer worden versterkt dan wanneer het pad langer was.

3. Het verdampen van de feromoon is zinnig om niet in een lokaal optimum te blijven hangen. Doordat de waarde in elke cel met een bepaalde verdampingsfactor (ρ) afneemt is er kans dat een andere (wellicht betere) route wordt gekozen.

$$\tau_{ij} = (1-\rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k$$

4. Om te beginnen moeten we wat dingen initializeren: een stel mieren en een feromoon-matrix. Vervolgens laten we elke mier een pad zoeken in een matrix waarin de feromoon-waarden in elke cel gelijk zijn.

   Nadat alle mieren het einde gevonden hebben (of getermineerd zijn omdat ze er veel te lang over doen), kiezen we de beste mier uit. We laten een bebaald deel $(1-\alpha) \cdot \tau_{xy}$ verdampen en tellen bij de cellen waar ons beste miertje geweest is een waarde $Q/|\text{ant}|$ op.

   Daarna gaat het weer opnieuw tot we het aantal iteraties hebben gehad.

Er moet in ieder geval een vergelijking komen. Wellicht ook met nog wat uitleg en hoe het geoptimaliseerd wordt

```
1  ants = [ant0, ant1, ... antN]
2  pheromone[height][width] = 0.1 # initialize with small value
3
4  for i = 0 ... iterations
5      foreach ant in ants
6          ant.find_naive_solution(halt_after_steps=10000)
7
8      best_ant = select_best(ants)
```

```
9
10      pheromone = (1 - alpha) * pheromone
11
12      for p in each best_ant.route
13          pheromone (p) += Q / length(best-ant)
```

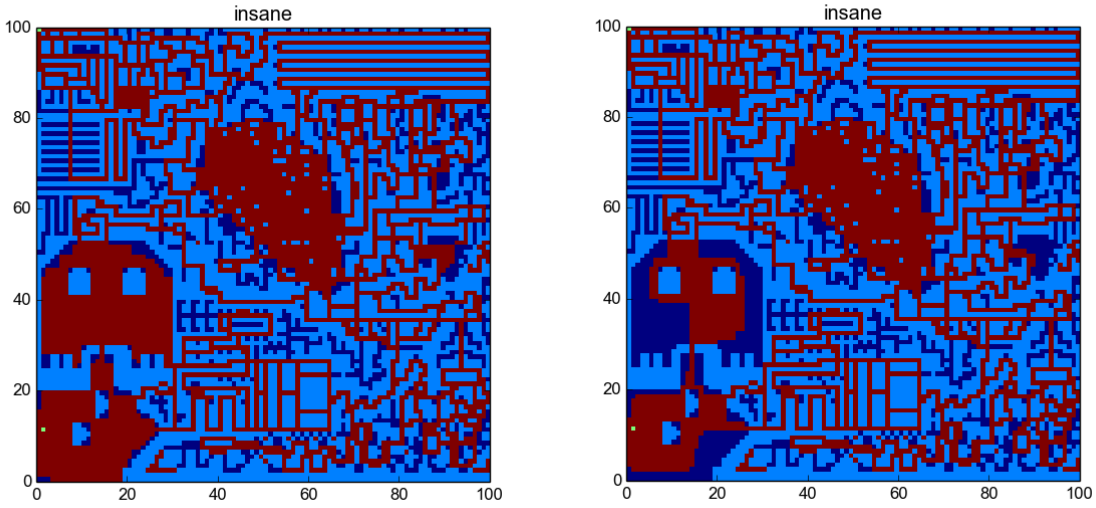
Ze vragen echt om een ant algorithm, maar hier wordt die als 'vanzelfsprekend' beschouwd en wordt de feromooncode geïllustreerd. ¡-

5. We hebben een aantal versnellingsmanieren bedacht en ook bijna allemaal geïmplementeerd. **Doolhofeliminatie**

   Doodlopende gangen Als een mier vanaf een bepaalde positie nog maar één kant op kan (terug), zit hij aan het einde van een doodlopende gang. We markeren die plek in de maze dan als muur, zodat er in het vervolg geen mier meer probeert in te lopen.

   Open ruimtes Hoekjes in een open ruimte kunnen worden weggestreept, als er iets bekend is over de aangrenzende cellen. Als een mier achtereenvolgens een hoekje en daarna twee cellen met voldoende ruimte tegenkomt waarbij de laatste op de diagonaal zit van het hoekje, kan het hoekje worden afgestreept.

   Merk op dat dit bij ons soms mis gaat doordat de mieren afzonderlijk te werk gaan in eigen threads. Het kan dus zijn dat twee of meer mieren er samen voor zorgen dat er een cruciale gang wordt afgesloten. Dit is wel op te lossen, maar gaat voorbij aan deze opdracht.



Figuur 1: Voorbeeld van eliminatie van doodlopende stukken. Links na de 1^e iteratie, rechts na de 30^e. Donkerblauwe vlakken zijn door de mieren 'uitgezet'.

Probeer min mogelijk over eigen pad te lopen
Het heeft niet veel zin om terug te keren waar we al geweest zijn. We geven daarom bij het kiezen van de volgende stap de voorkeur aan de posities waar we nog niet geweest zijn.

Alleen als het niet anders kan keren we weer terug op of over onze route.

Gebruik het pad van de beste mier over alle iteraties ook bij de feromoon-update
Door de beste mier in de run van het algoritme tot nu toe te onthouden en elke keer ook te gebruiken voor de update.

Optimalisatie van het gevonden pad
Vaak is een door een mier gevonden pad niet bijzonder efficient, zeker aan het begin. We kunnen zorgen dat we niet op te veel plekken feromoon neerleggen door voor dat te doen het pad van elke mier nog te optimaliseren:

- Als de mier langs het einde loopt, direct naar het einde lopen en de rest van de route overslaan.
- Als de mier weer langs start loopt het het eerste stuk overslaan.
- Als een mier een rondje loopt dat stuk overslaan.

Andere, niet-geimplementeerde ideeën
Naast de voorgaande ideeën hebben we nog meer ideeën, waar voor de implementatie helaas de tijd ontbrak:

- Alle mogelijke stappen hebben lengte 1, omdat we in een grid werken. We zouden dit kunnen transformeren naar een graaf waarbij we de kruispunten zien als nodes. Dat scheelt een hele hoop mogelijke stappen, en er onstaat variatie in lengte.

6. We hebben het algoritme laten lopen voor een aantal verschillende waarden van de parameters Q, `ant_count`, α and `optimize_ants`. Deze waarden hebben we 20 keer gevarieeerd voor elke doolhof terwijl de andere op een guestimated waarde stonden. Bij elke set van runs meten we drie dingen: run time, in welke iteratie de beste route voor het eerst voorkomt en de lengte van de beste route. Het was even rekenen (1000x easy maze, 760x medium maze, x hard maze), maar levert een hoop ($4 \cdot 3 \cdot 3$) zinnige plaatjes op.

   hard maze

   Easy maze Op grond van deze plaatjes maakt het niet zo veel uit wat we voor parameters gebruiken. Sowieso maakt het allemaal niet zoveel uit, want deze maze is zo klein dat de kortste route altijd wel vrij snel gevonden wordt.

   Medium maze

   Hard maze

   Insane maze

   Dat brengt ons uiteindelijk bij de volgende waarden:

	easy	medium	hard	insane
# iterations				
Q	38			
`ant_count`	50			
α				
`optimize_ants`	`True`			
Kortste gevonden route	38	141		

7.

medium en hard maze invoegen

Plaatjes maken voor insane maze en eventueel invoegen?

3.2: The Traveling Robot Problem

8. Het TSP (Reizende Verkoper Probleem) gaat over een verkoper die een aantal, bijvoorbeeld 20, steden moet bezoeken. Natuurlijk vindt hij het prettig om dit in een zo efficiënt mogelijke volgorde te doen. Het probleem is dat dit erg moeilijk is om uit te rekenen, want het aantal mogelijke volgordes is $20! = 2.4 \cdot 10^{18}$, wat hij logischerwijs niet allemaal door kan rekenen.

9. Het verschil tussen een gebruikelijk TSP en het Reizende Robot Probleem is dat de afstanden tussen de steden van het TSP bekend en duidelijk zijn, terwijl het zich voor ons probleem in een doolhof afspeelt, waarbij verschillende routes tussen twee punten genomen kunnen worden, die niet allemaal zo makkelijk te berekenen zijn.

10. CI-technieken zijn erg handig om dergelijke problemen op te lossen, doordat ze de rekenkracht van een computer gebruiken, maar op een slimme manier, waardoor het optimumveel sneller gevonden kan worden dan met enkel brute rekenkracht. Door de complexiteit van het doolhof zou het een mens veel langer kosten dan een computer om een goede route te vinden.

11. De chromosomen bestaan uit de nummers 1 tot en met 18 en de volgorde waarop deze voorkomen in een chromosoom representeert de volgorde waarop de items opgepakt moeten worden door de robot.

12. De fitness functie van het TSP is één gedeeld door de totale afstand die de robot van begin tot eind aflegt als hij alle producten oppakt.

13. De ouders worden geselecteerd met een kans die evenredig is met de fitness. Alle oorspronkelijke chromosomen worden gesorteerd op fitness. Vervolgens wordt er een normale kansverdeling gebruikt met verwachtingswaarde 0 en standaardafwijking 75. De absolute waarde hiervan geeft aan welke chromosoom uit de gesorteerde reeks genomen zal worden.

14. De nieuwe chromosomen worden gecreëerd door zowel cross-over als mutatie toe te passen. Cross-over houdt in dat van een ouderchromosoom een aantal genen gekopiëerd wordt, wat vervolgens de basis is van het kindchromosoom. De missende genen worden aangevuld door deze bij het kind in te plakken in de volgorde waarop ze voorkomen bij een tweede ouderchromosoom.
Mutatie betekent dat van een ouderchromosoom een willekeurig aantal genen geknipt, omgedraaid en teruggeplakt wordt. Dit wordt gedaan omdat op deze manier stukjes van dicht bij elkaar liggende items niet uit elkaar worden gescheurd, maar alleen in de omgekeerde volgorde bezocht worden, wat kan zorgen voor een verbetering.

15. De oorspronkelijke chromosomen worden zó gecreëerd dat ze precies alle items eenmaal bevatten. De manieren waarop gereproduceerd wordt, zijn zo ontwikkeld dat het niet kan voorkomen dat een kindchromosoom wel tweemaal hetzelfde getal bevat en een ander niet. Bij het mutatie- en cross-overproces wordt namelijk enkel de volgorde van de items veranderd, niet de items zelf.

16. Doordat er een normale verdeling gebruikt wordt om de ouders te selecteren, bestaat er altijd een kans dat er een slechter presterend chromosoom gepakt wordt. Dit chromosoom kan in een enkel geval wel betere kinderen geven, waardoor er voorbij een lokaal minimum gegaan kan worden. Om de optimale oplossing te vinden, moet echter een oneindig aantal iteraties plaatsvinden, want je weet niet wanneer de optimale oplossing gevonden gaat worden, maar enkel dát hij ooit gevonden wordt.

17. Elitisme houdt in dat een aantal van de fitste chromosomen niet sterven maar ook weer meedoen in de volgende generatie. Op deze manier voorkom je dat er alleen maar minder fitte chromosomen komen. Een gevaar is echter dat er op deze manier te vroeg een lokaal minimum gevonden wordt en niet verder wordt gezocht. Wij hebben elitisme gebruikt in zoverre dat het mogelijk is dat er bij mutatie slechts een rij van één getal wort geïnverteerd, wat geen verandering in het chromosoom aanbrengt, waardoor de ouder dus ongeschonden naar de volgende generatie doorgaat.

Als je de hele afstandstabel hebt heb je toch in principe hetzelfde als classic TSP?

Herschrijven als: '... op te lossen, doordat ze de rekenkracht van een computer op een slimme manier gebruiken

Je kan niet garanderen dat je het optimum daadwerkijk vindt toch?

Er staat niet letterlijk wat een 'gene' is.

Formule erbij doen?

Hoe kom je op 75? (Volgens mij was je al van plan daar wat over te gaan schrijven)

Doe je altijd een vast aantal, of gebruik je een be-