Algoritmes

Het eerste algoritme is het **Nearest Neighbour** algoritme die aan de hand van de volgende constraints een lijnvoering maakt:

- Startpunt = uithoeken.
- Als alle uithoeken geweest zijn: Kies de eerste station die nog niet geweest is in de lijst.
- Alle stations geweest? Kies eerste station wiens verbinding nog niet geweest is.
- Ga dan naar Nearest Neighbour.
- Als station al genomen is: Kies spoor dat nog niet gereden is, meerdere sporen niet gereden? Kies nearest Neighbour.
- Alle sporen bereden? Kies nearest Neighbour.
- Als alleen terug weg mogelijk is? Kies terug weg. (terug weg is laatste keus).
- Traject stopt bij 120 minuten.
- Als alle sporen & alle stations al zijn geweest: break.
- Pas scorefunctie toe, als het niet nodig was een extra traject te maken, maar het wel gebeurd is scorefunctie: trajecten(t) -1.

Het tweede algoritme dat wordt toegepast is een **'hill climber'**: het verschil met het Nearest Neighbour algoritme is het start punt:

- Eerst kiest het algoritme de uithoeken.
- Als alle uithoeken gekozen zijn kies per te maken traject: een random station.
- Maak traject a.d.h.v. constraints van algoritme 1.
- Pas scorefunctie toe, als het niet nodig was een extra traject te maken, maar het wel gebeurd is scorefunctie: trajecten(t) -1.
- Bewaar de betere oplossing.
- Itereer x aantal keer.

Het derde algoritme dat wordt toegepast is **'simulated annealing'**: het verschil met het Hill Climber algoritme is dat niet altijd de betere oplossing bewaard wordt:

- Eerst kiest het algoritme de uithoeken.
- Als alle uithoeken gekozen zijn kies per te maken traject: een random station.
- Maak traject a.d.h.v. constraints van algoritme 1.
- Pas scorefunctie toe, als het niet nodig was een extra traject te maken, maar het wel gebeurd is scorefunctie: trajecten(t) -1.

- Formuleer een kans functie P(t). Deze functie schaalt tussen 0 en 1 en is afhankelijk van één variabele: temperatuur.
- Formuleer een juiste waarde voor de temperatuur. De temperatuur neemt af per iteratie.
- De kans om een betere oplossing te bewaren = P(T) x 100%. Als T hoog is , is P(T) laag. Dus: Als er nog relatief weinig iteraties zijn geweest, willen we liever een slechtere oplossing bewaren.
- Itereer x aantal keer.

Verklaring nearest neighbour

De reden waarom wij gekozen hebben voor het nearest neighbour algoritme is dat dit algoritme zeer goed tot de verbeelding spreekt. Het was een efficiënte manier om op spelende wijze kennis te maken met de case. Uiteindelijk bleek uit experimentatie dat het algoritme erg goed scoorde, vandaar dat we het algoritme hebben gehouden in onze top 3.

Verklaring hill climber

Het hill climber algoritme kan binnen korte tijd een goede score opleveren voor de scorefunctie van het op te lossen probleem, en niet moeilijk te programmeren is. Een nadeel van het algoritme is dat er geen zekerheid is voor het vinden van een maximum van de scorefunctie. In tegendeel, de hill climber staat bekend om het bereiken van een lokaal maximum van de scorefunctie (Russell et al, *2003)*. Dit hill climber algoritme hebben wij gekozen omdat we hiermee onze nearest neighbour kunnen optimaliseren. We zullen niet de beste oplossing hiermee vinden, maar we zullen zeker een verbetering van het nearest neighbour algoritme vinden. Bovendien is de stap van een hill climber naar een simulated annealing algoritme relatief makkelijk gezet, lees verder voor meer informatie daarover.

Verklaring simulated annealing

Het simulated annealing algoritme staat erom bekend dat het in staat is om het maximum van een scorefunctie te vinden. Waar de hill climber blijft haken op een lokaal maximum, kan een goed geïmplementeerd simulated annealing algoritme gegarandeerd de beste oplossing vinden (Cerny, 1985). We hadden er ook voor kunnen kiezen om een ander algoritme te gebruiken dat ook de beste oplossing zou kunnen vinden, maar omdat we al een werkende hill climber hadden was dit de makkelijkste keuze.

Via de 3 bovenstaande algoritmes konden we steeds stapsgewijs onze scorefunctie verbeteren, wat uiteindelijk mogelijkheid biedt tot het vinden van de beste oplossing van ons probleem; de oplossing met de hoogste score op de scorefunctie.

Citaties:

- *Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, pp. 111–114, ISBN 0-13-790395-2*
- Černý, V. (1985). "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm". *Journal of Optimization Theory and Applications*. **45**: 41–51. doi:10.1007/BF00940812