RailNL

Amber, Julia & Josje

The Pythons

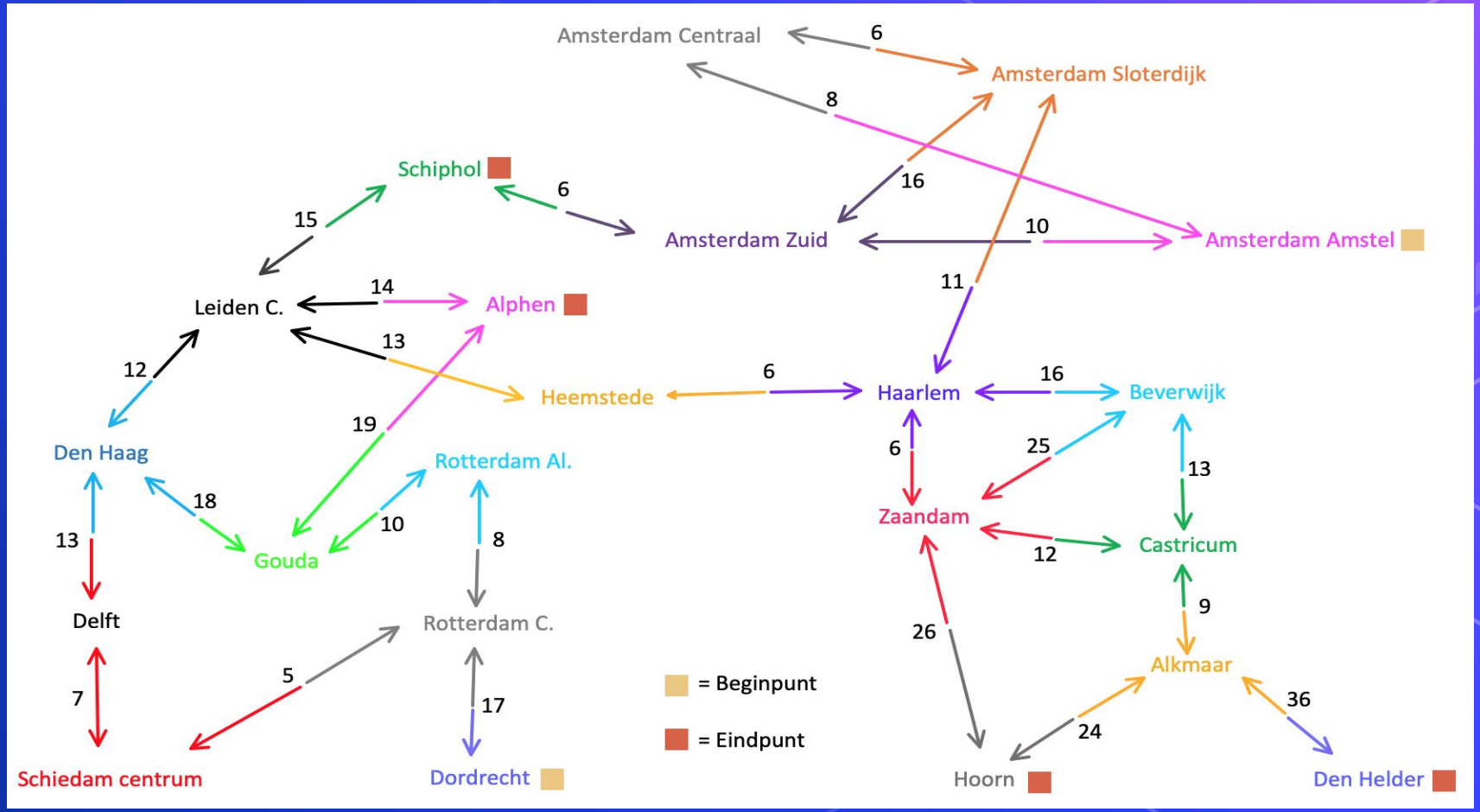
Opdrachtbeschrijving

- Lijnvoering van intercity treinen

  → Wat zijn de trajecten waarover de treinen gedurende de dag heen en weer rijden?

- Zet een aantal trajecten uit binnen een gegeven tijdsframe

Visualitatie



Doelfunctie

$$K = p * 10000 - (T * 100 + Min)$$

K = Kwaliteit lijnvoering

P = Fractie bereden verbindingen

T = Aantal trajecten

Min =Totale tijd

Doelfunctie

Holland:

Upperbound: K = 1/28 * 10000 - (7 * 100 + 840) = -1182
Lowerbound: K = 1*10000 - (1 * 100 + 0) = 9900

Nationaal:

Upperbound: K = 1/89 * 10000 - (20 * 100 + 3600) = -5487
Lowerbound: K = 1*10000 - (1 * 100 + 0) = 9900

State space

28^0 + 27^1 + 26^2 + 25^3 + ... + 1^27 + 0^28

= sum(x^i) (waarbij x=28 en per iteratie lager wordt, i = 0 en bij iedere iteratie hoger wordt)

4,96103909134443 1e18

State space wordt kleiner naarmate een nieuw traject is verbonden

Geprobeerde oplossingen

- Random algoritme
- Greedy algoritme
- Greedy Lookahead algoritme
- Hill climber algoritme

Overige implementaties

- Algoritme vaak runnen
- Keuze beginsteden van trajecten

Andere algoritmes vrij lastig omdat er geen einddoel is

Random

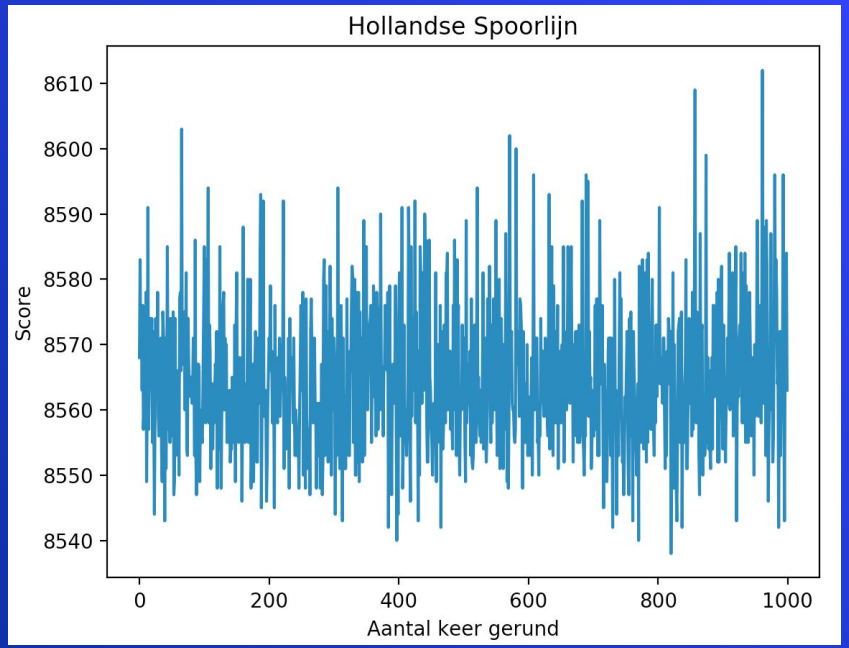
Pseudo code:

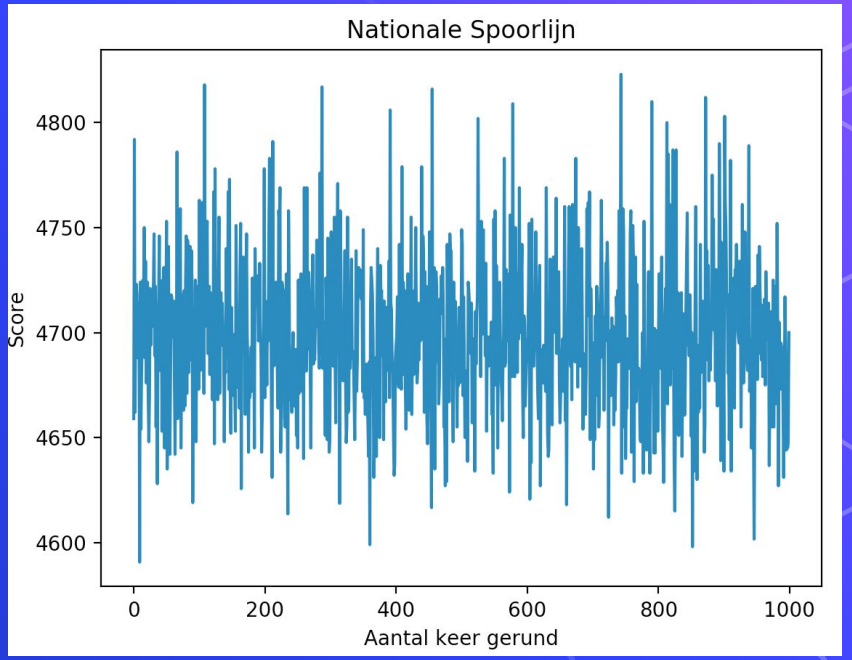
start_city = kies een random stad

find traject:
    kies random neighbour van start_city
    voeg neighbour toe aan het traject
    maak de neighbour de nieuwe start_city

begin aan een nieuw traject als tijdslimiet is overschreden

Random resultaten





Greedy

Pseudo code:

start_city =
- random stad
- stad uit lijst met minste buren
- stad uit lijst met meeste buren

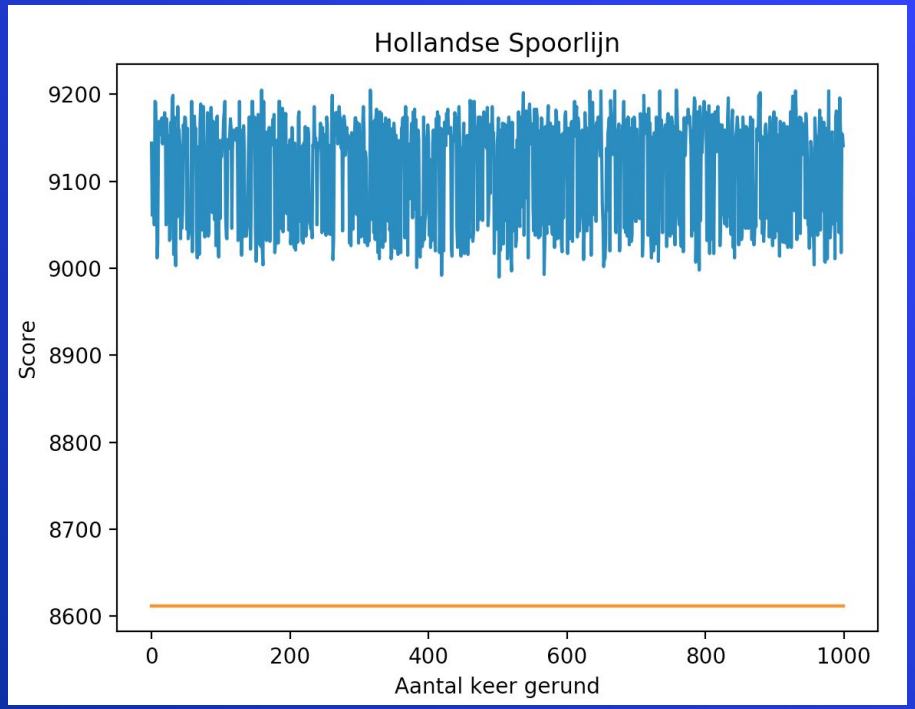
find traject:

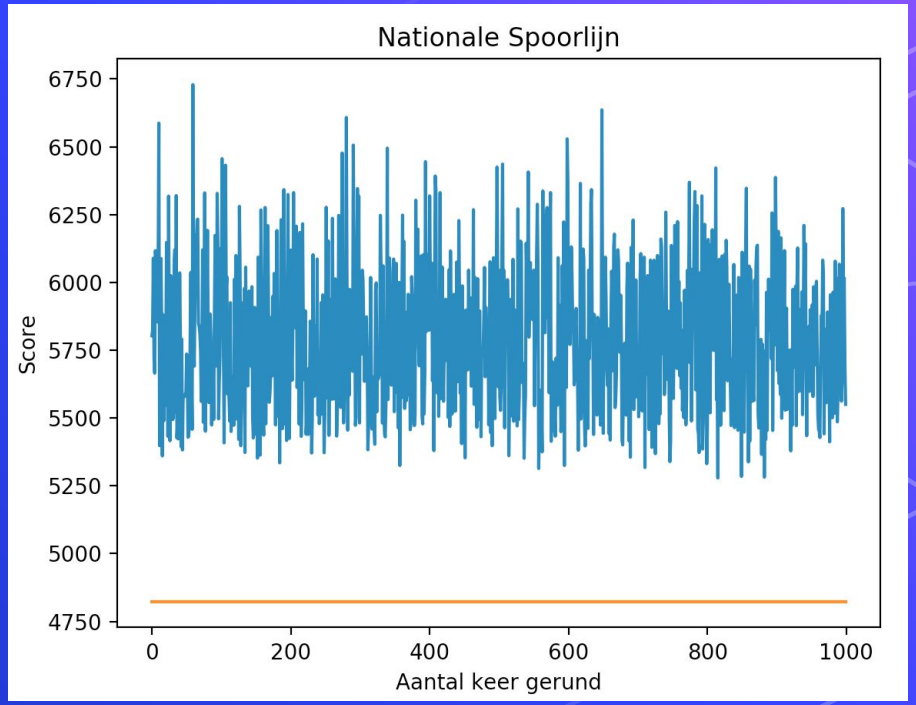
kies neighbour met kortste reistijd van start_city
voeg neighbour toe aan het traject
maak de neighbour de nieuwe start_city

begin aan een nieuw traject als tijdslimiet is overschreden

Greedy resultaten

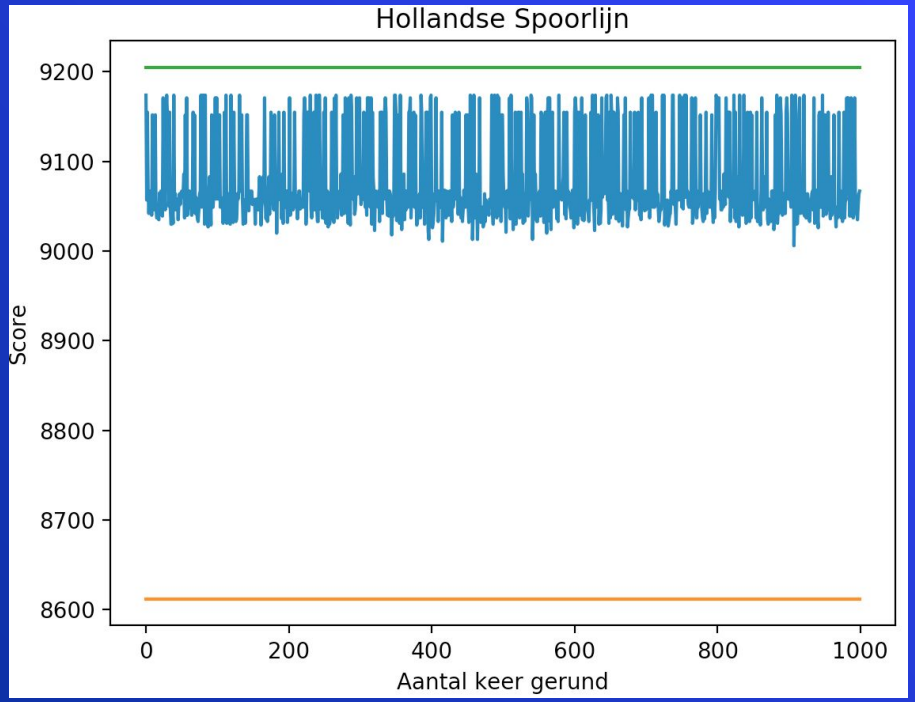
Random beginsteden

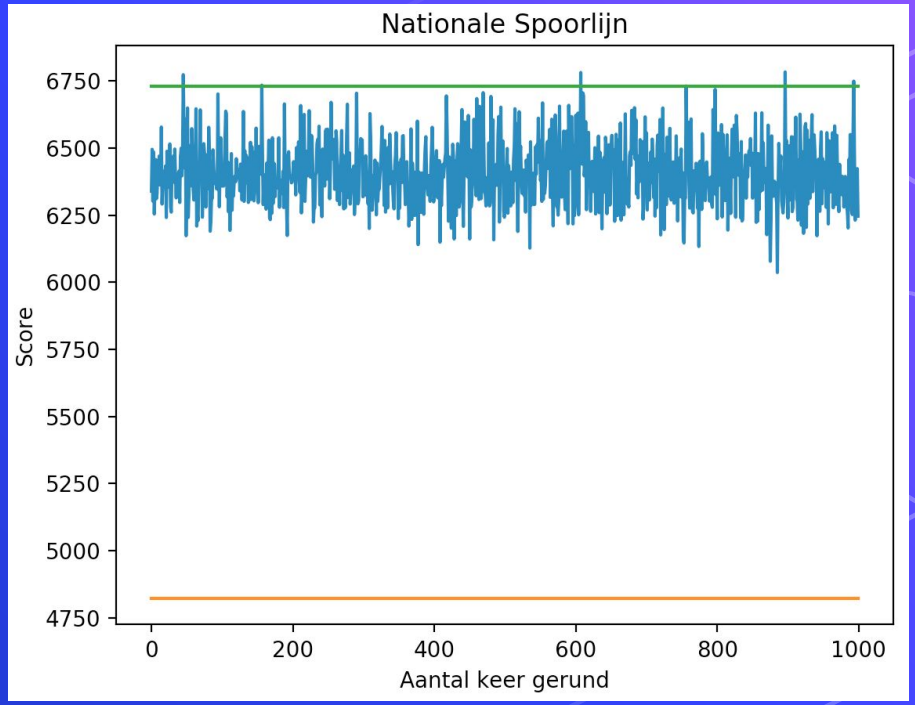




Greedy resultaten

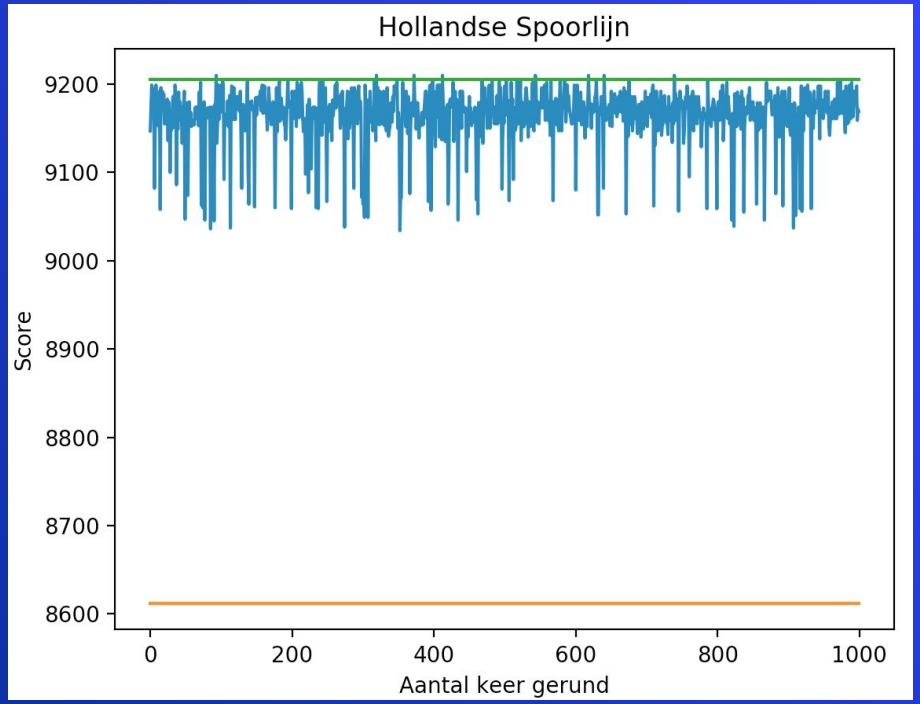
Meeste buren als beginsteden

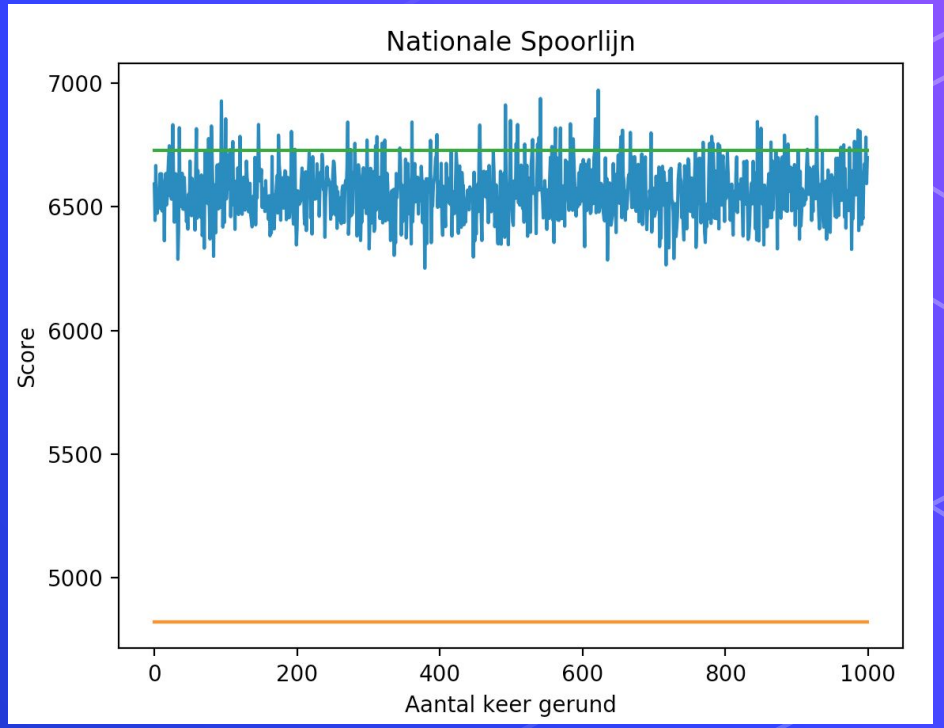




Greedy resultaten

Minste buren als beginsteden





Greedy Lookahead

Pseudo code voor stad uitkiezen:

steps = x
choose random neighbour

for trial in steps:

    traject = []
    for step in steps:

        - maak traject

    check of het langste is of check of het even lang is, maar met kortere tijd:

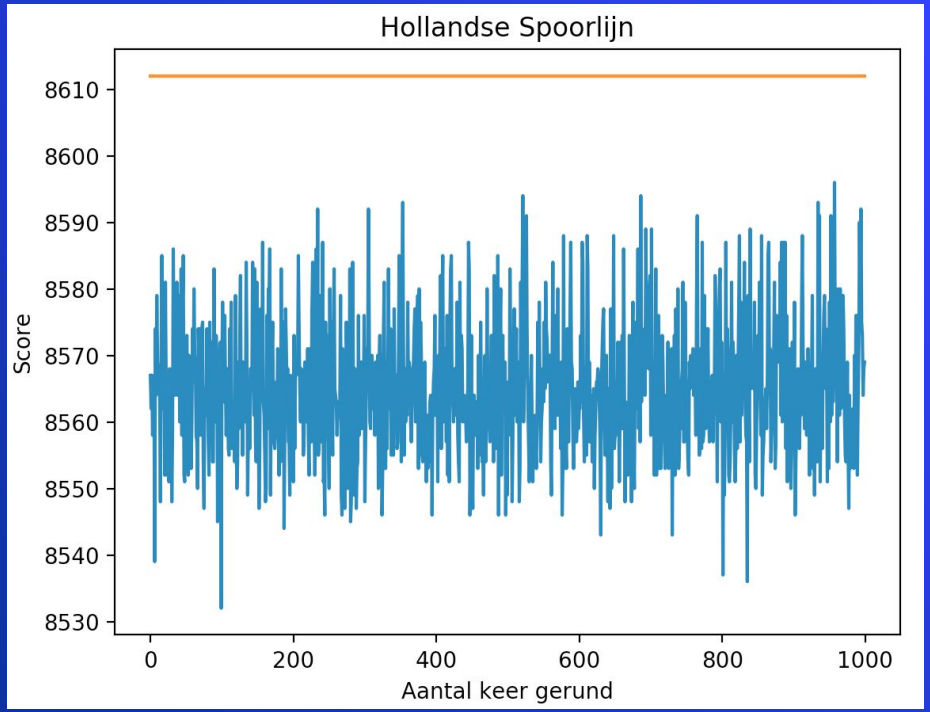
        - best_neigh = neigh

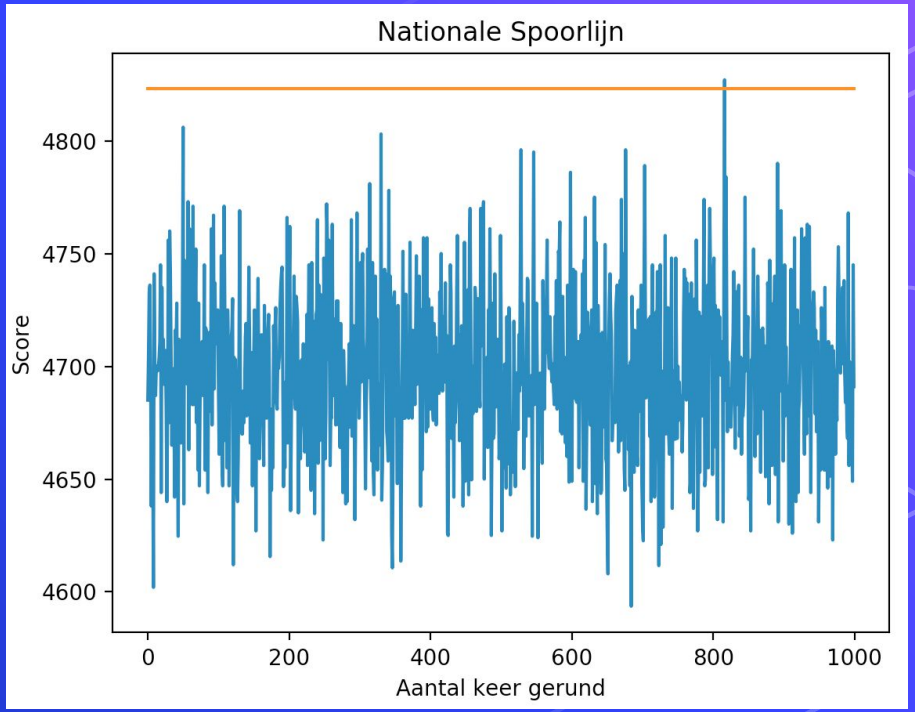
Kijkt naar:

- Langste traject
- Minste tijd

Greedy Lookahead resultaten

3 stappen





Hill Climber

- Extra keuze om algoritme te runnen
- Gebaseerd op random algoritme

- Weinig tot geen effect op Noord-Zuid Hollands spoorlijn
- Levert meestal 100 a 200 punten op voor nationale lijnvoering

pseudocode:

doe 100 keer:

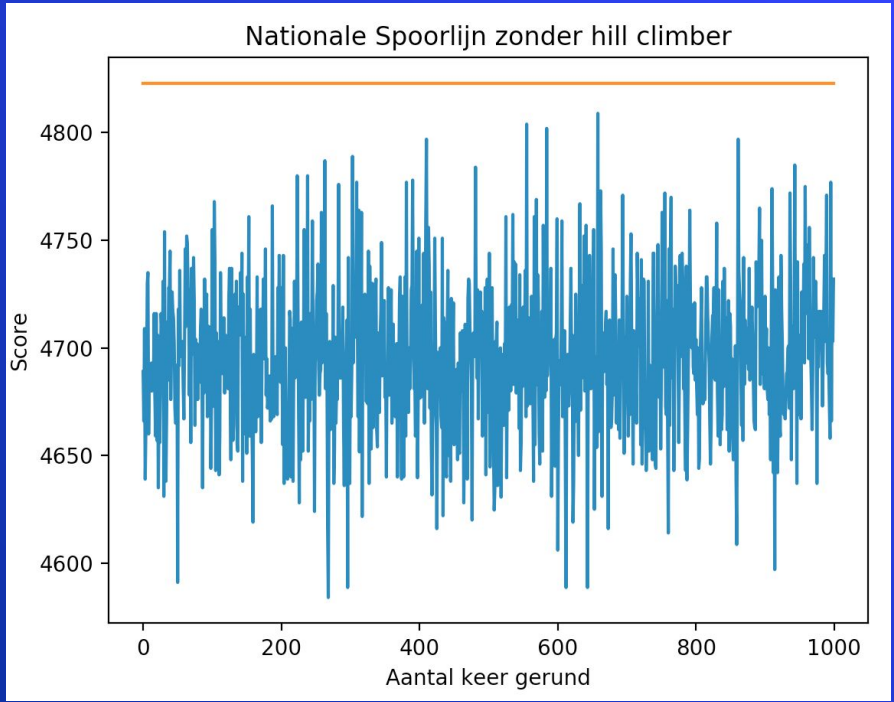
verwijder 2 trajecten
implementeer 2 nieuwe random trajecten

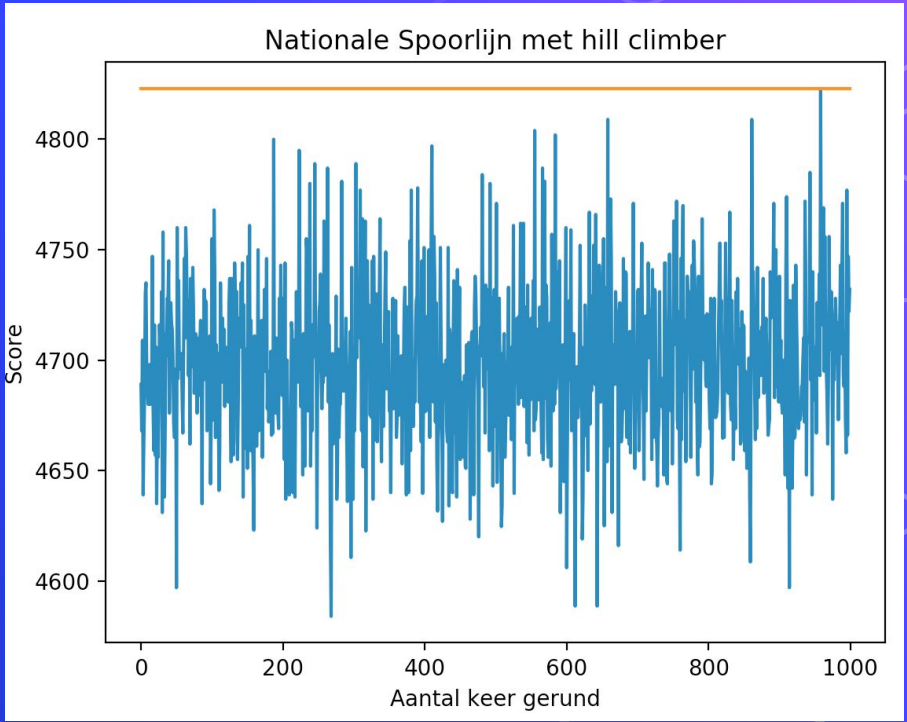
sla de nieuwe trajecten op in variabelen:

- score
- trajecten
- totale tijd
- gebruikte connecties

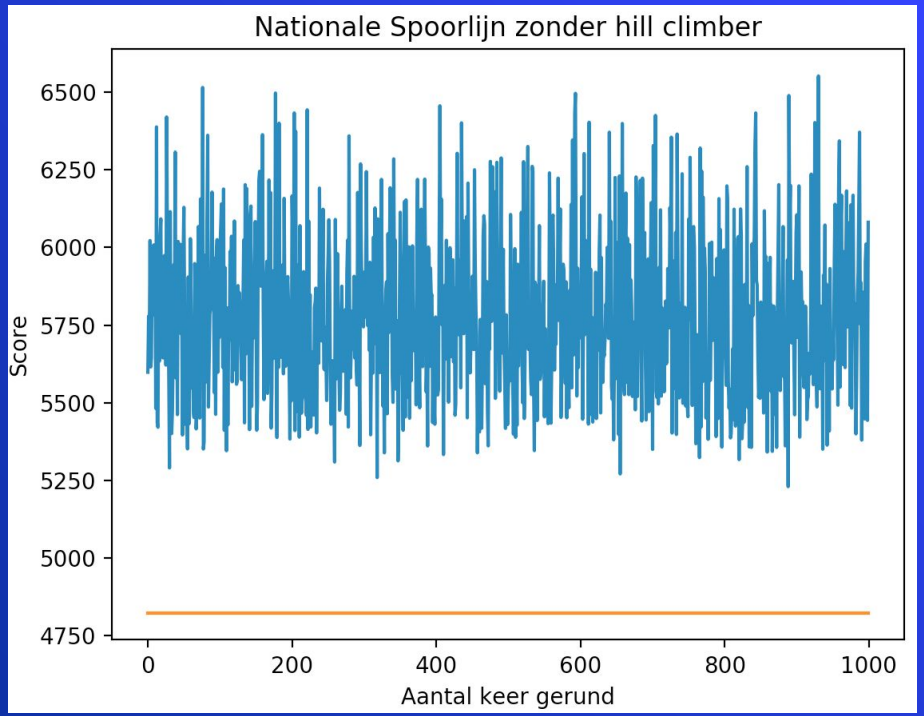
vergelijk de scores:
if nieuwe score > vorige score:
- de oude waarden worden vervangen door de nieuwe variabelen

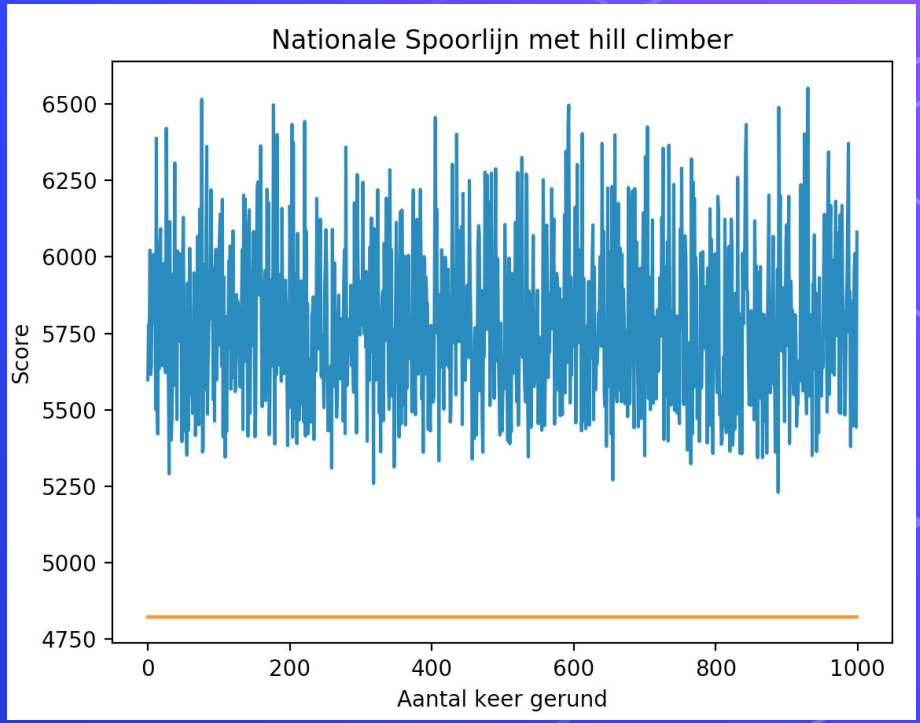
Effect Hill Climber op Random



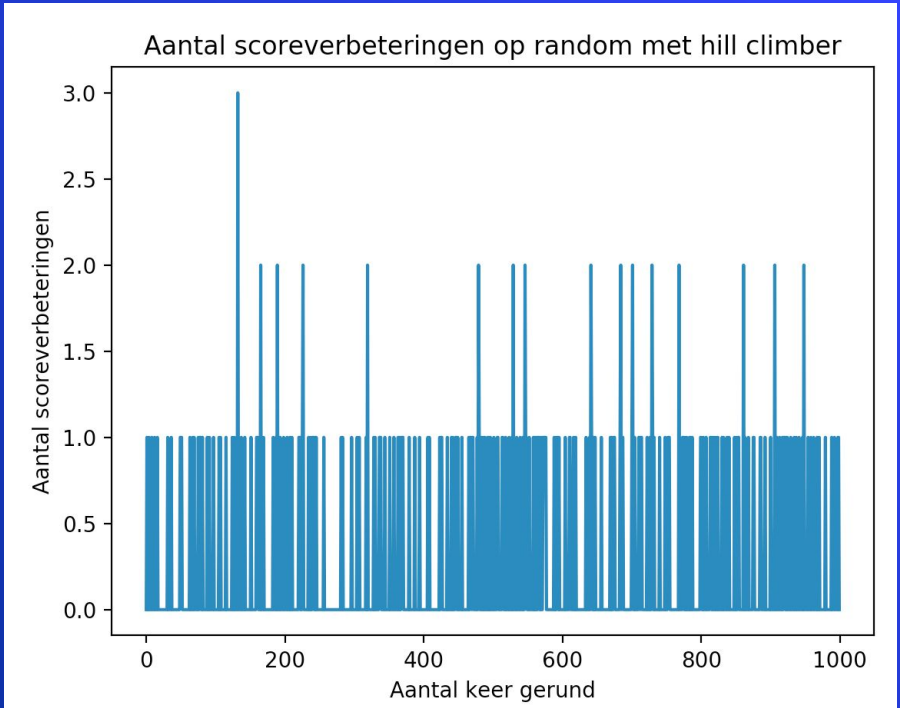


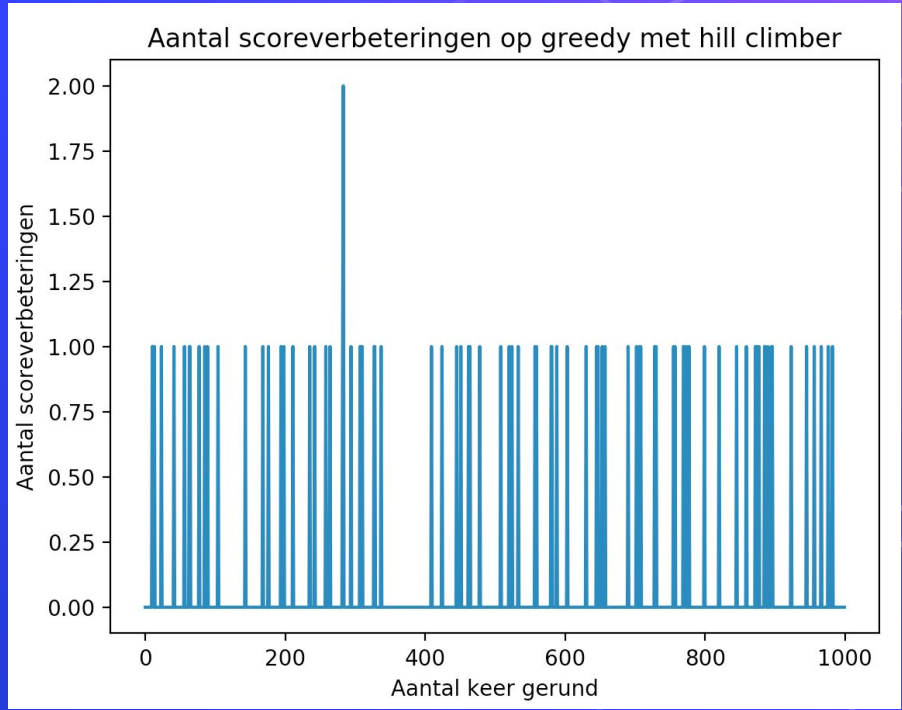
Effect Hill Climber op Greedy



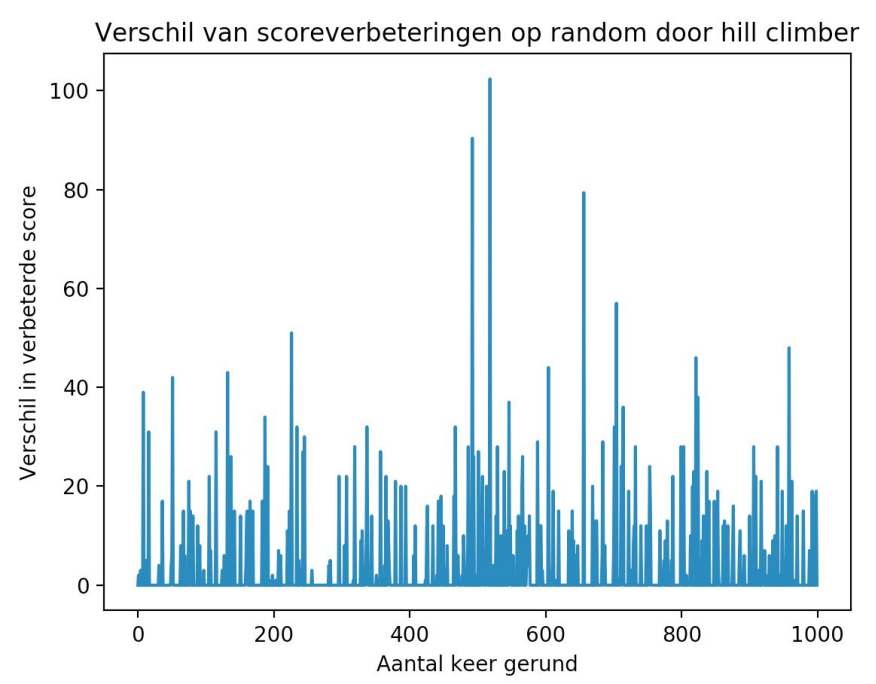


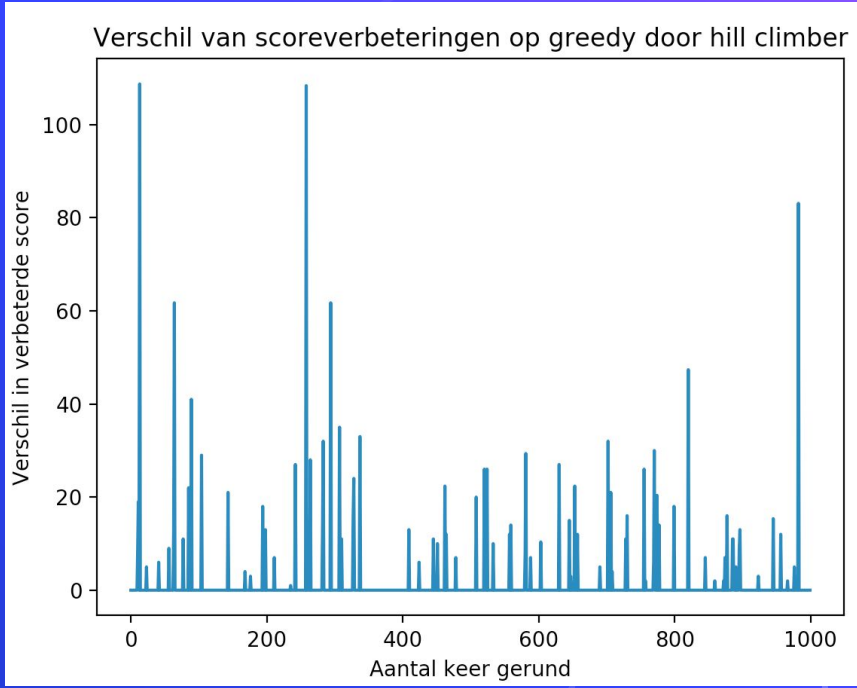
Hoe vaak vindt Hill Climber een verbetering?





Hoe groot is het verschil per verbetering?





Conclusie

- Welke algoritmes/implementaties werken beter?
- Waarom?

Algoritme	Holland	Nationaal	
Random	8610	4850	
Greedy random steden	9200	6700	
Greedy meeste buren	9180	6760	
Greedy minste buren	9210	6950	
Greedy Lookahead	8592	4900	
		Zonder HC	Met HC
Hill Climber op Random	x	4810	4850
Hill Climber op Greedy	x	6510	6510

Einde

Vragen?