Probleembeschrijving

's Winters kan het een keertje voorkomen dat het flink sneeuwt. Hierdoor kunnen de autowegen glad worden en dat is natuurlijk gevaarlijk. Vaak als er veel sneeuw op de wegen ligt worden er strooiwagens ingezet. Deze wagens strooien zout op de wegen zodat het sneeuw wegsmelt waardoor de wegen weer veiliger worden. Deze wagens moeten vaak ook in straten strooien. Het is niet zo handig als een strooi wagen, om bij een andere straat te komen, weer door eenzelfde straat heen moet rijden.

Dit kan komen doordat de bestuurders van de strooiwagens geen goede of duidelijke route krijgen. Het kan uiteraard lastig zijn om een zo'n goed mogelijke route uit te stippelen, vooral als er veel straten zijn waarin er moet worden gestrooid. En dit is dus precies mijn probleem: een algoritme bedenken dat voor strooiwagens de meest optimale route kan bepalen, zodat deze maar 1 keer langs alle straten hoeven te rijden.

De eisen

De eisen aan het algoritme zijn als volgt: ten eerste is het belangrijk dat het algoritme, des te meer punten je hebt, niet te snel toeneemt in tijd (een complexiteit van n! moet vermeden worden). Ten tweede moet het algoritme in ieder geval in de buurt komen van de meest optimale route. Dit hoeft dus niet per se *de* beste route te zijn, maar hij mag er ook niet te ver vanaf liggen. Tot slot moet het algoritme duidelijk weergeven wat dan de nieuwe, betere route is (bijvoorbeeld dat de code de route print zoals 'start bij A, ga vervolgens naar D' etc.

Beschrijving algoritme

Voor mijn probleem heb ik ervoor gekozen om gebruik te maken van Simulated Annealing in combinatie met Two-Opt. Dit omdat mijn probleem een beetje lijkt op het Traveling Salesman Problem.

Simulated Annealing

Simulated Annealing (of SA) is gebaseerd op het verhitten en laten afkoelen van metalen. Metalen smelten meestal bij erg hoge temperaturen.

Met SA start je dus met een hoge temperatuur en (in het geval met TSP) met een willekeurige route (momenteel is deze route de beste route). Tijdens iedere iteratie maak je een nieuwe route aan, die net een klein beetje anders is dan de vorige route. Daarna controleer je of de nieuwe route een kortere totale afstand heeft dan de huidige route.

Als dit het geval is dan wordt de nieuwe route gezien als de nieuwe huidige route. Nu zou je denken dat iedere slechtere route automatisch wordt afgekeurd. Dit is echter niet het geval met SA. Als een route slechter is dan de huidige route dan is er nog steeds een kans dat deze alsnog wordt gekozen als nieuwe huidige route. Er wordt een getal genaamd 'acceptatie' berekend. Als deze acceptatie groter is dan een willekeurig getal tussen 0 en 1 dan wordt de slechtere route als huidige route gekozen. Hoe hoger de temperatuur des te hoger de kans dat slechtere routes worden aangenomen.

De reden dat SA in het begin slechtere routes uitkiest is om te voorkomen dat het algoritme vast komt te zitten in een Lokaal Optimum. Een Lokaal Optimum is misschien wel een betere route dan de start route, maar is niet de beste route. De beste route is namelijk het Globaal Optimum. Door

slechtere routes aan te nemen vergroot je het gebied waarin het algoritme zoekt. Hierdoor wordt de kans op het vinden van het Globaal Optimum vergroot.

Pseudo Code algoritme [FOTO]: https://www.researchgate.net/figure/The-pseudo-code-of-simulated-annealing-algorithm_fig2_309537833

Paper algoritme [PDF]: https://projecteuclid.org/download/pdf_1/euclid.ss/1177011077

Two Opt

In TSP heb je te maken met edges. Een edge verbindt twee punten (of steden) met elkaar. Stel je hebt punten A en B die aan elkaar verbonden zijn en C en D. Two Opt wisselt de punten dan met elkaar. Zo krijg je edge A en C, en edge B en D. Daarna wordt gekeken of deze omwisseling een kortere route oplevert. Als dit het geval is dan neemt Two Opt de route aan. Is dit niet het geval dan wordt de route niet aangenomen. Een volledige Two Opt search zal alle mogelijke (valide) swap-combinaties uit proberen.

Pseudo Code algoritme [FOTO]: https://www.researchgate.net/figure/The-pseudo-code-of-simulated-annealing-algorithm_fig2_309537833

Evaluatie

Simulated Annealing (in combinatie met Two Opt) voor- en nadelen voor mijn probleem:

+ loopt minder risico om vast te blijven zitten in het Lokale Optimum (vergeleken met algoritmen zoals hill climbing en PSO)

+ doet er niet te lang over om een route te vinden (tussen de 2-8 seconden voor 10 punten)

- is niet helemaal polymorfisch, ieder punt moet genoeg verbindingen met andere punten hebben wil je dat het algoritme een kans heeft om een betere route te vinden.

- het invoeren van 'buur steden' moet handmatig worden gedaan en dat kan best wat tijd kosten

Overwegingen

Voor mijn SA algoritme heb ik gekozen om de temperatuur lineair af te laten nemen (1 graad per iteratie). Dit omdat alle (niet lineaire) formules die ik heb geprobeerd voor het laten afnemen van de temperatuur alleen maar resulteren in slechtere routes, en dat is natuurlijk niet de bedoeling.

De start temperatuur is 1000 x het aantal punten in de route. Ik heb gekozen voor 1000 omdat dit voor een optimale algoritme snelheid zorgt en bijna geen slechte resultaten oplevert (1 op de ongeveer 5 keer zal je een slechtere route terug krijgen). Ik heb dus een beetje effectiviteit gecompenseerd voor een hogere snelheid.

Echter wordt de snelheid beïnvloed door nog een andere parameter: namelijk het aantal verbindingen tussen alle punten. Hoe minder verbindingen, des te langzamer het algoritme wordt. Het algoritme heeft per iteratie (2 x aantal punten) de kans om een nieuwe valide route te maken (door middel van het omwisselen van 2 punten). Uiteraard wordt het aantal mogelijke valide omwisselingen kleiner des te minder verbindingen er zijn. Hierdoor moet het algoritme, in iedere iteratie, vaker gebruik maken van (bijna) al zijn kansen. Hierdoor neemt dus uiteindelijk de tijd toe.

De formule voor het minimum aantal verbindingen is: X . 2 + (wortel van (2^x))

X = aantal punten in de route

Let op: deze formule is niet helemaal juist aangezien het aantal verbindingen per punt ook invloed kan hebben op de performance van het algoritme.