ROUTE-OPTIMALISATIE-ALGORITMEN VOOR DHL

IPASS-project

Ole den Hertog
Studentnummer: 1830026

Inhoud

Inleiding

In dit verslag zal er een overzicht gegeven worden van wat het probleem is dat door de opdrachtgever is gesteld. Daarna wordt beschreven welke eisen de opdrachtgever gesteld heeft voor de oplossing. De gekozen algoritmes worden toegelicht en de voor- en nadelen worden besproken. Er wordt een evaluatie gedaan van de gekozen algoritmes en een advies gegeven.

Probleembeschrijving

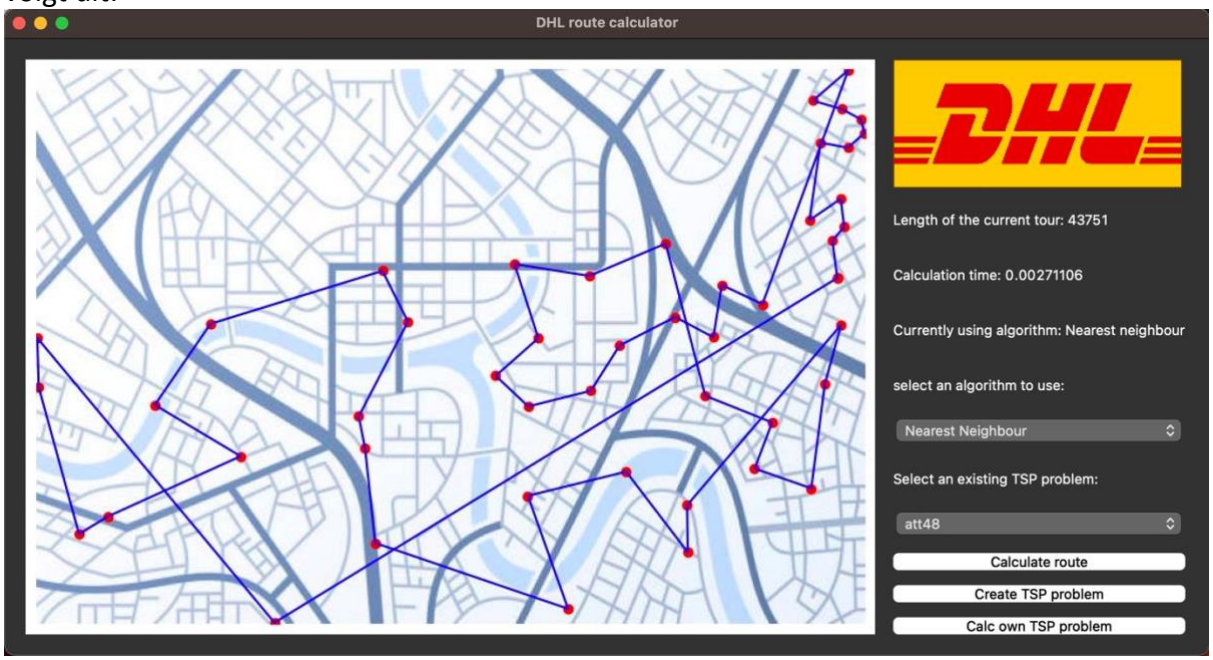
Er is door de pakket bezorg service DHL gevraagd om een implementatie te maken van het “Travelling Salesman Problem” (TSP). Dit probleem is een route probleem, de stelling is als volgt: vind voor een set steden de snelste route die alle steden precies één keer afgaat en weer eindigt bij het startpunt. DHL heeft elke dag met dit probleem te maken, zij moeten namelijk routes vinden tussen alle adressen waar pakketjes bezorgd moeten worden en weer eindigen bij het warenhuis. Er moet een applicatie worden ontwikkeld waarin er routes op een efficiënte manier berekend kunnen worden.

Eisen

De eisen van de opdrachtgever zijn dat er tenminste drie algoritmes geïmplementeerd moeten worden waarmee er routes berekend kunnen worden. Ook moet er een Graphical User Interface (GUI) gemaakt worden om de werking van de algoritmes te demonstreren. De gekozen algoritmes moeten vergeleken kunnen worden.

Applicatie

Er is een GUI gemaakt om de werking van de algoritmes te demonstreren. De GUI ziet er als volgt uit:



Figuur 1

Er kan tussen de drie verschillende algoritmes gekozen worden. Er kan gekozen worden tussen verschillende sets van steden om routes tussen te berekenen. Ook kan ervoor gekozen worden zelf een set met steden in te voeren in de applicatie met de knop “create TSP problem”. Verder is er een visuele weergave, berekentijd en afstand te zien van de gevonden route.

Algoritmes

Hier zal beschreven worden welke algoritmes gekozen zijn en hoe deze werken. De gekozen algoritmes zijn, nearest neighbour, 2-opt en een evolutionair algoritme.

Nearest Neighbour

Het nearest neighbour algoritme is een simpel algoritme. Het algoritme werkt door een random stad te kiezen als beginstad. Daarna gaat het algoritme alle nog niet bezochte steden af en kiest degene die het dichtstbij is bij de huidige stad. Deze stad wordt toegevoegd aan de tour en is nu de huidige stad. Dit wordt herhaald tot alle steden bezocht zijn (Kizilates, & Nuriyeva, 2013).

2-opt

Het 2-opt algoritme werkt door te beginnen met een random gekozen route. Daarna wordt er geïtereerd over elke paar van steden, bijvoorbeeld A-D en C-B. De lengte tussen de huidige paar van steden wordt berekend (afstand van A-D + C-B). Ook wordt de nieuwe afstand berekend tussen A-B en C-D. Als (A-B + C-D) - (A-D + C-B) < 0, dan wordt de 2-opt wissel uitgevoerd. Dit wordt gedaan door de volgorde van de steden tussen het paar om te draaien. In ons voorbeeld houdt dat in dat: ADBC -> ABDC. Hieronder is dezelfde wissel schematisch weergegeven.

```
- A   B -              - A - B -
    ×   |       ==>            |
- C   D -              - C - D -
```

Deze stappen worden herhaald tot er geen verbeteringen meer mogelijk zijn (Karagül et al., 2016).

Evolutionair

Het evolutionair algoritme werkt op basis van principes van natuurlijke selectie. Elke route is een individu en bij elke route hoort een afstand. Er wordt een populatie gegenereerd met 100 routes (het Nearest Neighbour-algoritme wordt gebruikt om de initiële populatie te genereren). Bij elke generatie worden de 50% beste individuen (met de kortste afstand) geselecteerd om ouders te worden. Deze ouders genereren 500 kinderen door middel van kruising (cross-over). Elk kind heeft een kans van 10% om te muteren. Een mutatie bij een kind houdt in dat twee willekeurige steden in de route worden verwisseld. Vervolgens worden van de 500 kinderen plus de 100 individuen uit de huidige generatie de 100 beste individuen geselecteerd om te overleven naar de volgende generatie. Dit proces wordt herhaald tot een bepaald aantal generaties is bereikt, waarna het fitste individu (route) wordt geselecteerd als winnaar (Larrañaga et al., 1999).

Evaluatie

Voor het testen van de werking van de algoritmes is er gekozen om alle data van elke berekening te schrijven naar CSV-bestand. Zo kan daarna gemakkelijk de gemiddelde waarden over meerdere oplossingen gevonden worden. Er is getest met verschillende testcases. Elke testcase heeft een andere hoeveelheid steden. Dit is een zeer belangrijke parameter, want hoe meer steden hoe lastiger het probleem wordt. Bij het evolutionair algoritme is ervoor gekozen om bij alle testcases een populatie van grootte 100 aan te houden en 200 generaties te berekenen. De gevonden resultaten zijn hieronder zichtbaar.

wi29	**Time**	**Distance**
Nearest Neighbour	0,000710779	34942,88889
2-opt	0,008035719	28102
Evolutionary/Genetic	4,806157351	29233,6

att48	**Time**	**Distance**
Nearest Neighbour	0,002172012	41648,16107
2-opt	0,028263936	34875
Evolutionary/Genetic	11,27051681	37080,77778

berlin52	**Time**	**Distance**
Nearest Neighbour	0,001833776	9399,471698
2-opt	0,022394115	8542
Evolutionary/Genetic	9,374043056	8127,928571

ch130	**Time**	**Distance**
Nearest Neighbour	0,010149564	7615,607143
2-opt	0,106879732	6812
Evolutionary/Genetic	37,94413102	7153

a280	**Time**	**Distance**
Nearest Neighbour	0,054758733	3296,090909
2-opt	0,162106403	2819
Evolutionary/Genetic	148,7574012	3138

Figuur 2

Zoals verwacht zijn er grote verschillen tussen de drie algoritmes. Er vallen een paar dingen direct op. Het nearest neighbour algoritme is altijd het snelst, maar geeft ook altijd het slechtste antwoord vergeleken met de andere algoritmes.
Het evolutionaire algoritme is veruit het traagst. Dit komt omdat dit algoritme de meeste berekeningen moet doen. Er vallen bij dit algoritme veel parameters aan te passen, zoals: hoeveelheid generaties, hoeveelheid kinderen, mutatie kans en de hoeveelheid

geselecteerde ouders. Al deze parameters hebben groot effect op de berekentijd en de afstand van de gevonden route. Als bijvoorbeeld het aantal generaties hoger is wordt de gevonden route beter, maar de berekentijd ook groter. Als er dus genoeg tijd is kan dit algoritme langer gedraaid worden en kunnen er dus betere resultaten gevonden worden. Het 2-opt algoritme is een mooie tussenweg. De berekentijd van dit algoritme is significant sneller dan het evolutionaire algoritme, maar wel iets langer dan het nearest neighbour algoritme. Wat ook opvalt is dat het 2-opt algoritme veel betere routes vindt dan het nearest neighbour algoritme. Een belangrijk verschil tussen de drie algoritmes is dat 2-opt altijd dezelfde route vindt gegeven een set steden. Bij het nearest neighbour algoritme is dit ook het geval wanneer dezelfde beginstad gekozen wordt. Bij het evolutionaire algoritme is dit compleet anders, het algoritme is gebaseerd op verschillende random gebeurtenissen wat betekend dat elke gevonden route een andere afstand heeft.

Advies

Van de drie beschreven algoritmes wordt er voor DHL aangeraden het 2-opt algoritme te gebruiken voor het berekenen van bezorg routes. Dit algoritme geeft constante betrouwbare antwoorden, de gevonden route is altijd beter dan het nearest neighbour algoritme en de berekentijd is altijd sneller dan het evolutionaire algoritme.

Bronnen

Kizilates, G., Nuriyeva, F., & 3rd International Conference on Computational Science, Engineering and Information Technology, CCSEIT 2013 Konya, TUR 2013 06 07 - 2013 06 09. (2013). On the nearest neighbor algorithms for the traveling salesman problem. *Advances in Intelligent Systems and Computing, 225(1),* 111–118. https://doi.org/10.1007/978-3-319-00951-3_11

Karagül, K., Aydemir, E., & Tokat, S. (2016). Using 2-Opt based evolution strategy for travelling salesman problem. *International Journal of Optimization and Control : Theories & Applications*, *6*(2), 103–113. https://doi.org/10.11121/ijocta.01.2016.00268

Larrañaga P, Kuijpers, C. M. H., Murga, R. H., Inza, I., & Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: a review of representations and operators. *Artificial Intelligence Review : An International Science and Engineering Journal, 13(2)*, 129–170. https://doi.org/10.1023/A:1006529012972