26 januari 2024

Individuele onderdeel; update technische beschrijving

Team RAM

UML:
https://github.com/mercedezvdw/ram_ah

Yessin Radouane

Ik zal het tweede base line algoritme maken. De eerste baseline was erg slecht en volledig random. Ik zal een algoritme maken die dus een random batterij kiest voor elk huis en dan een pad vindt d.m.v. een keuze te maken tussen de mogelijke paths die niet het meeste afstand creëert tussen de huidige positie van de kabel en het huis.
Eventueel ook een algoritme die een random batterij kiest voor elk huis en daarbij een greedy path kiest. Dit houdt dus in dat er een randomise_v2 en greedy_random.py script gemaakt moet worden.
Daarnaast zal ik functionaliteiten in het main programma maken om sneller en efficiënter verschillende algoritmen aan te roepen. Hierbij hoort dus een class AlgoRunner() die een algoritme naam als argument in de init functie neemt zodat het weet dat dit algoritme gerund moet worden. Zo hoef je dus niet elke keer de main te veranderen om een algo aan te roepen.

Update:

Het tweede baseline algo was een random algorithm die alle mogelijke moves simuleert. Vervolgens kiest het een random move van de twee beste opties. Ook wordt er random een batterij gekozen per huis. Verder heb ik nog een greedy algoritme gemaakt. Dit algoritme kiest een move die de afstand minimaliseert tussen de huidige positie van de kabel en het huis dat het zoekt. Ik ben niet echt onverwachte dingen tegen gekomen hier.
Dit is toegevoegd in de volgende commit:
https://github.com/mercedezvdw/ram_ah/commit/7e1e56e1732ca2529f206f97c4379296cc0861a6

Ik heb ook een argparse toegoevoegd in de main.py. Daarnaast heb ik een class gemaakt die alle algo's zal runnen. Deze class heet Engine. Hij neemt algo, district en plot als init arguments. Hij leest dan zelf de house en battery objects van de csv. Vervolgens doe je engine.run() om het geselecteerde algoritme te runnen. Ook hier zijn er geen onverwachte dingen tegen gekomen. Het was alleen even puzzelen met hoe boolean ar
Dit is toegevoegd in deze commit:
https://github.com/mercedezvdw/ram_ah/commit/32ce585e3ff5e58a55e3ff79f281ee02f30ffdbb

Mercedez van der Wal

Ik zal een algoritme uitwerken die op zoek gaat naar de kortste route naar een batterij. Op basis van Euclidische afstand wordt gekeken welke batterij het meest dichtbij is en daarna wordt er een heuristiek gebruikt die kijkt naar de dichtstbijzijnde kabel in de buurt. Aan de hand hiervan wordt een kabel gelegd. Het leggen van de kabel wordt ook gedaan door een heuristiek, namelijk de meest eenvoudige/snelste route.

Functionaliteiten:
- Bereken dichtstbijzijnde batterij
- Zoek dichtstbijzijnde kabel
- Vergelijk bovenste twee om kabels te kunnen leggen of verbinden

Functies:
- calculate_distance(coordinate1, coordinate2)
- find_closest_battery():
- find_closest_cable():
- compare_results():
- create_cable():

Met behulp van deze functionaliteiten kunnen wij resultaten genereren en vergelijken hoe optimaliserend dit algoritme werkt.

Update

Er zijn een aantal problemen waar ik tegenaan ben gelopen. Nadat ik mijn algoritme voor het eerst ging runnen, kwam ik erachter dat er geen rekening werd gehouden met de maximumcapaciteit. Na het implementeren hiervan, kwam ik op een nieuw probleem terecht. Namelijk dat de capaciteitsverdeling niet helemaal optimaal was. Bij het toewijzen van de laatste huizen, kwam het dan voor dat het laatste huis niet meer kon worden toegewezen. Voorbeeld huis A heeft een output van 60 en batterij 1 en batterij 2 hebben beide nog een capaciteit over van 30. Om dit op te lossen heb ik een nieuwe functie geïmplementeerd die dan telkens een andere kabel weghaalt om ervoor te zorgen dat het laatste huis verbonden kon worden. Dit wordt gedaan in een loop totdat alle huizen verbonden zijn.

Daarnaast heb ik het berekenen van de afstand veranderd naar het berekenen op basis van de Manhattan distance.

Mijn commits:
https://github.com/mercedezvdw/ram_ah/commits?author=mercedezvdw&since=2023-12-31&until=2024-01-26

Rembrand Ruppert

Eerste beschrijving:

Ik heb en zal een algoritme schrijven dat een ‘heatmap’ maakt van de dichtheid van huizen. Dit algortime heb ik al een simpele basis van gemaakt, maar doet niks anders als van individuele huizen de dichtheid op een grid bepalen. Als dit werkt kan ik vervolgens een algoritme schrijven dat een grove indeling van subdistricts zal maken, wat wij als team kunnen gebruiken om de hoeveelheid kabels te minimaliseren en om beter te voldoen aan de batterij capaciteit restrictie, gezien een deel huizen binnen een subdistrict alleen op 1 batterij aan kunnen sluiten als het goed werkt i.p.v. alle huizen op 1 batterij. Ook ben ik verantwoordelijk geweest voor het visualiseren van de districts door het gebruik te maken van matplotlib.pyplot. Dit is nog een erg slordige code, dus naast de heatmap zal ik bezig zijn met deze code net en simpel te navigeren maken. In de visualisatie kunnen kabels die gelegd zijn door algoritmen vrij simpel getekend worden, hoe deze ook liggen.

Update:

Een verdeling van subdistricten maken door het gebruik van een heatmap werkte niet goed. Door de verdeling van alle huizen was er niet genoeg ruimte tussen bepaalde gebieden om op deze manier subdistricten in te vullen. Wat ik nu heb gedaan is een toepassing van een ‘k-means’ algoritme waardoor ik vanuit elke batterij een selectie van de dichtstbijzijnde huizen kon maken en deze hieraan verbinden. Dit is nog niet volledig uitgewerkt, ik moet nog steeds een overlap van kabels voorkomen, maar de basis is er. Dit algoritme zal waarschijnlijk het beste werken in de ‘advanced’ opgave, waar wij zelf de batterijen plaatsen. Ook ben ik bezig geweest om data op te kunnen slaan en te kunnen schrijven. Oorspronkelijk was ik van plan om dit in .csv formaat te doen, maar het blijkt dat een .json formaat wordt verwacht, wat ik zelf nog nooit eerder heb gebruikt, maar zal niet voor veel moeilijkheden zorgen denk ik.

Commits:

- https://github.com/mercedezvdw/ram_ah/commit/21b1c47b8fd800419276df5ca55d7e8065eeef2c

Een grote eerste commit om alles op te schonen en klaar te maken om aan te werken: “added my own contribution SADDA and DCA, made a class for @mercedezvdw 's NBH algorithm, and rewrote the plotting and csv reading functions to have their own files and in separate classes. main.py is now way more roomy”

- https://github.com/mercedezvdw/ram_ah/commit/e63d0dc891297c6bcd82b083086e08454f8babfe

eerste lees/schrijf bestand setup toegevoegd

- https://github.com/mercedezvdw/ram_ah/commit/362254e731318e51585f6de50eda39bfdfb8204a

functies hieraan toegevoegd

- https://github.com/mercedezvdw/ram_ah/commit/641463cac29fc344be017834a111928a1046d902

grote verandering in SADDA + eerste files voor experiment data

- https://github.com/mercedezvdw/ram_ah/commit/2fe25f42271bf5b864aea70385f87cb6cef161e0

SADDA grotendeels afgemaakt

- https://github.com/mercedezvdw/ram_ah/commit/11f3dbe2a3ad4c978c65219f1c2ea75e24377d8c

extra comments toegevoegd aan SADDA

- https://github.com/mercedezvdw/ram_ah/commit/be6eb5860a386ce50f1d5e040ab1b5179df64fa7

simpele testcase aangemaakt

- https://github.com/mercedezvdw/ram_ah/commit/6d13eda860e91e27c6cffadc8bfee0e345548926

meer data opslag wijzigingen en files klaarmaken voor .csv opslag

- https://github.com/mercedezvdw/ram_ah/commit/8af40c6a0a0196c95a7200375924d16ec79b47fa

pad naar files toegevoegd in lees/schrijf code

- https://github.com/mercedezvdw/ram_ah/commit/c1f4408f10221a4e30f829b7ea83277644e0eb8d

SADDA en lees/schrijf files wijzigingen