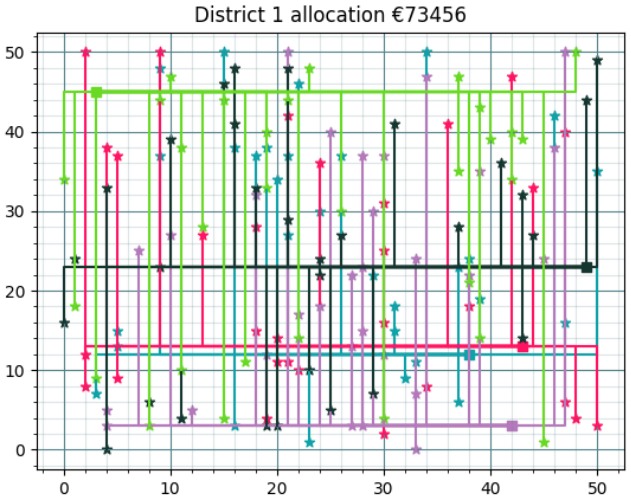
1 Theorie: Tweede algoritme

Onze basis oplossingen worden gegenereed door een greedy en random algoritme. Bij greedy kijken we naar alle dichtsbijzijnde huizen per batterij en alloceren we deze per batterij in een bepaalde volgorde. In het random algoritme alloceren we 30 willekeurige huizen per batterij. Omdat het random algoritme gegarandeerd een oplossing geeft, hebben we besloten deze allocatie die hieruit komt door te geven aan hill climber.

In hill climber starten we met de initiële connecties tussen huizen en batterijen en het aantal iteraties. Tijdens elke iteratie maken we een swap tussen twee huizen, en als de swap een verbetering oplevert, accepteer je deze en zo niet, dan maak je de swap ongedaan. In tegenstelling tot simulated annealing kan dit algoritme ervoor zorgen dat we vast komen te zitten in een lokaal minimum, omdat we slechts verbeteringen accepteren en zo niet uit een lokaal minimum kunnen springen. Het accepteren van verbeteringen is niet meer aan kans overgelaten.

Op basis van trial-and-error is de waarde van het aantal iteraties gekozen. Het algoritme verloopt sneller zodra de kabels niet gedeeld worden. We kunnen na het runnen van dit algoritme kiezen om met de vernieuwde allocatie met gedeelde kabels te plotten. Omdat we het algoritme runnen in het geval dat kabels niet gedeeld kunnen worden, is het sneller te bepalen wat ongeveer de beste aantal iteraties is. We merken dat 100, 1000 en zelfs 10000 iteraties toch tot verbeteringen in kosten leiden, dus hebben we dit aantal op 100000 gezet, zodat we zo goed mogelijk tot het (globale) minimum vinden.

Om echt te besparen op kosten, is het het beste om een algoritme met unieke kabels mee te geven, omdat we hiermee het grootst het prijzenverschil zien van het algoritme.



Deze allocatie is geheel willekeurig en kost €73456. Nadat we het hill climber algoritme erop los hebben gelaten krijgen we een optimalere oplossing: