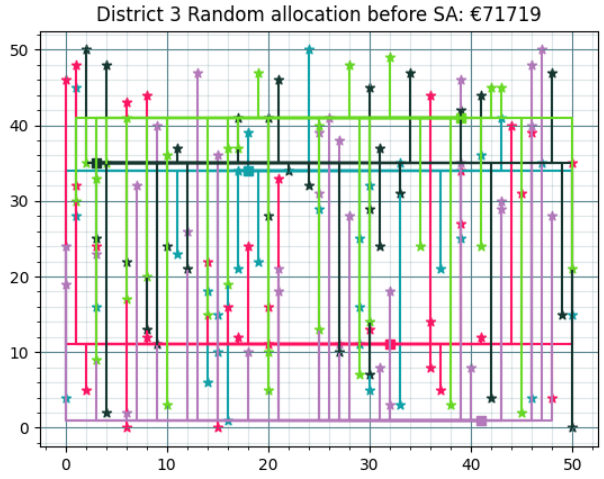
1 Theorie: Eerste algoritme

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 simulated annealing.

In simulated annealing starten we met een aantal variablen: de initiële connecties tussen huizen en batterijen, de start temperatuur en het aantal iteraties. We merken al snel dat het niet echt uitmaakt welke initiële connecties je meegeeft, als de parameters goed zijn dan convergeren de kosten naar ongeveer hetzelfde getal. Dat is natuurlijk ook de bedoeling, dat we zo dicht mogelijk bij het globale minimum komen. Wat ook verschil maakt, is het accepteren van niet feasible allocaties en hier een "boete" aan geven. In het begin van het algoritme zal het soms voorkomen dat de computer deze niet feasible allocatie accepteert, maar naarmate de temperatuur daalt zal dit niet meer zo snel geaccepteerd worden.

Op basis van trial-and-error zijn de waardes van de parameters gekozen. Bij het geven van boetes voor niet feasible allocaties convergeren de kosten naar een minimum. Deze boete verschilt per district. Dit is waarschijnlijk een lokaal minimum, aangezien de state space van dit probleem onvoorstelbaar groot is.

We geven simulated annealing de volgende allocatie mee:



Deze allocatie is geheel willekeurig en kost €71719,-. Zeker omdat lijnsegmenten niet gedeeld kunnen worden is dit relatief duur. Maar nadat we het simulated annealing algoritme erop los hebben gelaten krijgen we een veel optimalere oplossing, zoals hieronder te zien.