Operations Research 5

Discrete Improving Search

Mexx Dirkx & Michiel Kwantes

Table of Contents

1 Inleiding

In dit verslag beschrijven wij de aanpak voor de OR5 Programming Assignment 2025-2026. Het probleem gaat over het bezorgen van kranten door 4 bezorgers in Manhattan. Het doel is om de laatste krant zo vroeg mogelijk te bezorgen.

We werken in Python aan twee oplossingsmethodes:

1. Een greedy constructieve heuristiek, die een eerste oplossing opbouwt.
2. Een metaheuristiek (of meerdere), die een betere oplossing zoekt door systematisch de bestaande oplossing te verbeteren.

2 Bestanden en deliverables

In de meegeleverde zip zijn alle Python- en Excel-bestanden opgenomen.
De andere Python-bestanden laten zien hoe we stap voor stap tot de uiteindelijke oplossing zijn gekomen.

De definitieve VRP-oplossing komt uit:

- **Week_6_Multi_Route_VRP_quadrant_NN_2_Opt+SA.py** of **Final.py**
  (zelfde bestand)

Dit script leest **`newspaper problem instance.xlsx`** in, bouwt en verbetert de vier routes (kwadranten → NN → 2-Opt → SA → 2-Opt)
en exporteert de oplossing naar **`solution.xlsx`** in het vereiste formaat *(Newspaper boy, Sequence number, Customer number)*.

Gebruik:

1. Zorg dat Python is geïnstalleerd met de volgende packages:
   pandas, matplotlib en openpyxl.
2. Plaats newspaper problem instance.xlsx in dezelfde map als het script.
3. Run:
   ```
   python Week_6_Multi_Route_VRP_quadrant_NN_2_Opt+SA.py
   ```

   Het script gebruikt een vaste random.seed en schrijft de oplossing naar **`solution.xlsx`**.

3 Voorbereiding

We zijn begonnen met het coderen van een basismodel. Hiervoor hebben we:

- Een functie gemaakt die afstanden berekent tussen locaties en een afstandsmatrix genereert.
- Een nearest neighbor heuristiek geïmplementeerd die een eerste tour maakt.
- Een 2-opt verbeteringsprocedure toegevoegd om die eerste oplossing lokaal te verbeteren.
- Visualisaties ontwikkeld waarmee we routes grafisch kunnen weergeven, zodat we oplossingen beter begrijpen en vergelijken.

Hoewel dit model nog werkt met een TSP-benadering (één bezorger), gebruiken we het als stap naar het probleem met meerdere bezorgers. De functies die we nu hebben (afstandsmatrix, nearest neighbor, 2-opt) kunnen we hergebruiken en uitbreiden voor het scenario met 4 bezorgers.

4 Aftrap Ontwerpkeuzes

- We zijn gestart met Euclidische afstand om de functies te testen. Dit was eenvoudiger om visueel te controleren. Voor de uiteindelijke versie schakelen we over naar de Manhattan-afstand wat vereist wordt in de inleverbare opdracht.
- We gebruiken nearest neighbor als greedy aanpak, omdat deze eenvoudig een eerste oplossing oplevert.
- Voor verbetering kiezen we voor 2-opt als eerste stap, omdat dit een klassieke en effectieve manier is om tours korter te maken. Dit zal ons een eerste globaal idee opleveren van het gehele probleem, en helpt ons met het begrijpen van de volgende stappen.

5 Ontwerpkeuzes Eindcode

Na het inlezen van het Excel-bestand wordt stap voor stap een oplossing opgebouwd en verbeterd:

1. **Kwadrantindeling:**
   De coördinaten worden gesplitst bij x = 280 en y = 275, bepaald via trial-and-error voor een zo gelijk mogelijke verdeling. Elke krantenbezorger krijgt één eigen regio.
2. **Constructieve fase:**
   Binnen elk kwadrant wordt de volgorde van klanten bepaald met de Nearest Neighbor-heuristiek, zodat steeds de dichtstbijzijnde klant als volgende wordt bezocht.
3. **Lokale verbetering:**
   Voor elke route wordt de 2-Opt-procedure uitgevoerd. Deze verwijdert kruisingen en verkort de totale afstand binnen één route.
4. **Metaheuristiek – Simulated Annealing:**
   Om uit lokale minima te ontsnappen, passen we per route Simulated Annealing toe.
   Hierbij gebruiken we een begin-temperatuur $T_0 = 1000$, een afkoelingsfactor $\alpha = 0.995$ en 5000 iteraties.
   Slechtere oplossingen kunnen tijdelijk worden geaccepteerd, wat leidt tot robuustere resultaten.
5. **Fine-tuning:**
   Een laatste 2-Opt-ronde zorgt dat alle routes lokaal optimaal zijn na de metaheuristische fase.
6. **Visualisatie en export:**
   De routes worden weergegeven in een grafiek met verschillende kleuren per bezorger (inclusief depot), en geëxporteerd naar solution.xlsx voor verdere analyse.

6 Capaciteitsconstraint – Research Question 3

Als derde onderzoeksvraag hebben we onderzocht hoe verschillende capaciteitslimieten de kwaliteit en balans van de routes beïnvloeden.
We wilden weten of het beperken van het aantal adressen per bezorger leidt tot een efficiëntere of juist minder optimale verdeling.

Na het toepassen van de Nearest Neighbor-heuristiek hebben we voor elke capaciteit ook de 2-Opt-verbetering uitgevoerd, zodat de resultaten consistent zijn met onze andere experimenten.

Om te vergelijken met de standaard Nearest Neighbor-oplossing (zonder beperking), hebben we vier capaciteitsniveaus getest: 30, 35, 40 en 45 adressen per bezorger.

Capaciteit	Minimale tijdseenheid bezorger	Maximale tijdseenheid bezorger	Totale Afstand Afgelegd	Aantal bezorgers met limiet bereikt
30 adressen	936.00	2503.00	5540.00	4
35 adressen	1027.00	1907.00	5271.00	2
40 adressen	805.00	1693.00	5117.00	2
45 adressen	805.00	1835.00	4954.00	0

Analyse

Hierboven vind je de tabel met de behaalde resultaten en nodige informatie over wat te analyseren is voor deze capaciteitsconstraint.

Zoals we kunnen zien, bij een capaciteit van 45, houd je het standaard nearest-neighbour solution over die we in het begin gevonden hebben aan het begin van het proces.

Bij 40 en 35 adressen als capaciteit zitten 2 bezorgers aan hun limiet van bezorgen, maar kunnen we opmerken dat bij 35 adressen het verschil tussen minimale en maximale tijd groter is.

Dit vertelt alleen niet heel het verhaal, aangezien voor 40 adressen capaciteit, 2 bezorgers vrijwel aan of boven de 1600 zaten qua tijd, terwijl bij een capaciteit van 35 resulteerde in een groot verschil tussen 1 krantenjongen ten opzichte van de andere 3 krantenjongens, namelijk het kleinste tijdsverschil tussen de 2 hoogste tijden ongeveer 700 tijdseenheden is.

Bij een capaciteitsconstraint van 30 adressen, voldoen alle krantenjongens aan het capaciteits limiet, maar zit er een zeer groot verschil tussen de maximale tijd en de andere tijden, met een minimaal verschil van ongeveer 1500 tijdseenheden.

Conclusie

We kunnen concluderen dat hoewel in realiteit een capaciteits limiet handig zou kunnen zijn in verband met draagvermogen, het zeker geen optimale oplossing zal opleveren. Het kan zorgen voor grote verschillen in tijden gereden, hoewel de capaciteit niet altijd maximaal is.

Aangezien het doel is om tijd te minimaliseren, is een capaciteit niet de meest optimale oplossing. Wel zouden we kunnen zeggen dat bij een capaciteit van 40 adressen, de bezorgers er gemiddeld het minst werk hebben.

7 Pseudocode

1. Hoofdprogramma

Doel: Routes bouwen per kwadrant en stap voor stap verbeteren met heuristieken.

```
1. Lees klantdata in
   names, coords = read_instance(INPUT_XLSX)

2. Verdeel klanten in 4 kwadranten
   routes = build_quadrant_routes(coords, depot_idx=0, split_x=280,
split_y=275)
   → elke krantenjongen krijgt eigen regio

3. Herordenen met nearest neighbor
   routes = reorder_all_routes_nearest_neighbor(routes, coords)
   → binnen elk kwadrant steeds dichtstbijzijnde klant

4. Verbeter met 2-opt
   routes = two_opt_all_routes(routes, coords)
   → draai segmenten om als dat korter is

5. Verbeter verder met simulated annealing
   routes = sa_all_routes(routes, coords, T0=1000, alpha=0.995,
iters=5000)
   → accepteer soms slechtere routes om lokale minima te vermijden

6. Fine-tune opnieuw met 2-opt
   routes = two_opt_all_routes(routes, coords)

7. Bereken totale afstand, exporteer en visualiseer resultaat
```

2. Functie: build_quadrant_routes()

Doel: Klanten verdelen in 4 kwadranten rond een snijpunt.

```
Voor elke klant (x, y):
    Bepaal kwadrant:
        Q1: links-boven
        Q2: links-onder
        Q3: rechts-boven
        Q4: rechts-onder
Voeg depot toe aan begin van elke route
```

3. Functie reorder_route_nearest_neighbor()

Doel: De volgorde van klanten binnen een kwadrant maken.

```
Start bij depot
Zolang er nog onbezochte klanten zijn:
    kies klant met kleinste afstand (Manhattan)
    voeg toe aan route
```

4. Functie two_opt_single_route()

Doel: Lokale verbeteringen in volgorde van klanten.

```
Herhaal:
    draai segment [i:j] om
    bereken nieuwe afstand
    als korter → accepteer
Tot geen verbetering meer
```

Effect: Verwijdert kruisingen in de route en verkort zo de afstand.

5. Functie sa_single_route() (Simulated Annealing)

Doel: Route verbeteren en lokale minima vermijden door soms slechtere oplossingen toe te laten.

```
1. Start
   curr_route ← route
   curr_dist  ← afstand(curr_route)
   best_route ← curr_route
   best_dist  ← curr_dist
   T ← T0

2. Iteraties
   Voor t = 1 tot iters:
       Kies willekeurig i, j met 1 ≤ i < j < len(route)
       neighbor ← curr_route met segment [i:j] omgedraaid
       neigh_dist ← afstand(neighbor)
       Δ ← neigh_dist − curr_dist

3. Acceptatie
       Als Δ < 0:
           curr_route ← neighbor
           curr_dist ← neigh_dist
       Anders als random(0,1) < exp(−Δ / T):
           curr_route ← neighbor
           curr_dist ← neigh_dist

4. Beste route bijhouden
       Als curr_dist < best_dist:
           best_route ← curr_route
           best_dist ← curr_dist

5. Temperatuur verlagen
       T ← α × T
       Als T < min_T → stop

6. Retourneer best_route
```

Effect:

- Bij hoge temperatuur (T): meer willekeurige veranderingen → groter zoekgebied.
- Naarmate T daalt: algoritme accepteert bijna alleen verbeteringen.

8 Research Questions

RQ1 – Vergelijking heuristieken

Vraag: Hoe presteren onze metaheuristiek en onze constructieve heuristiek in termen van kwaliteit van de oplossing en rekentijd?

Aanpak:

- Run beide algoritmes (greedy en metaheuristiek) meerdere keren.
- Vergelijk de bezorgtijd van de laatste krant en de rekentijd.
- Analyseer verschillen en trek conclusies.

Antwoord:

Methode	**Beschrijving**	**Langste afstand** (time units)	**Verbetering t.o.v. vorige** (time units)	**Opmerkingen**
Alleen NN	Basisoplossing met enkel nearest neighbor	1857		Routes zijn ongelijk verdeeld; één route veel langer.
NN + 2-Opt	Lokale verbetering per route	1835	−22	Iets korter, maar verdeling tussen routes nog scheef.
Kwadranten + NN + 2-Opt	Klanten eerst verdeeld over 4 regio's	1503	−332	Grote verbetering; routes beter gebalanceerd.
Kwadranten + NN + 2-Opt + SA	Extra optimalisatie met simulated annealing	1350	−153	Beste resultaat; afhankelijk van random seed (hier seed = 46).

Conclusie:
De combinatie van kwadrantindeling, 2-opt, en simulated annealing levert de kortste en meest gebalanceerde routes op.
Nearest Neighbor alleen is snel, maar geeft een slechte verdeling; SA zorgt uiteindelijk voor de grootste kwaliteitsverbetering tegen beperkte extra rekentijd.

RQ2 – Parameterinstellingen metaheuristiek

Vraag: Welke parameterinstellingen werken het best voor onze metaheuristiek?

Aanpak:

- Test verschillende instellingen (temperatuur, aantal iteraties, cooling schedule, etc.).
- Vergelijk resultaten op basis van kwaliteit en stabiliteit van oplossingen.

Antwoord:

Parameter	**Waarde gebruikt**	**Uitleg / waarom deze keuze**
T0 (starttemperatuur)	1000	Hoge T laat in het begin ook slechtere moves toe. Dit helpt om uit slechte lokale minima te ontsnappen. Te lage T0 maakte SA “te voorzichtig” en gaf slechtere resultaten.
alpha (cooling rate)	0.995	Langzaam afkoelen. Temperatuur daalt heel geleidelijk, waardoor er genoeg tijd is om de ruimte te verkennen voordat SA ‘strak’ wordt. Bij lagere alpha (sneller koelen) zagen we minder verbetering.
iters (aantal iteraties)	5000	Genoeg iteraties per route zodat SA meerdere keren stukken kan omdraaien en echt verschil kan maken bovenop 2-opt. Veel hoger maakt het langzamer, veel lager gaf minder verbetering.
min_T (stoptemperatuur)	1e-3	We stoppen als de temperatuur praktisch “koud” is. Daarna verandert de route bijna niet meer, dus doorgaan is verspilling.

Conclusie:
De combinatie van een hoge starttemperatuur (T0 = 1000), langzame afkoeling (alpha = 0.995) en voldoende iteraties (5000) leverde de beste balans op tussen kwaliteit, stabiliteit en rekentijd.
Met deze waarden kon Simulated Annealing de routes duidelijk verbeteren bovenop 2-opt, en gaf het de meest consistente resultaten in onze experimenten.

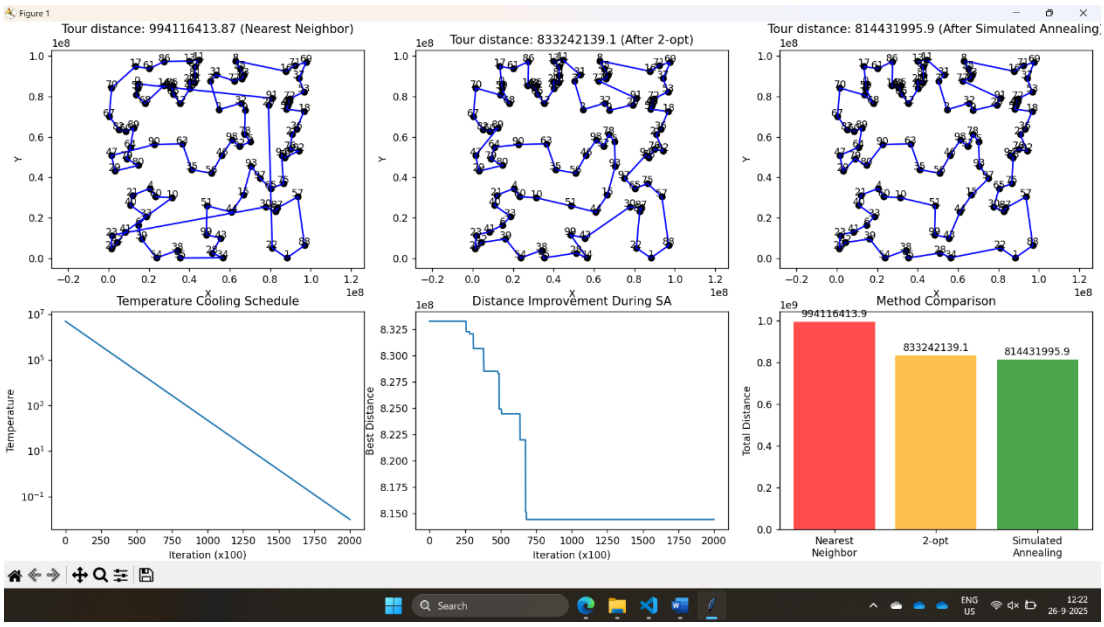
RQ 3: Capaciteitsconstraint – Research Question 3

9 Heuristieken vs SA

De figuur laat zien hoe de oplossingen verbeteren (1 route):

- **Nearest Neighbor** start met een lange tour (994M).
- **2-opt** verkort dit tot 833M.
- **Simulated Annealing** levert de beste tour op (814M), zichtbaar in de grafieken met afstandsverbetering en vergelijking.

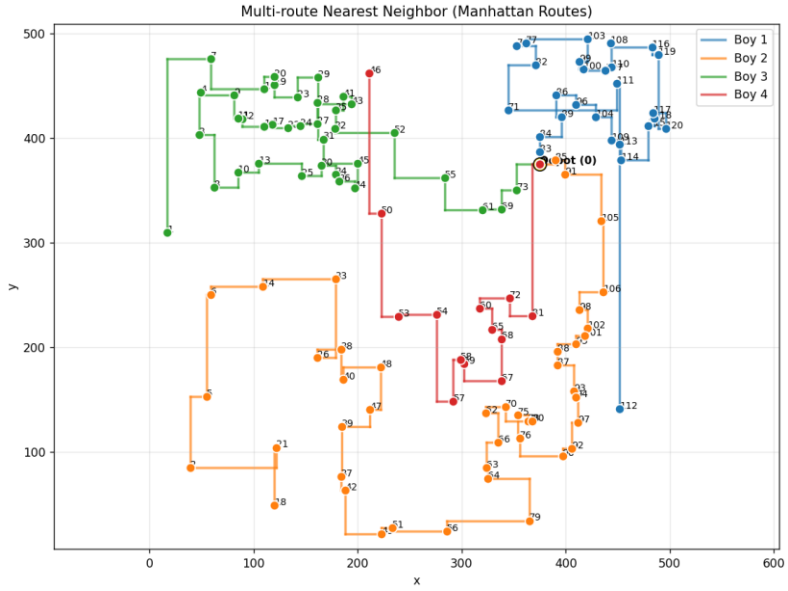
Opmerking: Dit voorbeeld is nog geen VRP (Vehicle Routing Problem), maar een eenvoudig TSP (Travelling Salesman Problem) dat ik in de les heb gemaakt. Het is wel handig om hiermee het verschil in kwaliteit tussen de heuristieken te laten zien.



Figuur 1 - TSP vergelijking heuristieken

10Visualisaties

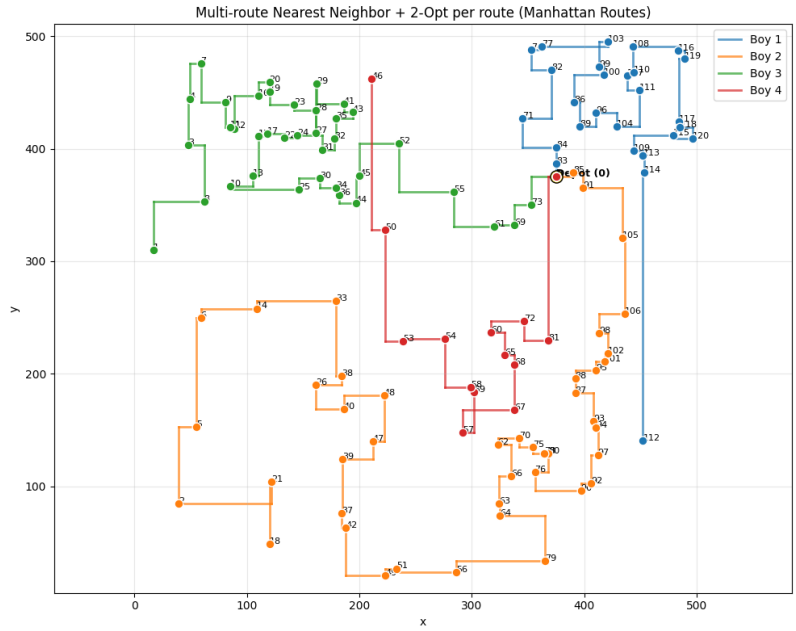
10.1 Alleen NN



Figuur 2 - VRP Nearest Neighbor

Langste afstand van 4 routes = 1857

10.2 NN & 2-Opt

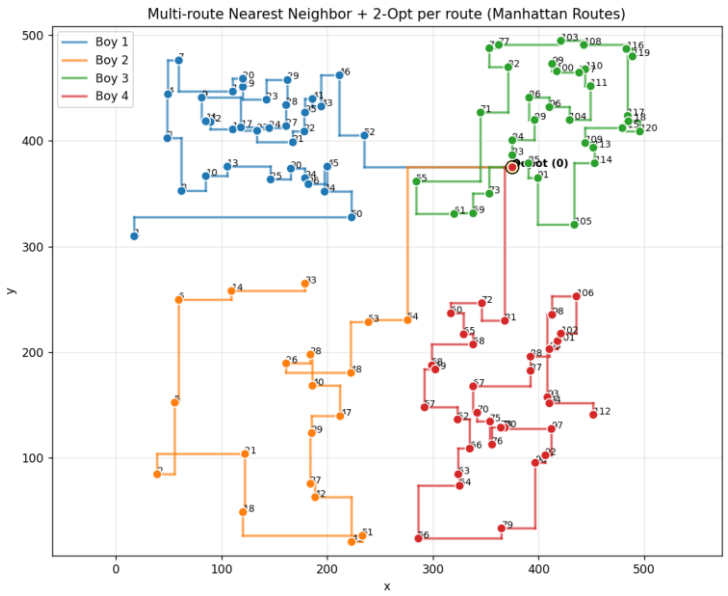


Figuur 3 - VRP Nearest Neighbor and 2-Opt

Langste afstand van 4 routes = 1835

- Dit is al een verbetering, maar we zien nog steeds dat de 2e route veel meer klanten bezoekt dan de andere 3. Dit hebben we opgelost door de map op te delen in kwadranten.

10.3 Kwadranten + NN & 2-Opt

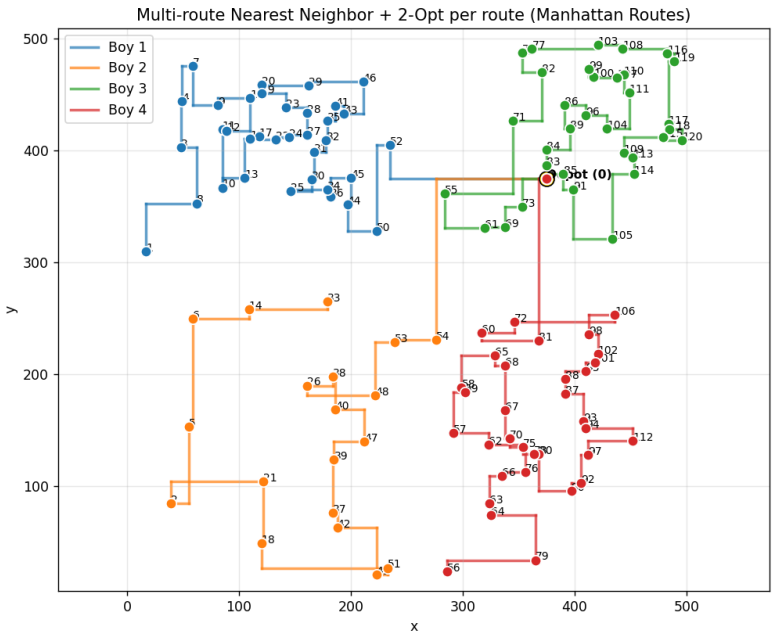


Figuur 4 - VRP Kwadranten + (NN & 2-Opt)

Langste afstand van 4 routes = 1503

- Dit is al een grote verbetering, hier kan je zien dat de klanten beter verdeeld zijn over de 4 krantenjongens waardoor de langste afstand van 4 routes ook korter wordt.
- Nu kunnen we de individuele routes verder optimaliseren aan de hand van SA.

10.4 Kwadranten + NN & 2-Opt & SA



Figuur 5 - VRP Kwadranten + (NN & 2-Opt & SA)

Langste afstand van 4 routes = 1350

- Door simulated annealing is de oplossing nog beter, het resultaat is hier wel afhankelijk van de random.seed().
- Random.seed = 46

11 Output

Hieronder is de output van het Python-script weergegeven:

```
Totale afstand van alle routes: 5201.00
  Route 1: 1337.00 (met 35 klanten)
  Route 2: 1350.00 (met 19 klanten)
  Route 3: 1196.00 (met 33 klanten)
  Route 4: 1318.00 (met 33 klanten)
Gereed. Oplossing weggeschreven naar: solution.xlsx
```

De totale afstand over alle routes bedraagt 5201 eenheden.
De verschillen in route-lengte zijn beperkt, wat aangeeft dat de kwadrantmethode in combinatie met SA effectief balans creëert.

Zoals je ziet wordt de oplossing ook weggeschreven naar een Excel bestand:

	A	B	C
1	Newspaper boy	Sequence number	Customer number
2	1	1	83
3	1	2	84
4	1	3	71
5	1	4	82
6	1	5	74
7	1	6	77
8	1	7	103
9	1	8	99
10	1	9	100
11	1	10	86
12	1	11	89
13	1	12	96
14	1	13	104
15	1	14	111
16	1	15	107
17	1	16	110
18	1	17	108
19	1	18	116
20	1	19	119
21	1	20	117
22	1	21	118
23	1	22	120
24	1	23	115
25	1	24	109
26	1	25	113
27	1	26	114
28	1	27	112
29	2	1	85

Figuur 6 - Solution.xlsx (output)

12 Conclusie en reflectie

De combinatie van kwadranten, Nearest Neighbor, 2-Opt en Simulated Annealing levert de kortste en meest gebalanceerde oplossing.
Door verschillende iteraties en parameterinstellingen te testen, hebben we inzicht gekregen in de werking en stabiliteit van onze metaheuristiek.
Een volgende stap zou zijn om inter-route verbeteringen (zoals relocate-moves) te onderzoeken, om de balans tussen routes verder te optimaliseren.

13 Figurentabel