Parallel ALNS met Anytime Performance voor Real-Time Oceaanreiniging

Bram Leisink

December 2025

1 Probleemformulering

Het Ocean Cleanup-probleem [1] wordt geformaliseerd als een *prize-collecting vehicle routing problem* met tijdsvensters. Gegeven een grid $G = (V, E)$ met plastic-distributie $p : V \to \mathbb{Z}^+$, asymmetrische bewegingskosten $c : E \to \{5, 7\}$, en constraints $\mathcal{C}$ (5 dagen, 50 km/dag, unieke opname), zoeken we een route $R = (v_1, \ldots, v_n)$ die de winst maximaliseert:

$$\max_{R \in \mathcal{F}} \sum_{v \in R} p(v) \quad \text{s.t.} \quad R \text{ voldoet aan } \mathcal{C} \tag{1}$$

Met drie richtingskeuzes per stap en $n \approx 45$, omvat de zoekruimte $\mathcal{O}(3^n) \approx 10^{21}$ configuraties, wat exacte enumeratie onmogelijk maakt.

2 Methodologie

2.1 Adaptive Large Neighborhood Search

ALNS optimaliseert routes door iteratieve destructie en reconstructie. Het proces selecteert een destroy-operator $d_i \in \mathcal{D}$ en een repair-operator $r_j \in \mathcal{R}$ op basis van prestatie-gewichten $\mathbf{w}_t$. Hierin is tevens *simulated annealing* verwerkt.

$$s_{t+1} = r_j(d_i(s_t)) \quad \text{met } (i, j) \sim \text{RouletteWheel}(\mathbf{w}_t) \tag{2}$$

$$\text{accept}(s_{t+1}) \sim \exp\left(\frac{f(s_{t+1}) - f(s_t)}{T_t}\right), \quad T_t = T_0 \cdot \alpha^t \tag{3}$$

2.2 Operator-Implementatie

Destroy-operators:

- d_1 (Segment): Verwijdert aaneengesloten segmenten (20–40%), effectief voor dag-herplanning.
- d_2 (Random): Stochastische verwijdering (15–30%) ter fragmentatie.
- d_3 (Suffix): Truncatie van het einde, behoudt sterke start-segmenten.

Repair-operators:

- r_1 (Greedy): Kiest lokaal optimum ($\arg\max_m p(m)$).
- r_2 (Softmax): Probabilistisch $P(m) \propto \exp(p(m)/\tau)$ met $\tau = 1.5$.
- r_3 (Distance-biased): Softmax met afstandsboete om clustering te voorkomen.

[1] `https://www.uva.nl/programmas/bachelors/business-analytics/studieprogramma/data-challenge.html`

2.3 Parametertuning en Optimalisatie

Een parameter-sweep over 1200 configuraties (variërende seeds, gewichten, SA-schema's) identificeerde de optimale setting: gewichten $[30, 8, 3, 1]$ en SA-parameters $(50000, 1, 0.9993)$. Computationele last wordt gereduceerd door:

1. **Lazy evaluation**: Caching van $(s, f(s))$ voorkomt herberekening; update-complexiteit daalt naar $\mathcal{O}(|\Delta s|)$.
2. **Quick-mode**: Eliminatie van niet-essentiële checks reduceert evaluatietijd van 800 μs naar 250 μs.

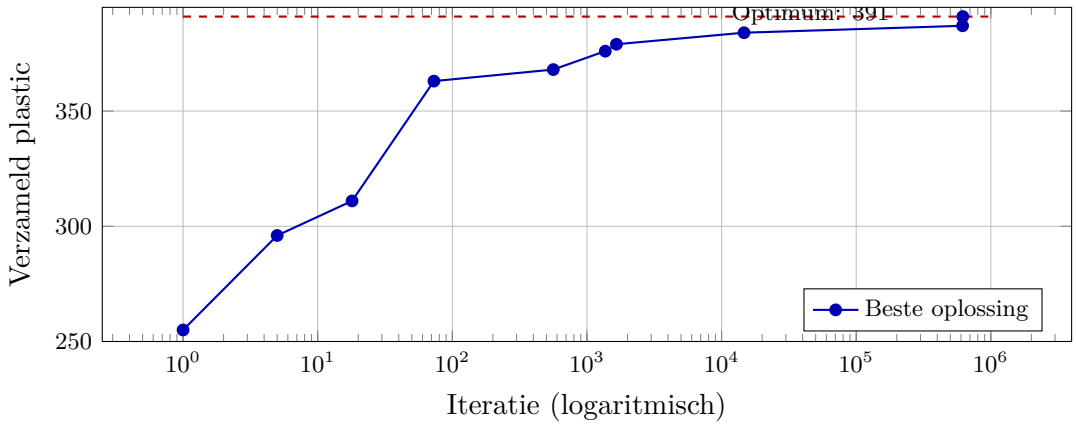
2.4 Parallelle Multi-Start Strategie

In plaats van één lange run, draaien $k = 8$ onafhankelijke instanties parallel. De globale oplossing is $s^* = \arg\max_i f(s_i^{\text{best}})$. Dit verhoogt de kans op het vinden van het globale optimum lineair met de rekenkracht.

3 Resultaten

3.1 Convergentie-Analyse

Figuur 1 toont het verloop van de beste run.



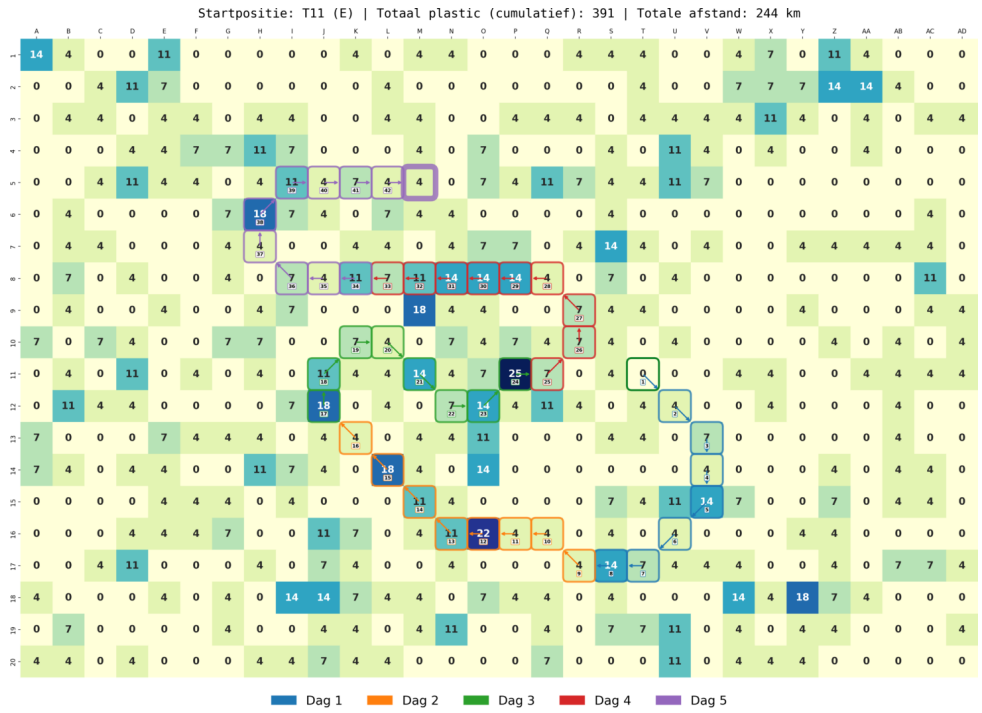
Figuur 1: Convergentie: snelle stijging naar 363 (<1s), gevolgd door asymptotische groei naar 391.

Drie fases zijn zichtbaar: (1) *Rapid exploration* tot 368 punten, (2) *Intensificatie* waarbij subroutes worden verfijnd tot 384, en (3) *Asymptotische fase* waarin zeldzame verbeteringen leiden naar het optimum van 391. Deze route is grafisch weergegeven in figuur 2.

3.2 Robuustheid en Validatie

Over 100 onafhankelijke runs blijkt de anytime-eigenschap van het algoritme (Tabel 1): op elk moment kan het algoritme gestopt worden met een geldige oplossing, waarbij de kwaliteit monotoon stijgt met beschikbare rekentijd. Uit cross-validatie met zwaardere settings (10^8 iteraties) bleek geen score hoger dan 391, wat suggereert dat dit het globale optimum is.

De finale route omvat 244 km (97.6% efficiëntie) en bezoekt 43 unieke cellen. Tests van deze methode op grids tot 200×200 tonen aan dat de methode lineair schaalbaar blijft.



Figuur 2: Geoptimaliseerde route met een score van 391.

Iteraties	**Tijd**	$P(\geq 379)$	$P(\geq 384)$	$P(= 391)$
2,000	0.2 s	0.82	0.13	0.00
50,000	5.0 s	1.00	0.91	0.00
1,000,000	1.7 m	1.00	1.00	1.00

Tabel 1: Probabilistische prestaties tonen betrouwbare convergentie bij langere rekentijd.

4 Innovatieve Toepassingen

Een belangrijk kenmerk van ALNS is de mogelijkheid tot progressive optimization tijdens operatie. Wanneer een schip uitvaart met een initiële route kan het algoritme parallel blijven draaien en verbeterde routes communiceren. Dit wordt mogelijk gemaakt door na een werkelijke verplaatsing de verplaatsingsbudetten te verbeteren. Hierbij helpt het om een heuristiek toe te voegen, waarbij de prioriteit ligt op het begin van de route.

Ten tweede biedt de inherente parallellisatie van de ALNS-structuur een robuuste basis voor *Multi-Vessel Coördinatie*. Het schaalt effectief naar complexe scenario's door het coördineren van een vloot (meerdere schoonmaakschepen). Dit gebeurt door het gebruik van gedeelde 'visited-sets' en dynamische afstandslimieten tussen schepen, waardoor overlap in de verzamelde gebieden wordt geminimaliseerd en de collectieve efficiëntie van de operatie wordt gemaximaliseerd.

5 Conclusie

Deze studie presenteert een parallelle ALNS-implementatie die consistent de optimale score van 391 behaalt. De methode combineert:

1. **Anytime performance**: Bruikbare routes binnen 0.2s, near-optimaal binnen 5s.
2. **Schaalbaarheid**: Geschikt voor grote grids door lokale operaties en parallellisatie.
3. **Flexibiliteit**: Ondersteunt real-time herplanning tijdens de vaart.