Applied Routing

VEHICLE ROUTING PROBLEMS IN PRACTICE

Solving real routing problems, in minutes.

An introduction to VRPs, heuristics, and PyVRP, a state-of-the-art open-source solver, with one case from the field.

Niels Wouda

ABOUT ME

Niels Wouda

I build optimisation software that helps organisations plan their delivery routes. I created **PyVRP** during my PhD at the University of Groningen and postdoc at the Erasmus University, and now develop it further at our start-up, **Applied Routing**.

PhD, University of Groningen

Founder, Applied Routing

ABOUT THIS TALK

Open-source vehicle routing

01

The problem

What VRPs are, from TSP to real-world constraints.

02

The solver

PyVRP: why it exists, and a bit about how it works.

03

The practice

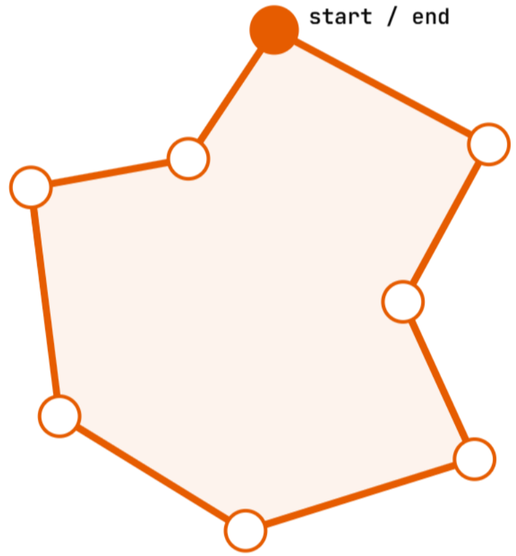
A case about waste collection in Groningen.

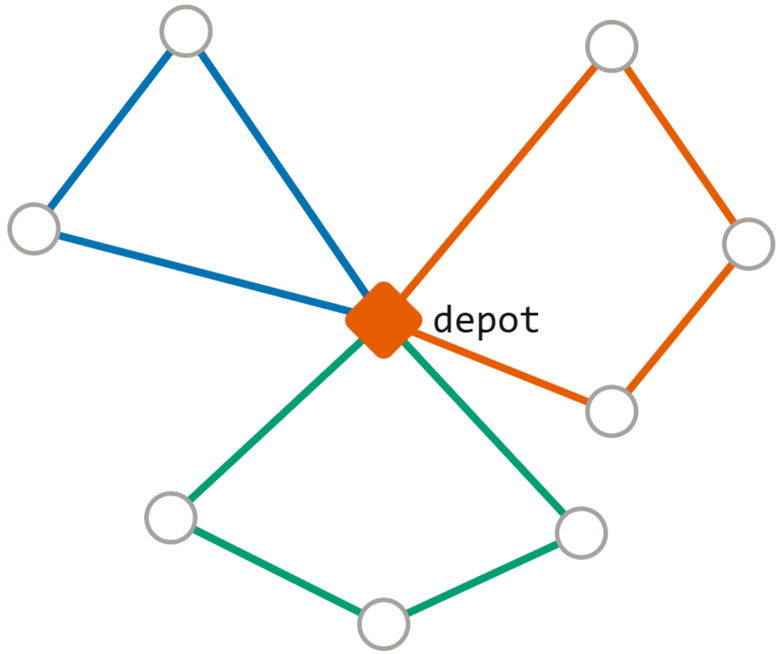
The takeaway: open-source optimisation software can unlock real, measurable improvements for many organisations, with sometimes up to 40% cost and time reductions.

The travelling salesman problem.

One vehicle needs to visit every stop exactly once. Find the **shortest possible route**.

This is already NP-hard, but not hopeless. ☺





The vehicle routing problem.

Now a **fleet** of vehicles operates from a depot. The goal is to assign each stop to one of the vehicles, at minimum cost.

TSP generalised to many vehicles

This is the problem countless organisations solve every single day, across package delivery, waste collection, or field services.

THE PROBLEM · 3 OF 3

Reality adds more constraints.

The textbook VRP is only the start. Real operations impose additional constraints that make the problem richer and harder.

Time windows

Each stop must be served within opening hours.

Capacities

Vehicles can only carry so much weight or volume.

Multiple depots

Fleets can start and end from several locations.

Heterogeneous fleet

Different vehicle sizes, speeds and costs.

Driver breaks

Mandatory rest, compliant with EU/US/etc. regulations.

Optional stops

Use rewards to prioritise certain stops over others.

Applied Routing

THE SOLVER

Meet PyVRP.

A state-of-the-art, open-source vehicle routing solver. Free, MIT-licensed, and used by routing teams worldwide.

```
$ pip install pyvrp
```

pyvrp.org · github.com/PyVRP/PyVRP

THE SOLVER

Why build another routing solver?

Commercial solvers are expensive black boxes.

Hidden behind licences, binaries you cannot inspect, or adapt to handle unusual constraints.

Research code doesn't ship.

Academic prototypes are fast but fragile: hard to install, extend, or run in production.

Existing open source is not good enough

OR-Tools, jsprit, VROOM are good software projects, but do not offer performance that is competitive with current commercial and academic solvers.

PyVRP does both.

State-of-the-art performance, packaged as clean, open, well-tested software anyone can use and extend.

THE SOLVER

A bit of history.

2021

Precursors

Started as a research project for VRPs with time windows, based on other academic software.

2022

A name!

The research project finishes 1st place in the 12th DIMACS challenge and EURO/NeurIPS 2022 challenge. “PyVRP” is coined.

2023

Public release

Released openly with multiple vehicle types, optional stops and the high-level modelling API.

2024–25

Real-world reach

Multiple depots, profiles, load dimensions, and mid-shift reloading.

2026

Scaling up

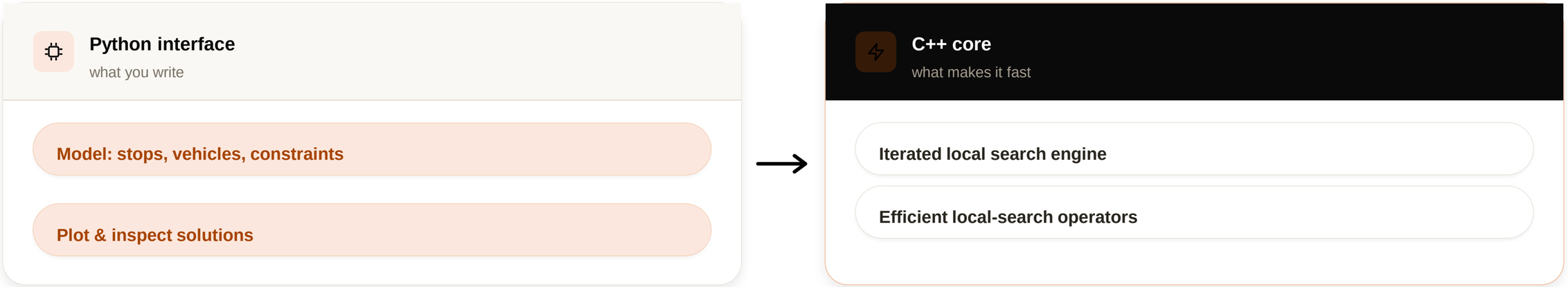
Pickup and delivery, regulatory break planning (enterprise).

Today PyVRP is maintained by Applied Routing and forms the engine behind our enterprise product.

THE SOLVER

How it fits together.

A friendly Python front end you work in, over a fast C++ core that does the heavy lifting.



The result: **speed**, with the ergonomics of a normal Python package.

THE SOLVER

Model-and-run

```
COORDS = [(0, 0), (0, 1), (2.5, 0)]
DELIVERIES = [4, 5]
```

```
m = pyvrp.Model()
m.add_vehicle_type(2, capacity=10)
m.add_depot(location=m.add_location(x=COORDS[0][0], y=COORDS[0][1]))

m.add_client(
    location=m.add_location(x=COORDS[1][0], y=COORDS[1][1]),
    delivery=DELIVERIES[0],
)

m.add_client(
    location=m.add_location(x=COORDS[2][0], y=COORDS[2][1]),
    delivery=DELIVERIES[1],
)

for idx_frm, frm in enumerate(m.locations):
    for idx_to, to in enumerate(m.locations):
        if idx_frm != idx_to:
            m.add_edge(frm, to, distance=1)  # unit distances

res = m.solve(stop=pyvrp.stop.MaxRuntime(1))  # one second of runtime
```

A handful of lines describes a full routing problem. The same model scales from toy examples to instances with **10,000+ stops**.

- **Pickup & delivery**
- **Multiple time windows**
- **Prize-collecting**
- **Reloading mid-shift**
- **Driver breaks**
- **Multiple depots**

< 60s to a near-optimal answer on real-world-sized instances.

CASE STUDY

Waste collection in Groningen.

A research collaboration with the municipality.

240,000 residents

850 container clusters

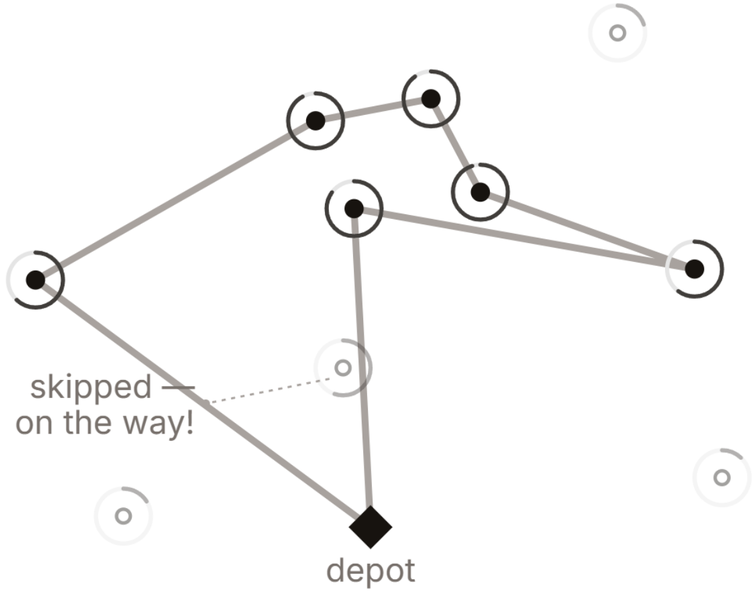
4 vehicles · 1 depot

GRONINGEN · CLUSTERS AND SCALE

Empty clusters before they overflow.

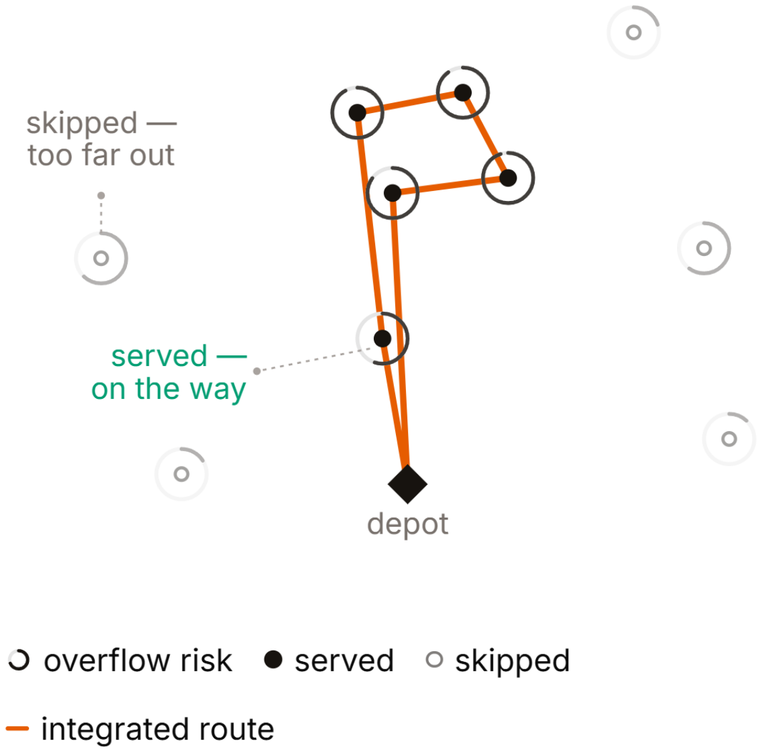
Residents drop waste in underground containers. Every morning the city must route trucks to empty the clusters most likely to overflow. **Fill levels aren't measured**, only deposit counts.

The catch: the baseline first *selects* ~250 clusters, then *routes* them, in two separate steps. Selection and routing interact, but the baseline does not capture that interaction.



overflow risk ● selected ○ skipped

— baseline route



Integrate selection and routing.

We assign each cluster a **prize** that captures its overflow urgency, and then solve a single *prize-collecting VRP* with time windows and driver breaks.

PyVRP weighs each prize against the routing cost of the detour. It serves high-risk clusters *and* convenient on-the-way ones, while skipping costly detours.

Solving takes about 60s.

GRONINGEN · THE RESULTS

Same service level. 40%+ less driving.

Averaged over year-long simulations on the city's own data (nearly two million deposits from Q1 2023).

−41%

Daily driving

251 km → 146 km

−44%

Route duration

6.8 h → 3.8 h

99.9%

Service level

unchanged

WRAPPING UP

Open optimisation, real impact.

VRPs are everywhere

From TSP to real routes with time windows, capacities and prizes — many operations solve them daily.

PyVRP is open and strong

Award-winning performance, free to use, and built to scale.

Better algorithms cut costs

And save a lot of time.

Try it yourself.

```
$ pip install pyvrp
```

pyvrp.org · github.com/PyVRP/PyVRP

appliedrouting.com

Applied Routing

Questions?

Thank you!

Niels Wouda

niels@appliedrouting.com