Stefan Røpke

AIROYoung 2026

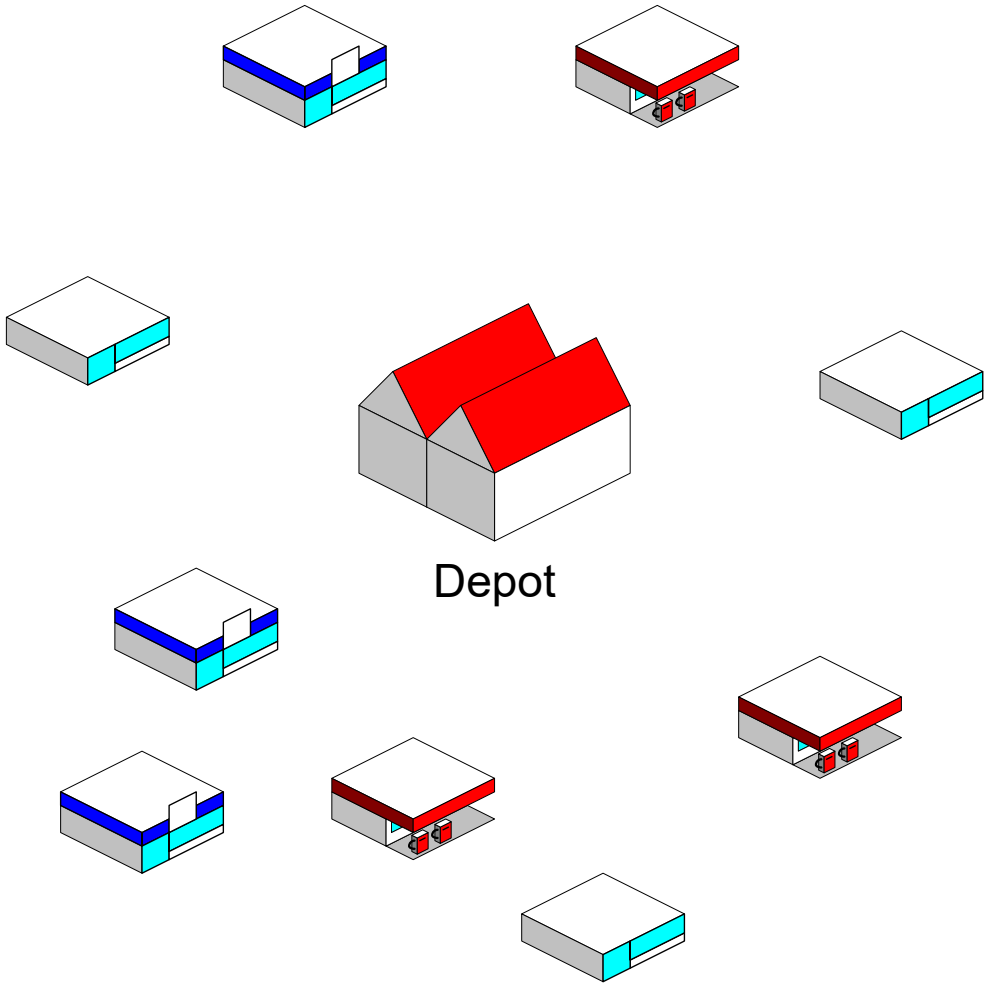
Vehicle routing problems

Agenda

- Short introduction to Vehicle routing problems
- State of the art in exact and heuristic solution methods
- ALNS
- Letting LLMs write ALNS heuristics

Vehicle routing problems

Our task: Suply supermarkets and gas stations with goods from a common warehouse using three trucks

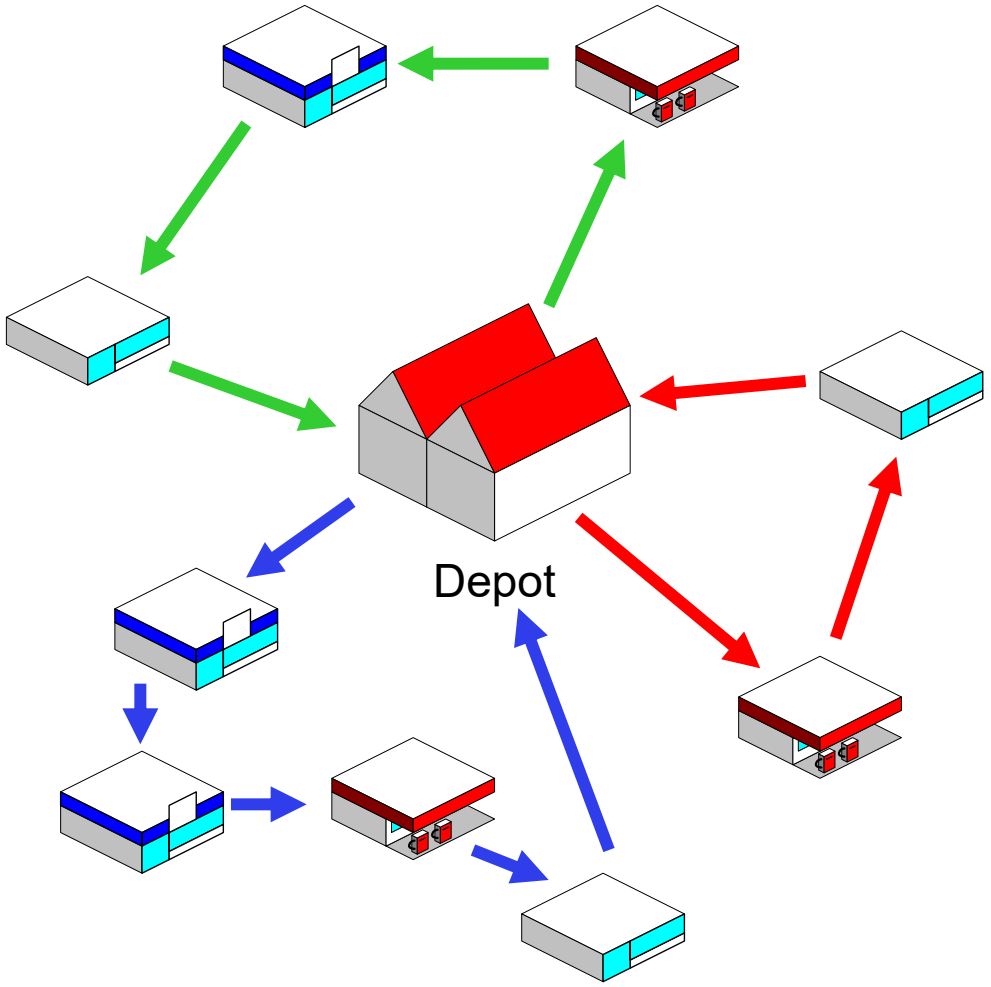


Vehicle routing problems

Our task: Suply supermarkets and gas stations with goods from a common warehouse using three trucks.

Terminology:

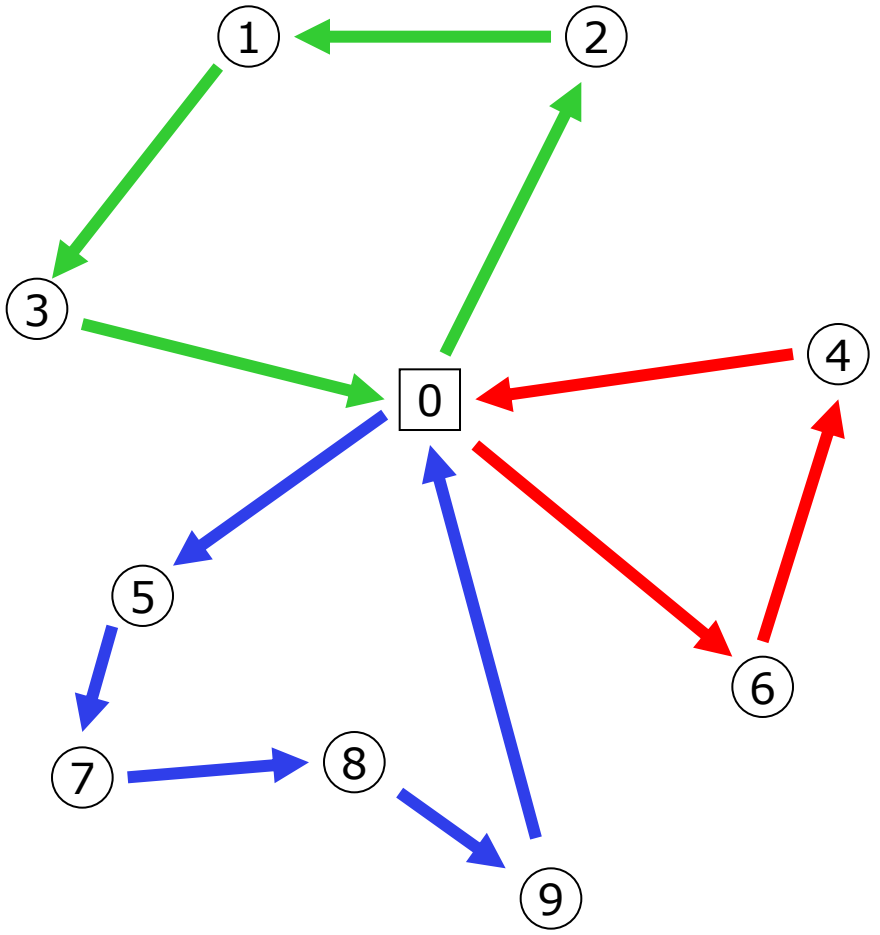
- Depot
- Customer
- Route
- Vehicle



Vehicle routing problems

Capacitated vehicle routing problem (**CVRP**):

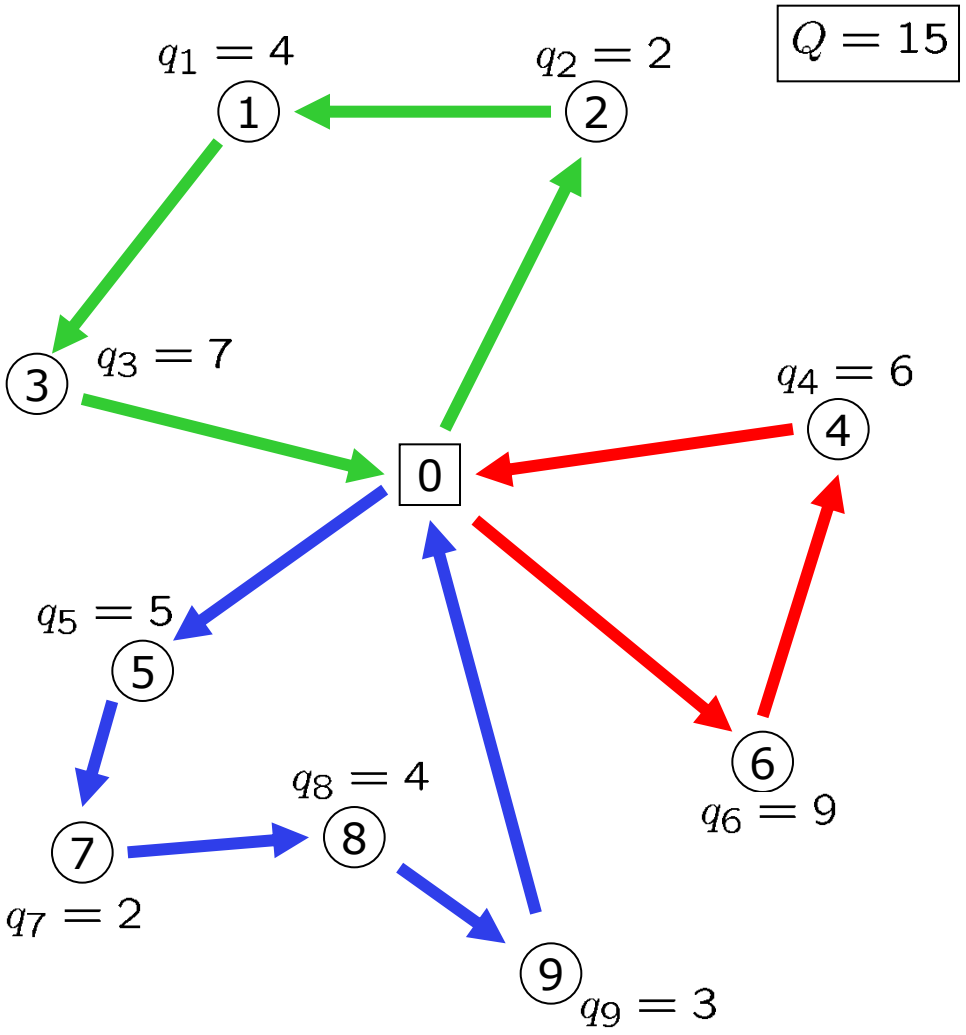
- Visit all customers once.
- Each customer has a demand q_i.
- Each vehicle has a capacity Q.
- Sum of demands on a route ≤ vehicle capacity
- Number of routes should be equal to K.
- **Or:** Number of routes is free
- **Objective:** Minimize cost of routes.



Vehicle routing problems

Capacitated vehicle routing problem (**CVRP**):

- Visit all customers once.
- Each customer has a demand q_i.
- Each vehicle has a capacity Q.
- Sum of demands on a route ≤ vehicle capacity
- Number of routes should be equal to K.
- **Or:** Number of routes is free
- **Objective:** Minimize cost of routes.



Vehicle routing problems

- Vehicle routing problems occur frequently in practice.
- Large savings can be realized by using computerized optimization of the distribution planning. 5-20%. (?)
- Many companies use some kind of software to control their fleet of vehicles.
- Many variants of the VRP motivated from real life applications.

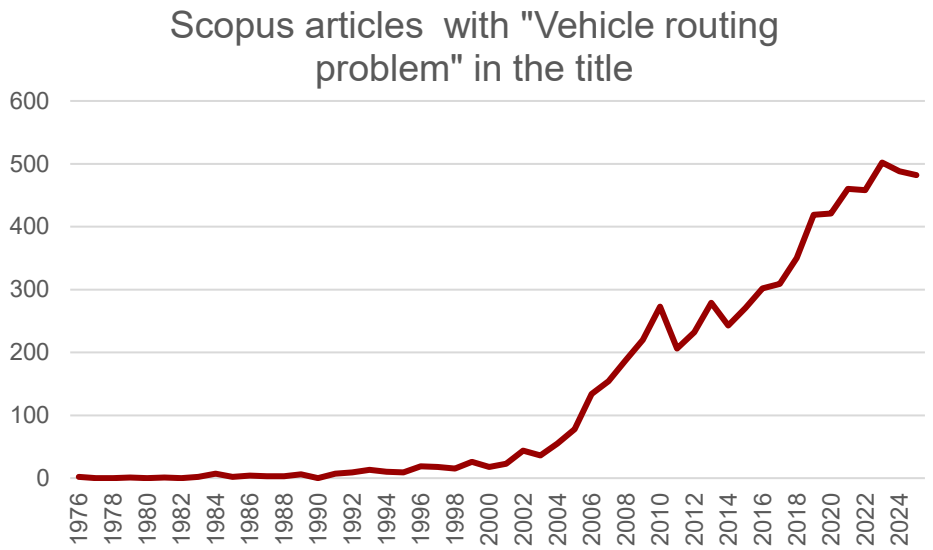
VRP Variants

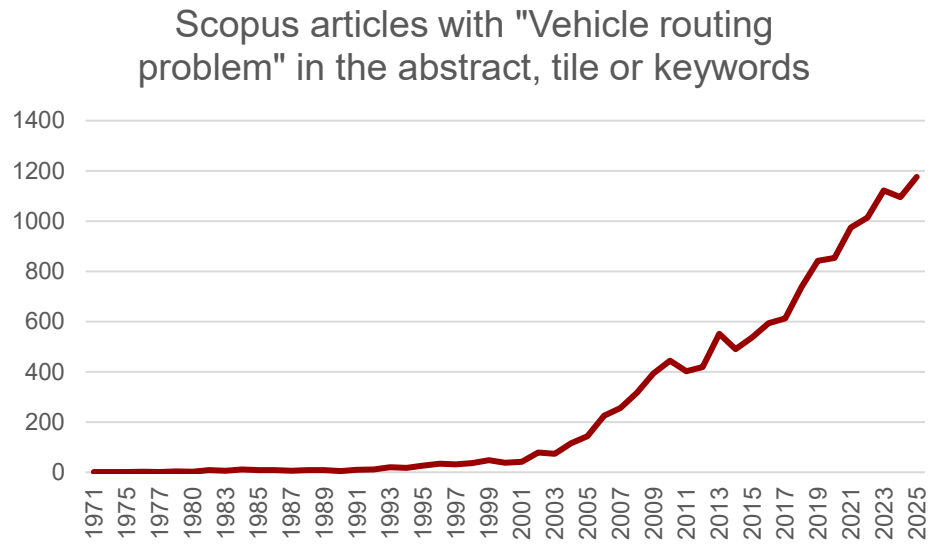
- **Deterministic / stochastic / dynamic problems**

- VRP with time windows
- Pickup and delivery problems
- Multi-depot problems
- 2-echelon problems
- VRP with drones
- VRP with robots
- Electric VRPs
- …

Popular subject

- In 1990 there were zero articles with ”vehicle routing problem” in the title published that year
- In 2002 there were 44
- In 2025 there were 482.





Three main model types

Vehicle routing problems – model type 1

- N: number of customers. Node 0 is the depot, $1, \ldots, N$ are customers
- c_{ij}: cost of going from node i to node j
- q_i: demand of customer i
- Q: vehicle capacity

Variables

- $x_{ij} \in \{0, 1\}$: 1 if arc (i, j) is used, 0 otherwise
- u_i : counter, keeps track of vehicle load

$$\min \sum_{i=0}^{N} \sum_{j=0}^{N} c_{ij} x_{ij}$$

subject. to

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall j = 1, \ldots, N$$

$$\sum_{i=1}^{N} x_{ji} = 1 \qquad \forall j = 1, \ldots, N$$

$$u_i + q_j \leq u_j + Q(1 - x_{ij}) \qquad \forall i = 0, \ldots, N, j = 1 \ldots, N$$

$$\sum_{i=0}^{N} x_{ii} = 0$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 0, \ldots, N, j = 0, \ldots, N$$

$$0 \leq u_i \leq Q \qquad \forall i = 0, \ldots, N$$

Vehicle routing problems – model type 2

- N: number of customers. Node 0 is the depot, $1, \ldots, N$ are customers
- c_{ij}: cost of going from node i to node j
- q_i: demand of customer i. $q_i > 0$
- Q: vehicle capacity

Variables

- $x_{ij} \in \{0, 1\}$: 1 if arc (i, j) is used, 0 otherwise

$$\min \sum_{i=0}^{N} \sum_{j=0}^{N} c_{ij} x_{ij}$$

subject. to

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall j = 1, \ldots, N$$

$$\sum_{i=1}^{N} x_{ji} = 1 \qquad \forall j = 1, \ldots, N$$

$$\sum_{i=0}^{N} x_{ii} = 0$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq \left\lceil \frac{\sum_{i \in S} q_i}{Q} \right\rceil \qquad \forall S \subseteq \{1, \ldots, N\}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 0, \ldots, N, j = 0, \ldots, N$$

Vehicle routing problems – model type 3

- $V_c = \{1, \ldots, N\}$. Set of customers.
- Ω : set of all feasible CVRP routes.
- c_p : cost of route $p \in \Omega$.
- a_{ip} : constant that is 1 if customer i is visited by route p and 0 othervise.
- y_p : binary variable that is 1 if and only if path $p \in \Omega$ is used in the solution.

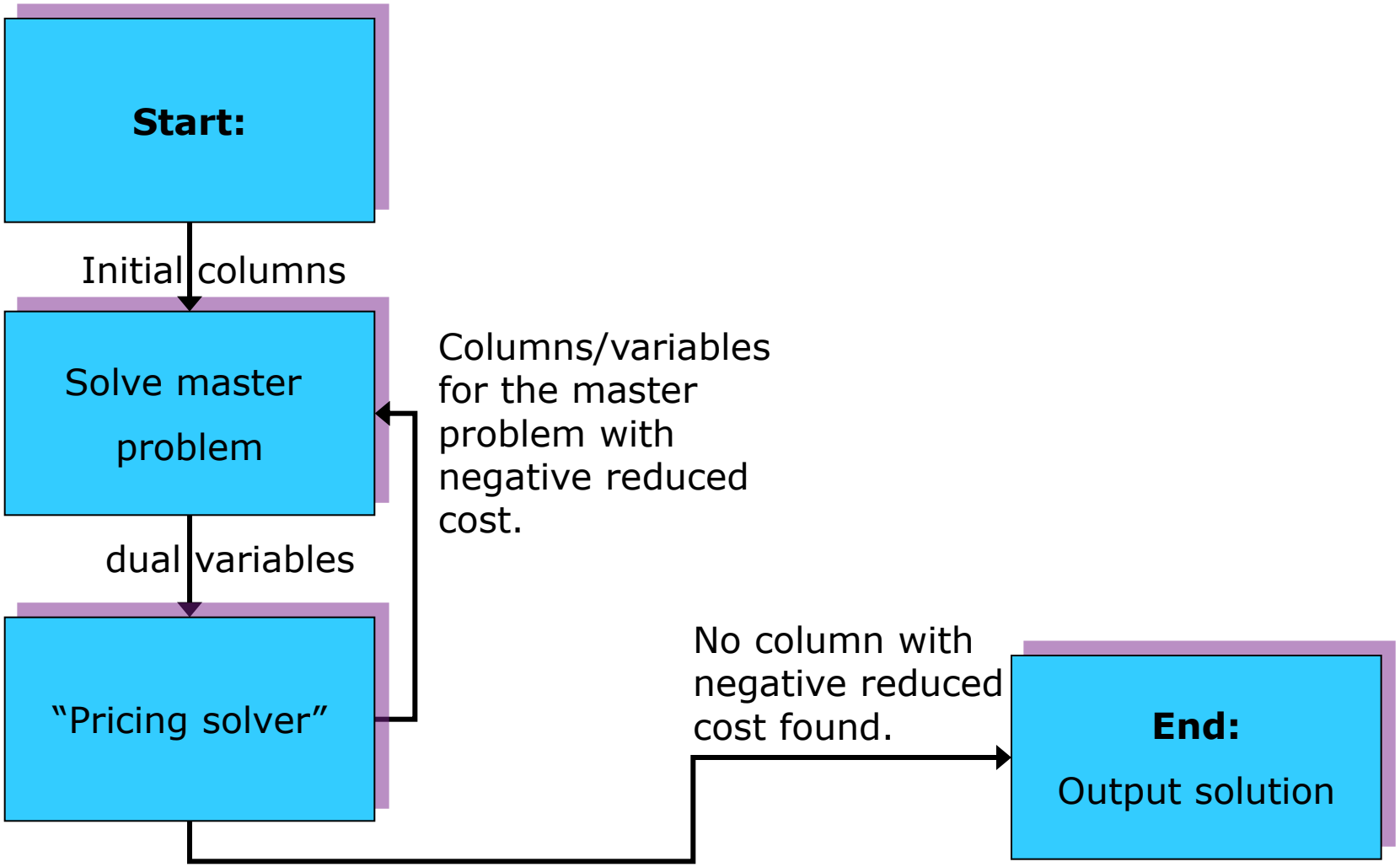
$$\min \sum_{p \in \Omega} c_p y_p$$

Subject to:

$$\sum_{p \in \Omega} a_{ip} y_p = 1 \qquad \forall i \in V_c$$

$$y_p \in \{0, 1\} \qquad \forall p \in \Omega$$

Column generation (minimization) The algorithm

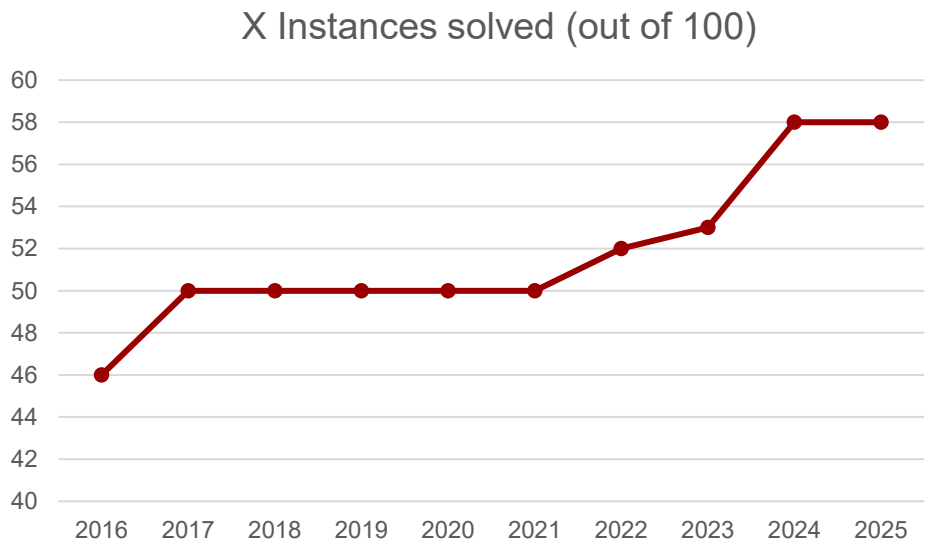


Some key papers – exact CVRP

- Baldacci, Roberto, Nicos Christofides, and Aristide Mingozzi. "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts." *Mathematical Programming* 115, no. 2 (2008): 351-385. Up to 121 customers
- Baldacci, Roberto, Aristide Mingozzi, and Roberto Roberti. "New route relaxation and pricing strategies for the vehicle routing problem." *Operations research* 59, no. 5 (2011): 1269-1283. Up to 121 customers
- Pecin, Diego, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. "Improved branch-cut-and-price for capacitated vehicle routing." *Mathematical Programming Computation* 9, no. 1 (2017): 61-100. Up to 360 customers
- Pessoa, Artur, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. "A generic exact solver for vehicle routing and related problems." *Mathematical Programming* 183, no. 1 (2020): 483-523. Up to 548 customers

Solving CVRP to optimality

- X instances
- Uchoa, Eduardo, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. "New benchmark instances for the capacitated vehicle routing problem." *European Journal of Operational Research* 257, no. 3 (2017): 845-858.
- 100 to 1000 customers (more small than large instances)



Heuristics

Results on new XL instances (1000-10000 customers)

2 hour time limit

	Mean % gap
AILS-II	0.07
FILO	0.25
FILO2	0.21
KGLS	1.00
HGS-CVRP	1.36
SISRs	0.50
LKH-3	5.47
OR-tools	6.81

AILS-II: An adaptive iterated local search heuristic for the large-scale capacitated vehicle routing problem

VR Máximo, JF Cordeau, MCV Nascimento

FILO/2 A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems

L Accorsi, D Vigo

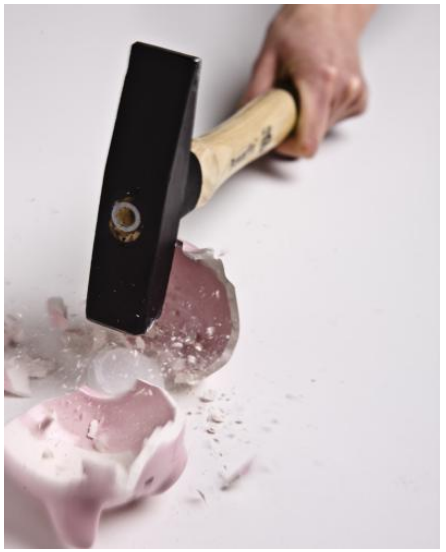
Source: Queiroga, Eduardo, Rafael Martinelli, Anand Subramanian, Eduardo Uchoa, and Thibaut Vidal. "The XL Instances for the Capacitated Vehicle Routing Problem." *arXiv preprint arXiv:2601.11467* (2026).

Large neighborhood search (LNS)

- Formalized by P. Shaw in 1998 and was originally used within the constraint programming community.

1. Input: an initial solution
2. Repeat until stop-criterion:
   a. Destroy part of solution
   b. Repair solution
   c. Accept or reject new solution
3. Return best solution encountered

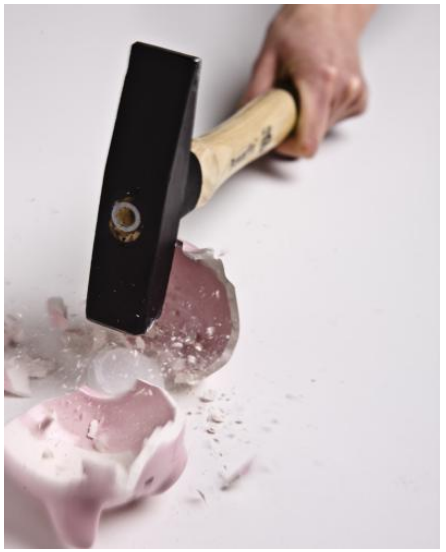
Shaw only accepted improving solutions



Large neighborhood search (LNS)

- Formalized by P. Shaw in 1998 and was originally used within the constraint programming community.

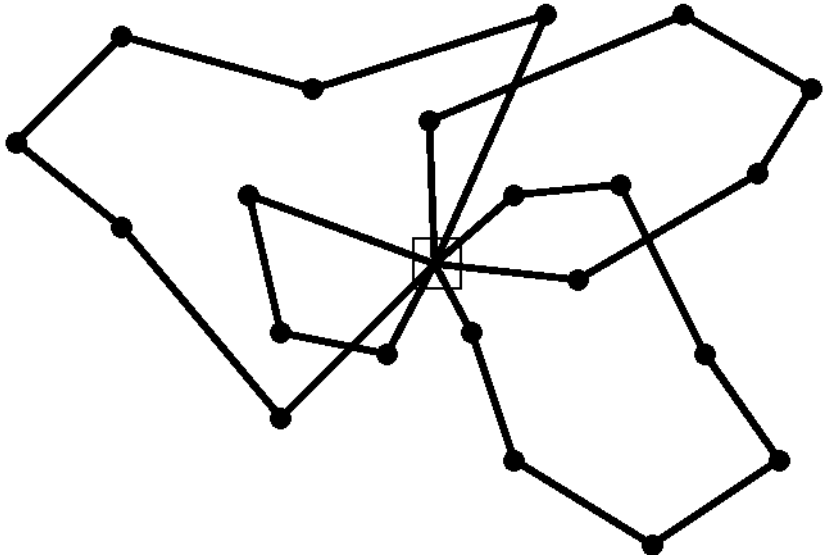
1. Input: an initial solution
2. Repeat until stop-criterion:
   a. Destroy part of solution
   b. Repair solution
   c. Accept or reject new solution
3. Return best solution encountered



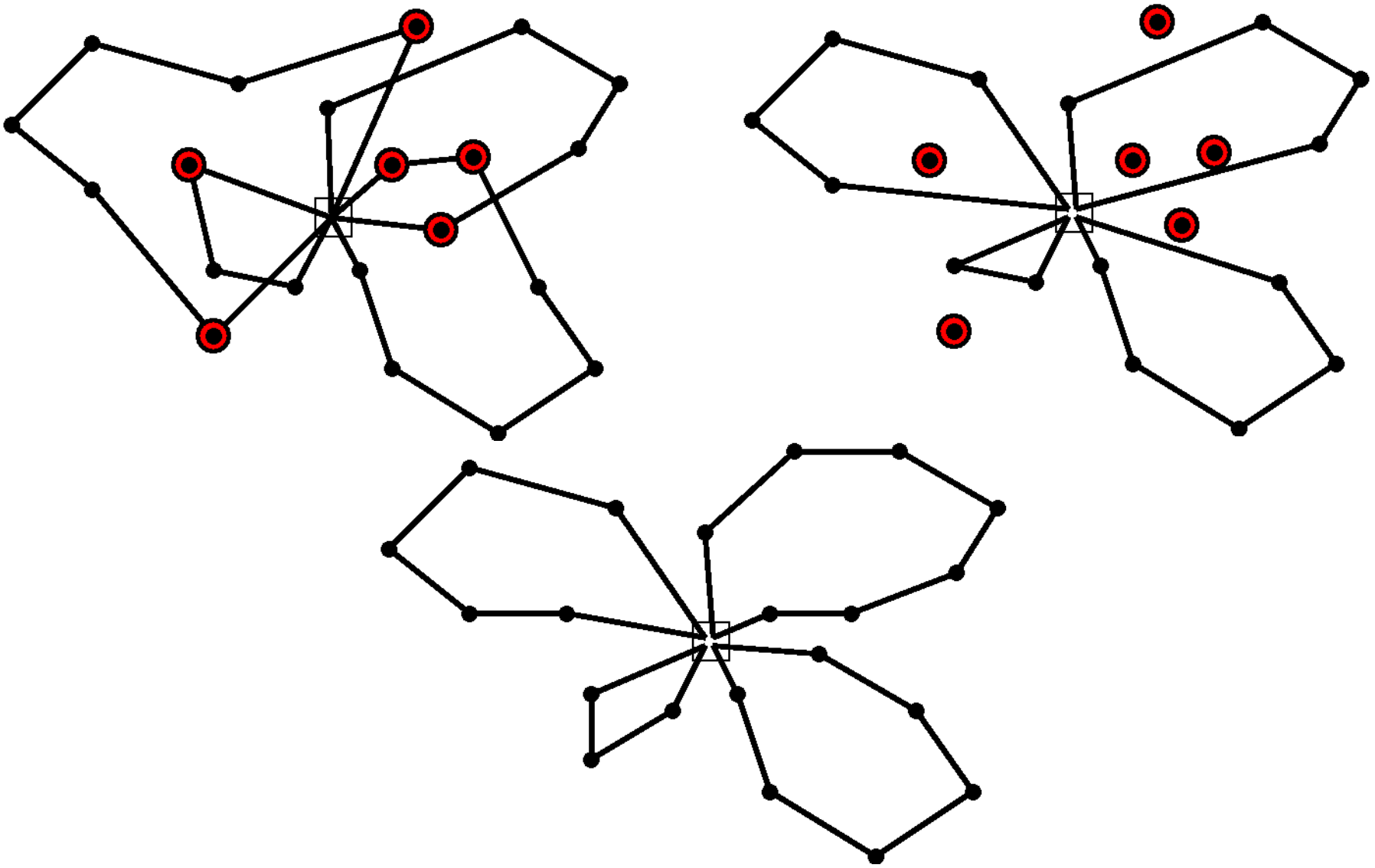
Algorithm keeps track of:

- Current solution
- Best solution
- Temporary solution

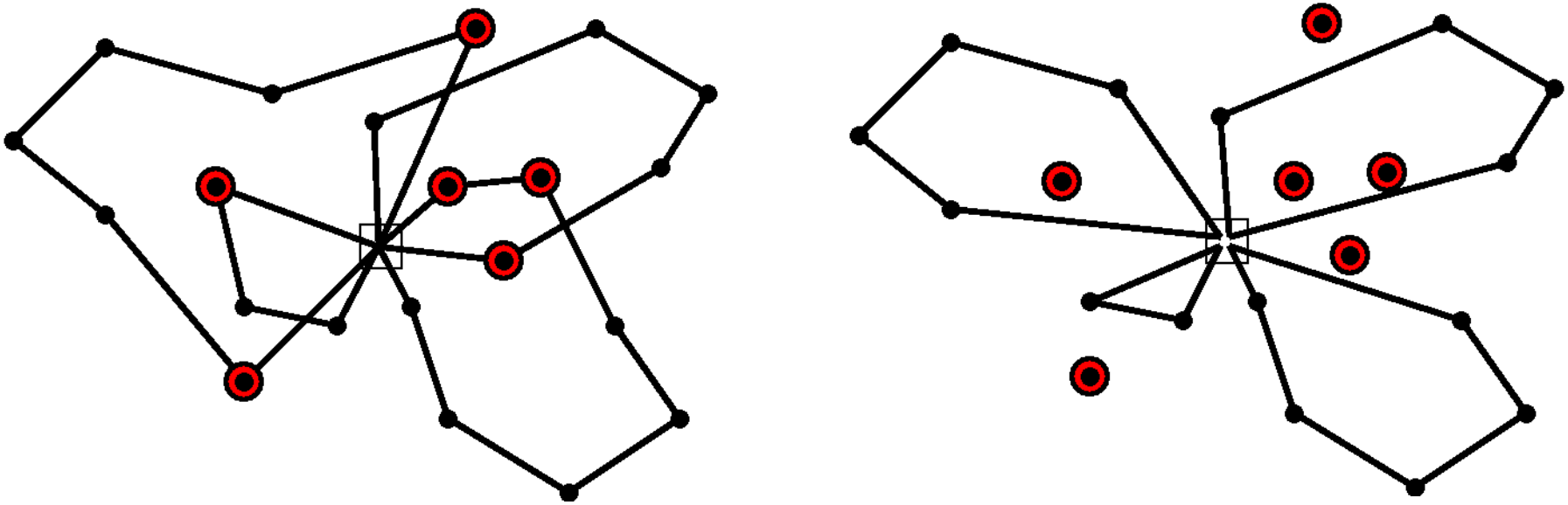
VRP destroy/repair example



VRP destroy/repair example



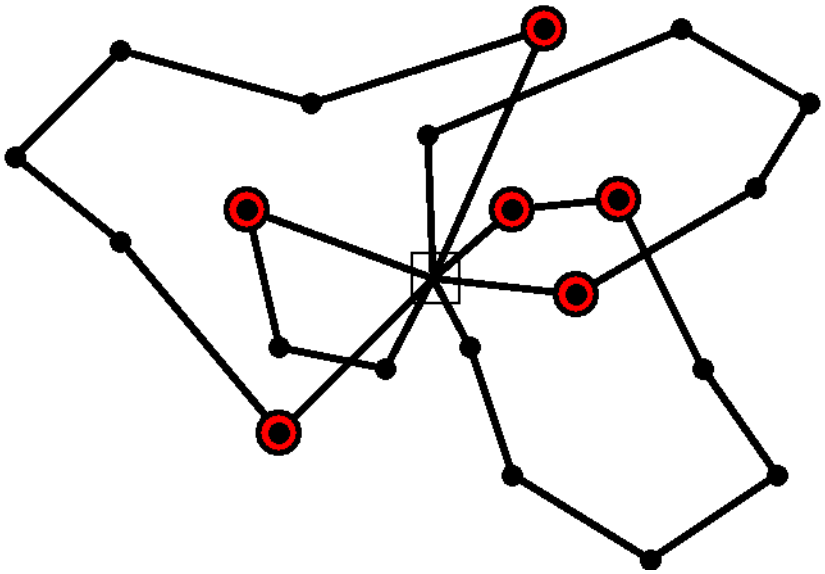
VRP destroy/repair example

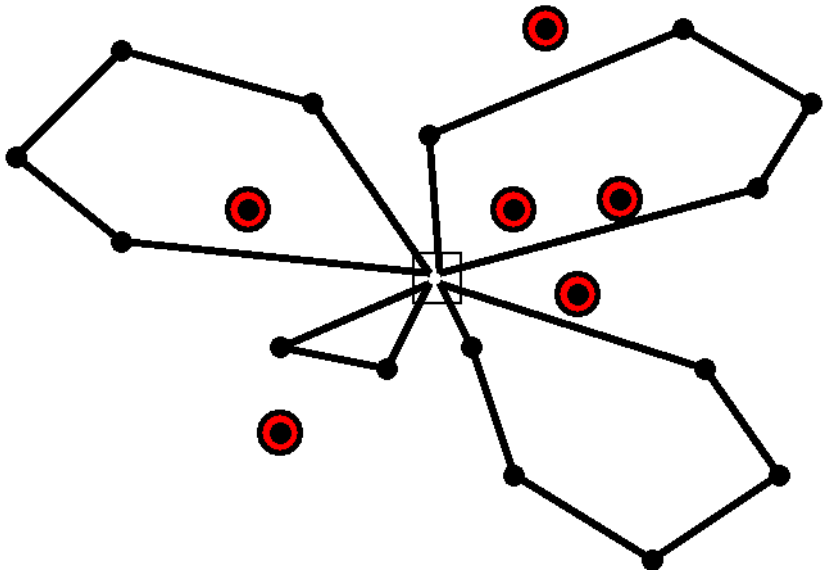


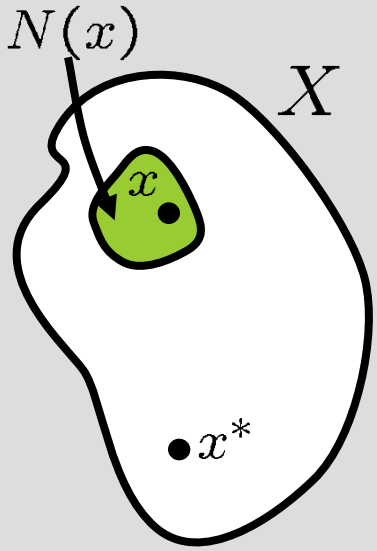
How to destroy and repair?

- This does not have to be complicated! One could simply:
- Destroy method could select customers at random.
- Repair method could sort removed customers according to some criteria and insert the one by one, each time inserting the customer in the route that fits best.

VRP destroy/repair example



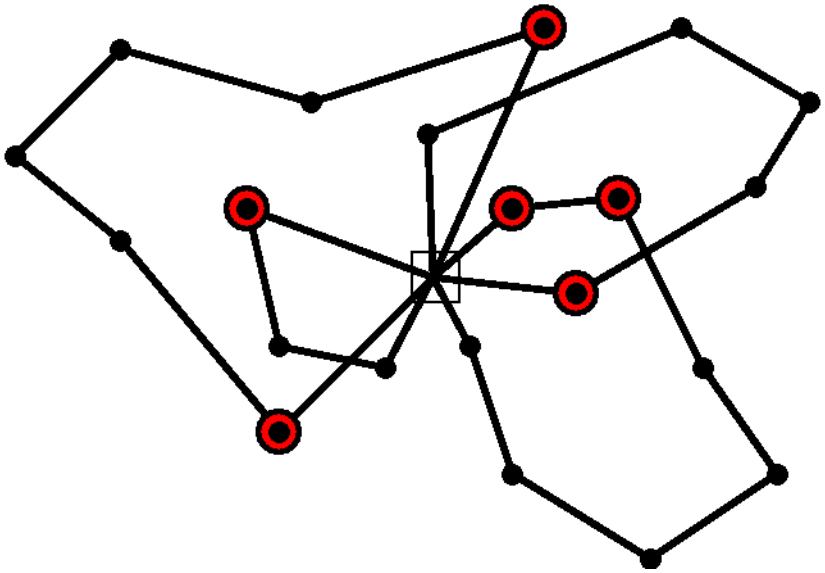


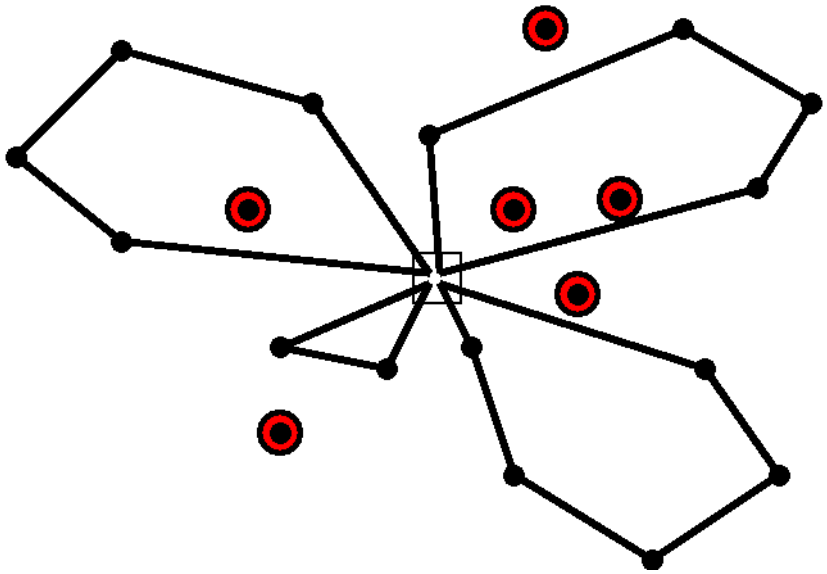


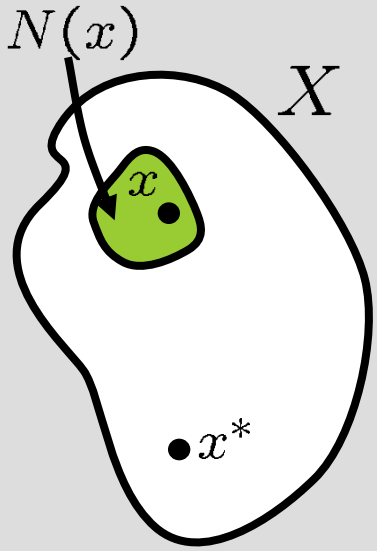
Relation to local search:

- The destroy and repair method **together** defines the neighborhood.
- *N(x)* contains all solutions that can be reached from *x* by first applying the destroy method and then the repair method.
- **Important**: we are only sampling the neighborhood. We cannot inspect all solution in *N(x)*.

VRP destroy/repair example



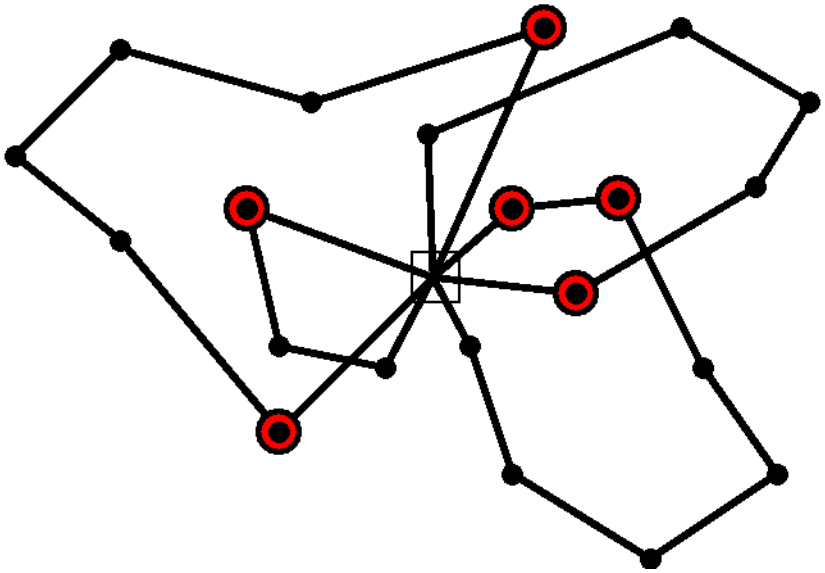


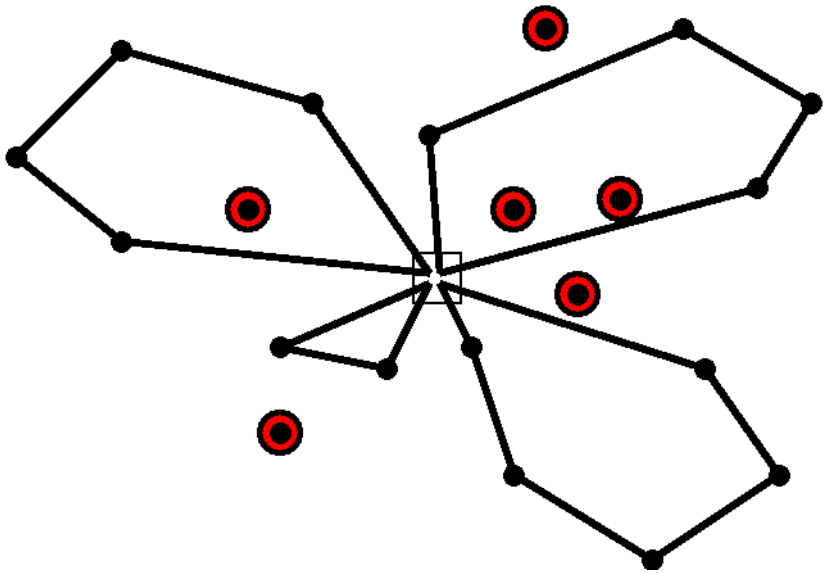


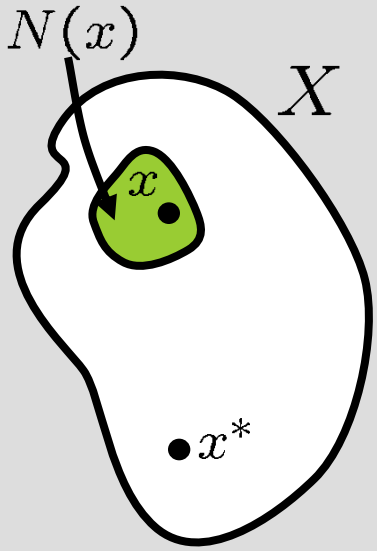
Why <u>Large</u> neighborhood search?
Why a member of the VLSN search family?

- The neighborhood *N(x)* is huge!
- Think of removing 20 customers out a 100 at random. More than $5 \cdot 10^{20}$ ways of doing so.
- For each removal we have to consider all possible ways of inserting the customers back into the solution.

VRP destroy/repair example



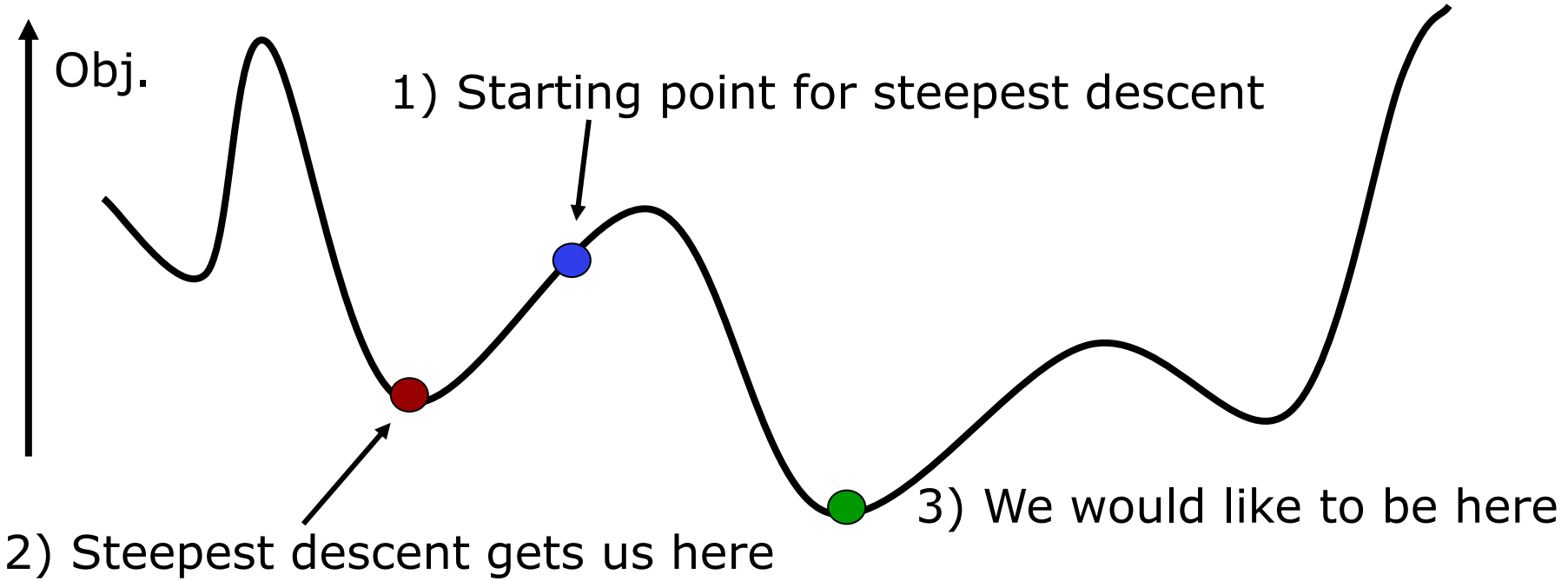




Destroy / repair.
Relax / re-optimize (Shaw ’98).
Ruin / recreate (Schrimpf et al. 2000).
Fix / optimize.

What are the potential advantages of LNS?

- Compared to smaller neighborhoods
  - we can get around in the solution space much faster because of large neighborhoods.
  - There is a smaller change of being stuck in a local minimum (but we do not know if it happens).



What are the potential advantages of LNS?

- Compared to smaller neighborhoods
  - we can get around in the solution space much faster because of large neighborhoods.
  - There is a smaller change of being stuck in a local minimum (but we do not know if it happens).
  - Disadvantage: cannot perform as many iterations (thousands compared to millions).
- Easy to adapt to new problem types.
- Some ”early” examples of succesful LNS algorithms:
  - Shaw 1998 (VRPTW).
  - Bent and Van Hentenryck 2004 (VRPTW).
    - Both used a randomized destroy method that removed ”related” customers and a repair method based on partial enumeration.

Adaptive large neighborhood search (ALNS)

1. Input:
   - an initial solution
   - A set of destroy methods Ω-
   - A set of repair methods Ω+
2. Repeat until stop-criterion:
   a. Choose a destroy method from Ω- and a repair method from Ω+
   b. Destroy part of solution
   c. Repair solution
   d. Accept or reject new solution
   e. Update destroy and repair weights
3. Return best solution encountered

Ropke, Stefan, and David Pisinger. "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows." *Transportation science* 40, no. 4 (2006): 455-472.

Choosing an insertion/removal heuristic

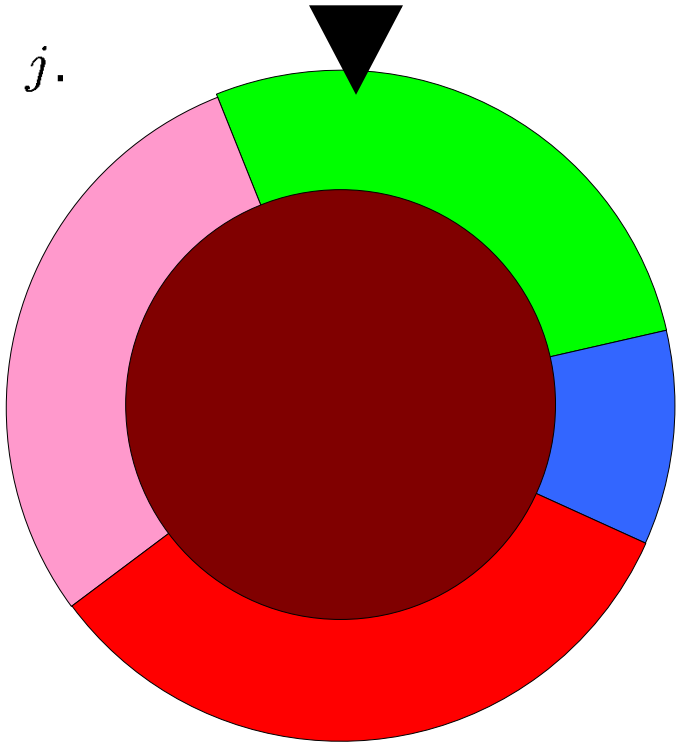
- Assign a weight to each method and choose a method randomly according to weight (*Roulette wheel* selection).
  - w_j : Weight of method j.
  - p_j : Probability of choosing method j.

$$p_{j=} \frac{w_j}{\sum_{i=1}^{k} w_i}$$

Choosing an insertion/removal heuristic

- Assign a weight to each method and choose a method randomly according to weight (*Roulette wheel* selection).
  - w_j : Weight of method j.
  - p_j : Probability of choosing method j.

$$p_{j=} \frac{w_j}{\sum_{i=1}^{k} w_i}$$



Choosing an insertion/removal heuristic

w_j are updated dynamically every time method j has been used.

It is increased if the method did well and decreased otherwise.

$$p_{j=} \frac{w_j}{\sum_{i=1}^{k} w_i}$$

- w_j : Weight of method j.
- p_j : Probability of choosing method j.


Development of the weights of three destroy methods.
7
6
5
4
3
2
1
0
0
5000
10000
15000
20000
25000

Motivation

- Multiple destroy/repair methods:
  - More robust solution method: handle instances with different characteristics better.
  - Synergy: Combined we are stronger.
- Adaptive weight adjustment:
  - Let the heuristic adapt to the instance at hand.
  - Let the heuristic adapt to the current state of the search (initial phase, end phase, etc.)

Possible destroy methods

- Completely random
- Destroy related customers
- Destroy ”expensive” customers
- Use history: destroy customers that do not seem to fit in.
- Destroy whole routes

Repair methods

- Typically based on existing construction heuristics
- Could be more complex (for example exact).
- Sometimes it pays off to randomize repair method.

- Repair method can be boosted by ordinary (small neighborhood) local search.

ALNS vs. LNS

- Case: pickup and delivery problem with time windows (a kind of VRP)
- ALNS uses 3 destroy methods and 4 repair methods
- LNS uses one destroy and one repair method. The destroy and repair method was selected as the best combination among the methods available to the ALNS in a test on a separate set of instances.

	Average gap (%)	Relative time
LNS	1.75	2.1
ALNS	1.39	1.0

Impact of #destroy methods
Impact of adaptive weight adjustment

- Case: Several VRP variants, all featuring ”backhauls”. 338 instances in total.
- ALNS-3D: ALNS with 3 destroy methods
- ALNS-6D: ALNS with 6 destroy methods
- LNS-6D: LNS with 6 destroy methods, all given equal weight

	Average gap (%)	#best
ALNS-3D	0.81	201
ALNS-6D	0.50	248
LNS-6D	0.62	234

Impact of adaptive weight adjustment?

European Journal of Operational Research 292 (2021) 423–442

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search

Renata Turkeš [a,*], Kenneth Sörensen [b], Lars Magnus Hvattum [c]

[a] Department of Mathematics and Computer Science, University of Antwerp, Belgium
[b] Department of Engineering Management, University of Antwerp, Belgium
[c] Faculty of Logistics, Molde University College, Norway

ARTICLE INFO

Article history:
Received 8 November 2019
Accepted 29 October 2020
Available online 5 November 2020

Keywords:
Metaheuristics
Meta-analysis
Adaptive large neighborhood search

ABSTRACT

Research on metaheuristics has focused on (novel) algorithmic development and on competitive testing, both of which have been frequently argued to yield little generalizable knowledge. The main goal of this paper is to promote meta-analysis — a systematic statistical examination that combines the results of several independent studies — as a more suitable way to obtain problem- and implementation-independent insights on metaheuristics. Meta-analysis is widely used in several scientific domains, most notably the medical sciences (e.g., to establish the efficacy of a certain treatment). To the best of our knowledge, this is the first meta-analysis in the field of metaheuristics.

To illustrate the approach, we carry out a meta-analysis to gain insights into the importance of the adaptive layer in adaptive large neighborhood search (ALNS). Although ALNS has been widely used to solve a broad range of problems, it has not yet been established whether or not adaptiveness actually contributes to the performance of an ALNS algorithm.

A total of 134 studies were identified through Google Scholar or personal e-mail correspondence with researchers in the domain, 63 of which fit our eligibility criteria. After sending requests for data to the authors of the eligible studies, we obtained results for 25 different implementations of ALNS, which were analysed using a random effects model.

On average, the addition of an adaptive layer in an ALNS algorithm improves the objective function value by 0.14% (95% confidence interval 0.06–0.21%). Although the adaptive layer can (and in a limited number of studies does) have an added value, it also adds complexity and can therefore only be recommended in some specific situations. These findings underline the importance of evaluating the contribution of metaheuristic components.

Large language models

- ”Attention is All you Need“ paper, 2017 (Google research)
- GPT 1.0 June 2018 - 117 million parameters
- GPT 2.0 November 2019 – 1.5 billion paramters
- GPT-3.0 May 2020 - 175 billion parameters
- **GPT-3.5** November 2022
  - 100 million users in two months, fastest-growing consumer software application in history.
  - Perhaps mostly used as entertainment and light text editing?
- GPT-4.0 March 2023
- GPT-4.5 February 2025
- GPT-5 August 2025
- GPT-5.2 December 2025
- GPT-5.3-codex February 2026

Gemini

Claude

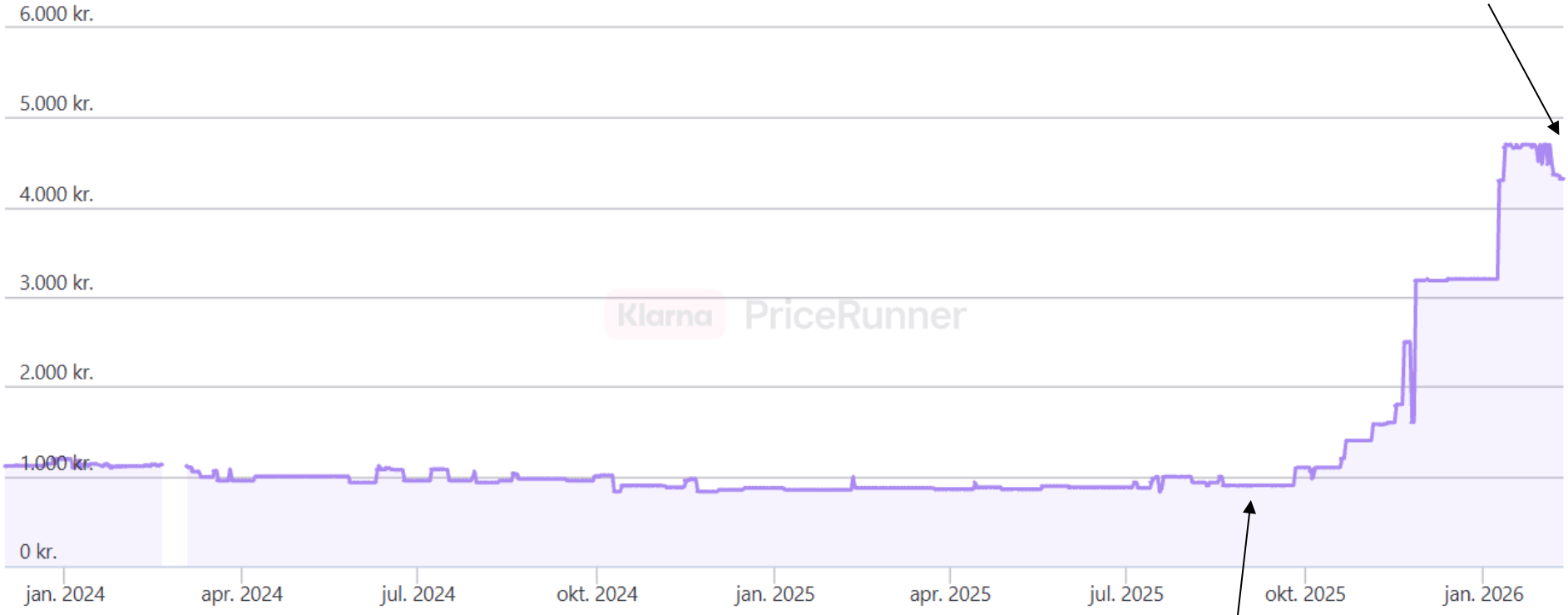
Qwen

LLaMA by Meta

Large language models
Price of memory (RAM)

Price of 2x16GB DDR5 RAM (what you use in your computer)

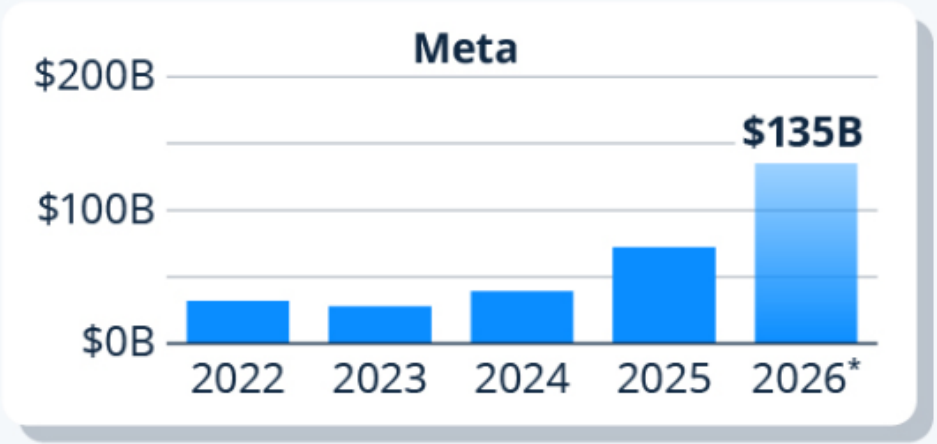
Today
4317 DKK
578 Euro

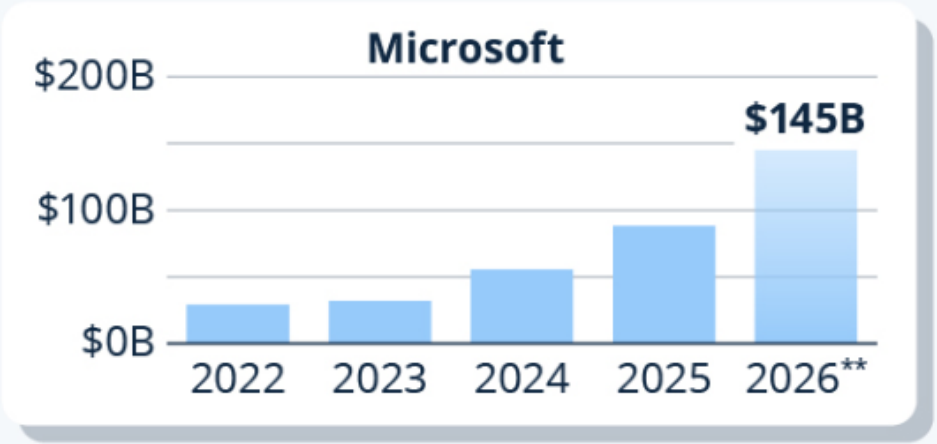


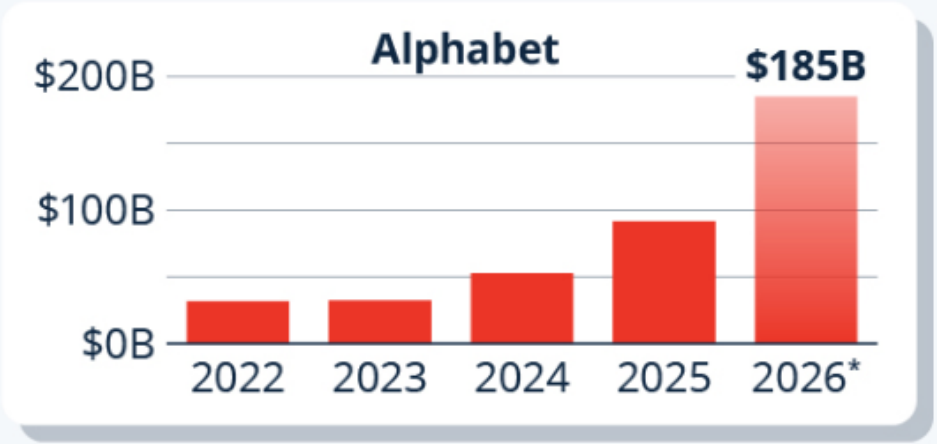
Sept 2025
899 DKK
120 Euro

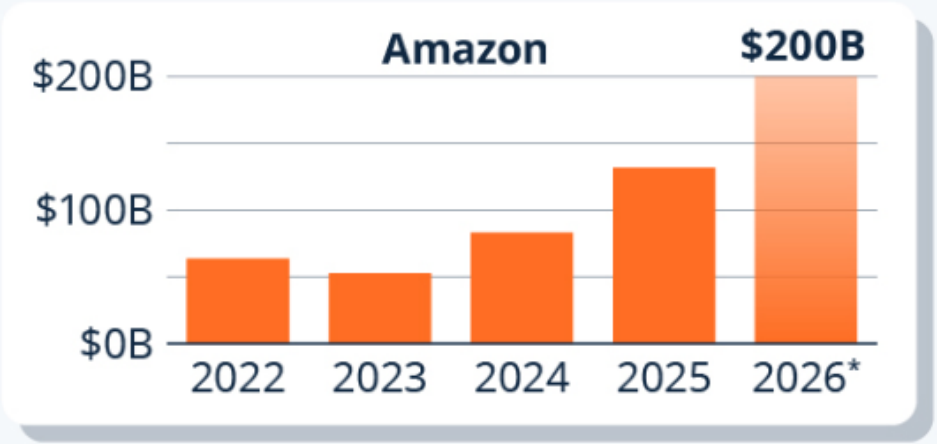
Price of phones, tablets, computers will rise as a result

Biggest US companies are pouring money into buying hardware for their cloud business. Mainly due to AI (LLMs mainly)









Capital expenditures. 2026 numbers are upper limit on companies' projections.

Source: https://www.statista.com/

Changing perception of LLMs?

- Has our perception of LLMs changed over the last 6-12 months?
- From: *they are just stupid next-word predictors repeating the text they have been trained on.*
- To some level of: *hmm… they are actually useful and seemingly smart in some areas.*
- ChatGPT got 8 out of 10 questions right from this winter's "Introduction to OR" exam at DTU (multiple choice, computation-oriented)
- Impact on teaching and learning?
- Impact on our cognitive abilities, especially the young generation?
- Impact on science?
- Use in academia is a taboo?

ALNS, VRPs and LLMs

- Can you use an LLM to write an ALNS heuristic?
- Motivation: create a many ALNS heuristic (5-10?) for different problems.
- Want to use this test different ways of selecting destroy and repair operators

ALNS, VRPs and LLMs

- **Starting point:** ALNS framework that implements an ALNS heuristic without knowing about the problem
- **Starting point:** An example implementation of an ALNS heuristic for the Generalized assignment problem
- Asks ChatGPT to implement an ALNS heuristic for the CVRP using the same framework.

ALNS, VRPs and LLMs

- **Prompt (.jl = Julia file – all implementation is done in Julia):**

I have an ALNS framework (ALNS.jl) and an example application, that implements a heuristic for the generalized assignment problem (ALNS-GAP.jl). I have attached the code.

Can you try to implement a heuristic for the capacitated vehicle routing problem? Please put the new code in a module like in ALNS-GAP.jl.

Instances should be read from files that uses a standard VRP format similar to the TSPLIB format. I have attached an example. It's just a normal text file.

Result

- ChatGPT thinks for about 1 minute

- Working ALNS heuristic for the CVRP (“compiles first time”)
  - Random customers
  - Worst contribution
  - Random routes

  - Greedy insertion
  - Regret-2 insertion

  - Intraroute 2-opt

Result

- ChatGPT improves code based on profile of a run
- Uses agentic coding in Vscode to improve further based on LLMs “ideas”
  - Adds regret-3 repair
  - Adds two more destroy methods
  - Adds two more local search methods (swap customer, 2-opt*)
- This improvement phase take less than an hour (?)

Results – mean

	#customers	ALNS-LLM	FILO	Time ALNS	Time FILO
Small	1-250	0.39%	0.18%	1-2 minutes	1-2 minutes
Medium	250-500	0.88%	0.39%	1-3 minutes	1-2 minutes
Large	500-1000	1.3%	0.5%	3-6 minutes	1-2 minutes

VRPTW

- Similar prompt
- First version after 7 minutes. Compiles first time.
- Destroy and repair functions similar to CVRP
- Feasibility check took O(n) [n length of route]
- The LLM change that to O(1) once it was pointed out
- Results doesn’t seem stelar

Single liner shipping service design

- Plum, Christian EM, David Pisinger, Juan-José Salazar-González, and Mikkel M. Sigurd. "Single liner shipping service design." *Computers & Operations Research* 45 (2014): 1-6.

Single liner shipping service design

$$\min \sum_{a \in A} c_a x_a - \sum_{k \in K} r^k \sum_{a \in \delta^-(s^k)} f_a^k$$

subject to

$$x(\delta^-(i)) = 1, \quad i \in V$$

$$x(\delta^+(i)) = 1, \quad i \in V$$

$$x(\delta^+(S)) \geq 1, \quad S \subset V$$

$$\sum_{a \in \delta^+(i)} f_a^k = \sum_{a \in \delta^-(i)} f_a^k, \quad k \in K \quad \text{and } i \in V \backslash \{s^k, d^k\}$$

$$\sum_{k \in K} f_a^k \leq Q x_a, \quad a \in A$$

$$\sum_{a \in \delta^-(s^k)} f_a^k \leq F^k, \quad k \in K$$

$$\sum_{a \in A} t_a f_a^k \leq t^k \sum_{a \in \delta^-(s^k)} f_a^k, \quad k \in K$$

$$f_a^k \geq 0, \quad a \in A \quad \text{and } k \in K$$

$$x_a \in \{0, 1\}, \quad a \in A$$

Single liner shipping service design

$$\min \sum_{a \in A} c_a x_a - \sum_{k \in K} r^k \sum_{a \in \delta^-(s^k)} f_a^k$$

subject to

$$x(\delta^-(i)) = 1, \quad i \in V$$

$$x(\delta^+(i)) = 1, \quad i \in V$$

$$x(\delta^+(S)) \geq 1, \quad S \subset V$$

$$\sum_{a \in \delta^+(i)} f_a^k = \sum_{a \in \delta^-(i)} f_a^k, \quad k \in K \quad \text{and } i \in V \setminus \{s^k, d^k\}$$

$$\sum_{k \in K} f_a^k \leq Q x_a, \quad a \in A$$

$$\sum_{a \in \delta^-(s^k)} f_a^k \leq F^k, \quad k \in K$$

$$\sum_{a \in A} t_a f_a^k \leq t^k \sum_{a \in \delta^-(s^k)} f_a^k, \quad k \in K$$

$$f_a^k \geq 0, \quad a \in A \quad \text{and } k \in K$$

$$x_a \in \{0, 1\}, \quad a \in A$$

- Lets say a commodity k has demand $F^k = 100$
- We only send 50 units
- The commodity is sent through 3 arcs with travel times 10, 12, 14
- Highlighted constraint becomes
- $10 \cdot 50 + 12 \cdot 50 + 14 \cdot 50 \leq t^k \cdot 50$ or
- $(10 + 12 + 14) \cdot 50 \leq t^k \cdot 50$
- This is only feasible if t^k is at least 36

Single liner shipping service design

- LLM is only presented with Julia implementation of model and data reading functions

```
@variable(model, x[1:n_customers, 1:n_customers], Bin)
@variable(model, f[1:n_customers, 1:n_customers, K] >= 0) # Flow variable
@variable(model, seq[V] >= 0) # sequence variable

# Self-loops are not allowed
@constraint(model, [i in V], x[i,i] == 0)
@constraint(model, [i in V, k in K], f[i,i,k] == 0)
# don't send flow back to the source node
@constraint(model, [i in V, k in K], f[i,s[k],k] == 0 )

…Continues like this…
```

```
struct SLSSDInstance
    n_customers::Int64
    ship_capacity::Float64
    time_matrix::Matrix{Float64}
    cost_matrix::Matrix{Float64}
    request::Vector{Request}
end
```

```
struct Request
    source::Int64
    destination::Int64
    load::Float64
    time::Float64
    unit_revenue::Float64
end
```

Single liner shipping service design

- LLM output (part of it)

- This is a **single-vehicle routing + multi-commodity flow** model with **optional transportation requests** and **profit maximization** (implemented as cost minus revenue).
- You can think of it as:
- **Choose a Hamiltonian tour over all customers, and decide which transport requests to carry along that tour, subject to capacity and time constraints, to maximize profit.**
- It's *not* a standard VRP — it's closer to a **profitable pickup-delivery flow on a fixed TSP tour**.

Single liner shipping service design

- LLM output (part of it)
- Misunderstands the “difficult” constraint

5. Time constraint per request

julia Copy code

```
sum(t[i,j]*f[i,j,k]) <= request[k].time * sum(f[s[k],j,k])
```

This is clever and subtle:

- If request `k` is not served → RHS = 0 → constraint inactive
- If served → average travel time per unit flow must be ≤ allowed time

So time feasibility is enforced **conditionally on serving the request**.

Single liner shipping service design

- Need to explain to the LLM what the constraint does.
- Further bug fixes and improvements. All done by LLM with human guidance.
- Takes a couple of hours (on/off work).

Single liner shipping service design

- **Random edge destroy** - pick random edge, destroy adjacent nodes, repeat
- **Worst-edge destroy**
- **Bad-requests endpoint destroy -** *Targets high-value requests that are currently time-infeasible on the tour. We score infeasible requests by (unit_revenue*load) * (pathTime - T). Then remove endpoints of the top fraction of such requests.*
- **Congesting request destroy -** only removes requests-

- **repairCheapestInsertion** – insert nodes at cheapest position, trying to take potential gain by requests into account
- **repairRegret2Insertion** – Looks at best and second best insertion and inserts node with highest regret (difference)

- After node insertion tries to insert requests in a greedy manor (looks at revenue and number of edges “covered”)

Single liner shipping service design

Statistics for instance with 25 nodes and 15 requests:

- Random tour destroy, attempts: 11307, avg. time: 2.0e-6
- Worst-edge tour destroy, attempts: 12198, avg. time: 4.0e-6
- Bad-requests endpoint destroy, attempts: 43828, avg. time: 2.0e-6
- Congesting request destroy, attempts: 44028, avg. time: 6.0e-6

- Cheapest insertion repair, attempts: 55058, avg. time: 0.000247
- Regret-2 insertion repair, attempts: 56301, avg. time: 0.000267
- Number of iterations: 111357

Single liner shipping service design

140 instances, up to 25 nodes and 600 requests

Heuristics run for 30 seconds

Gurobi run for 7200 seconds

Dario Palasgo coded the hand-craftet heuristic (ILS). Not his latest results!

ALNS vs. ILS	count
Equal	93
ALNS best	10
Hand-coded heuristic best	37

ALNS vs Gurobi	count
Equal	59
ALNS best	59
Gurobi best	22

LLM generated ALNS heuristics - outlook

- Fast way to get solutions of reasonable quality
- How to automate further?
- Can we generate heuristic fully or semi automatically from Julia model and data?
  - Compare heuristic solutions to solver solutions for small instances?
  - Check feasibility by translating solutions back to model (fixing variables)
- Should we write our models in a more expressive way? MIP models are not the easiest to read for humans. Constraint programming? Hexaly models? Plain text and human feedback?

Conclusion

- Vehicle routing is still an interesting research field and very much alive!
- LLMs can write ALNS heuristics pretty well!
- What will the future bring?