Optimization

Leon Lan
Maandag 7 april 2025
l.lan@vu.nl

Wie ben ik?

- Leon Lan
- BSc Amsterdam University College
- MSc Operations Research
- 4e jaars PhD @ VU Amsterdam
  - Algorithms and software for routing and scheduling

Wie zijn jullie?

- Naam
- Studie
- Huidige opdracht
- Wat optimaliseer jij graag?

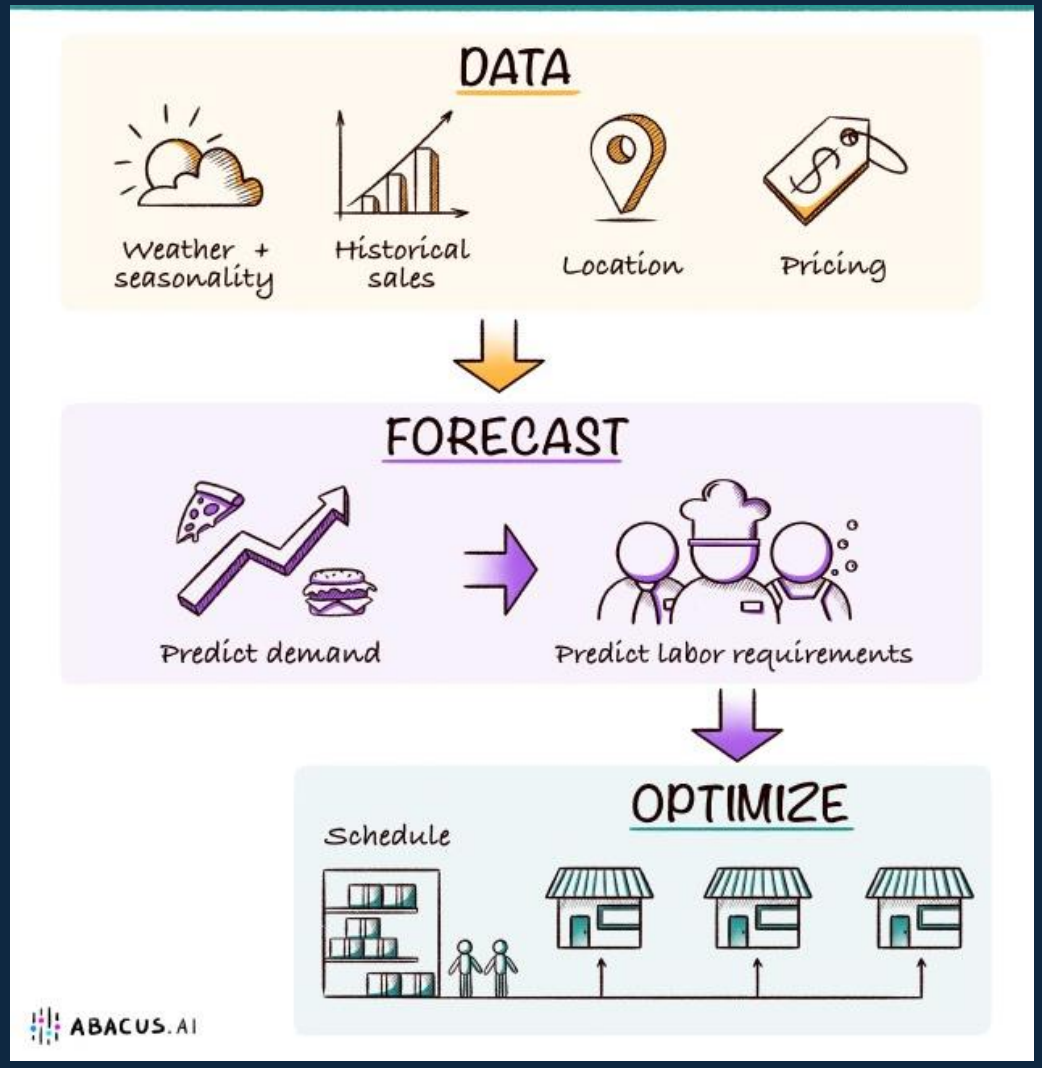
De data analyst

SQL

Azure

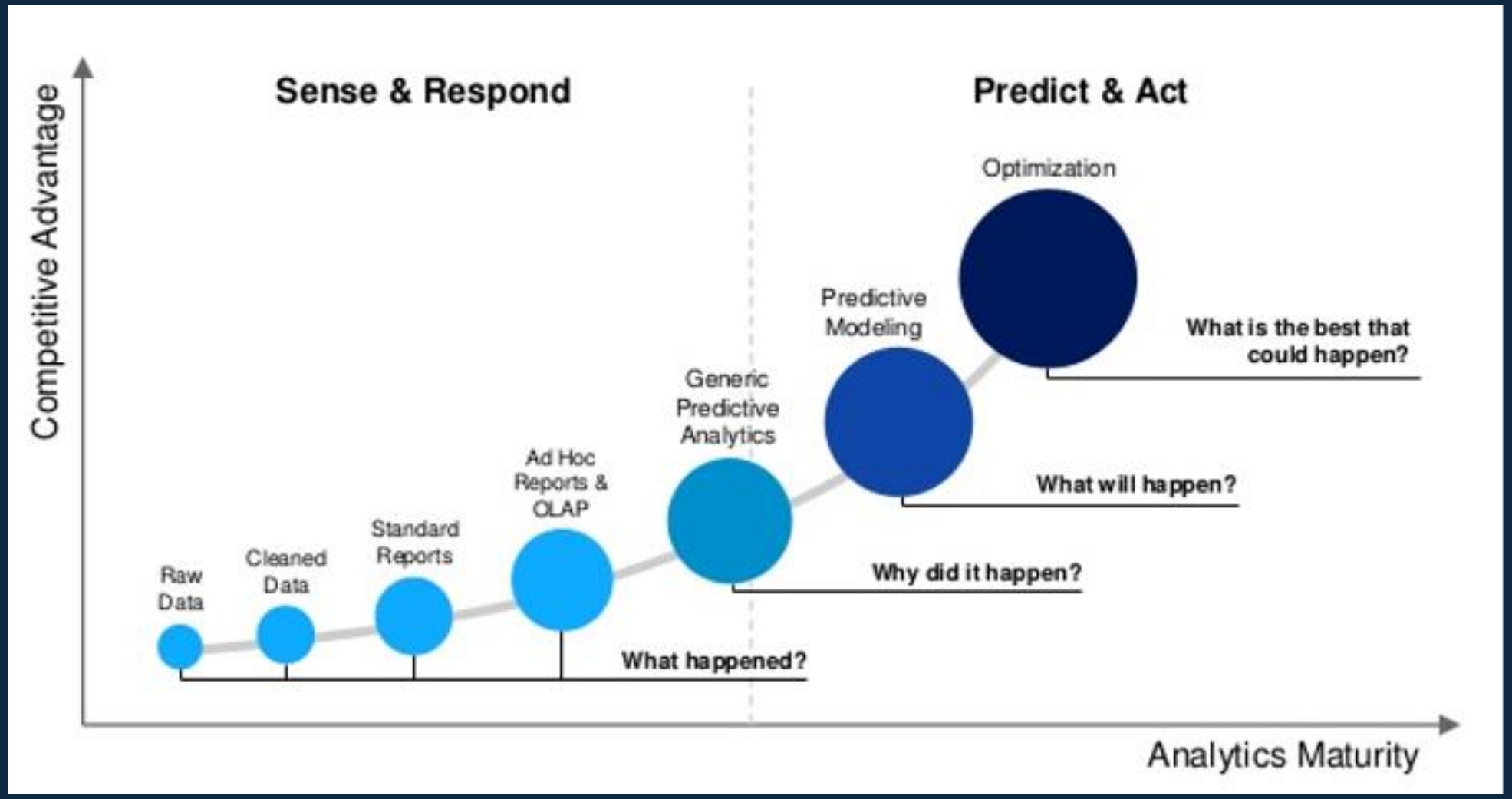
pandas

Van data naar optimization



- 📅 Welke medewerker werkt wanneer zodat we de bezetting maximaliseren?
- 🚚 Welke routes zijn het efficiëntst zodat we kilometers minimaliseren?
- 🏢 Hoeveel en wanneer bestel ik bij zodat de kosten geminimaliseerd worden?

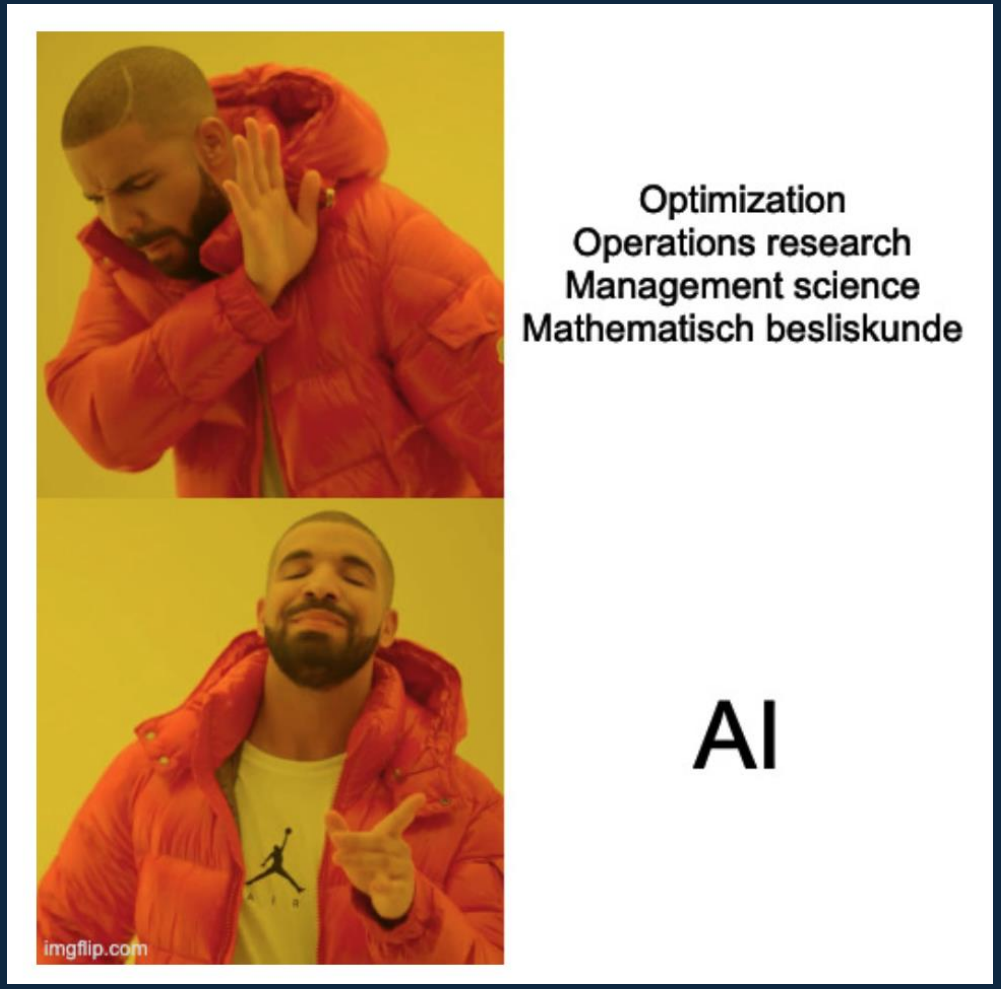
De ontwikkeling van data-gedreven organisaties



Wat is *geen* optimization?

- Data schoonmaken
- Beschrijvende statisiek
- SQL queries schrijven
- Dimensiereductie
- Outlier detection
- Clusteranalyse/segmentatie
- Forecasting

Het marketing probleem van optimization



Google Cloud

Optimization AI

Wat zijn de leerdoelen van vandaag?

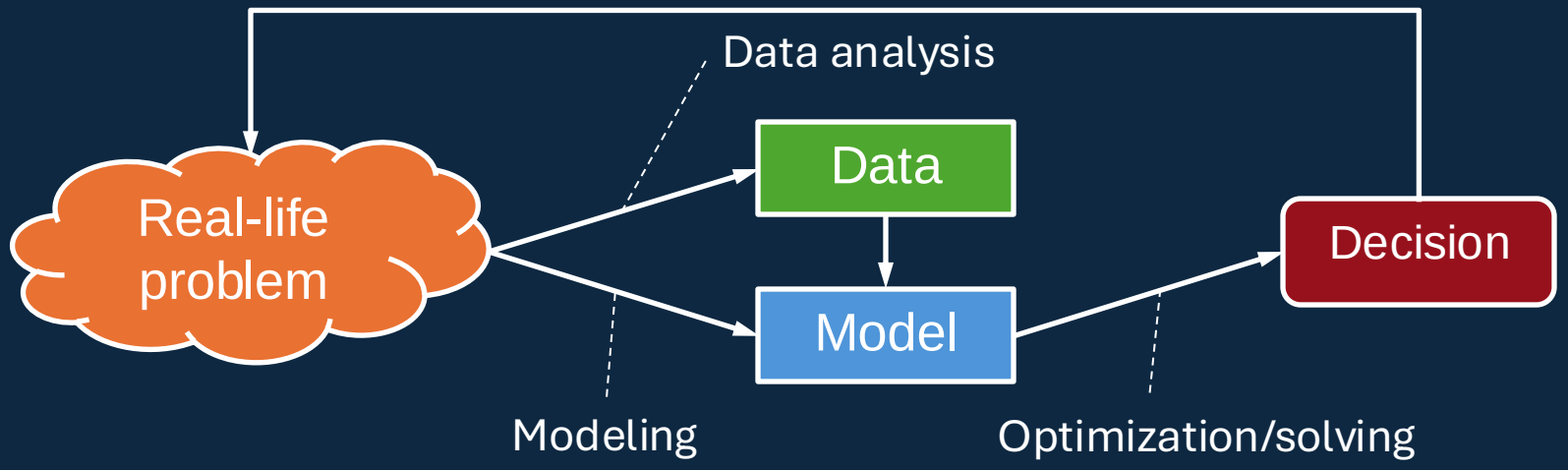
1. Uitleggen wat optimalisatie is en hoe het gebruikt kan worden toegepast in jouw bedrijf
2. Hands-on ervaring met optimalisatiealgoritmes bouwen
3. Ontwikkelen van een optimalisatie dashboard

Overzicht

10:00 – 11:00:	*Wat is optimization?*
11:00 – 12:15:	*Bouw je eigen optimalisatie algoritme*
12:15 – 13:15:	*Lunch pauze*
13:15 – 14:30:	*Gebruiken van optimalisatie solvers*
14:30 – 16:15:	*Optimization challenge*
16:15 – 16:30:	*Afsluiting en evaluatie*

Wat is optimization?

Optimization lifecycle



Optimization lifecycle

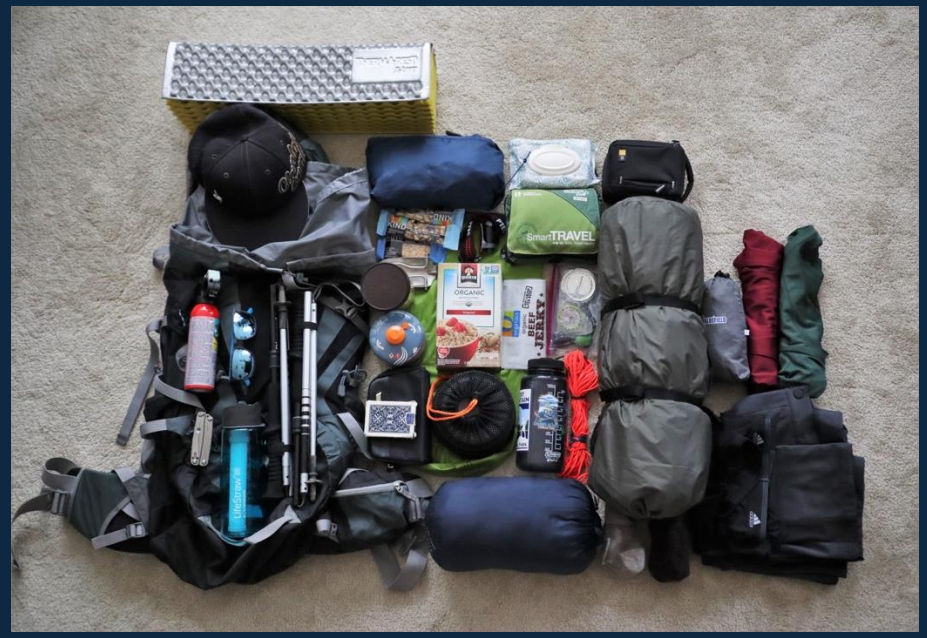
Real-life problem

Camino Portugues 2023
11 daagse wandeling

Vraagstukken
- Welke spullen neem ik mee?
- Rugtas heeft een capacitieit max 10KG

Optimization lifecycle

Data



Item	Gewicht	Waarde
Regenbroek	300g	10
Donsjas	250g	5
Slaapzak	500g	9
…	…	…
…	…	…

Optimization lifecycle

Model

Knapsack problem

Article Talk

From Wikipedia, the free encyclopedia

The **knapsack problem** is the following problem in combinatorial optimization:

> *Given a set of items, each with a weight and a value, determine which items to include in the collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.*

Optimization lifecycle

Model

1. **Objective**
   - Maximaliseer de totale waarde
2. **Decision variables**
   - Neem ik item i wel of niet mee?
3. **Constraints**
   - Rugzak capaciteit van max 10KG

Optimization lifecycle

Model

$$x_i = copies\ of\ each\ kind\ of\ item$$

$$v_i = value$$

$$w_i = weight$$

$$W = maximum\ weight\ capacity$$

$$i = items\ numbered\ 1..n$$

$$maximize \sum_{i=1}^{n} v_i\, x_i$$

$$subject\ to \sum_{i=1}^{n} w_i\, x_i \leq W, x_i \in \{0,1\}$$

Optimization lifecycle

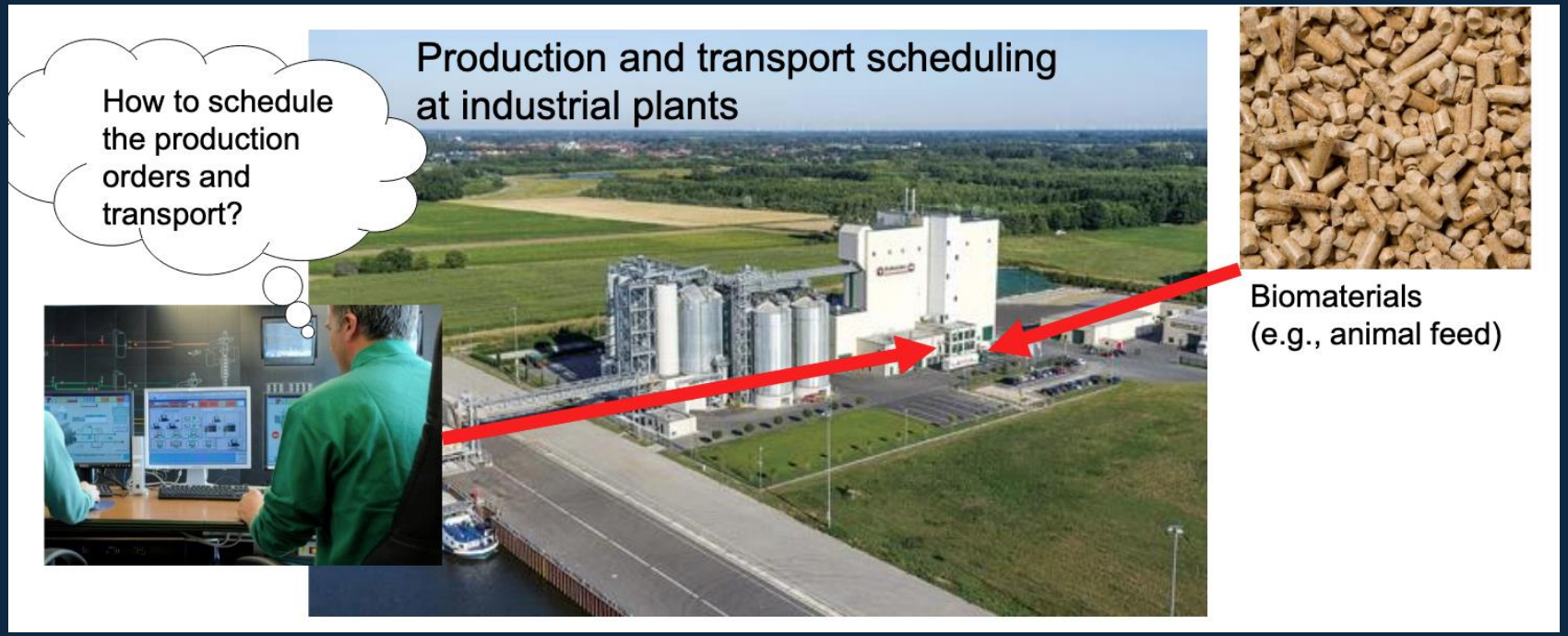
Decision

Greedy algorithm:

- Vul rugzak in volgorde van hoogste “waarde gedeeld door gewicht”
  1. Regenbroek: 10/300 = 0.033
  2. Slaapzak: 9/500 = 0.018
  3. Donsjas: 5/300 = 0.0167
- Totdat de maximum capaciteit bereikt is.

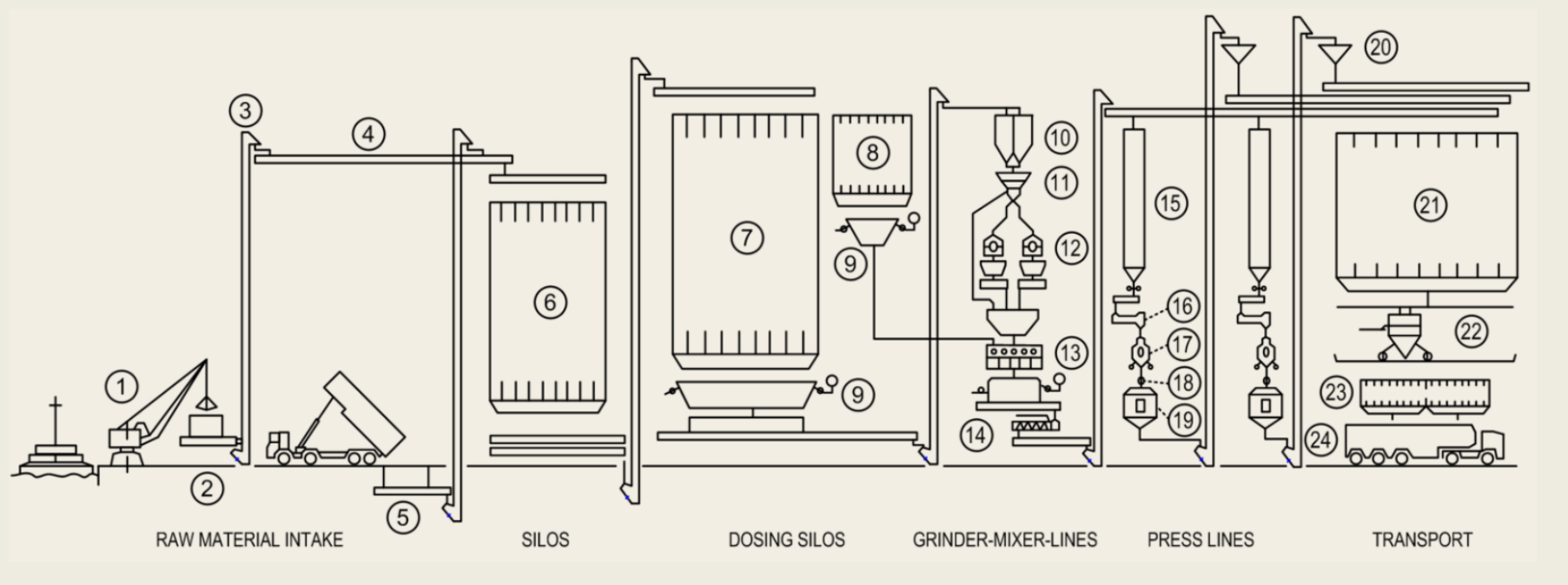
Optimization lifecycle

Real-life problem



Optimization lifecycle

Data



Klant	Product	Hoeveelheid	Gewenste levertijd
Gerritsen Frans	9304A	10.000kg	Dinsdag
Gerritsen Frans	9410	10.000kg	Dinsdag
Pluimveebedrijf Plomp	1111	100.000kg	Vrijdag

Optimization lifecycle

Model

1. **Objective**
   - Minimaliseer de fabriekstijd (*makespan*)
   - Minimaliseer de klant-deadline overschrijdingen
2. **Decision variables**
   - Wanneer produceer ik een batch van welk product?
   - Start en eindtijden + aangewezen machine
3. **Constraints**
   - Produceer batches van hetzelfde product back-to-back
   - Bepaalde machines kunnen alleen bepaalde producten produceren
   - ….

Optimization lifecycle

Model

Integer linear optimization (ILO) problem

Production schedules

Minimize tardiness and makespan

Objective

Constraints

$$\min_{s_{iu},\, \bar{s}_{iu},\, p_{ij},\, \bar{p}_{lj},\, \delta_{ik},\, \bar{\delta}_{l_1,l_2},\, \gamma_r,\, F_{\max}} \quad \sum_{r=1}^{\eta} \gamma_r \cdot w_r \;+\; \epsilon \cdot F_{\max}$$

Subject to

$$\sum_{j \in M_i} p_{ij} = 1, \qquad i = 1, \ldots, n.$$

$$s_{k1} - s_{i1} \le \delta_{ik} \cdot T, \qquad i \in \Theta_{l_1},\ \ k \in \Theta_{l_2},\ \ l_1, l_2 = 1, \ldots, \kappa,\ \ l_1 \ne l_2,$$

$$\delta_{ik} + \delta_{ki} = 1, \qquad i < k,\ \ i \in \Theta_{l_1},\ \ k \in \Theta_{l_2},\ \ l_1, l_2 = 1, \ldots, \kappa,\ \ l_1 \ne l_2,$$

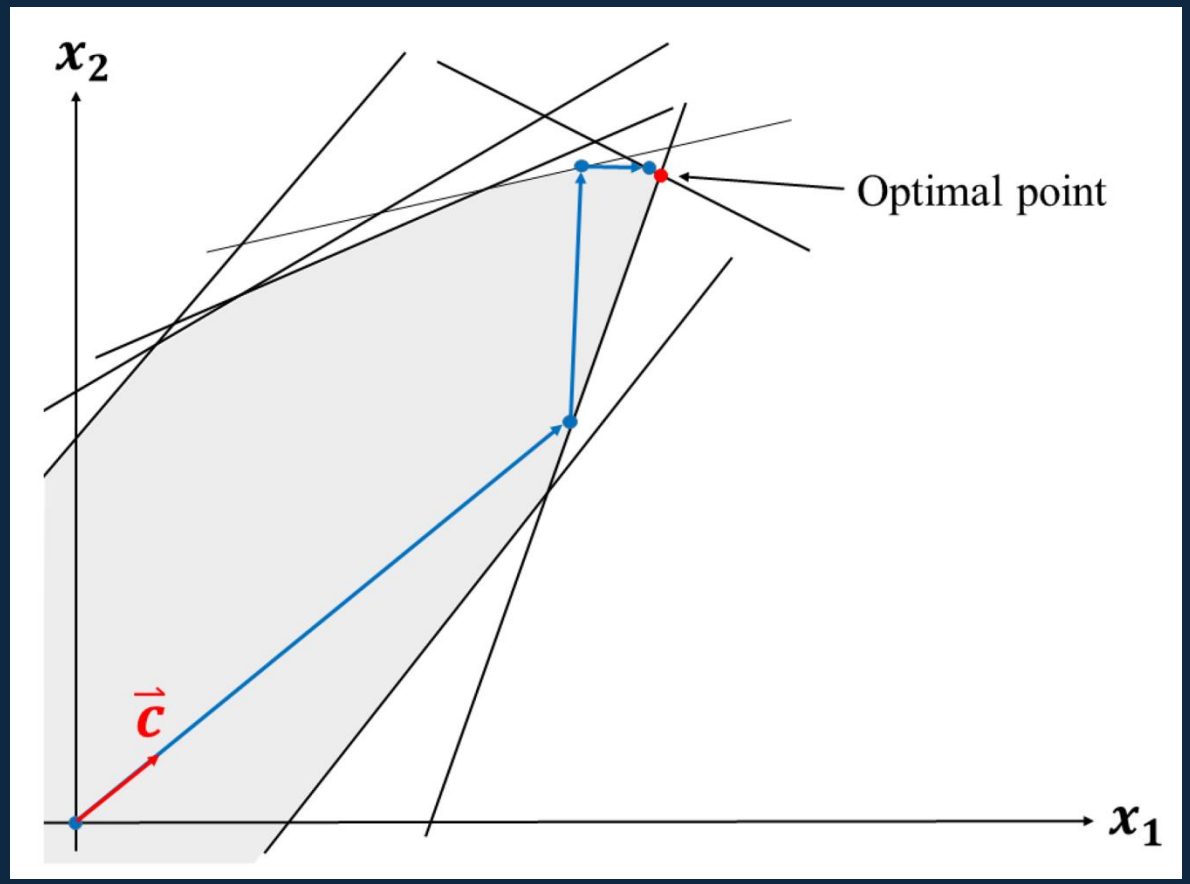
$$s_{i,u+1} - s_{ku} \le T \cdot (2 - p_{ij} - p_{kj}), \qquad i, k \in \Theta_l,\ \ i < k,\ \ l = 1, \ldots, \kappa,\ \ u = 1, \ldots, U-1,\ \ j \in M_i\ (= M_k),$$

$$s_{i,u+1} + t_{ikuj} - s_{ku} \le T \cdot (1 - \delta_{ik} + 2 - p_{ij} - p_{kj}), \qquad i \in \Theta_{l_1},\ \ k \in \Theta_{l_2},\ \ l_1, l_2 = 1, \ldots, \kappa,\ \ l_1 \ne l_2,\ \ u = 1, \ldots, U-1,\ \ j \in M_i \cap M_k,$$

$$s_{iU} + \tau_{iUj} - s_{kU} \le T \cdot (2 - p_{ij} - p_{kj}), \qquad i, k \in \Theta_l,\ \ i < k,\ \ l = 1, \ldots, \kappa,$$

Optimization lifecycle

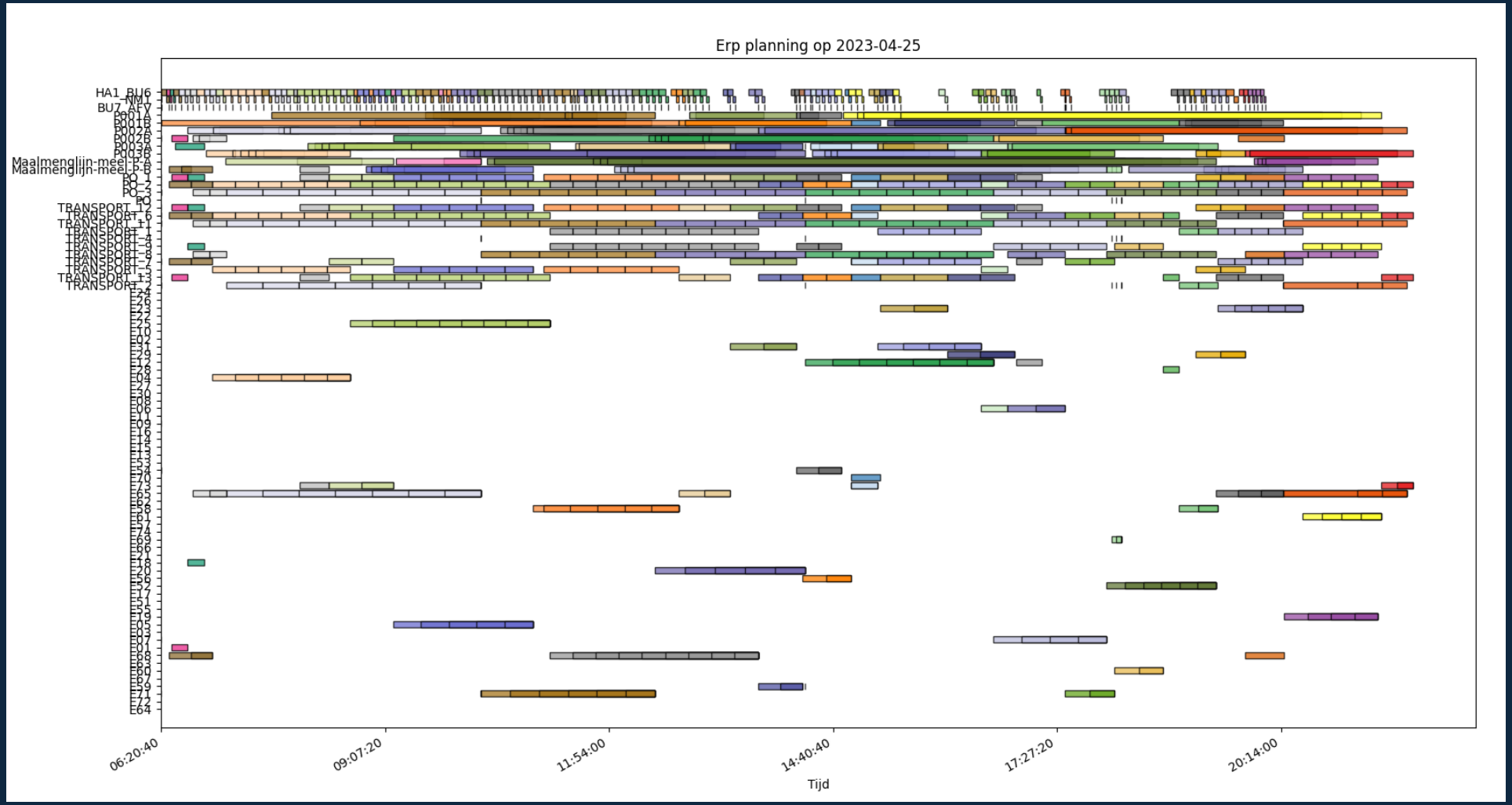
Decision



Interior point methods
Branch and bound
Constraint programming

Optimization lifecycle

Decision



Optimization problems

Real-life problem

ALPIQ

Een duidelijk overzicht van de resultaten

Wie is er populair? Wie wordt er gepest? Wie voelt zich prettig? En wie voelt zich juist onveilig?

Met ons rapport krijg je in een handomdraai antwoord op deze vragen. Je ontvangt een overzichtelijk rapport met daarin informatie over het gedrag en de sociale positie van elke leerling, diverse sociogrammen en normscores. Maken jouw collega's ook gebruik van Stoeltjesdans? Wij zorgen dan ook voor een duidelijke rapportage over de sociale veiligheid op jouw school!

Start nu! ▸

stoeltjesdans

	Sociale status				Positief gedrag			Negatief gedrag			Slachtoffer van negatief gedrag		
	Beste vrienden	Meest aardig	Minst aardig	Meest populair	Minst populair	Goed samenwerken	Hulp bij problemen	Verdedigen bij pesten	Zijn gemeen of schelden	Roddelen of sluiten buiten	Pest anderen*	Wordt uitgescholden of gemeen tegen gedaan	Wordt over geroddeld of buitengesloten	Wordt gepest
Emma	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Noah	8	0	31	8	0	38	31	23	8	23	0	0	15	0
Yara	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tess	15	15	8	0	23	54	54	23	0	0	17	0	8	0
Liam	8	8	8	92	0	54	85	54	15	23	14	0	0	0
Luca	54	8	23	0	0	15	0	8	38	0	29	23	0	0
Mo	31	23	8	0	8	77	85	62	8	15	17	0	8	0
Mees	31	31	0	0	23	77	77	46	0	8	14	0	8	0
Daan	8	15	8	0	23	46	31	23	0	0	14	0	31	8
Finn	43	7	0	0	7	50	43	29	7	0	0	0	7	0
Timo	38	38	0	0	0	54	23	38	15	0	0	0	0	0
Sarah	46	15	0	0	0	54	69	31	0	15	0	0	8	0
Boaz	23	23	8	0	31	54	62	46	0	0	0	0	0	0
Floris	46	23	0	0	0	77	69	46	8	0	0	0	8	0
Gijs	31	0	23	0	8	23	15	23	8	0	0	0	0	0
Omar	54	15	0	0	8	85	54	23	0	0	0	0	0	0
Rayan	31	0	23	0	23	23	31	23	23	0	0	15	15	0

Factories

packing, inventory

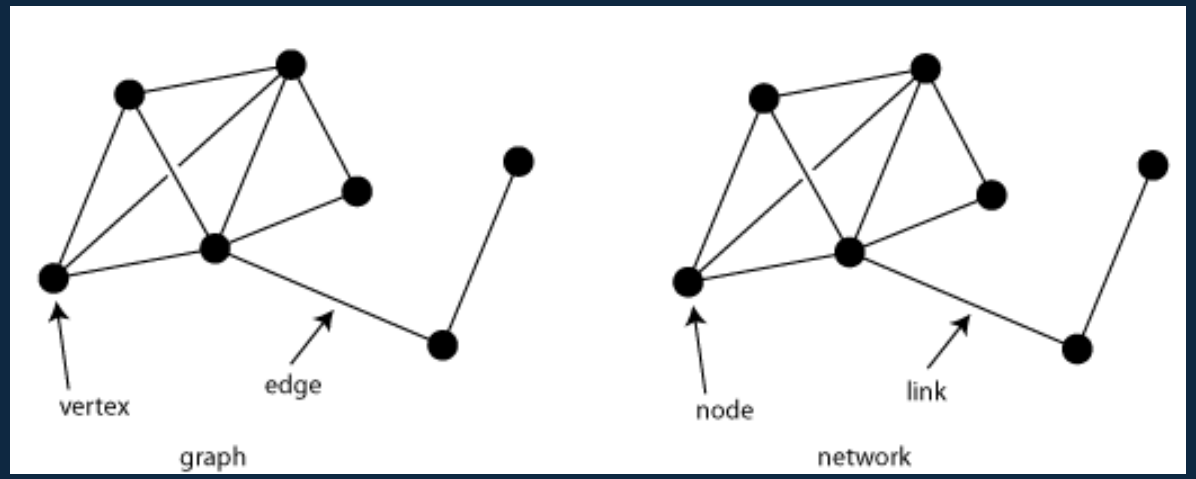
5

Optimization models

Model

Item	Choose?
1	Yes
2	No
3	Yes

$$(\mathcal{P}) \begin{cases} \max \quad z(x) = & 2x_1 + x_2 - 2x_3 & \\ & x_1 - x_3 \geq 2 \\ & -3x_1 + x_2 + 2x_3 \leq 8 \\ & x_1 + x_2 + 2x_3 = 6 \\ & x_i \geq 0 \quad i = 1, 2, 3 \end{cases}$$



NOT

x	*x'*
0	1
1	0

AND

x	*y*	*xy*
0	0	0
0	1	0
1	0	0
1	1	1

OR

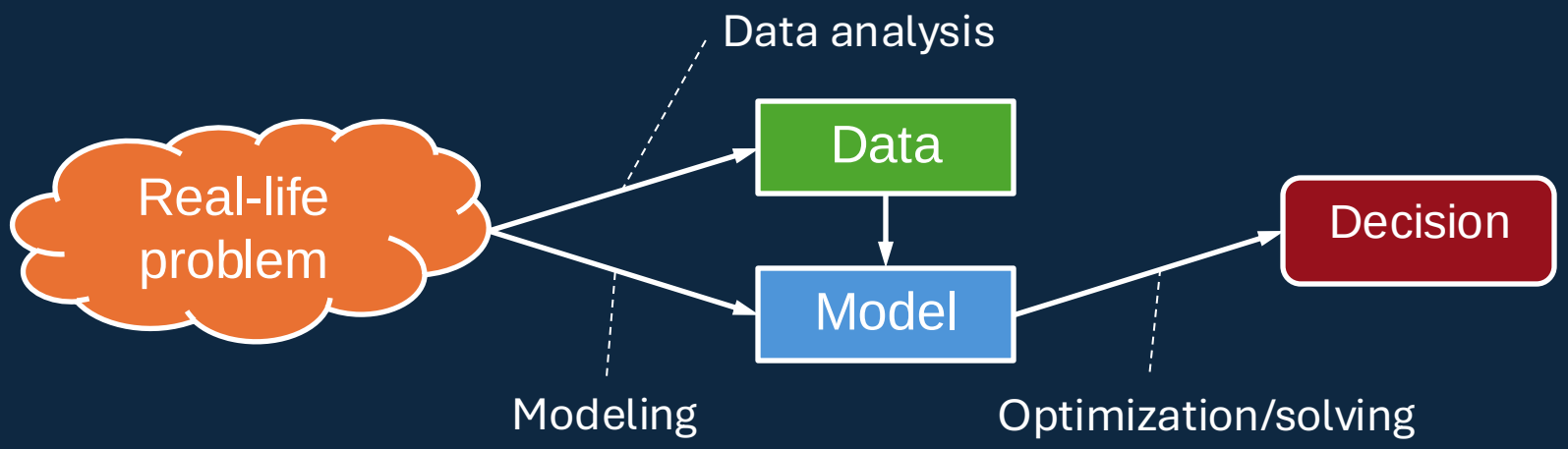
x	*y*	*x+y*
0	0	0
0	1	1
1	0	1
1	1	1

Optimization solvers

Decision

In de praktijk worden *solvers* gebruikt om (complexe) optimalisatie problemen op te lossen.

Discussie: Optimization toegepast op jouw bedrijf



Bouw je eigen algoritme!

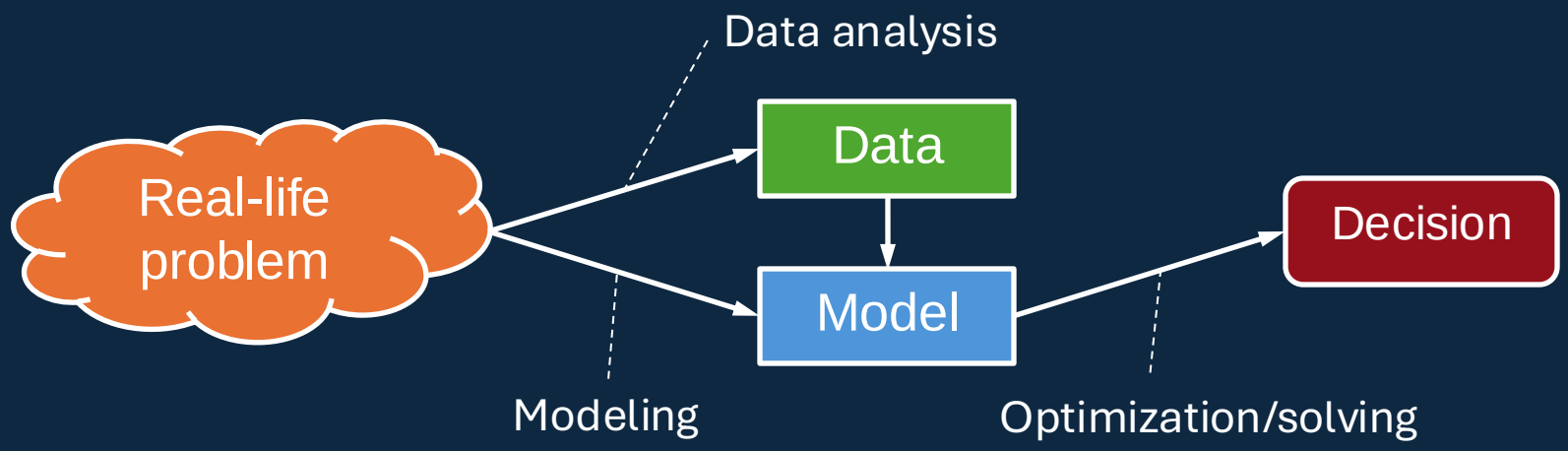
BREINSTEIN DELIVERY

Delivering young talent to professional organizations!

BREINSTEIN DELIVERY

Delivering young talent to professional organizations!

Breinstein Delivery



Breinstein Delivery (v1)

Real-life problem

Salespitch langs verschillende organisaties in Nederland in de kortst mogelijke tijd.

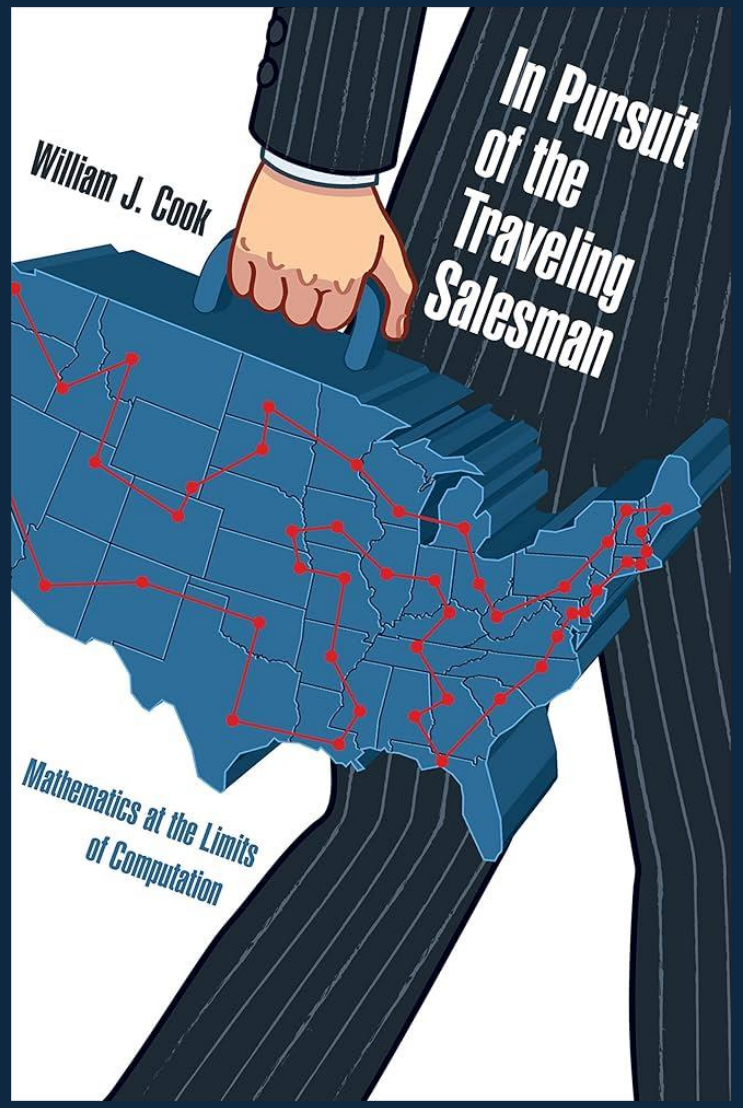
Traveling salesman

Data

- Lijst met steden om te bezoeken
- Coordinaten van steden

Model

- **Objective**: bezoek alle steden met minste totale reistijd
- **Decisions**: kies de volgorde van steden
- **Constraints**:
  - Elke stad mag maar 1x bezocht worden
  - Begin en eindig op dezelfde plek



Number of solutions of TSP

Decision

- Alle mogelijke permutaties van 10 steden
  - [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
  - [1, 2, 3, 4, 5, 6, 7, 8, 10, 9]
  - [1, 2, 3, 4, 5, 6, 7, 10, 8, 9]
  - ...

- # oplossingen: 10! = **3,628,800**

50!

= 30414093201713378043612608166064768844377641568960512000000000000

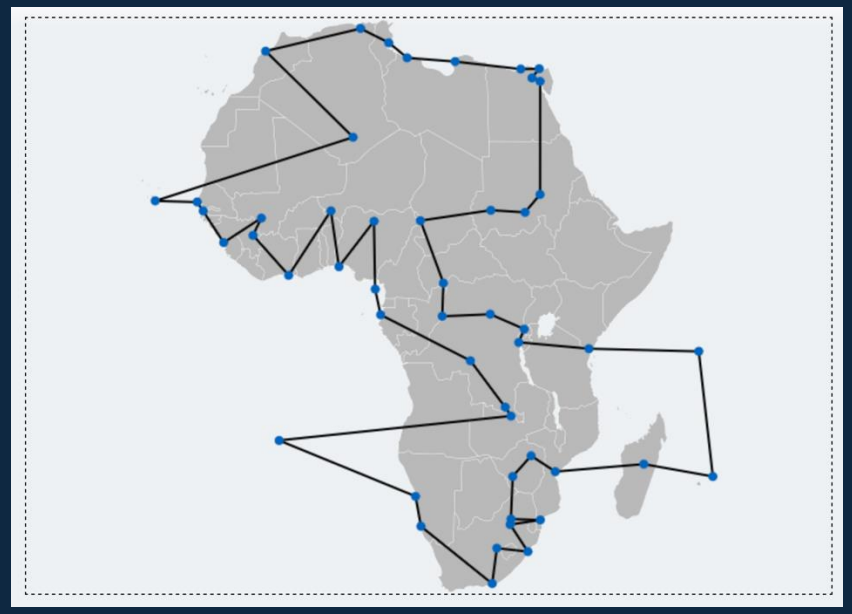
= 3.04×10^{64}

Challenge: Kun jij de beste tour maken?

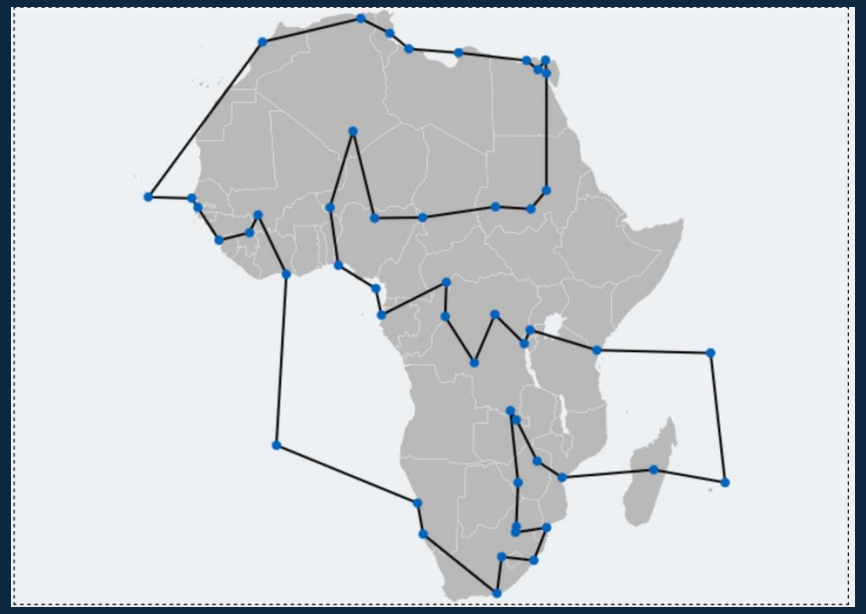
- https://algorithms.discrete.ma.tum.de/graph-games/tsp-game/index_en.html
- Code: E050AF0b1cd84a467d1c0d883e53277f33152d5

Hoe moeilijk is TSP?

- We gebruiken vaak een greedy aanpak
- Maar greedy aanpak is niet optimaal...

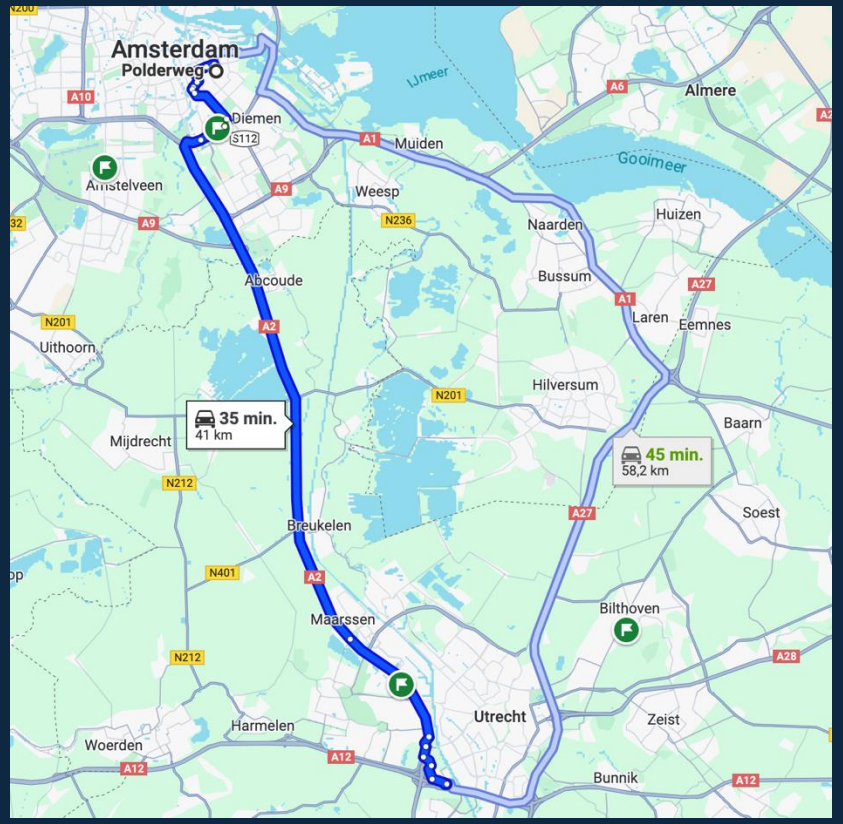


43669.6 km

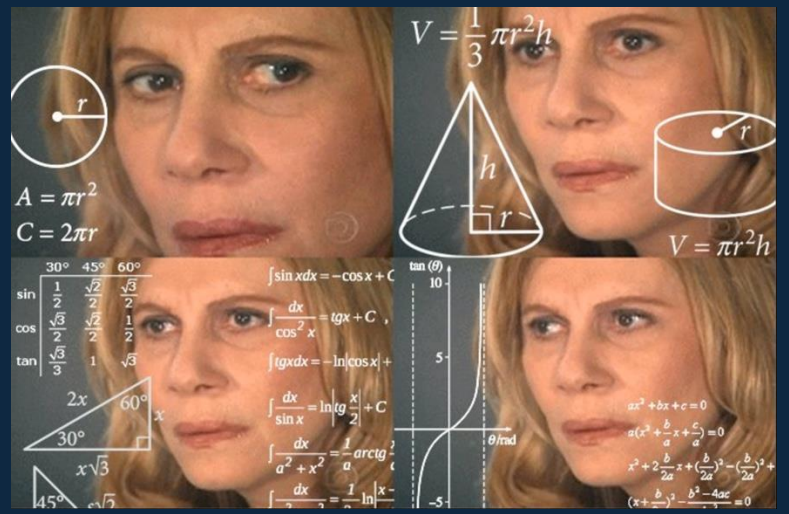


40078.8 km

Shortest path problem

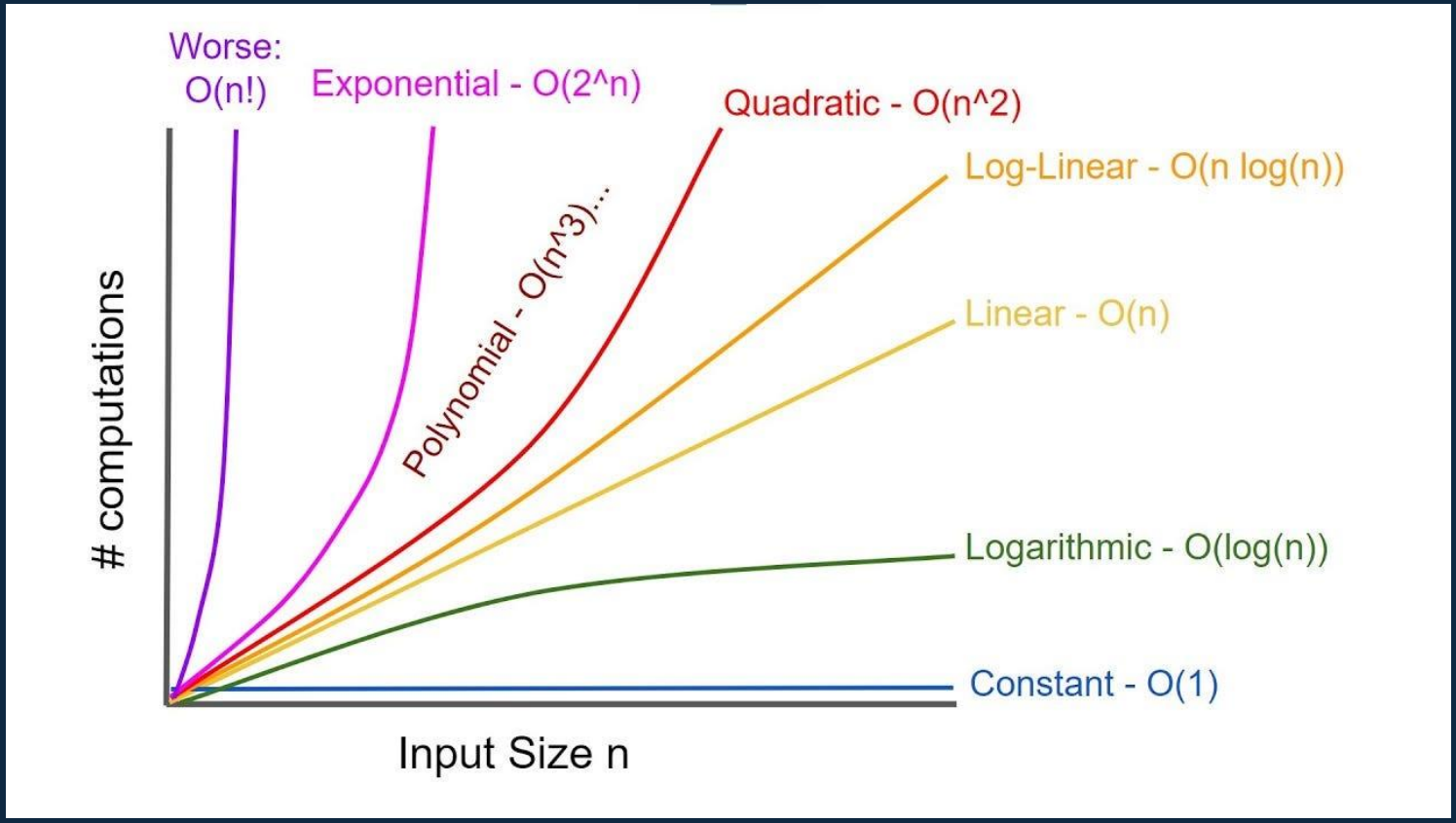


Dijkstra's algoritme lost dit *optimaal* op in $O(n^2)$

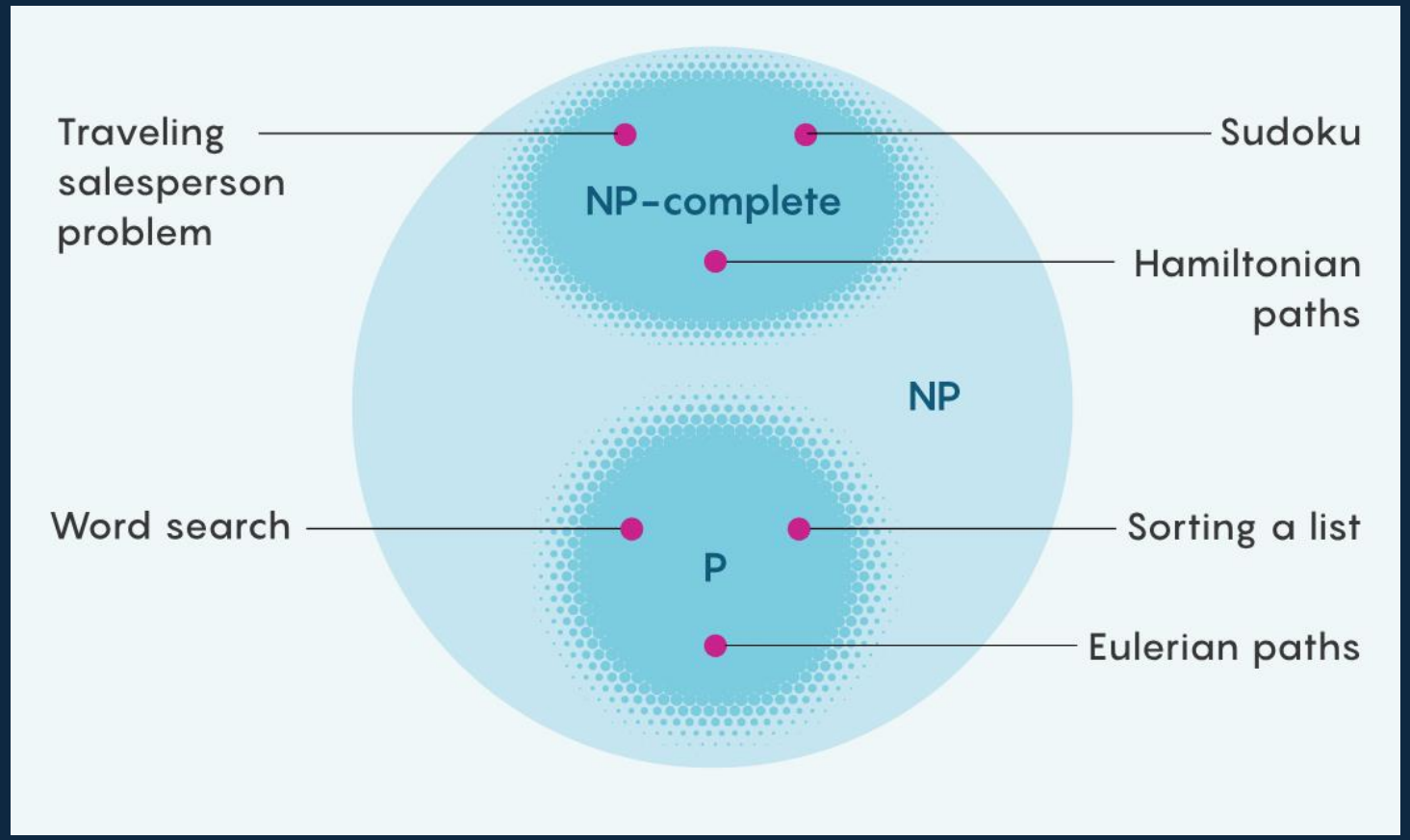


Longest path problem
niet optimaal oplosbaar
in polynomiale tijd!

Algorithmic time complexity



P vs. NP vs. NP-complete



Twee verschillende soorten algoritmes

Exact and heuristic solution

Exact	Heuristic
An exact algorithm is typically deterministic and proven to yield an optimal result.	A heuristic has no proof of correctness, often involves random elements, and may not yield optimal results.
Lot of iterations, lot of constraints → Big computation resources → Long time	Does not explore all possible states of the problem → short time
Exact solution	Optimal (Good solution)

What to use? When?

Theorie vs. praktijk

- In business maakt optimaliteit vaak niet uit (1-2% verschil is geen probleem)
- Veel belangrijker: snelheid, robuustheid, automatisering

Focus van vandaag: heuristieken

Het belang van accurate data

- Optimale route hangt ook af van goeie data
- Distance matrix: afstand of reistijd tussen twee locaties
- Haversine distance
- Map data providers

Nearest neighbor algorithm

Decision

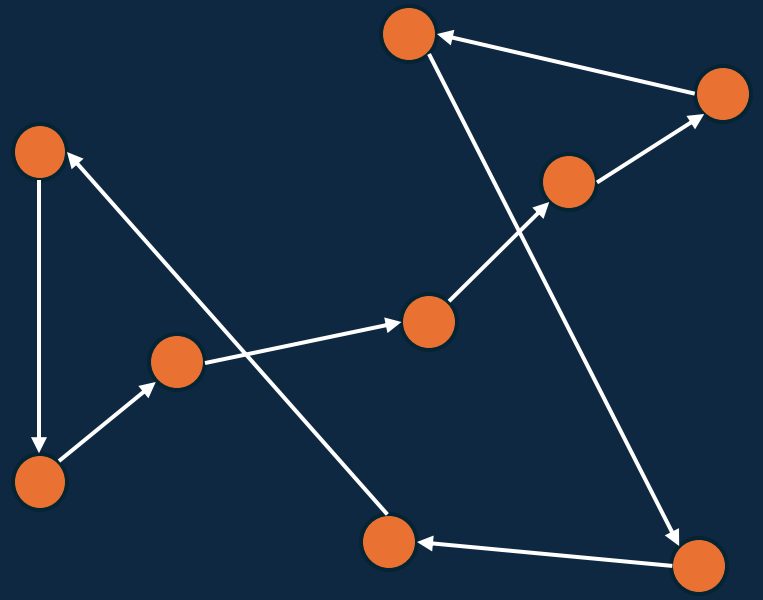
```python
def nearest_neighbor(data):
    current = ... # select random starting point
    tour = [current]

    while ...: # no more unvisited nodes
        neighbor = ... # select nearest neighbor
        tour.append(neighbor)
        current = neighbor

    return tour
```

colab

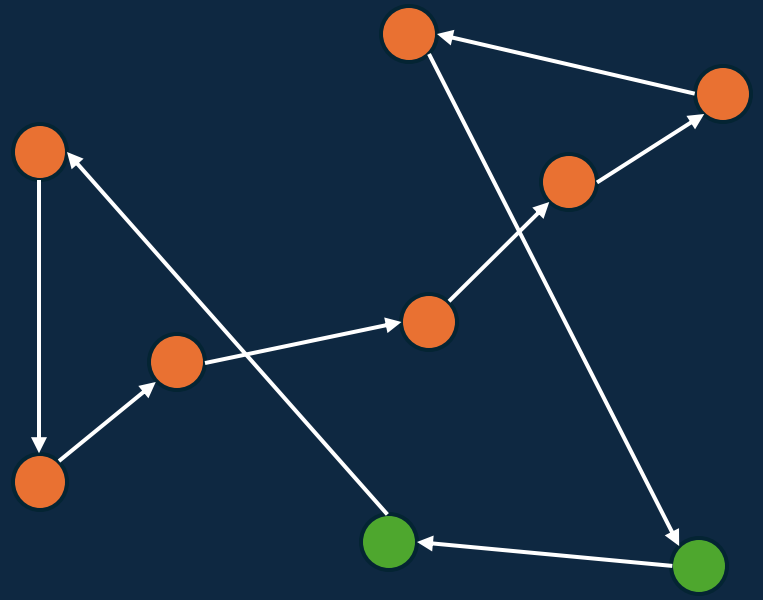
Nearest neighbor algorithm

- Suboptimale oplossing maar wel snel $O(n^2)$
- Meer tijd → geen betere oplossing
- Wat als we het principe van greedy behouden, maar lokaal toepassen?

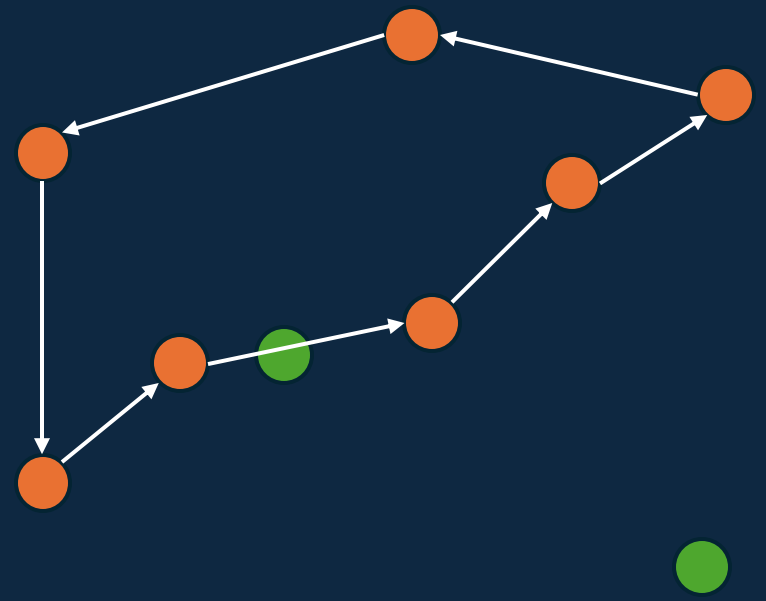
Large neighborhood search



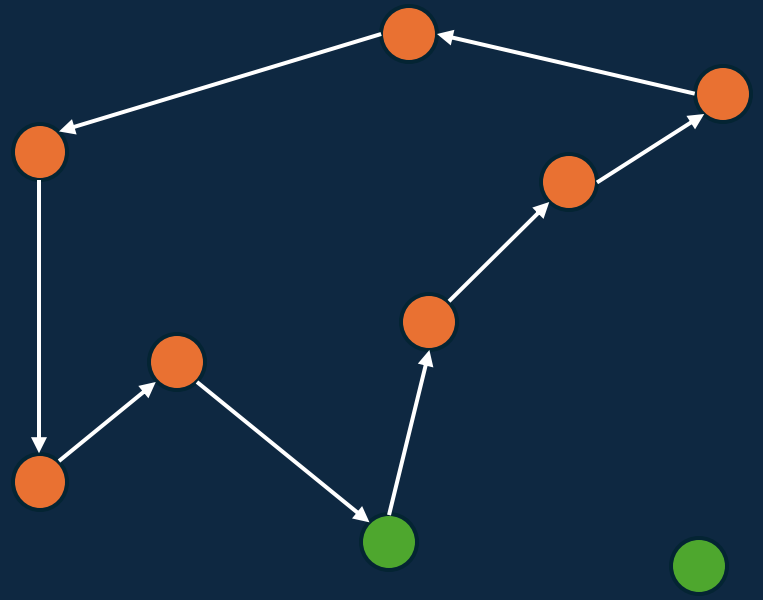
Large neighborhood search



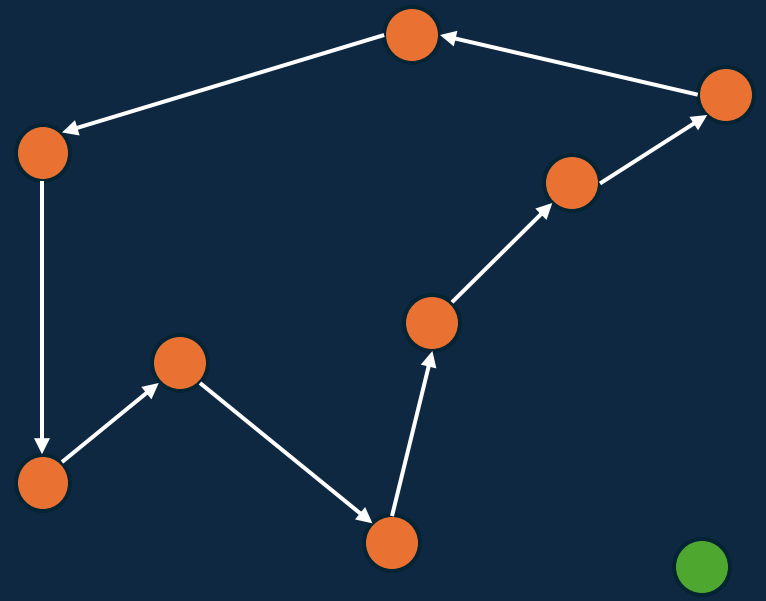
Large neighborhood search



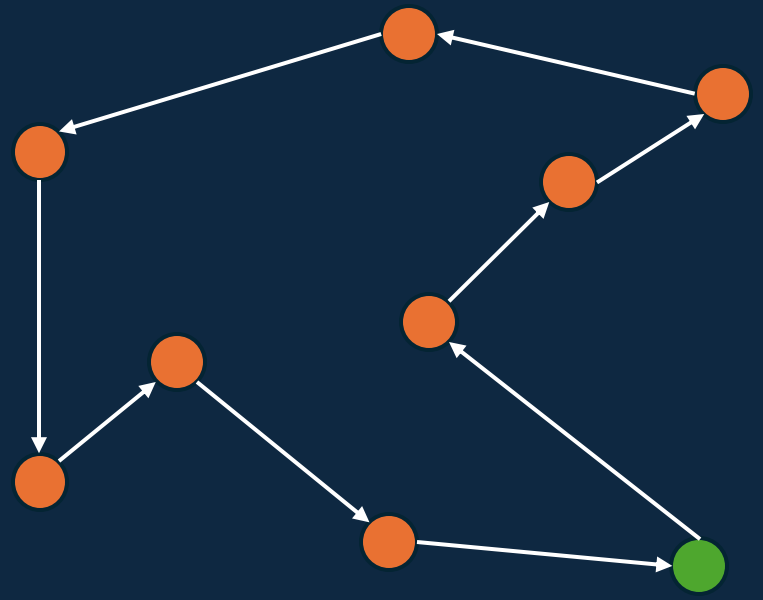
Large neighborhood search



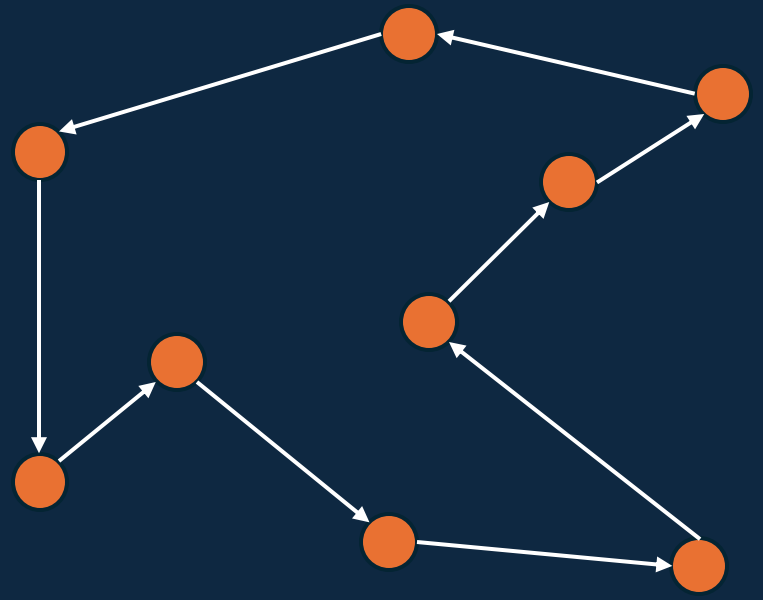
Large neighborhood search



Large neighborhood search



Large neighborhood search



Large neighborhood search

Destroy
- Verwijder een aantal klanten

Repair
- Voeg ze daarna zo goed mogelijk terug in de tour

Accept?
- Beter dan vorige oplossing? Vervang oplossing
- Niet beter? Begin weer opnieuw met de vorige oplossing

Large neighborhood search (basic)

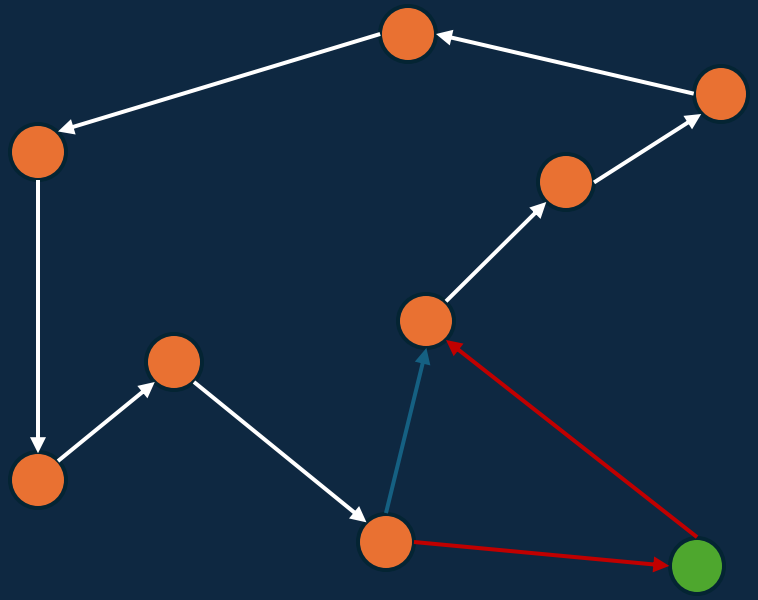
Decision

```python
def large_neighborhood_search(
    data: Data,
    num_iterations: int,
    num_removals: int = 5,
) -> list[int]:
    curr_tour = nearest_neighbour(data)
    curr_cost = cost(curr_tour, data)

    for _ in range(num_iterations):
        new_tour = copy(curr_tour)
        removed = ... # remove random clients

        for client in removed:
            new_tour = ... # re-insert removed clients

        new_cost = cost(new_tour, data)
        if new_cost < curr_cost:
            curr_cost = new_tour
            curr_tour = cand_tour

    return curr_tour

```

colab

Large neighborhood search (fast)

Efficient insertion evalueren: O(1)



$$d_{ijk} = d_{ik} + d_{kj} - d_{ij}$$

TSP is niet realistisch

- Meerdere voertuigen
- Capaciteit aan voertuigen
- Verschillende start locaties
- Werkduur restricties
- Tijdsvensters
- ...

Breinstein Delivery (v2)

Real-life problem

Meerdere* account managers willen een salespitch houden langs verschillende organisaties in Nederland in de kortst mogelijke tijd.

Vehicle routing problem

Model

- Depots
- Clients
- Vehicles
- Distances and durations

Vehicle routing problem solver

PyVRP 

https://github.com/PyVRP/PyVRP/

colab

Gebruik PyVRP voor de volgende data set

- Pas PyVRP toe op de data set `depots.csv`, `clients.csv` en `vehicles.csv`
- Gebruik de haversine distance
- Neem aan dat 60km p/u wordt gereden

colab

Dashboard

- Raw input data
- Input data voor solver
  - Bijv. distance matrix
- Solver runnen
- Solver output mappen naar gewenste output
- Visualisatie

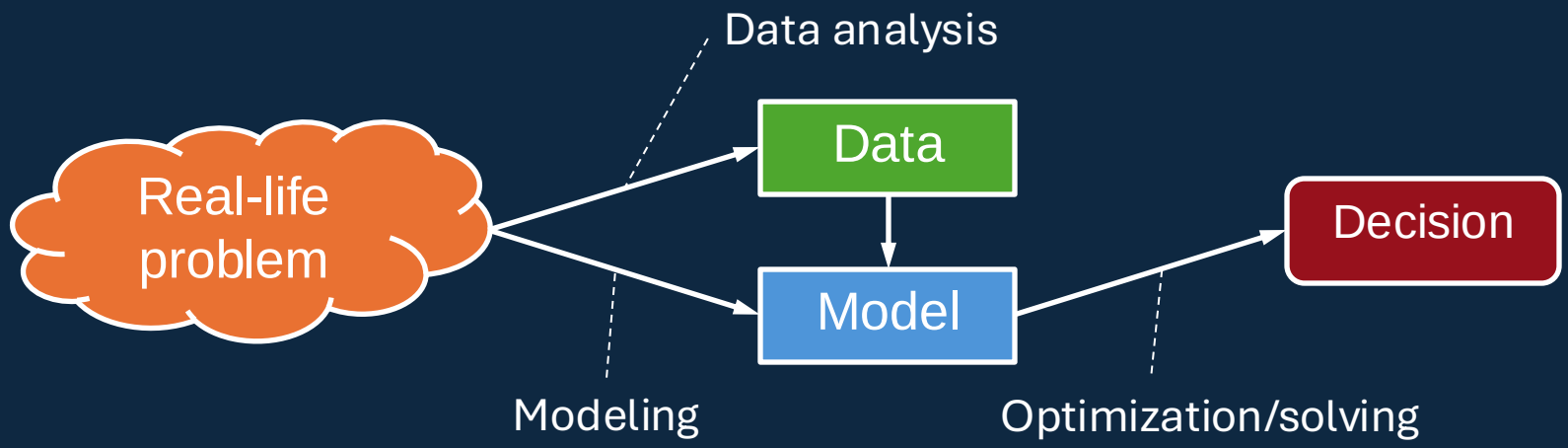
Uitleg van de optimalisatie challenge

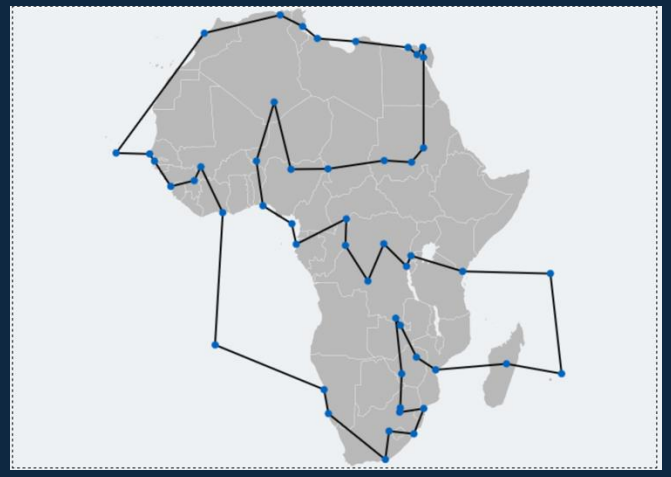
Large-scale VRP probleem (5K nodes)
PyVRP is gemaakt voor 1K nodes
Los dit probleem zo goed mogelijk op!

Tips!

- Gebruik decompositie methodes → clustering?
- Denk na over time limit!
- Begin eerst met de kleine dataset (100 nodes)

Wat hebben we vandaag geleerd?

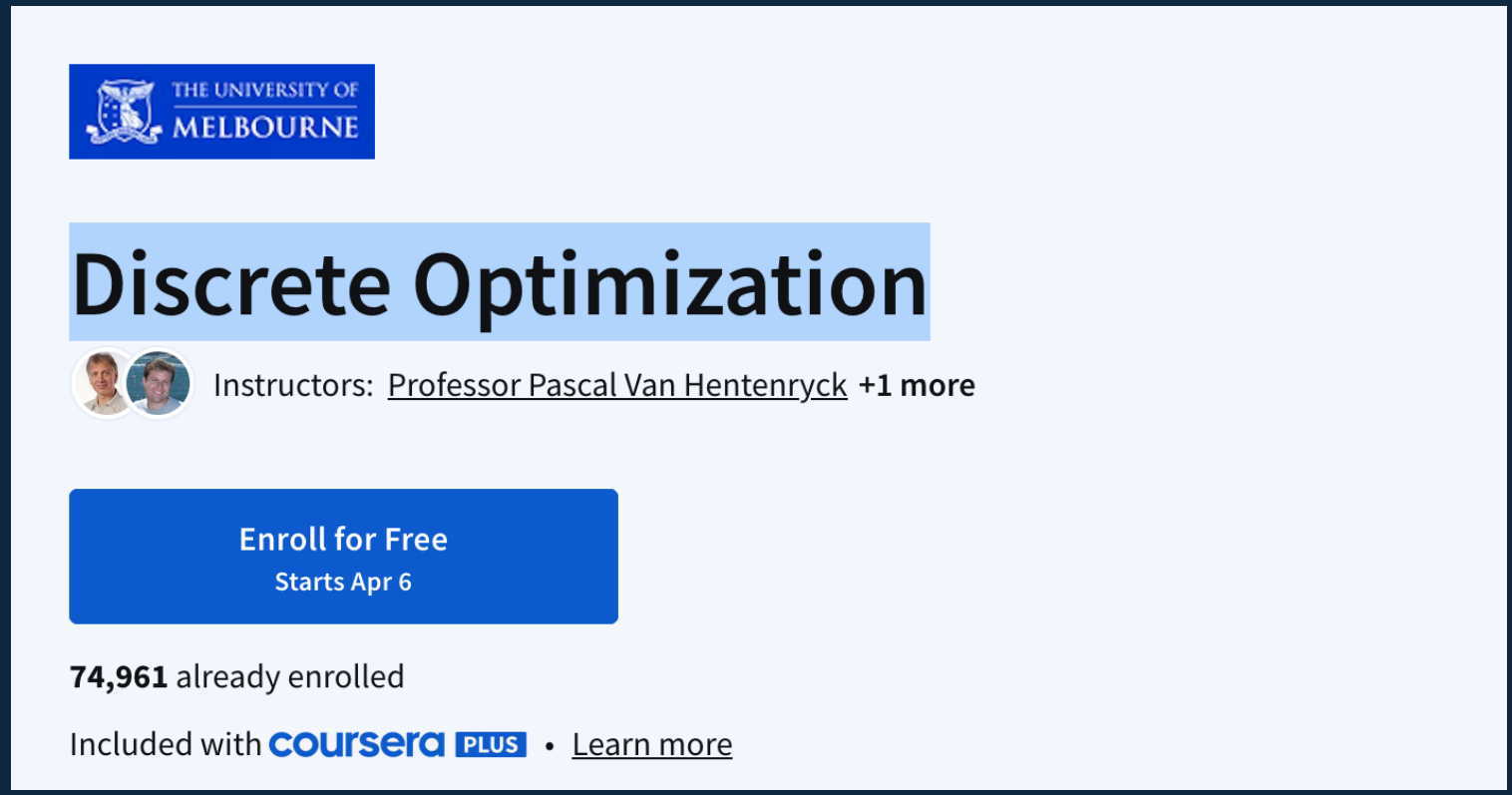




Meer leren over optimalisatie?

https://gurobi.com/

Meer leren over optimalisatie?



https://www.coursera.org/learn/discrete-optimization

Optimization

Leon Lan
Maandag 7 april 2025
l.lan@vu.nl

Evaluatieformulier