



A JOURNEY THROUGH UNCERTAIN OPTIMISATION

Thibaut Cuvelier

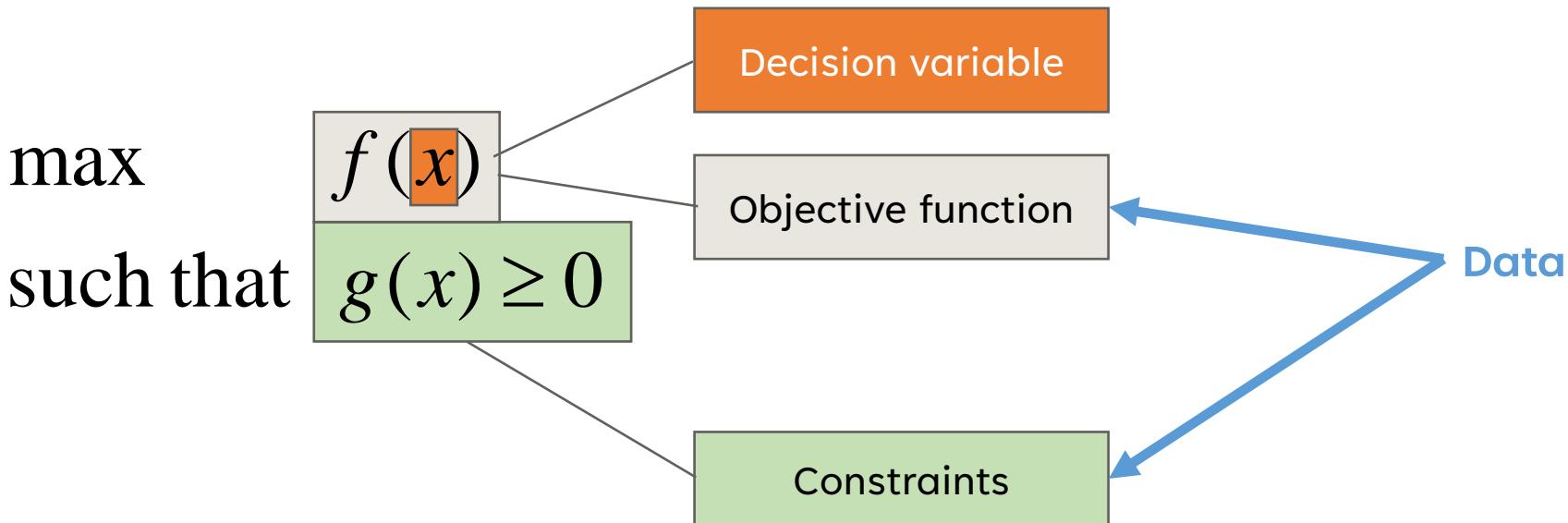
Operations research?

- Machine learning: how do I predict the amount of sales for Christmas 2023?
- Nice. What do you do with that? A dashboard?
- How to bring that product efficiently to clients? Use operations research!
- Other questions:
 - How to get the largest amount of treats in a minimum amount of time for Halloween?
 - From which distribution centres should you serve the demand?
 - How many trucks to serve a given neighbourhood?
 - How to pack parcels in a truck?
 - Where to build a new distribution centre?
 - How many machines of what type for a new data centre and future computational load?
 - ...



Operations research: what does it look like?

- Typically, your problem will have:
 - Decision variables: actions you can take
 - Objective function: minimise the cost, maximise the reliability, etc.
 - Constraints: demand to fulfil, technical possibilities, etc.



Why uncertainty in optimisation?

Perfect
demand
forecast?

Perfect
price
forecast?

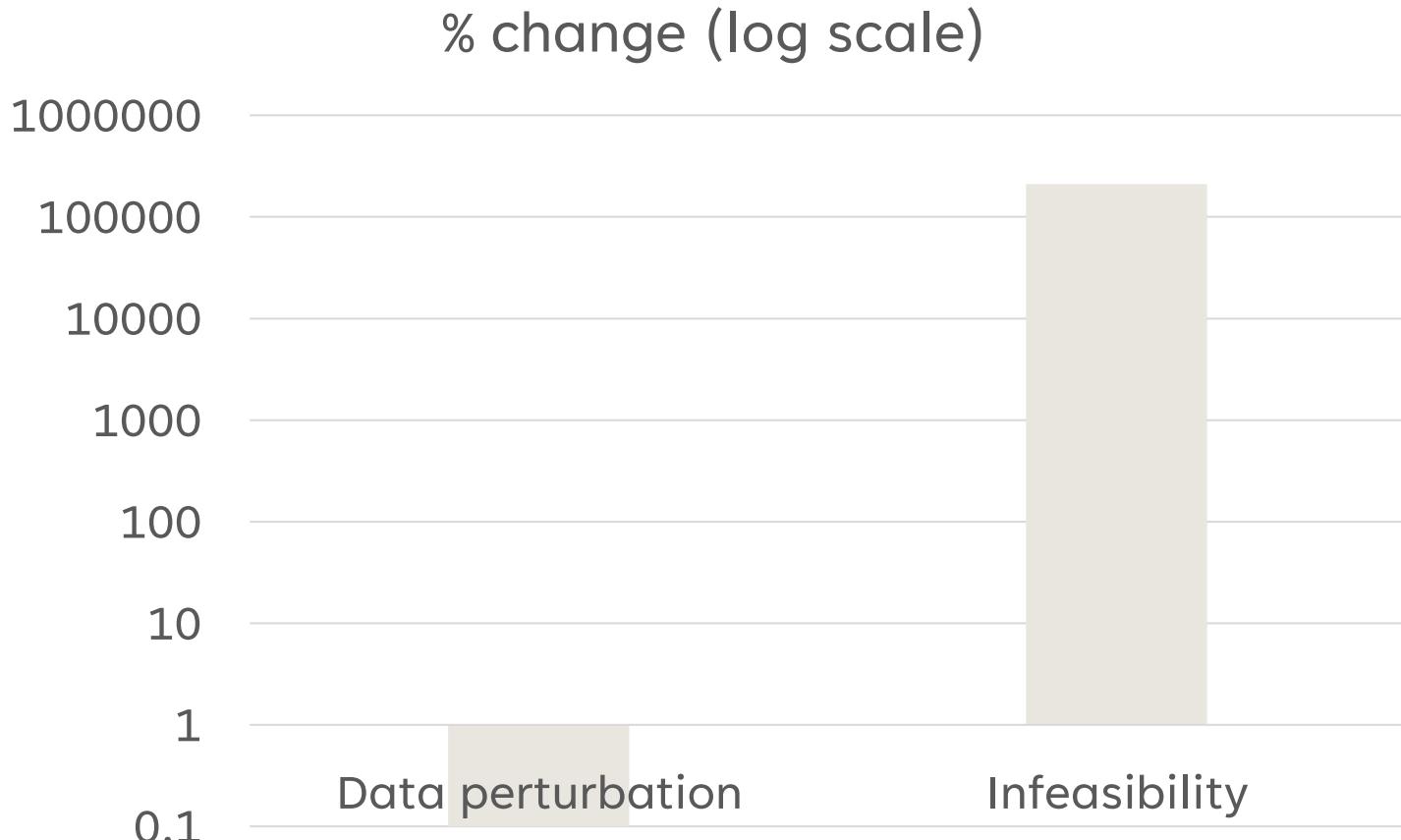


Perfect
travel-time
forecast?

Perfect human
implementation?

Why uncertainty in optimisation?

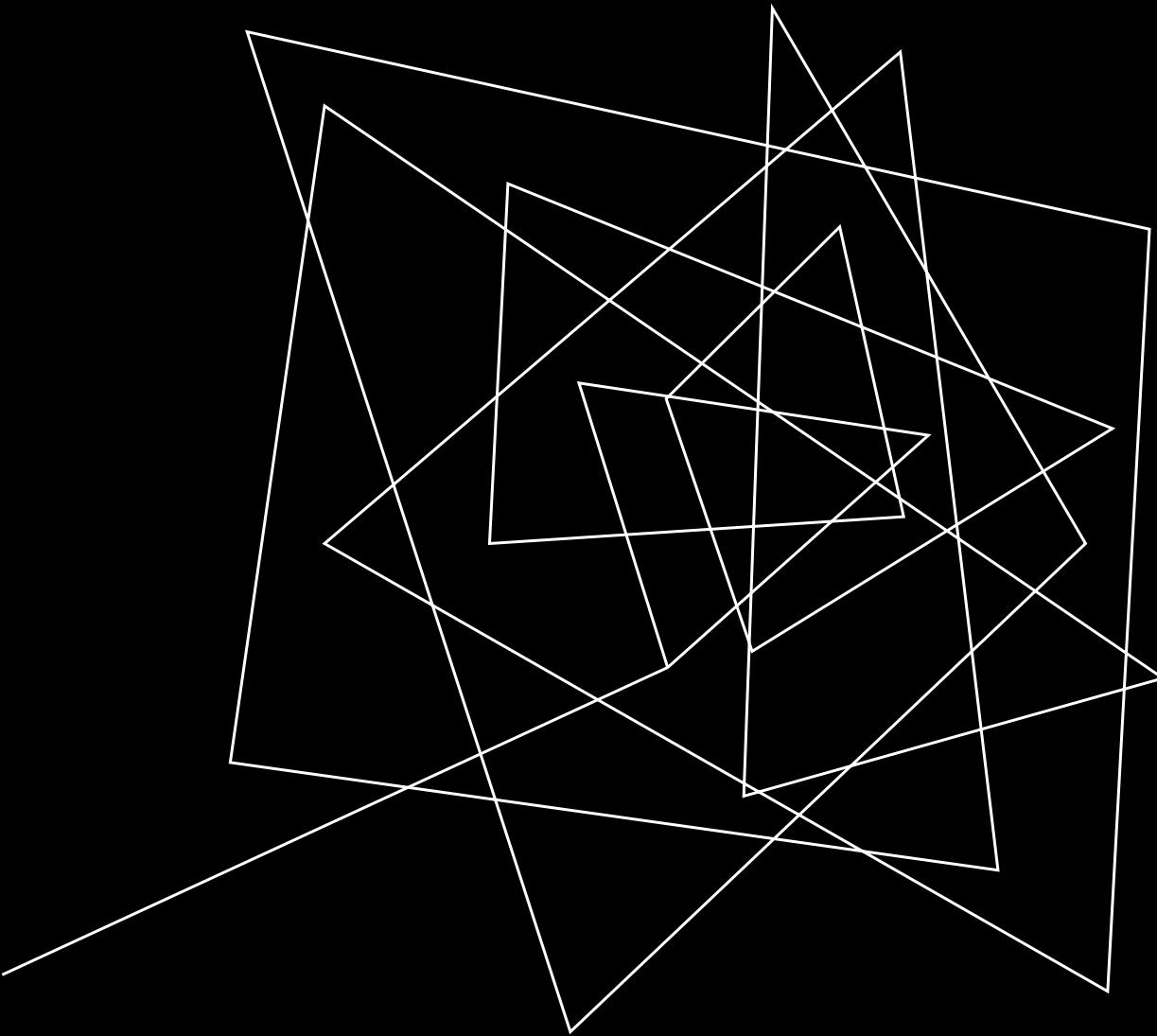
- **Small** perturbations can lead to **highly infeasible** solutions
- NETLIB data set (PILOT4):
 - 0.1% change in a coefficient
 - 210,000% constraint violation



Source: A. Ben-Tal, L. E. Ghaoui, and
A. Nemirovski. Robust Optimization. 2009
(introduction)

A Journey through Uncertain Optimisation

- How to model uncertainty?
- How to scale uncertain models?
- How to learn from the uncertainty?
- How to use uncertainty to solve faster?

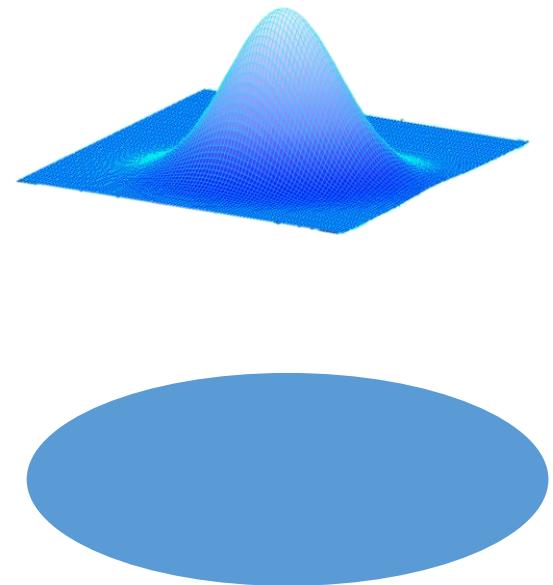


How to model
uncertainty?

How to model uncertainty?

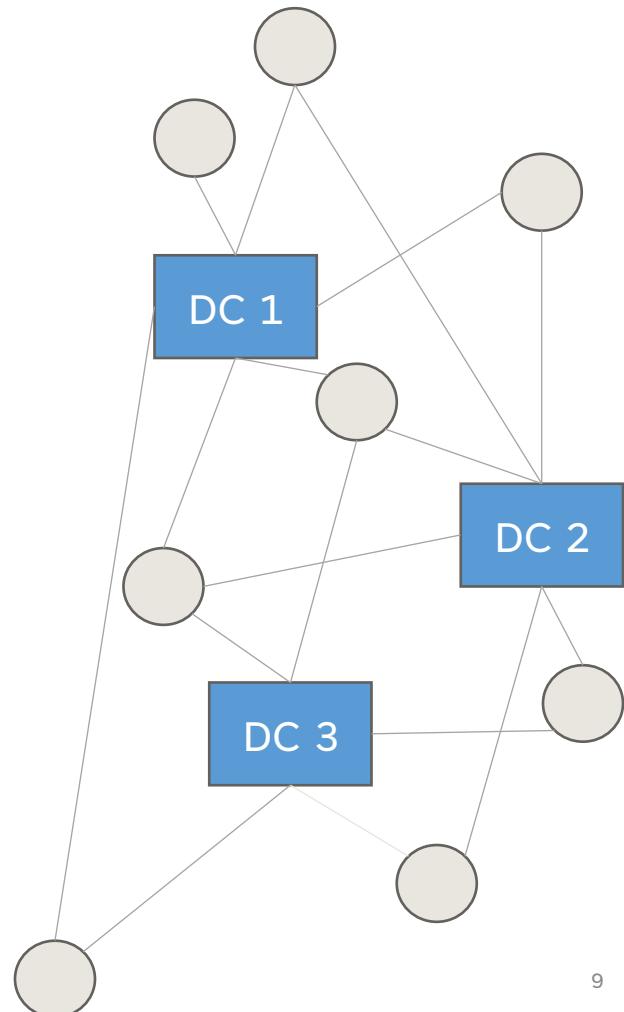
Two major approaches:

- Based on probabilities: **stochastic optimisation**
 - Oldest: dates back from 1955
 - Use a probability distribution for the uncertain parameters
 - Optimise for the average cost, the 95th percentile, etc.
 - Discretise based on scenarios
- Based on uncertainty sets: **robust optimisation**
 - Inspired by game theory
 - Define the plausible values for the uncertain parameters
 - Optimise for the worst-case scenario in the set of plausible values
 - Designed to be computationally lighter than stochastic optimisation



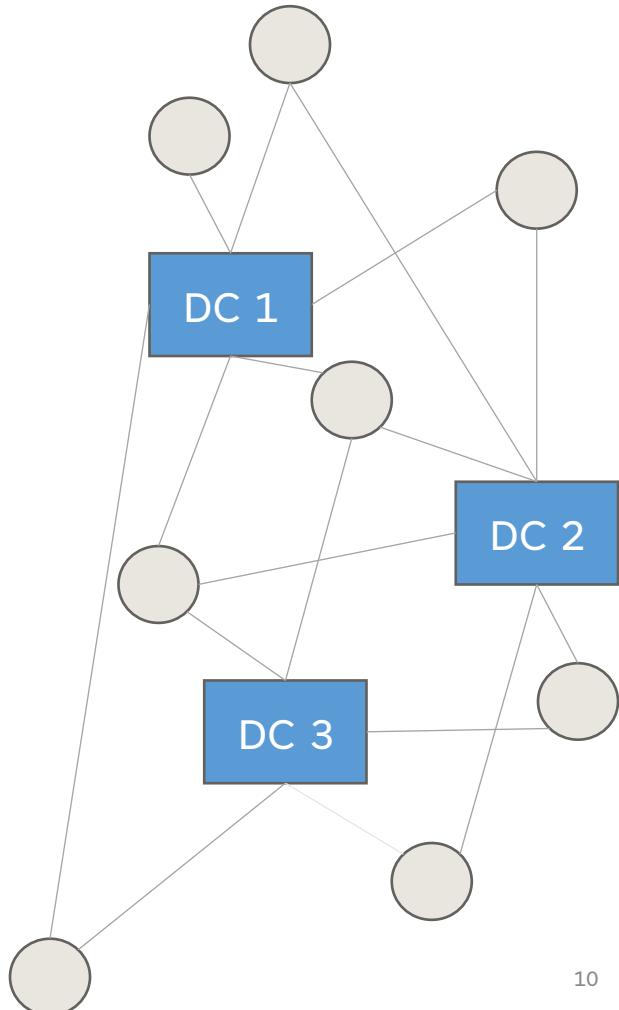
Example: capacitated facility location

- Decision variables: open a distribution centre (DC) i , x_i
- Dependent variables: serve a city j from a DC i , $y_{i,j}$
- Objective: minimise the costs of opening DCs and serving cities
- Data:
 - Demand for each city
 - Capacity for each DC
 - Cost to serve a city from a DC
 - Cost to open a DC
- Constraints:
 - Serve demand from open DC only up to their capacity
 - Don't exceed the DC capacities



Example: capacitated facility location

$$\begin{aligned} \text{min} \quad & \sum_i f_i x_i + \sum_i \sum_j c_{i,j} y_{i,j} \\ \text{s. t.} \quad & \sum_j y_{i,j} \leq \text{capacity}_i x_i, \text{ for all DCs } i \\ & \sum_i y_{i,j} \geq \text{demand}_j, \text{ for all cities } j \\ & x_i \in \{0,1\} \end{aligned}$$



Example: capacitated facility location

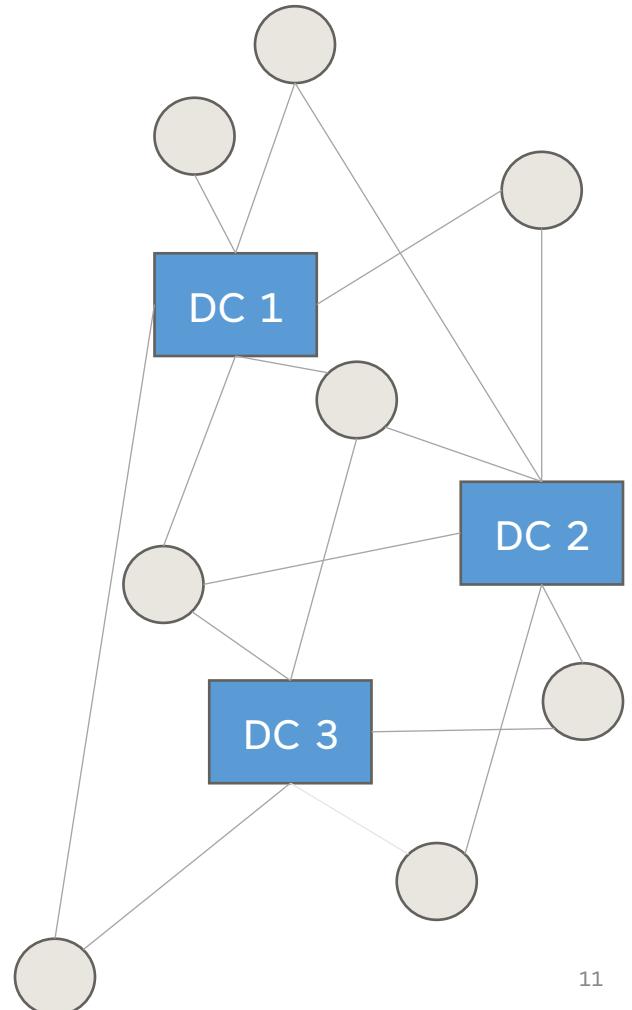
- Stochastic model: several demand scenarios
 - “Here and now” decisions: which DCs to open **now**
 - “Wait and see” decisions: amounts from DCs to cities
 - Minimise **expected** cost
 - Represent probability distribution with scenarios (index: s)

$$\min \quad \sum_i f_i x_i + \sum_s p_s \sum_i \sum_j c_{i,j} y_{i,j}^s$$

$$\text{s. t.} \quad \sum_j y_{i,j}^s \leq \text{capacity}_i x_i, \text{ for each DC } i, \text{ for each scenario } s$$

$$\sum_i y_{i,j}^s \geq \text{demand}_j^s, \text{ for each city } j, \text{ for each scenario } s$$

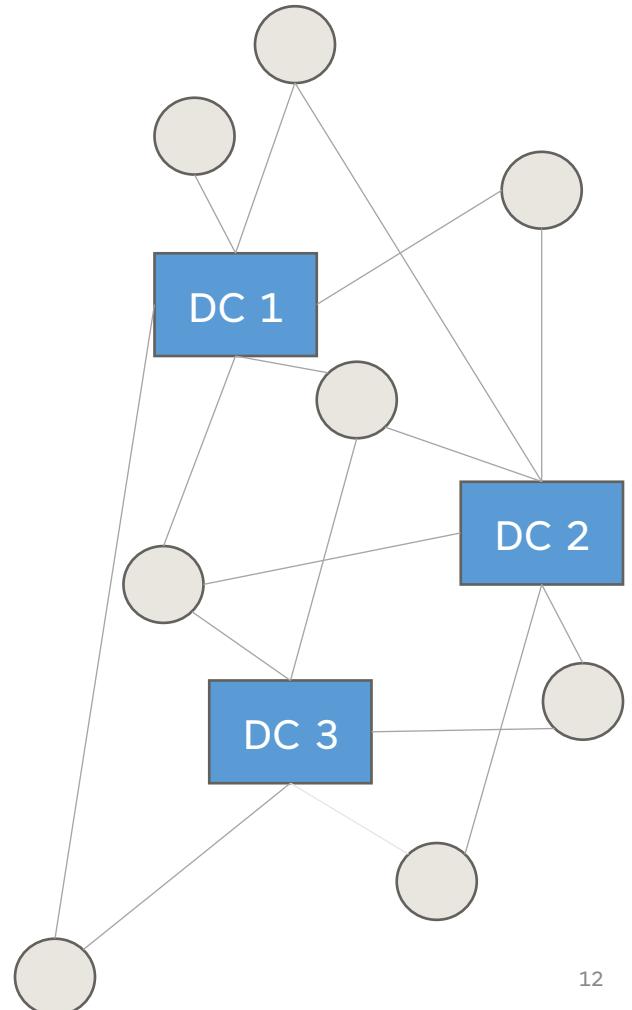
$$x_i \in \{0,1\}$$



Example: capacitated facility location

- Robust model: one uncertainty set per city
 - Soyster (“box”) set: $\text{demand}_j \in [\underline{\text{dem}}_j, \overline{\text{dem}}_j]$
 - Worst case: every city has the maximum demand, $\overline{\text{dem}}_j$

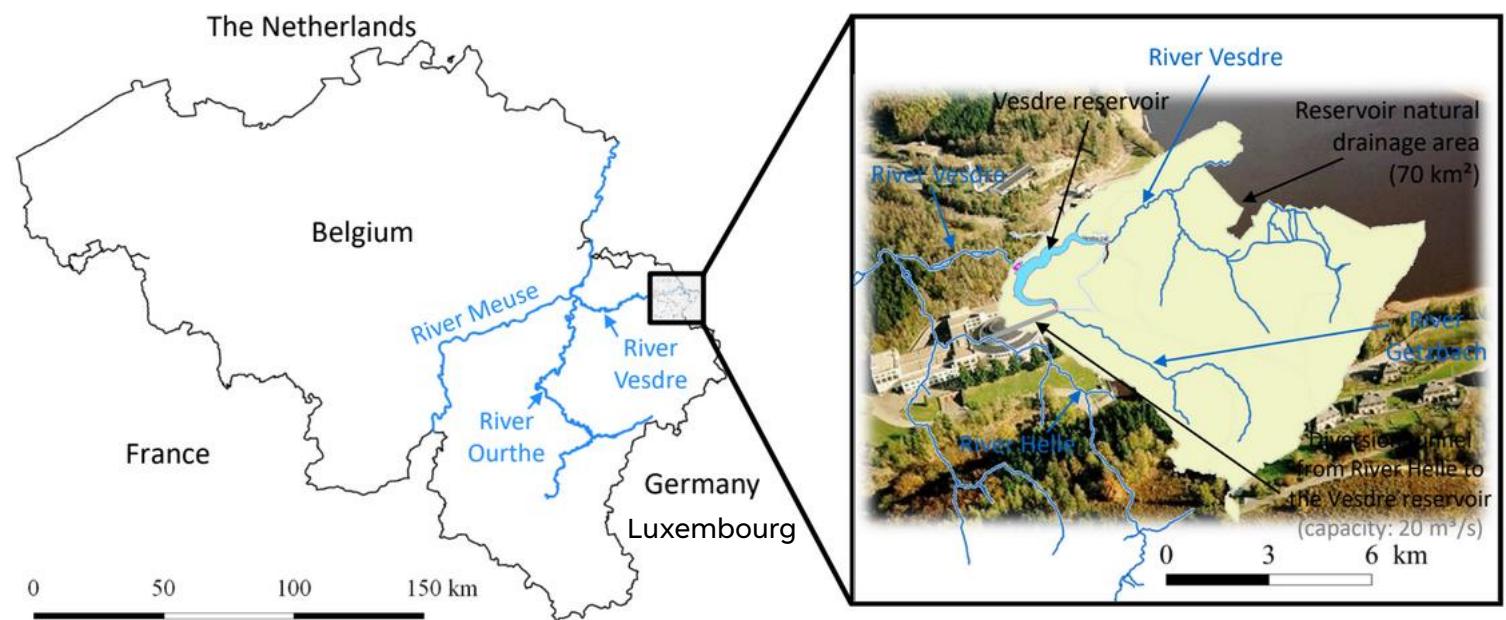
$$\begin{aligned} \min \quad & \sum_i f_i x_i + \sum_i \sum_j c_{i,j} y_{i,j} \\ \text{s. t.} \quad & \sum_j y_{i,j} \leq \text{capacity}_i x_i, \text{ for all DCs } i \\ & \sum_i y_{i,j} \geq \overline{\text{dem}}_j, \text{ for all cities } j \\ & x_i \in \{0,1\} \end{aligned}$$



Case study: water-reservoir management

How to manage a water reservoir in a **very** risk-averse way?

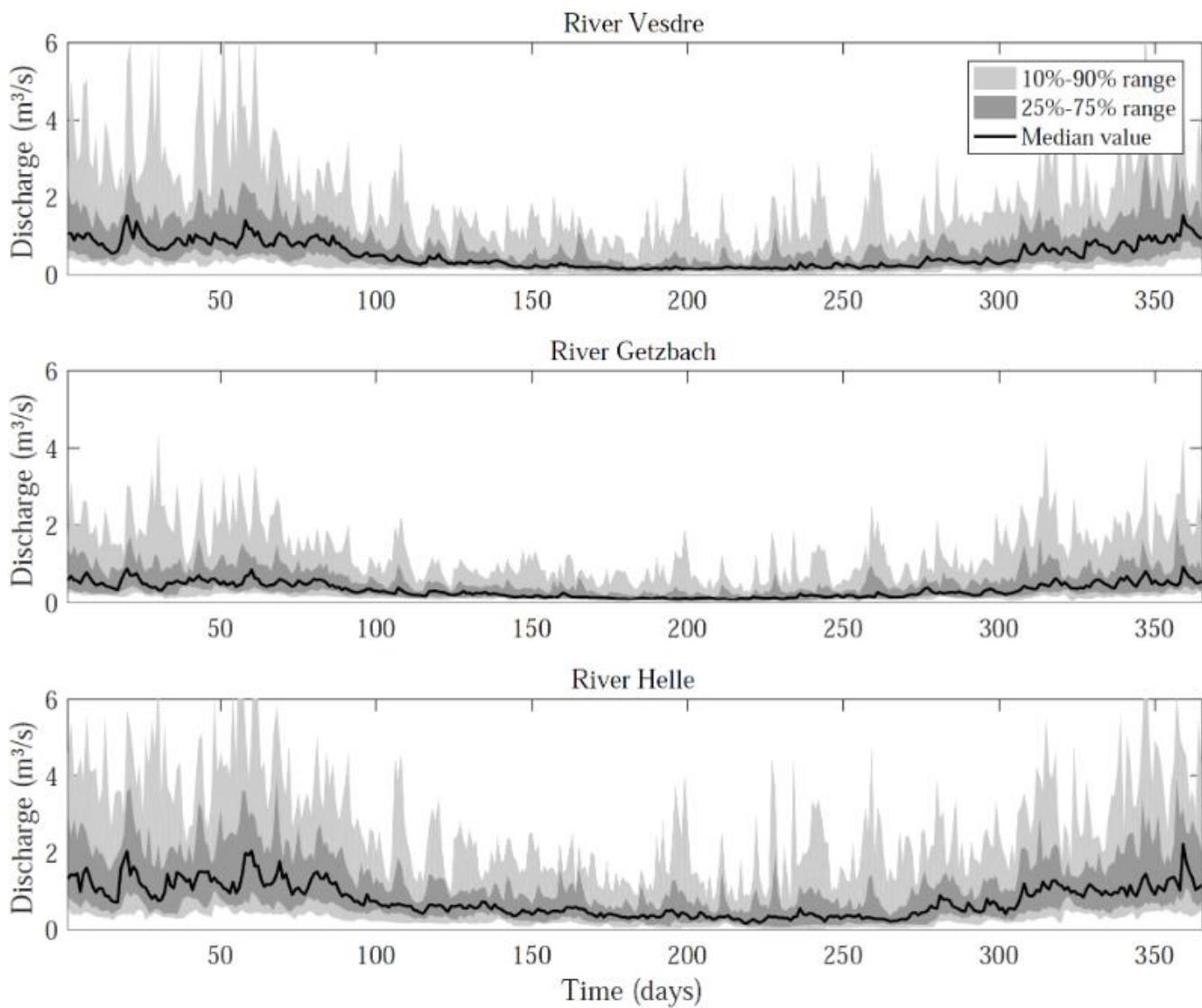
- Uncertainty: rain
- Expected result: a new minimum-water-depth *rule curve*
- Major constraint: drinking water supply for two years, whatever the weather conditions



Total drainage area $\sim 100+$ km²

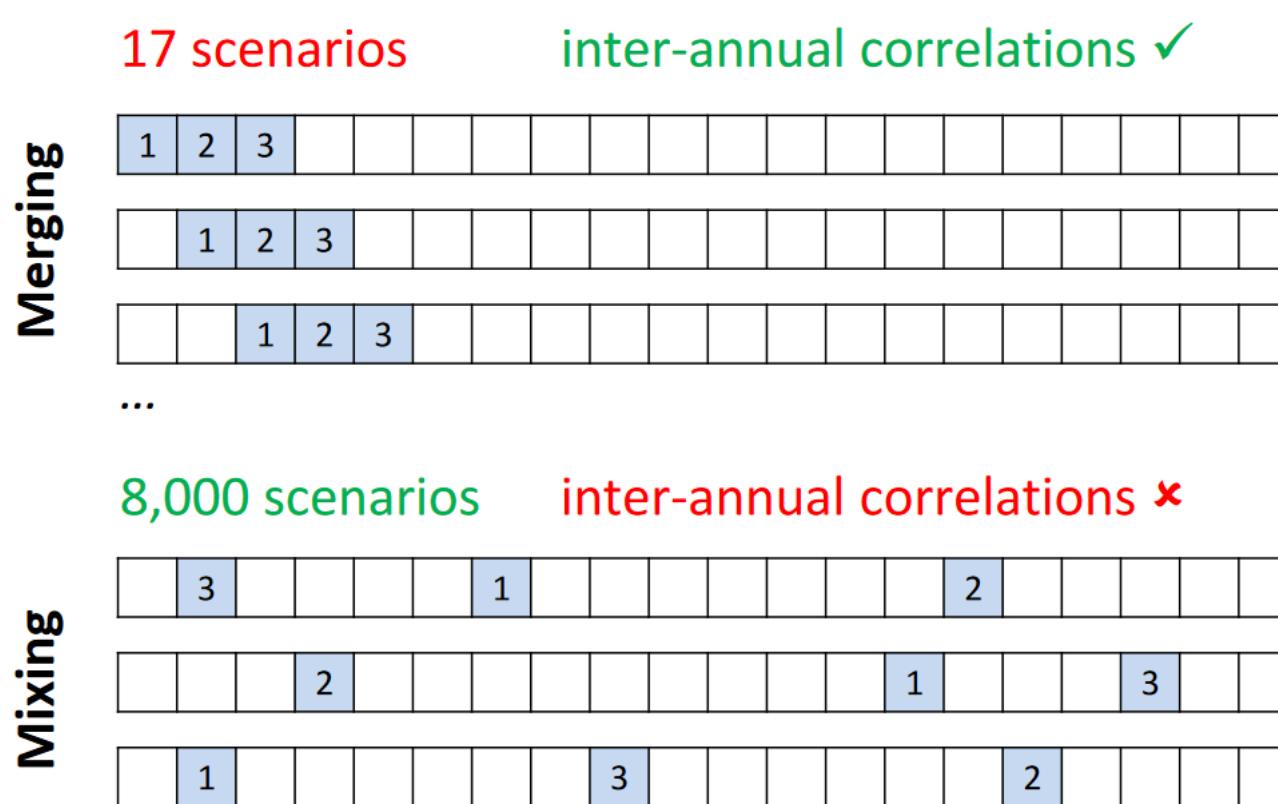
Case study: water-reservoir management

- Stochastic approach:
 - Use historical inflow data: only 20 years of measurements
 - Upper envelope of the water depths
- Robust approach:
 - For each week of the year, use 95% confidence intervals around the average inflow
- Special trick: receding horizon control
 - Solve once per day/week with a horizon of two years



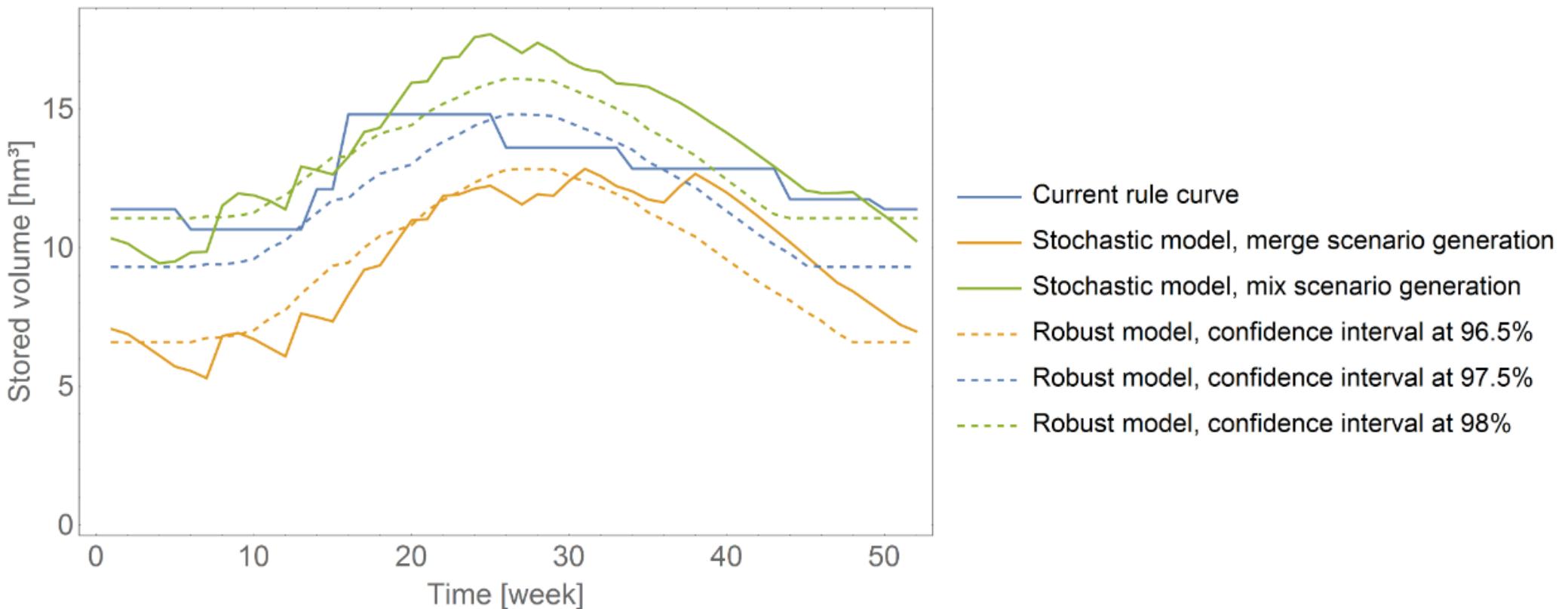
Case study: water-reservoir management

- Data augmentation: merging and mixing
- Based on hydrological years (starting in October)
- Always keep intra-year correlations



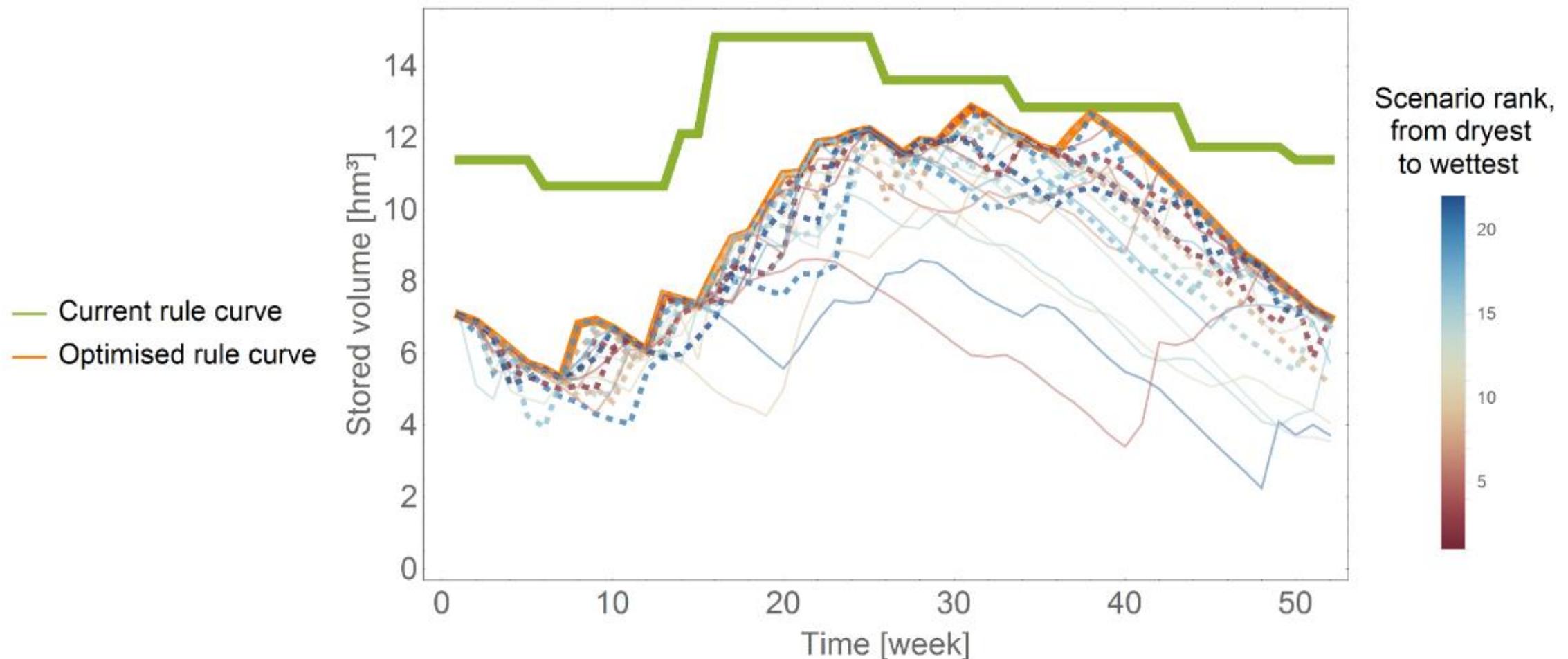
Case study: water-reservoir management

- Mixing requires the deepest water levels
- Robust evaluation of stochastic solutions: closest confidence interval



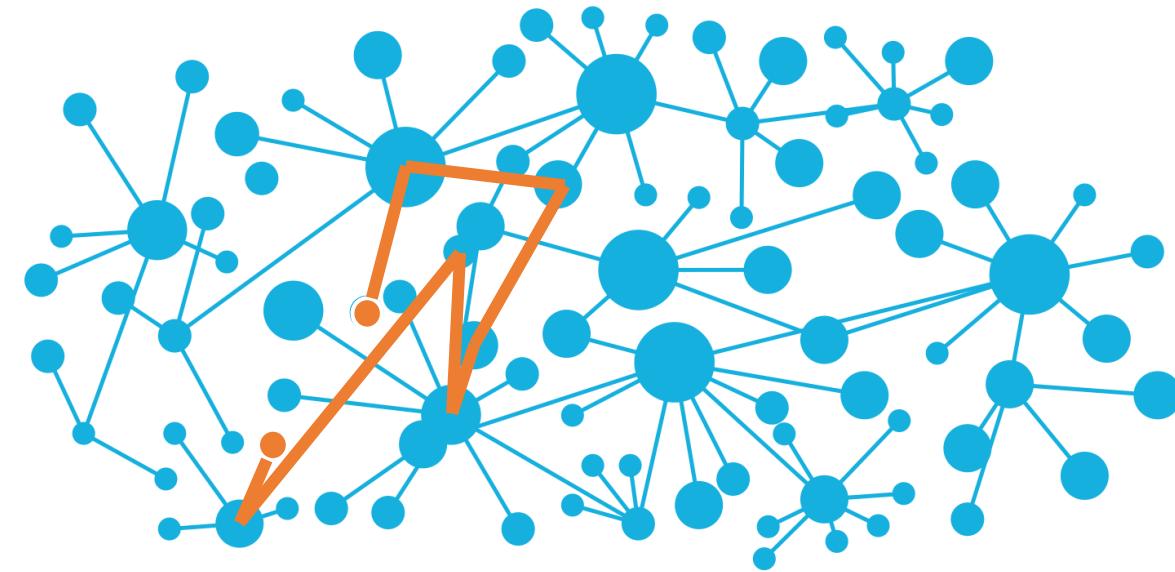
Case study: water-reservoir management

Stochastic evaluation: what scenario contributes to each point on the rule curve?



Case study: network optimisation

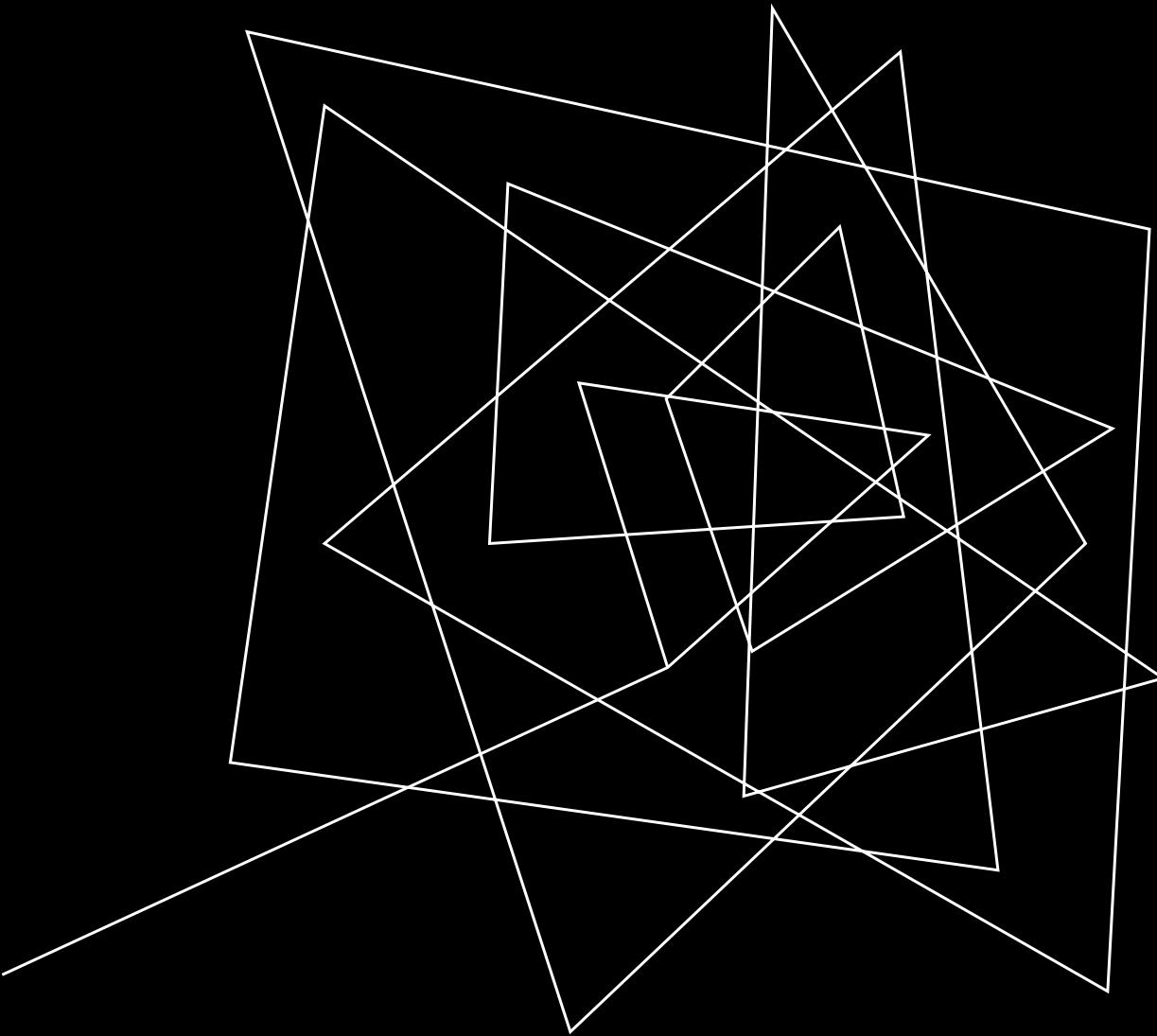
- Given: a network topology and pairs of nodes that communicate
- How to route the traffic while having the lowest congestion?
 - Formally: minimise the maximum load in the network
 - Load: throughput divided by capacity
- Source of uncertainty: traffic!
 - Not easy to forecast
 - Patterns evolve over time: Netflix, large-scale ML, etc.
- Bonus points: don't change the routing too often



Case study: network optimisation

- Major problem: **optimising the routing without uncertainty is not easy**
- How to model uncertainty?
 - Scenarios? Quickly very costly to solve!
 - Robust? Hard to cover the interesting cases!
 - Oblivious? Cover **all** cases!
- Oblivious routing: best congestion whatever the traffic conditions
 - Theoretical guarantee: performance compared to the best routing for given conditions





How to solve
uncertain
problems at
scale?

How to solve uncertain problems at scale?

Solving uncertain problems can be hard:

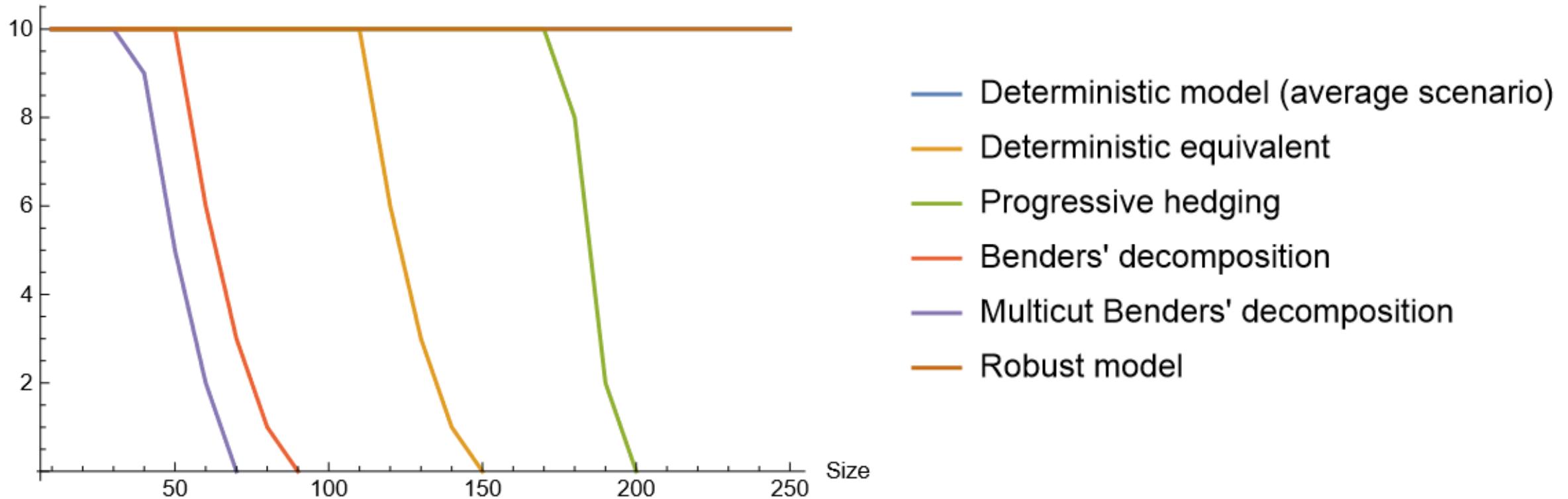
- Stochastic: extremely large
- Robust: no longer linear

How to improve the running times?

- Stochastic optimisation: decomposition along scenarios
 - Exact: Benders' decomposition
 - Approximate: progressive hedging
- Robust optimisation: rewrite the worst-case scenario
 - Iterative process: determine one worst-case scenario at a time
 - Reformulation: implicitly characterise the worst-case scenario

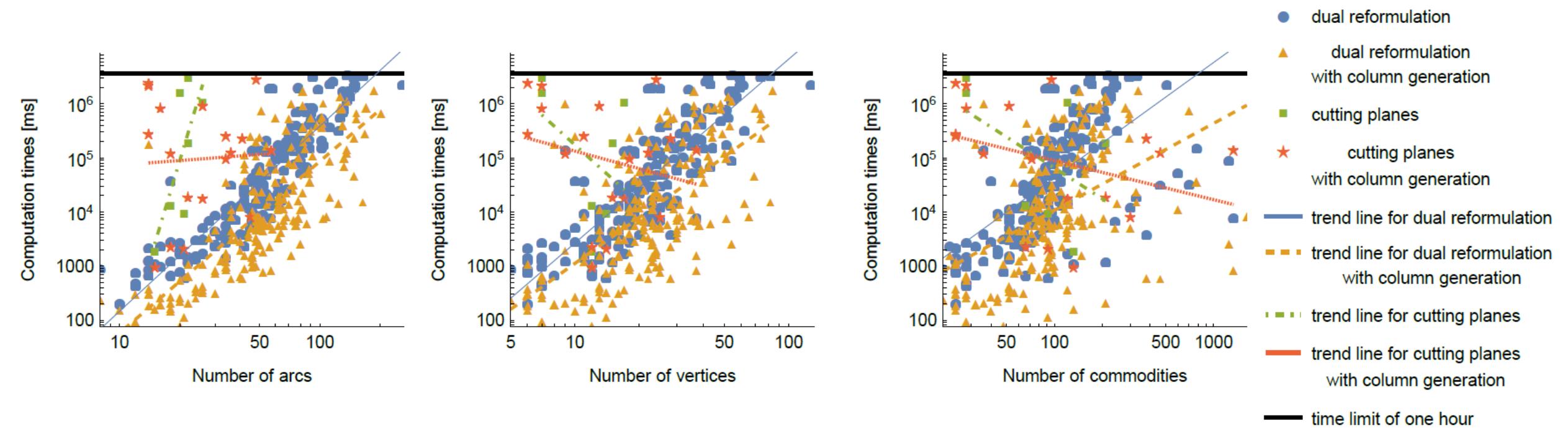
Capacitated facility location

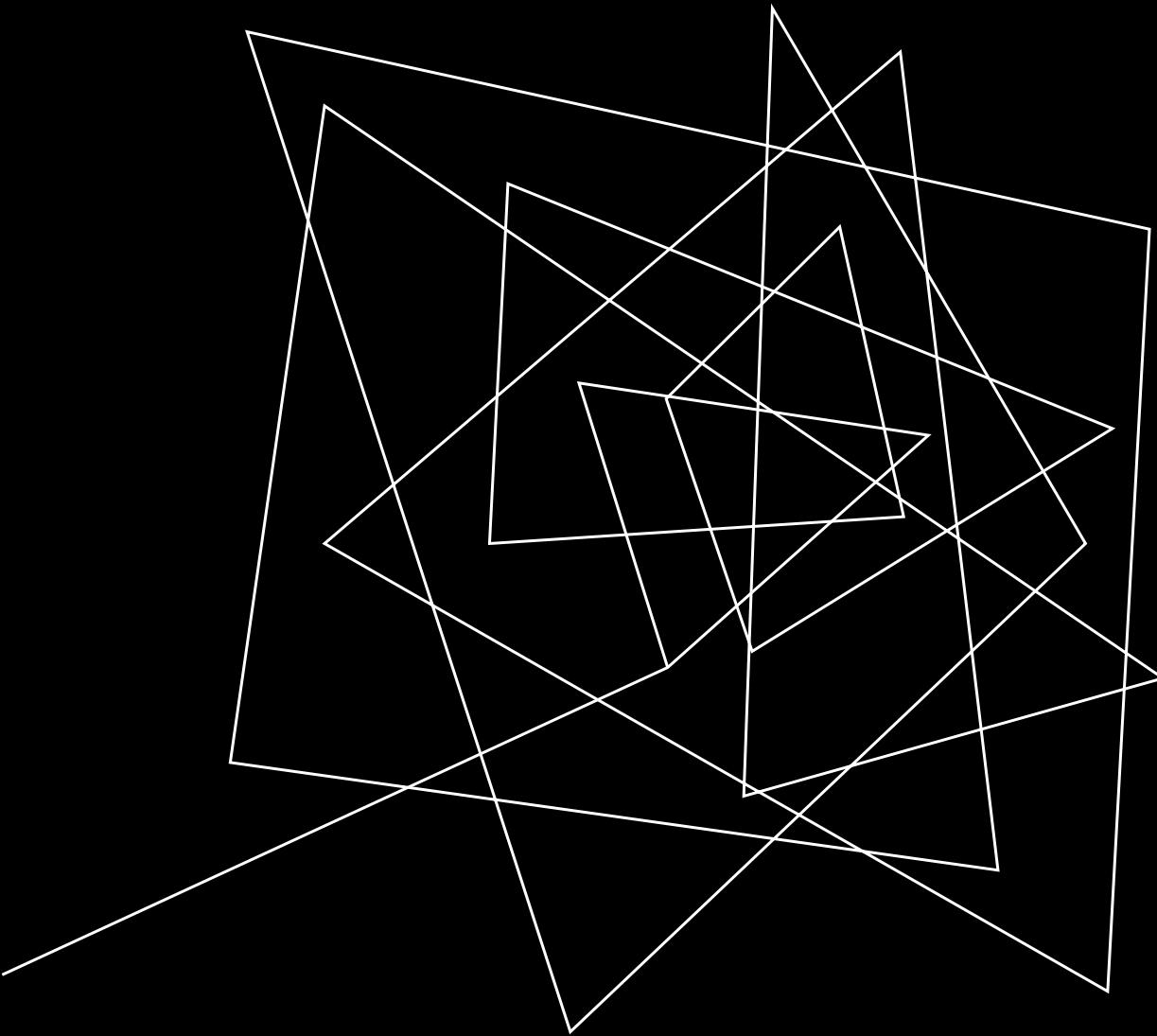
Instances solved



Time limit: two hours
 L^∞ -ellipsoid

Oblivious routing: numerical results





How to learn from
the uncertainty?

Combinatorial bandits

- Combinatorial bandit: solve an optimisation problem **without knowing the costs**
- Many applications:
 - Choose the ads to display:
online knapsack/matching
 - Find your way in a transit network:
online shortest path
 - Schedule work on unreliable machines:
online machine scheduling



Combinatorial bandits

What algorithms can you use?

- Time to make a decision: computational complexity
 - Low: solve one combinatorial problem (possibly polynomial time)
 - Medium: solve one modified combinatorial problem (possibly polynomial time)
 - High: exponential time in all cases
- Number of rounds before learning the real costs (“regret”): sample efficiency
 - Low: asymptotically suboptimal, $\mathcal{O}(\sqrt{T})$ or worse
 - High: $\mathcal{O}(\ln T)$ or asymptotically optimum

Name	Complexity	Sample efficiency
Thompson sampling	Low 😊	Unknown
CUCB	Low 😊	Low 😞
ESCB, OSSB	High 😞	High 😊
AESCB, GLPG	Medium 😐	High 😊

AESCB, high efficiency in polynomial time

- Why are combinatorial bandits hard?

You need to solve a nonlinear optimisation problem in polynomial time:

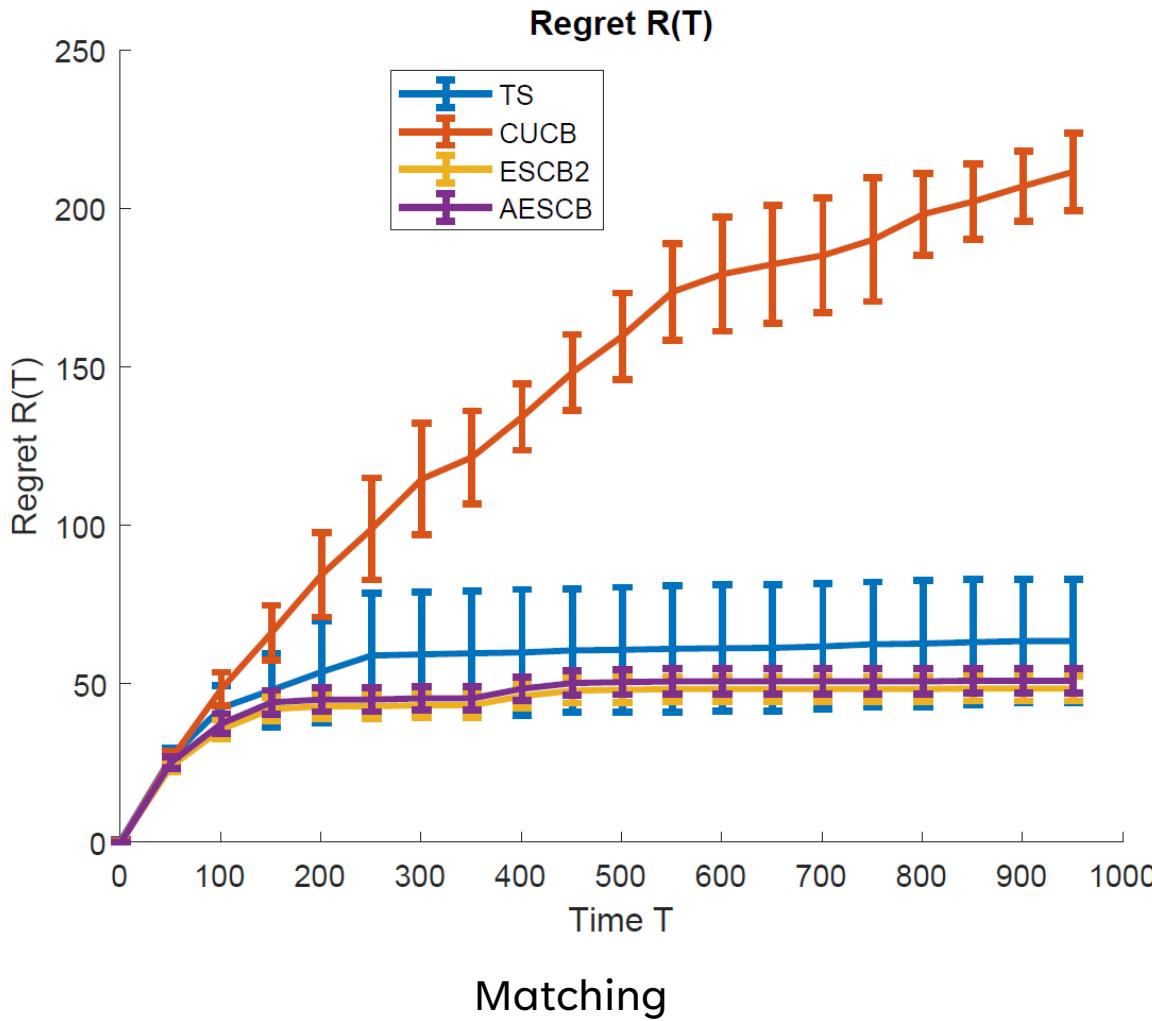
$$\max_{x \in \mathcal{X}} \mu^T x + \sqrt{x^T \sigma^2}$$

- μ : average reward
- σ^2 : inverse of the number of times an arm is played (akin to a variance)
- Major idea: use budgeted combinatorial problems with a threshold t

$$\max_{x \in \mathcal{X}} \mu^T x \text{ such that } \sqrt{x^T \sigma^2} \geq t$$

- How to solve a budgeted combinatorial problem in polynomial time?
 - Integer coefficients (rounding)
 - Dynamic programming
- It works for knapsacks, shortest paths, spanning trees, matchings, etc.

AESCB: high efficiency in polynomial time



AESCB: high efficiency in polynomial time

Combinatorial set	ESCB through CPLEX	AESCB
At most 16 elements among 50	1.24 ± 0.03 s	0.10 ± 0.03 s
Path in a 190-node graph	0.11 ± 0.04 s	0.05 ± 0.00 s
Spanning tree in a 190-node graph	0.20 ± 0.03 s	0.04 ± 0.01 s
Matching in a 25-25-bipartite graph	0.26 ± 0.06 s	0.18 ± 0.01 s

GLPG, optimum efficiency in polynomial time

- Based on a mathematical property of the problem: the *Graves-Lai bound*
 - In the long term, what is the minimum degree of exploration needed to ensure that only the best solutions are played?

$$\begin{array}{ll} \min & \sum_{x \in \mathcal{X}} \alpha_x \Delta_x \\ \text{subject to} & \sum_{i=1}^d \frac{x_i}{\sum_{y \in \mathcal{X}} y_i \alpha_y} \leq \Delta_x^2 \quad \forall x \in \mathcal{X} \\ & \alpha_x \geq 0 \quad \forall x \in \mathcal{X} \end{array}$$

The diagram illustrates the components of the GLPG optimization model:

- Frequency to play x :** α_x (green box)
- Gap of x :** average reward of x — optimum reward (yellow box)
- Total regret:** $\sum_{x \in \mathcal{X}} \alpha_x \Delta_x$ (grey box)
- Distinguish optimum solutions from others:** $\sum_{i=1}^d \frac{x_i}{\sum_{y \in \mathcal{X}} y_i \alpha_y} \leq \Delta_x^2$ (grey box)

GLPG, optimum efficiency in polynomial time

- Intuitive meaning:
 - If you explore less than this: you might think a solution is optimal when it is not
 - If you explore more than this: too much regret for the same level of confidence you have found the optimum solution
- Computational problems:
 - Large number of variables — exponential
 - Large number of constraints (but convex) — exponential
- **GLPG to the rescue!**

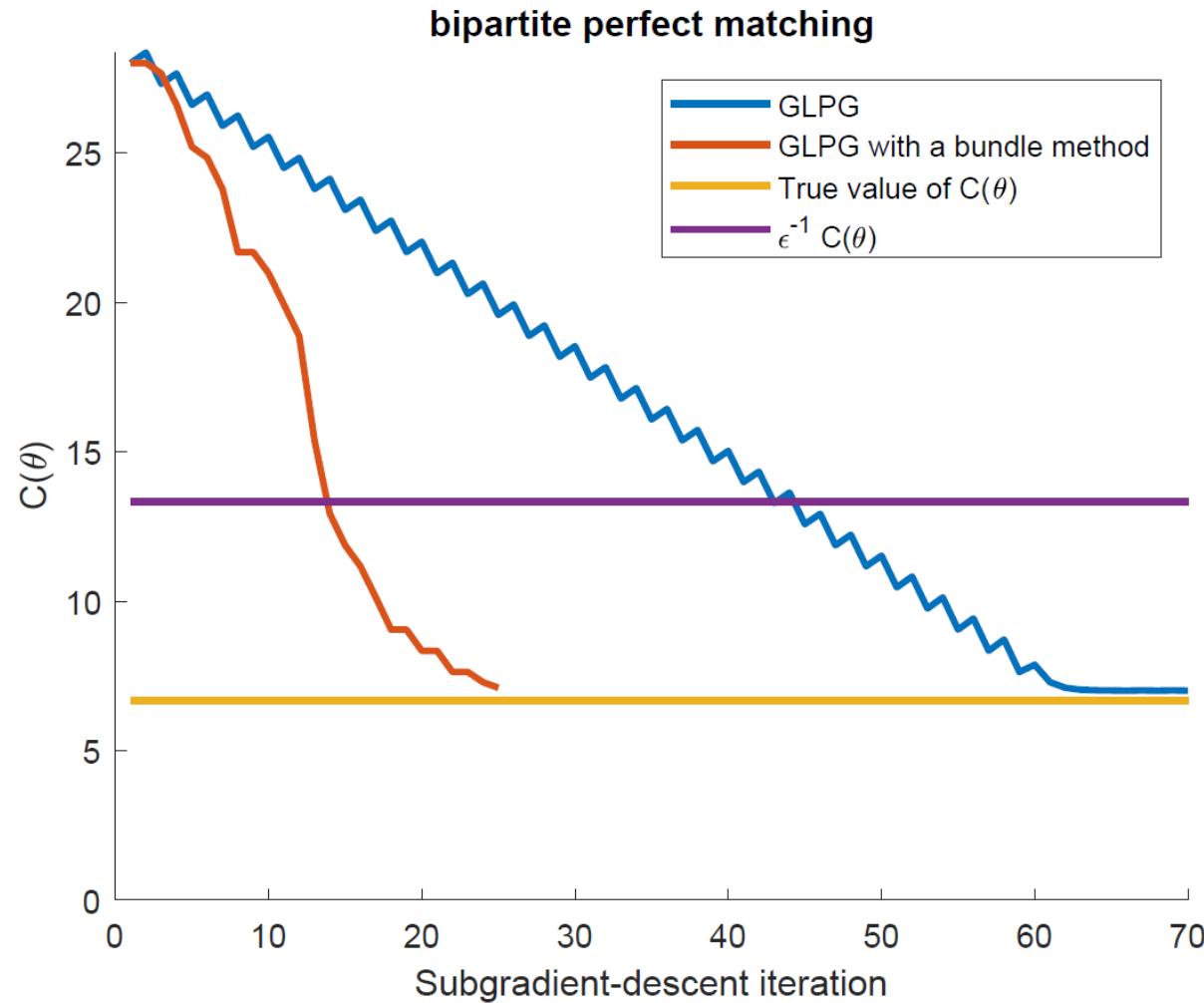


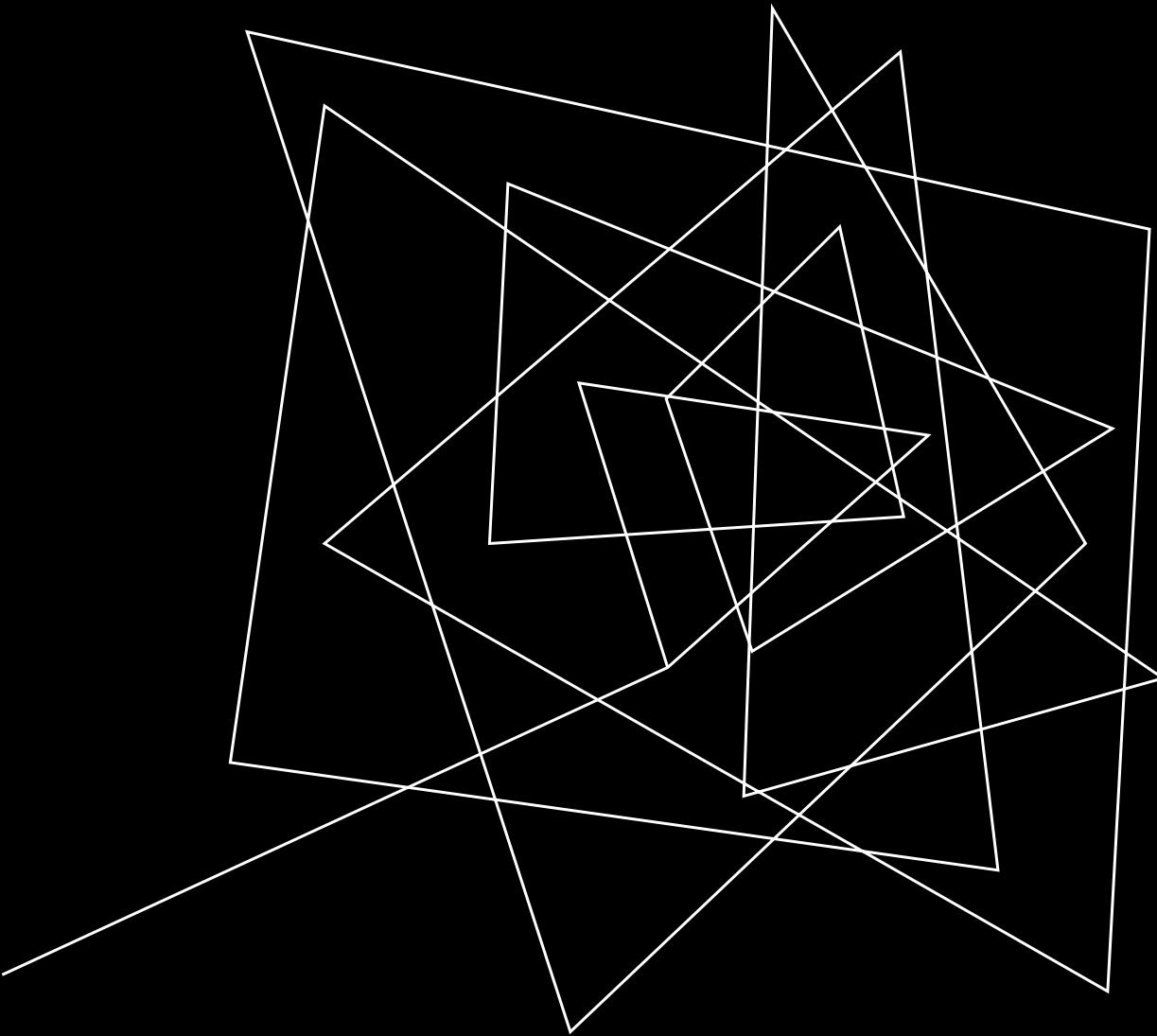
GLPG, optimum efficiency in polynomial time

- The Graves-Lai problem has a lower *intrinsic* dimensionality
 - *Change variables*: use subarm frequency as variables
 - *Use a nonsmooth constraint* instead of many smooth constraints
- Penalise the nonsmooth constraint
 - If the weight λ is large enough, the constraint will be satisfied
- New problem: convex nonsmooth objective, linear constraints

$$\begin{aligned} \min \quad & q^T w + \lambda \left[\max_{x \in \mathcal{X}} \left\{ \sum_{i=1}^d \frac{x_i}{w_i} - \Delta_x^2 \right\} \right]^+ \\ \text{subject to} \quad & M w = 0 \end{aligned}$$

GLPG, optimum efficiency in polynomial time

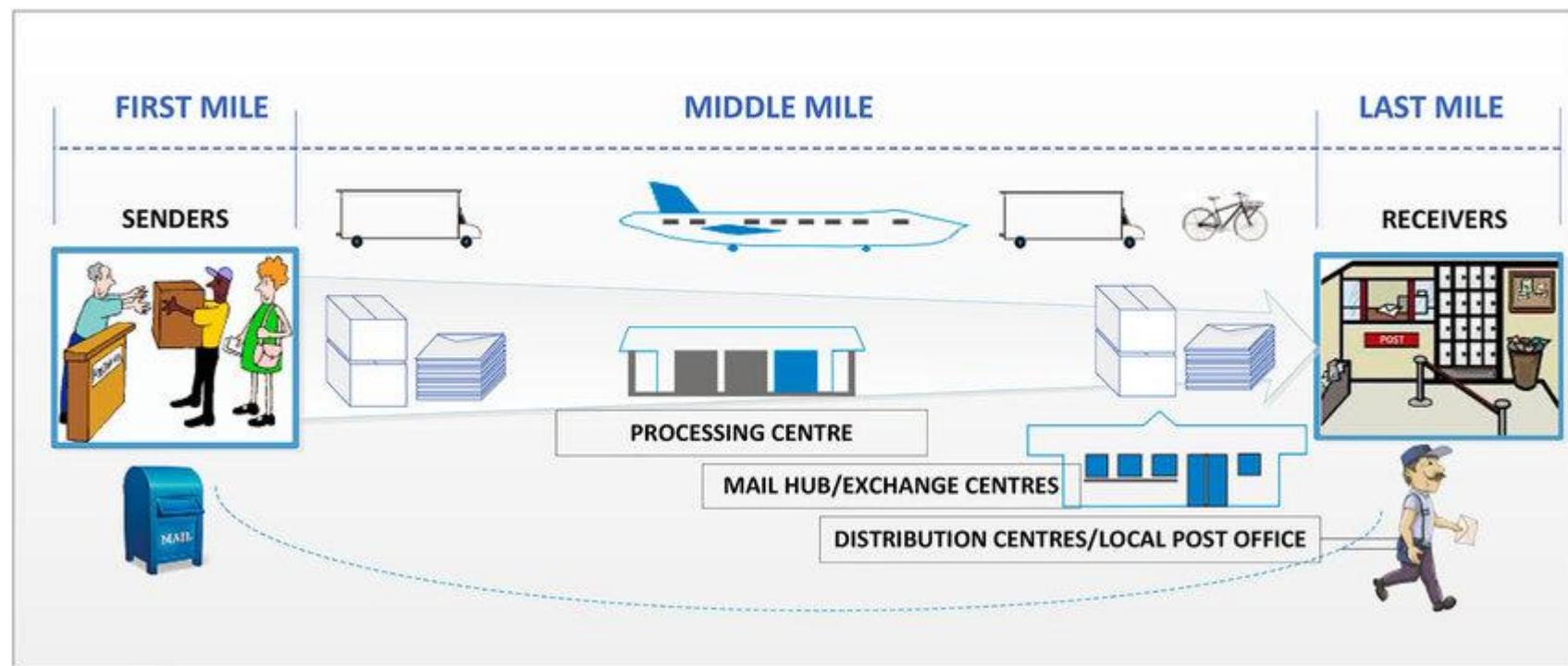




How to use
uncertainty to
solve faster?

Life of a parcel

- A client orders an item from their favourite online store
- The store prepares a parcel with this item
- A delivery person comes and picks the parcel up and drop it at a warehouse: **first mile**
- The parcel is loaded onto a truck, then another, until it reaches its final warehouse: **middle mile**
- A delivery person picks the parcel up at the warehouse and delivers it: **last mile**
- The client receives their parcel (hopefully)



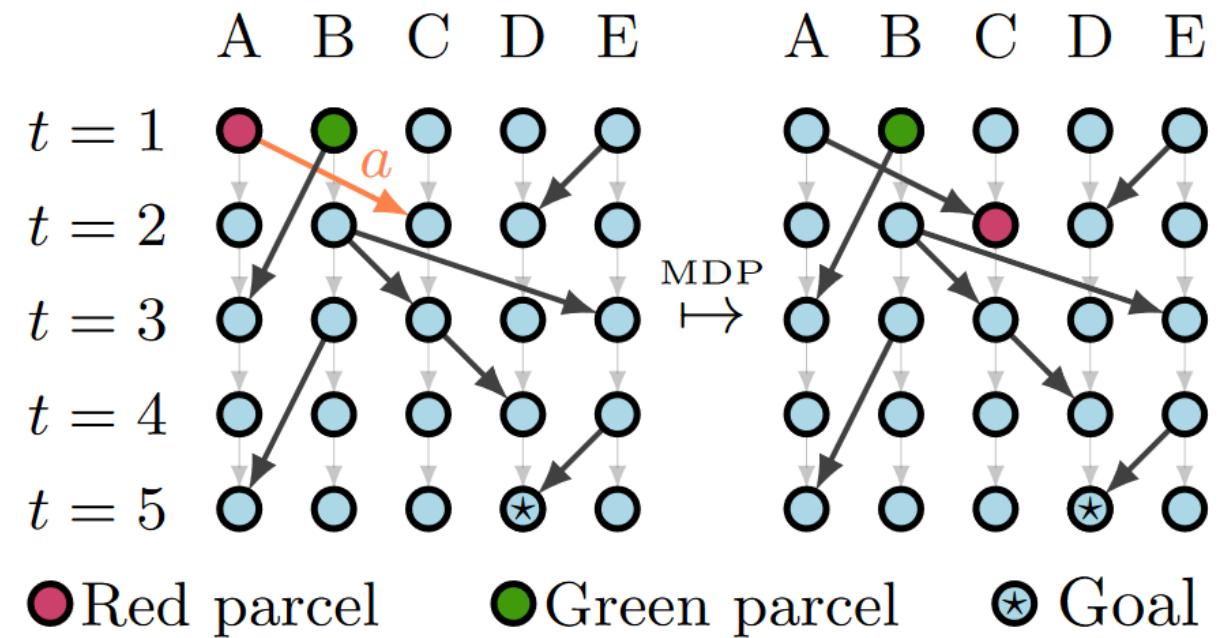
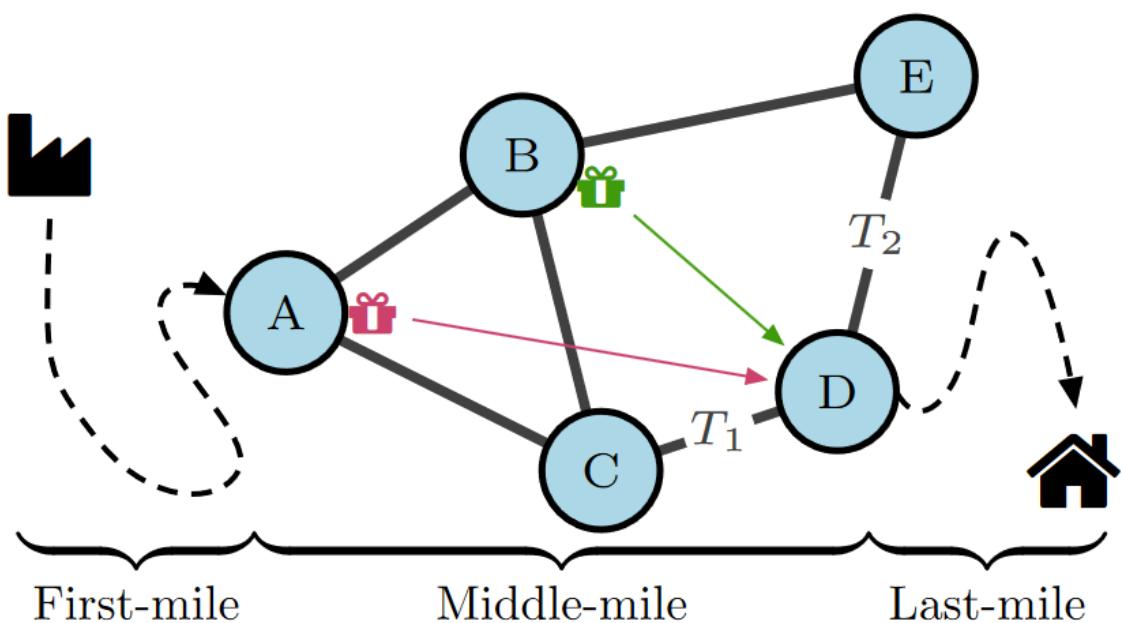
Source: Providing solution to last mile challenges in postal operations. Laseinde, O.T. & Mpofu, Khumbulani. International Journal of Logistics Research and Applications, 2017

Why is middle-mile logistics hard?

- Typical mathematical formulation: multicommodity flow with MIP solvers
 - It does not scale! In one hour: up to 20 warehouses and 400 parcels
 - It does not solve the right problem! Clients would prefer an online system
- How to do better?
 - Combinatorial bandits are not appropriate: presence of a state (trucks are not empty)
 - Full-fledged reinforcement learning (RL): the principles are a good fit!
- How to ensure solution feasibility?
 - Hard to encode constraints in RL
 - Careful modelling!

RL modelling for middle-mile logistics

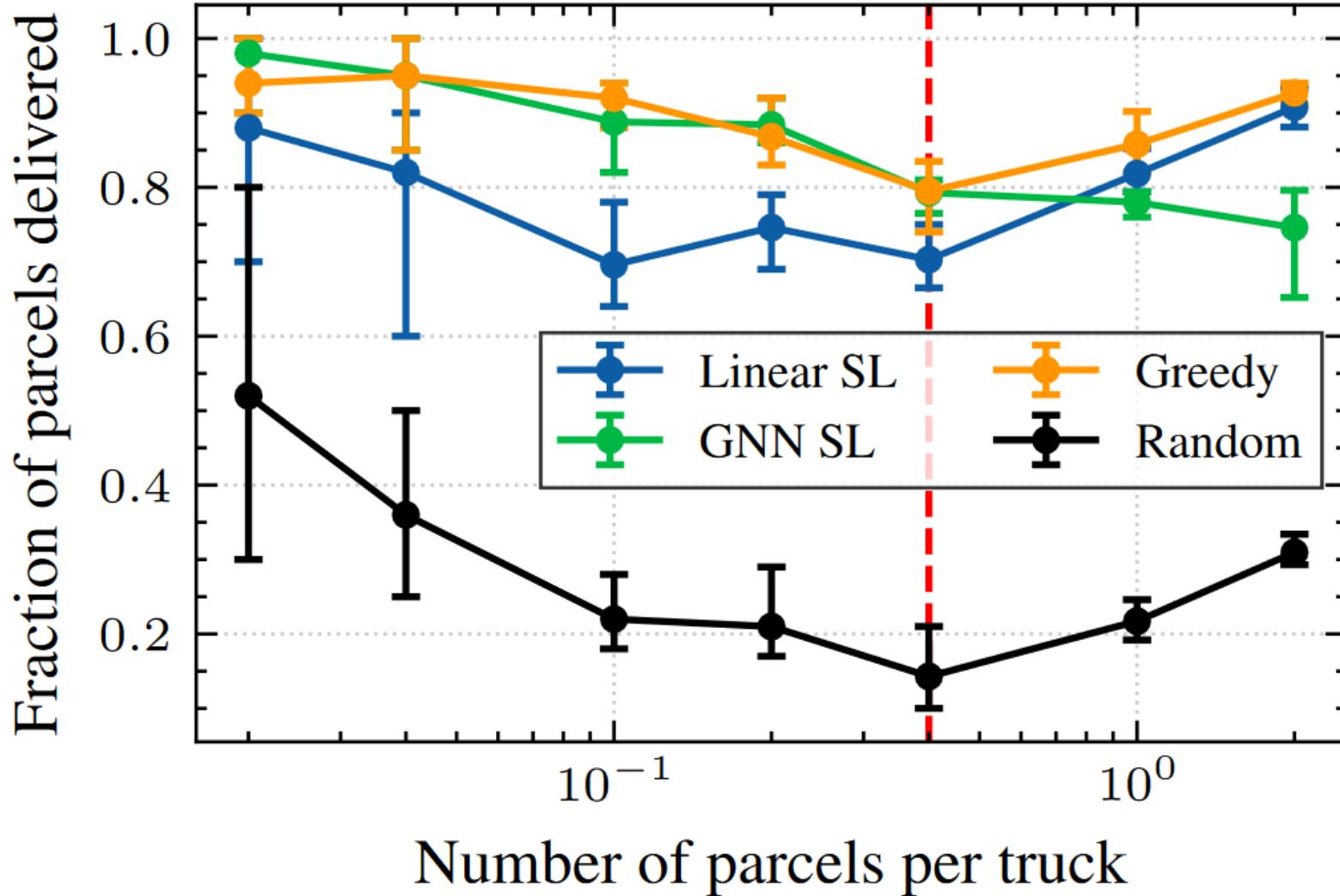
- Consider truck schedules known
- Time-expanded graph: warehouses in one dimension, time in the other
 - Edges: either a parcel stays in the current warehouse or moves with a truck
 - Nodes: warehouse-time combination, parcel



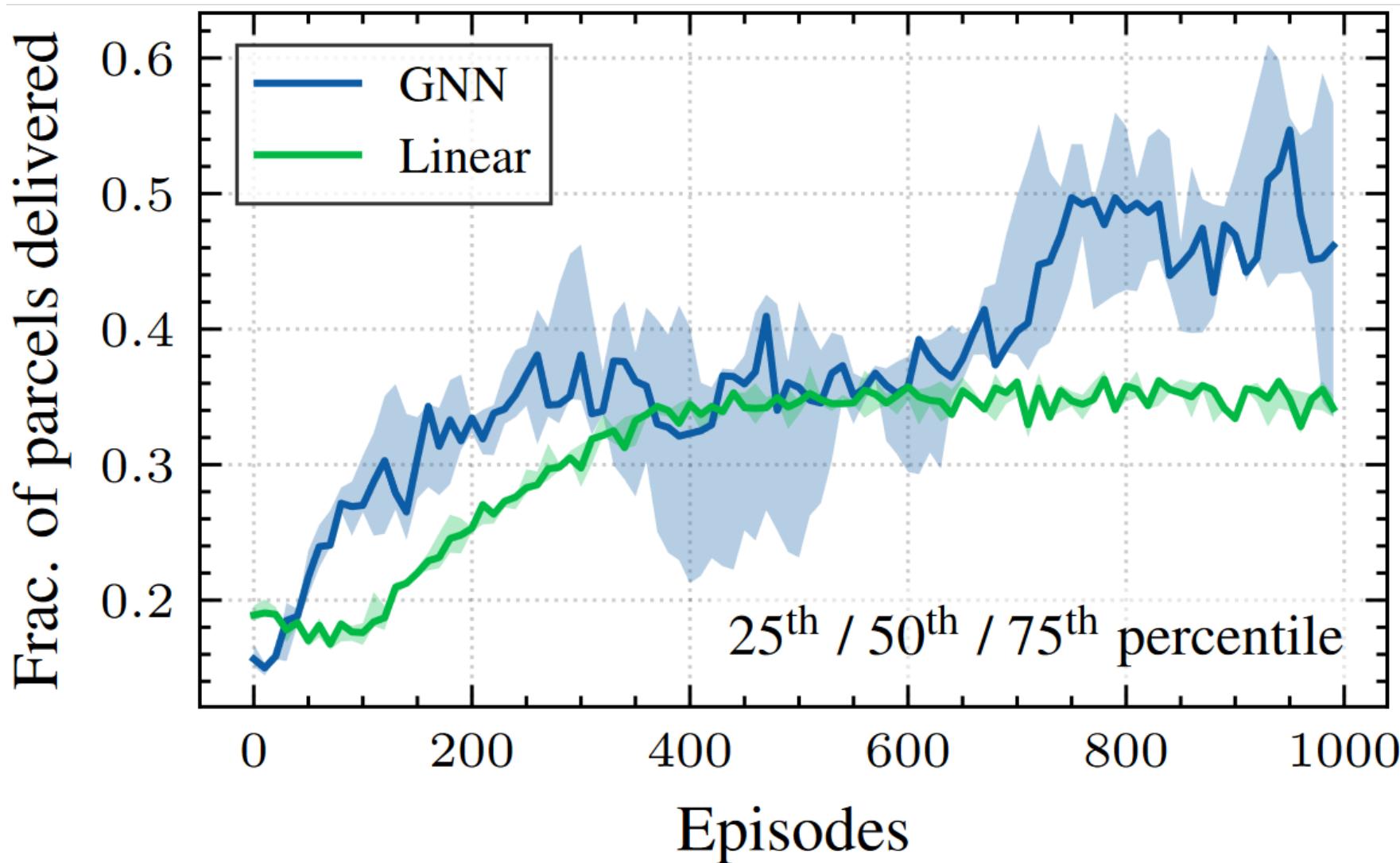
RL modelling for middle-mile logistics

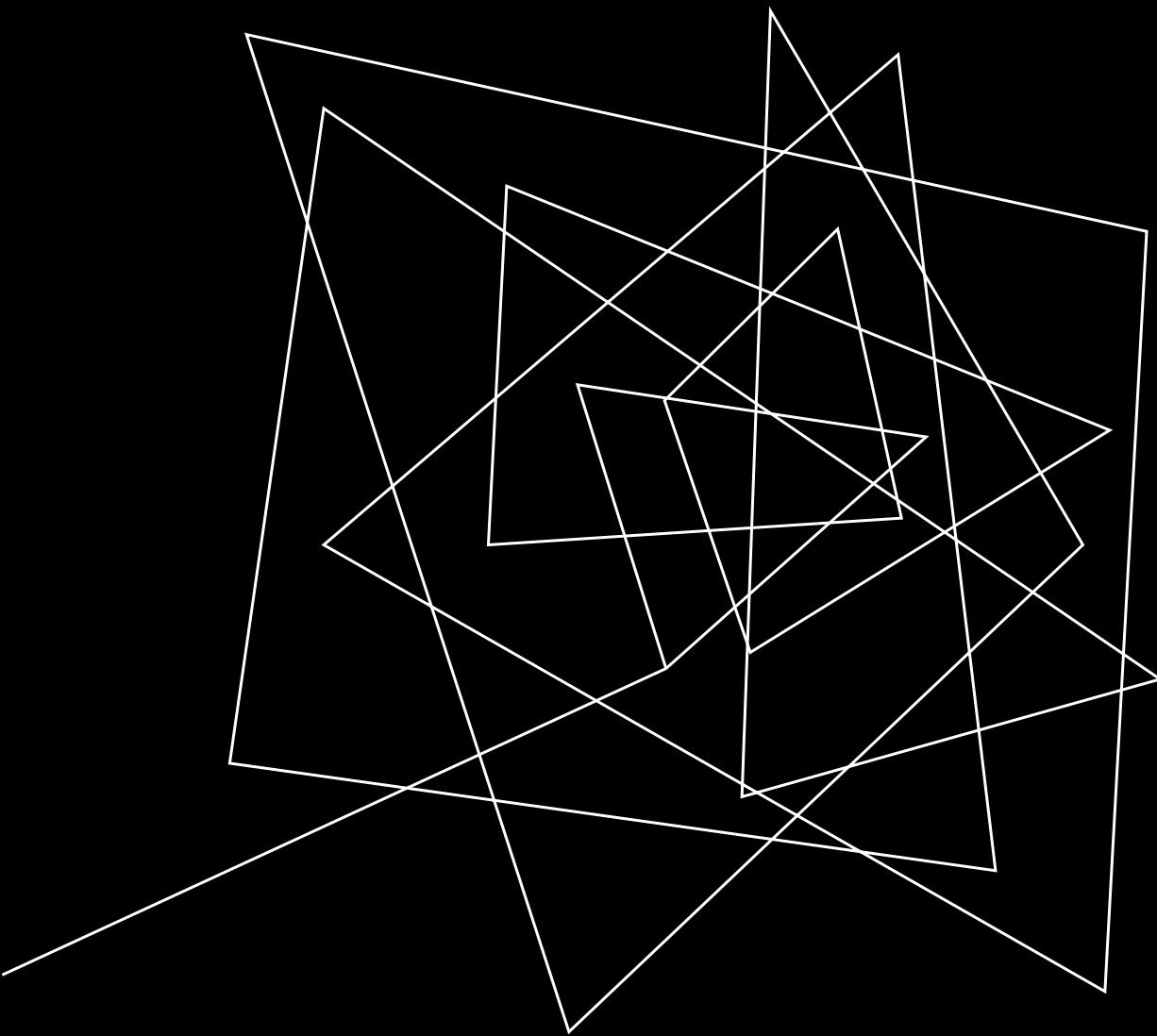
- How to make decisions on top of this graph?
 - At each time step, advance one parcel for one step
 - Choose the best edge according to the learnt value function
- Value function of a state: expected number of delivered parcels
 - Input: state graph (hence the use of graph neural networks)
 - Node and edge features: truck capacity, parcel weight, distance to goal

RL modelling for middle-mile logistics



RL modelling for middle-mile logistics





Time for questions!

I'm not so sure...

Related publications

- Implementing and Comparing Stochastic and Robust Programming. Thibaut Cuvelier. Engineering degree dissertation, university of Liège, June 2015.
<http://hdl.handle.net/2268/197090>
- Comparison Between Robust and Stochastic Optimisation for Long-term Reservoir Operations Under Uncertainty. Thibaut Cuvelier, Pierre Archambeau, Benjamin Dewals, Quentin Louveaux. Water Resources Management, vol. 32, no. 5, pp. 1599–1614, March 2018.
<http://hdl.handle.net/2268/219394>
- Oblivious Routing: Static Routing Prepared Against Network Traffic and Link Failures. Thibaut Cuvelier, Éric Gourdin. Network Traffic Measurement and Analysis (TMA) PhD School 2019, Paris (France).
<https://hal.science/hal-02161708>
- Asymptotically Optimal Strategies For Combinatorial Semi-Bandits in Polynomial Time. Thibaut Cuvelier, Richard Combes, Éric Gourdin. Algorithmic Learning Theory (ALT), Paris (France), March 2021.
<http://proceedings.mlr.press/v132/cuvelier21a.html>
- Statistically Efficient, Polynomial Time Algorithms for Combinatorial Semi Bandits. Thibaut Cuvelier, Richard Combes, Éric Gourdin. ACM SIGMETRICS 2021, Beijing (China), June 2021.
<https://hal.science/hal-03201526>
- Middle-Mile Logistics Through the Lens of Goal-Conditioned Reinforcement Learning. Onno Eberhard, Thibaut Cuvelier, Michal Valko, Bruno de Backer. Conference on Neural Information Processing Systems (NeurIPS) Goal-Conditioned Reinforcement Learning Workshop, New Orleans (USA), December 2023.