KONINKLIJK INSTITUUT VAN INGENIEURS

Engineer your career • Improve our society

Smart Algorithms for Smart Grids

dr. Mathijs de Weerdt & dr. Matthijs Spaan

Main message

The energy system is changing.

The future energy system is a rich environment with many challenges for computer scientists.

Overview

1. First example: EV charging
2. Power System Essentials for non-Electrical Engineers
3. CS Challenges in the Future Power Grid
4. *(10 minute break)*
5. Techniques for Decision Making
6. Concrete Examples of Smart Algorithms for Smart Grids
    1. Decision making with wind scenarios
    (Walraven & Spaan, 2015)
    2. Planning and coordination of thermostatic loads
    (de Nijs et al., 2015, 2017)
7. Discussion

Power system essentials I

- Most significant changes are in the electricity systems.
- Electricity systems support the generation, transport and use of electrical energy.
- They are large and *complex*.

Energy generated = *energy consumed* at all times

How it used to be...

- demand is predictable (at an aggregate level, day ahead)
- which *generators* are used is decided one day in advance (*unit commitment*), taking into account transmission constraints
- a market with 2–10 actors (energy retailers)
- minor corrections are made, based on frequency (primary control, secondary control, etc.)

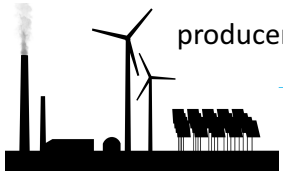
Power system essentials II

Electrical grid design

Specifically for this mode of operation (50–100 years ago):

- *transmission* at high voltage to transport large amounts over long distances (thick cables, little losses, redundancy, few nodes), actively measured and controlled,
- *distribution* to bring energy to the *end-users* (thinner cables, lower voltages, safer, cheaper, but higher losses, star topology, many nodes), passively operated,
- **over-dimensioned**: designed for peak demand (Christmas day)

Power system essentials III



producers

market operator

futures

day-ahead

adjustment

balancing

system operator

1 day

1 hour

retailers

consumers

clients

Power system essentials IV

However

- controllable carbon-based generators are being replaced by renewable energy from sun and wind, which is
  - intermittent
  - uncertain
  - uncontrollable
  - sometimes located in the distribution grid, and
  - has virtually no marginal costs
- numbers of non-conventional loads such as heat pumps, airconditioning, and electric vehicles are increasing, and these loads are
  - significantly larger than other household demand, and
  - more flexible (and therefore also less predictable)

CS Challenges I

This master class focuses on computational challenges regarding

1. Aggregators
2. (Wholesale) market operators
3. Distribution network operators

Other interesting related CS / IT issues...

- optimizing flexible energy use (multiple energy carriers?) for *single users* (factories, large data centers, cold warehouses)
- the old carbon-based suppliers (still needed, but much less)
- predicting generation (weather) and prices
- carbon emission markets
- security
- privacy

Challenges for Aggregators

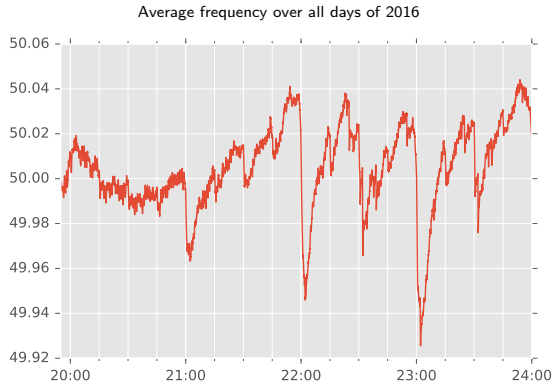
- Consumers do not want to interact with the market.
- Markets do not want every consumer to interact.
- But there is value in flexible demand.

Aggregator of flexible demand

e.g. charging electric vehicles, heat pumps, air-conditioning

1. design mechanism to interact with consumers with flexible demand
2. interact with both wholesale markets and distribution service operator
3. optimize use of (heterogeneous) flexible demand under uncertain prices and uncertain consumer behavior

Design errors in existing markets



- significant frequency deviations every (half) hour when generators shut down
- average imbalances of about 2 GW (one power plant)
- expensive reserves are being used to repair market design error

Challenges in Wholesale Market Design

Market Operators/Regulators and ISO/TSO

1. more accurate models for bidding and market clearing
   - use finer granularity, power-based instead of energy-based
   - include new flexibility constraints
   - model stochastic information explicitly

   but reasonable models are non-linear: interesting optimization problem
2. deal with intertemporal dependencies caused by shiftable loads (re-think combined day-ahead, adjustment, and balancing)
3. allow smaller, local producers and flexible loads (scalability)
4. interaction with congestion and voltage quality management in distribution network

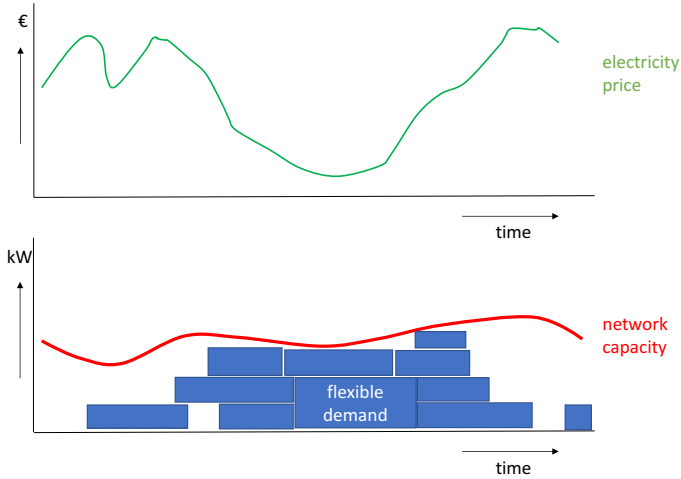
Challenges for DSOs

Distribution ~~network~~ system operators

Aim to avoid unnecessary network reinforcement by demand side management to resolve congestion and voltage quality issues

1. coordinate generation, storage and flexible loads of self-interested agents
2. complex power flow computations (losses and limitations more relevant in distribution)
3. stochastic information regarding other loads, local generation
4. communication may not be always reliable
5. there are many more agents than in traditional energy market
6. interaction with wholesale markets

Challenges for DSOs: Schedule flexible loads within network capacity



Computational Limitations: Complexity Theory

Sometimes problems are easy.

Example problem: Scheduling to minimize the maximum lateness

- Given a set of n jobs with for each job j length p_j and deadline d_j
- Schedule all of them such that the maximum lateness over jobs is minimized.
- The lateness of a job is $\max\{0, d_j - f_j\}$, where f_j is the finish time of a job in the schedule.

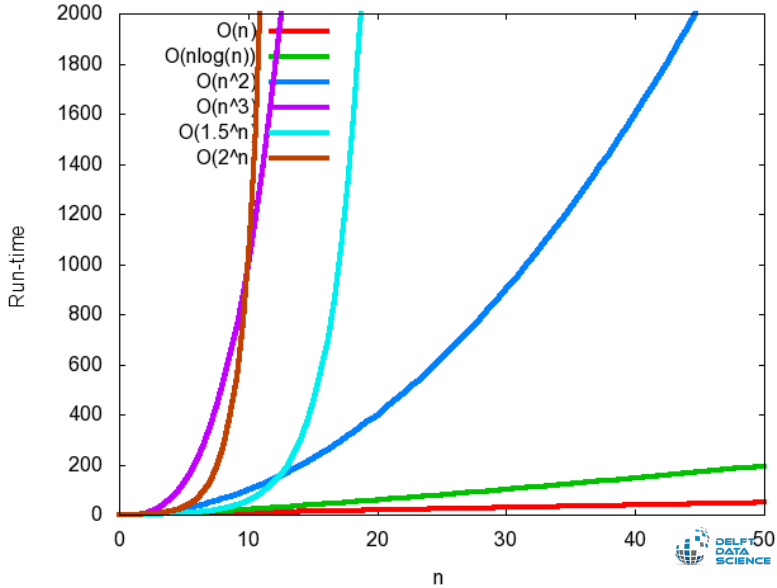
Scheduling to Minimize Maximum Lateness

Algorithm pseudocode

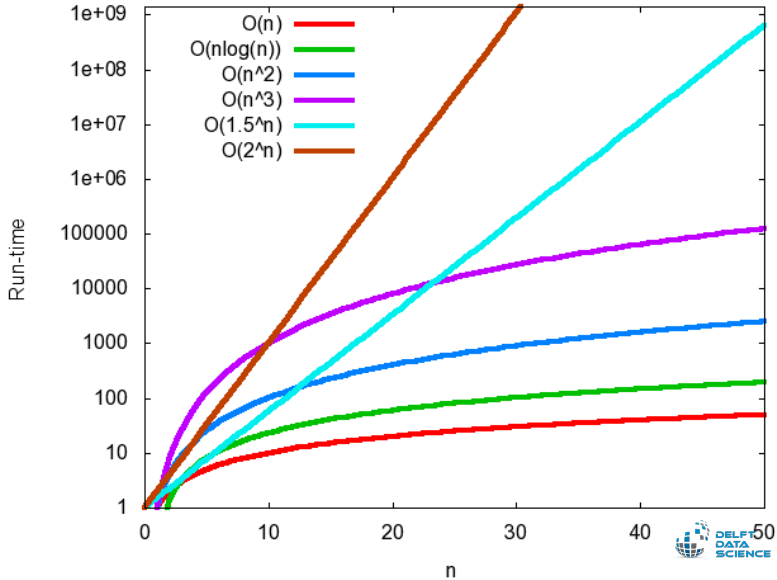
Sort jobs by deadline so that $d_1 \leq d_2 \leq \ldots \leq d_n$
$t \leftarrow 0$
for $j \leftarrow 1, 2, \ldots, n$ **do**
    $s_j \leftarrow t$ `// Assign job` j `to interval` $[t, t + p_j]$
    $f_j \leftarrow t + p_j$
    $t \leftarrow f_j$
Output intervals $[s_j, f_j]$

The runtime consists of sorting: $O(n \log n)$ and then n steps in the for-loop.

Runtime complexity



Runtime complexity (log scale)



P and NP-hard

If an algorithm is known with runtime of $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^3)$, ..., we say:

- the problem can be solved *efficiently* or is *tractable*
- the problem is in the class P

If no efficient algorithm is known, we say:

- the problem is intractable
- the problem is NP-hard

(This is usually proven by a reduction from a known NP-hard problem.)

Example of a known NP-hard problem

Traveling salesman problem

- What is the most efficient route to arrange a school bus route to pick-up children?
- What is the most efficient order for a machine to drill holes holes in a circuit board?

Given

- n cities with distances $d(i, j)$
- Find the shortest path from city 1 through all cities back to 1

Consequence of NP-hardness

We must sacrifice one of three desired features.

1. We cannot solve problem in polynomial time.
2. We cannot solve problem to optimality.
3. We cannot solve arbitrary instances of the problem.

Example: Complexity of Charging EVs in Constrained Smart Grid

Mathijs de Weerdt, Michael Alberts, and Vincent Conitzer (Duke University)

Results:

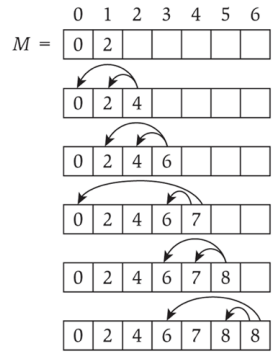
- equivalence to single-machine scheduling variants if charging speeds are identical
- hardness results if vehicles not always available ("gaps") or with complex demand/charging speeds
- *dynamic programs* for constant horizon problems

	gaps			*constant charging speed*			*unbounded charging speed*		
$\|T\|$	demand constant	demand polynomial	demand unbounded	demand constant	demand polynomial	demand unbounded	demand constant	demand polynomial	demand unbounded
$O(1)$	P^{T1}	P^{T1}	weak NP-c^{T1}	P^{T1}	P^{T1}	weak NP-c^{T1}	P^{T1}	P^{T1}	weak NP-c^{T1}
$O(n^c)$	strong NP-c^{T3}	strong NP-c^{T3}	strong NP-c^{T3}	?	?	? NP-c^{P2}	P^{T2}	P^{T2}	weak NP-c^{T2}

Dynamic Programming

Main idea: divide into subproblems, reuse solutions from subproblems

- Characterize structure of problem
- Recursively define value of optimal solution: $OPT(i) = \ldots$
- Compute value of optimal solution iteratively starting from smallest
- Construct optimal solution from computed information



Dynamic Programming for Charging EVs

- $|T|$ periods, n agents
- supply per period $t \in T$ of $m_t \leq M$
- demand per agent at most L encoded in constraints on possible allocation a
- $v_i(a)$ denotes value for agent i for allocation a

Optimal solution $OPT(m_1, m_2, \ldots, m_{|T|}, n)$

computed using:

$$
\begin{aligned}
OPT(m_1, m_2, \ldots, m_{|T|}, i) &= \begin{cases} 0 & \text{if } i = 0 \\ \max\left\{OPT(m_1, m_2, \ldots, m_{|T|}, i-1), o\right\} & \text{otherwise} \end{cases} \\
\text{where } o &= \max_{a_1,..,a_{|T|}} \left\{OPT(m_1 - a_1, \ldots, m_{|T|} - a_{|T|}, i-1) + v_i(a)\right\}.
\end{aligned}
$$

Runtime of dynamic programming implemenation: $O\left(n \cdot M^{|T|} \cdot L^{|T|}\right)$

Example: Complexity of Charging EVs in Constrained Smart Grid

Typical scenario: constraint in substation for 20–500, prices that differ per 15 minute
Conclusions:

- Scheduling of about 20 electric vehicles in a rolling horizon setting of 1.5 hours doable, but
- more than 10% EVs or look-ahead more than 1.5 hours, we need to sacrifice optimality.
- Without bounds on infrastructure or on charging speed, realistically-sized problems can be solved quickly.

Techniques for Decision Making

- Efficient algorithms: greedy, dynamic programming
- Approaches to NP-hard problems: integer programming, planning and multi-agent planning
- Concrete examples

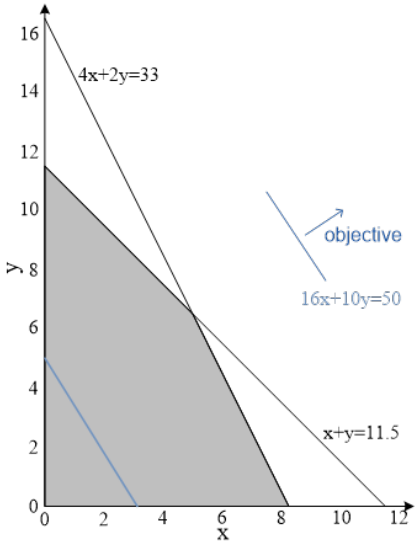
Integer Programming

$$z = \max_{x,y} 16x + 10y$$
$$\text{s.t. } x + y \leq 11.5$$
$$4x + 2y \leq 33$$
$$x, y \geq 0$$
$$x, y \in \mathbb{R}^+$$

Simplex Method (1947):

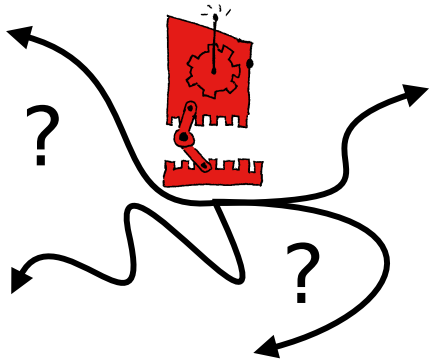
```
Iteration           Objective                In Variable
     1            132.000000                         x
     2            145.000000                         y
LP status(1): optimal
Cplex Time: 0.00sec (det. 0.01 ticks)

Optimal solution found.
Objective :          145.000000
```



Introduction to planning in Artificial Intelligence

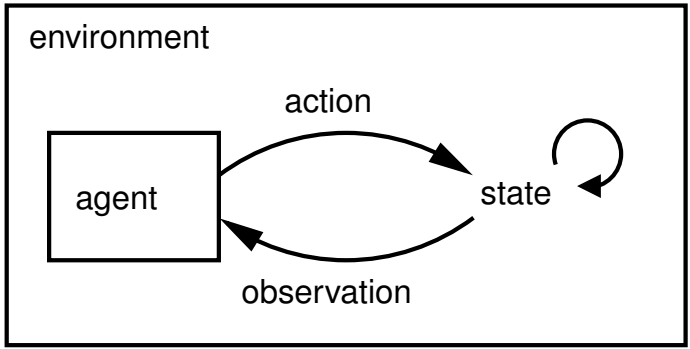
- Goal in Artificial Intelligence: to build intelligent agents.
- Our definition of "intelligent": perform an assigned task as well as possible.
- Problem: how to act?
- We will explicitly model uncertainty.



Agents

- An agent is a (rational) decision maker who is able to perceive its external (physical) environment and act autonomously upon it.
- Rationality means reaching the optimum of a performance measure.
- Examples: humans, robots, some software programs.

Agents



- It is useful to think of agents as being involved in a perception-action loop with their environment.
- But how do we make the right decisions?

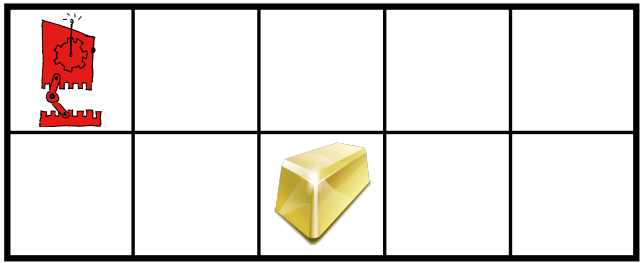
Planning

Planning:

- A plan tells an agent how to act.
- For instance
  - A sequence of actions to reach a goal.
  - What to do in a particular situation.
- We need to model:
  - the agent's actions
  - its environment
  - its task

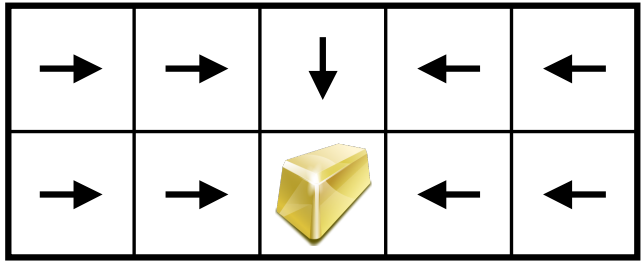
We will model planning as a sequence of decisions.

Classic planning



- Classic planning: sequence of actions from start to goal.
- Task: robot should get to gold as quickly as possible.
- Actions: $\rightarrow \downarrow \leftarrow \uparrow$
- Limitations:
  - New plan for each start state.
  - Environment is deterministic.
- Three optimal plans: $\rightarrow \rightarrow \downarrow, \rightarrow \downarrow \rightarrow, \downarrow \rightarrow \rightarrow$.

Conditional planning



- Assume our robot has noisy actions (wheel slip, overshoot).
- We need conditional plans.
- Map situations to actions.

Decision-theoretic planning

−0.1	−0.1	−0.1	−0.1	−0.1
−0.1	−0.1	10	−0.1	−0.1

- Positive reward when reaching goal, small penalty for all other actions.
- Agent's plan maximizes **value**: the sum of future rewards.
- Decision-theoretic planning successfully handles noise in acting and sensing.

Decision-theoretic planning

Optimal values (encode optimal plan):

9.7	9.8	9.9	9.8	9.7
9.8	9.9	10	9.9	9.8

Reward:

−0.1	−0.1	−0.1	−0.1	−0.1
−0.1	−0.1	10	−0.1	−0.1

Sequential decision making under uncertainty

- Uncertainty is abundant in **real-world planning** domains.
- **Bayesian** approach $\Rightarrow$ probabilistic models.

Main assumptions:

Sequential decisions: problems are formulated as a sequence of "independent" decisions;

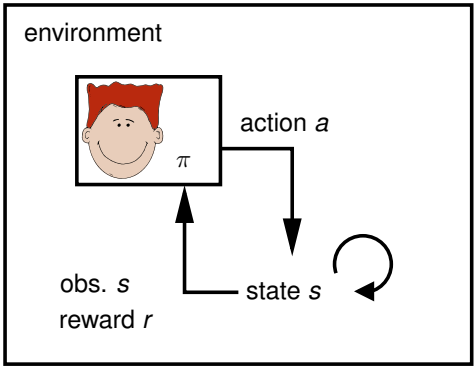
Markovian environment: the state at time t depends only on the events at time $t-1$;

Evaluative feedback: use of a reinforcement signal as performance measure (reinforcement learning);

Transition model

- For instance, robot motion is inaccurate.
- Transitions between states are **stochastic**.
- $p(s'|s, a)$ is the probability to jump from state s to state s' after taking action a.

MDP Agent



Problems and solutions at the distribution level

- Power generation of renewables is uncertain.
- Congestion in the network.

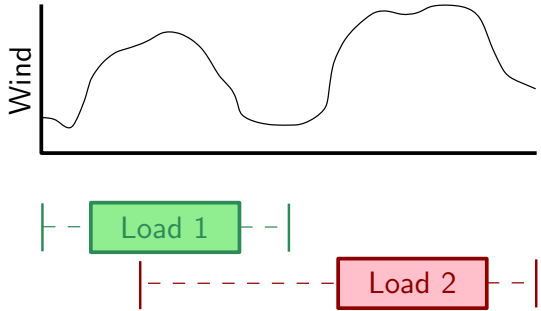
Solutions

- Grid reinforcements.
- Buffers and storage devices.
- Actively matching generation and consumption of local consumers in a smart grid $\rightarrow$ SDM research.

Why coordination and control is hard

- Thousands of loads and generators connected to the grid.
- Communication infrastructure might fail.
- Heterogeneous characteristics and objectives.
- Capacity constraints of the grid.

Scheduling of deferrable loads using MDPs

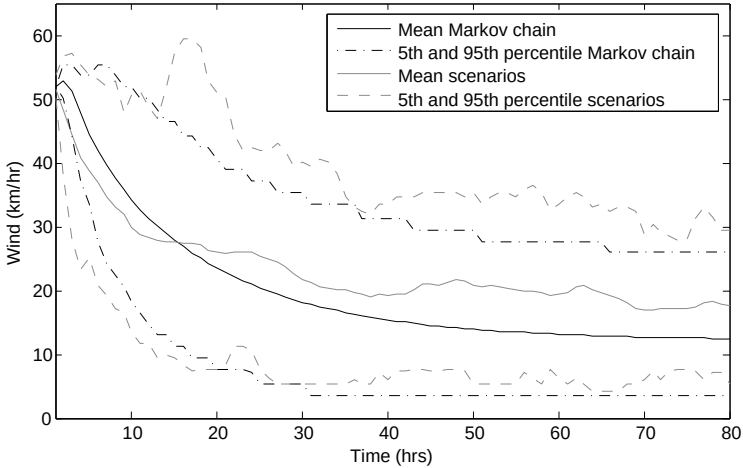


Modeling challenges

- Renewable wind supply generated in the future is uncertain.
- Supply is hard to model using a compact Markovian state.

Markov models for wind

Second-order Markov chains do not accurately model wind.



Modeling external factors in MDPs

In many planning domains it can be hard to model external factors, and therefore it can be difficult to predict future states.

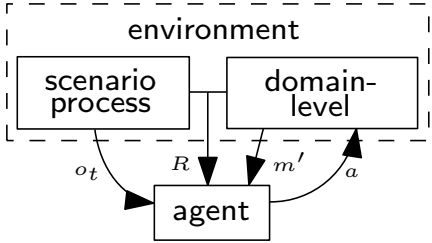
Modeling challenges

- Selecting the right state features.
- Obtaining an appropriate level of detail.
- Estimating transition probabilities.

Overview of our approach (Walraven & Spaan, UAI 2015)

- Scenario representation, including weights.
- POMDP model to reason about scenarios.

Agent-environment interaction



- Scenario process represents hard-to-model external factor.
- State of the scenario process is always represented by a numerical value (e.g., wind speed), and is observable.

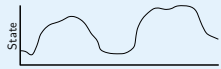
- Other process models domain-level state of the environment.
- Actions only affect domain-level state of the environment.

Scenario representation

We use sequences of states to predict future states of the uncertain wind process.

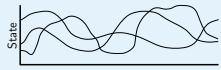
Scenario

Scenario $x = (x_1, x_2, \ldots, x_T)$ defines the states for time $1, 2, \ldots, T$.



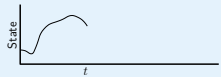
Scenario set

A scenario set X is an unordered set, where each $x \in X$ is a scenario.

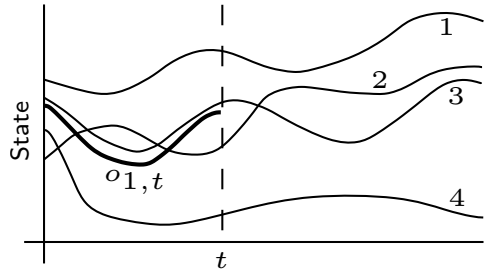


State observations

The sequence $o_{1,t} = (o_1, o_2, \ldots, o_t)$ defines the observed states for time $1, 2, \ldots, t$, where o_i is revealed at time i.



Assigning weights to scenarios



Algorithm: assigning weights

Input: scenario set X, state observations $o_{1,t}$
Output: weight vector w

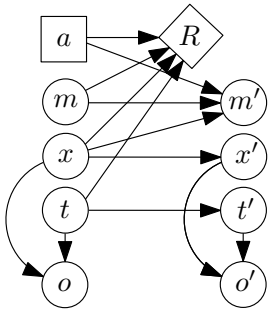
1. For each $x \in X$: compute distance between $o_{1,t}$ and x.
2. For each $x \in X$: assign weight w_x inversely proportional to distance.
3. Normalize w, such that the sum of weights equals 1.

Scenario-POMDP

Scenario-POMDP

A Scenario-POMDP is a POMDP in which each state s can be factored into a tuple

- m: observable domain-level state of the environment
- x: scenario of the scenario process, which is partially observable
- t: time index



In state $s = (m, x, t)$, scenario process state x_t is observed with probability 1, which is the state at time t in scenario x.

Planning with scenarios

- POMDP model which incorporates scenarios.
- We use the POMCP algorithm for planning.
- The algorithm samples scenarios from X based on weights, rather than sampling states from a belief state.

Scenario-POMCP

Given domain-level state m and $o_{1,t}$, the POMCP algorithm can be applied (almost) directly to select the next action.

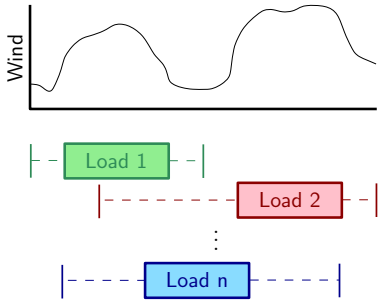
Algorithm: selecting an action at time t

1. Observe state o_t of the scenario process.
2. Given $o_{1,t}$ and scenario set X, compute weight vector w.
3. Run POMCP from 'belief' state (m, w, t) to select action a.
4. Execute action a in domain-level state m.

POMCP samples scenarios from X based on weights, rather than sampling states from a belief state.

Scheduling deferrable loads: problem formulation

- Domain-level state represents the state of the flexible loads.
- Actions correspond to starting or deferring loads.
- Scenario $x = (x_1, x_2, \ldots, x_T)$ encodes wind speed for T consecutive timesteps.

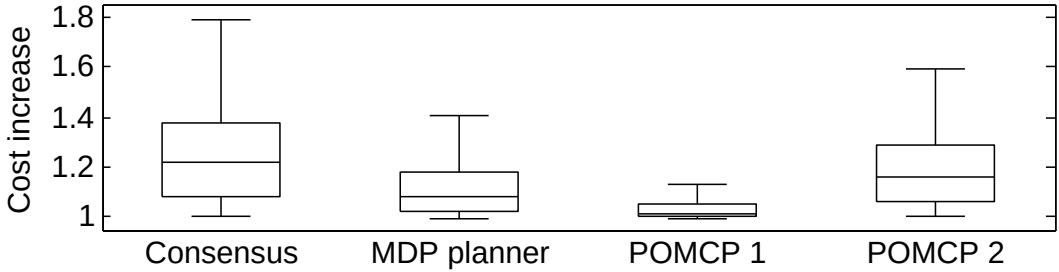


Objective: minimize grid power consumption by scheduling loads in such a way that wind power is used as much as possible.

Experiment

We obtained historical wind data from the Sotavento wind farm in Galicia, Spain.

Performance comparison with Markov chain, consensus task scheduling, omniscient schedules.



Summary

Scenarios can be used to model external factors that are typically hard to model using a Markovian state.

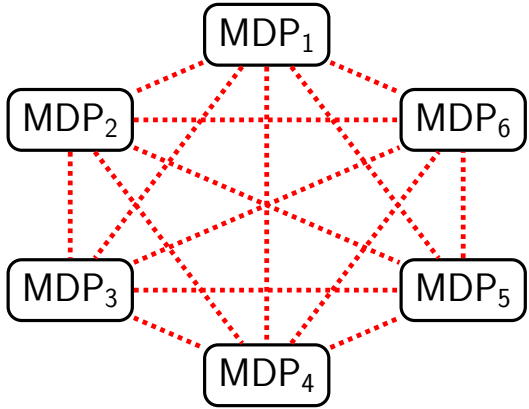
Benefits of scenarios

- Only requires a historical dataset of the external factor involved.
- Does not require estimates of transition probabilities.
- Can be easily combined with problems modeled as an MDP.
- May provide better long-term predictions than a Markov chain.

Disadvantages of scenarios

- We did not implement a Bayesian belief update yet.
- Scalability may become an issue in smart electricity grids.

Multiagent planning



Thermostats as Energy Storage (AAAI '15 and '17)

with Frits de Nijs, Erwin Walraven, and Matthijs Spaan (with Alliander)

De Teuge

- pilot sustainable district
- heatpumps for heating

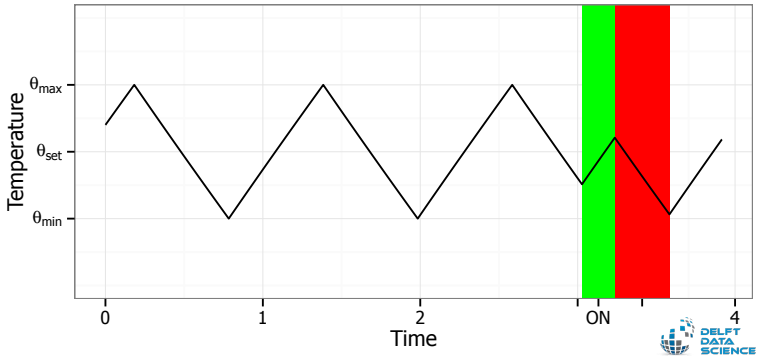
But: at peak (cold) times, overload of electricity infrastructure

Thermostats as Energy Storage

Stay within capacity of infrastructure by flexibility of demand.

Thermostatically controlling loads (TCLs) exhibit inertia:

- Energy inserted decays over time
- Behaves as one-way battery



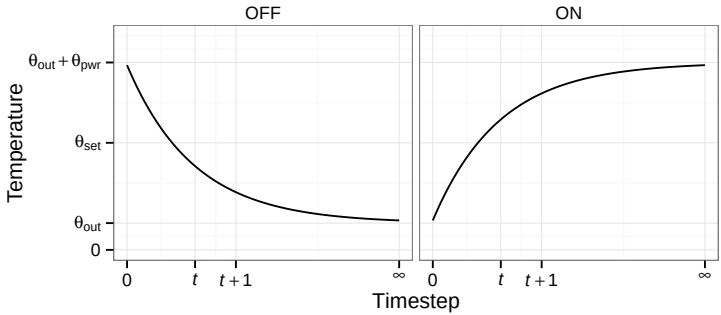
Trade-off: Comfort v.s. Capacity

Comfort (reward):

$$\theta_{i,t} \approx \theta_{i,t}^{\text{set}} \quad \forall i, t$$

Capacity (constraint):

$$\sum_i^n \text{action}_{i,t} = \text{ON} \leq \text{capacity}_t \quad \forall t$$



Optimisation Problem

Formulate as a mixed integer problem (MIP)

- decide when to turn on or off heat pump
- minimise discomfort (distance to temperature set point)
- subject to physical characteristics and capacity constraint

MIP formulation

$$\underset{[\,\mathbf{act}_0\ \mathbf{act}_1\ \cdots\ \mathbf{act}_h\,]}{\text{minimize}} \quad \sum_{t=1}^{h} \text{cost}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t^{\text{set}}) \qquad \text{(discomfort)}$$

$$\text{subject to} \quad \boldsymbol{\theta}_{t+1} = \text{temperature}(\boldsymbol{\theta}_t, \mathbf{act}_t, \theta_t^{\text{out}})$$

$$\sum_{i=1}^{n} \text{act}_{i,t} \leq \text{capacity}_t$$

$$\text{act}_{i,t} \in [\text{OFF}, \text{ON}] \qquad \forall i, t$$

This scales poorly (binary decision variables: houses $\times$ time slots).

But that is not the only problem...

Real-life Conditions

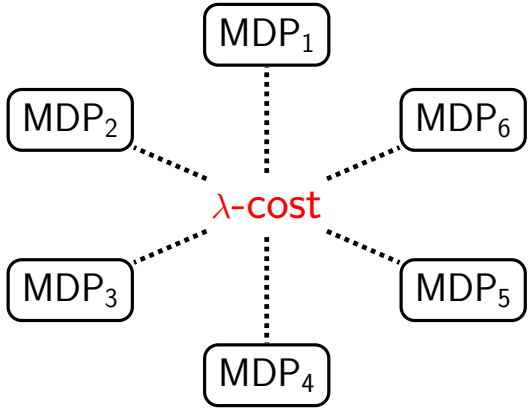
- Agents live in an **uncertain environment** (effect of actions, available capacity)
- Possibly operating **without communication** during policy execution

Approach: compute plans that are not conditioned on states of other agents by setting (time-dependent) limits initially per agent.

- However, a robust pre-allocation of available capacity (Wu and Durfee, 2010) gives poor performance in uncertain environments, and
- satisfying constraints in expectation (CMDPs, Altman 1999) gives violations about 50% of the time.

Our contribution: limit **violation probability** of constraints (e.g., by $\alpha = 0.05$)

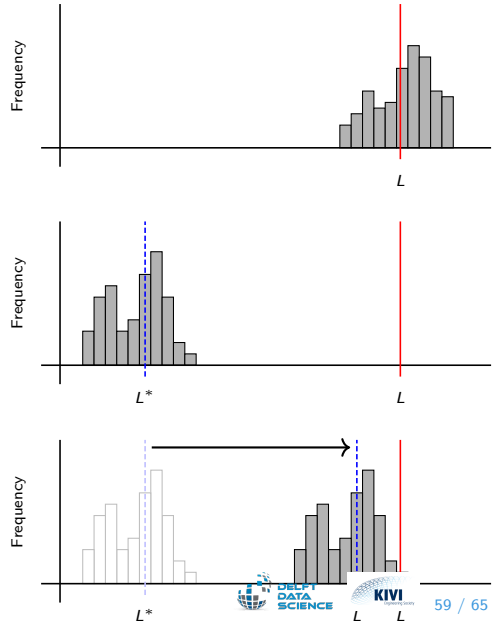
Decoupled multiagent planning

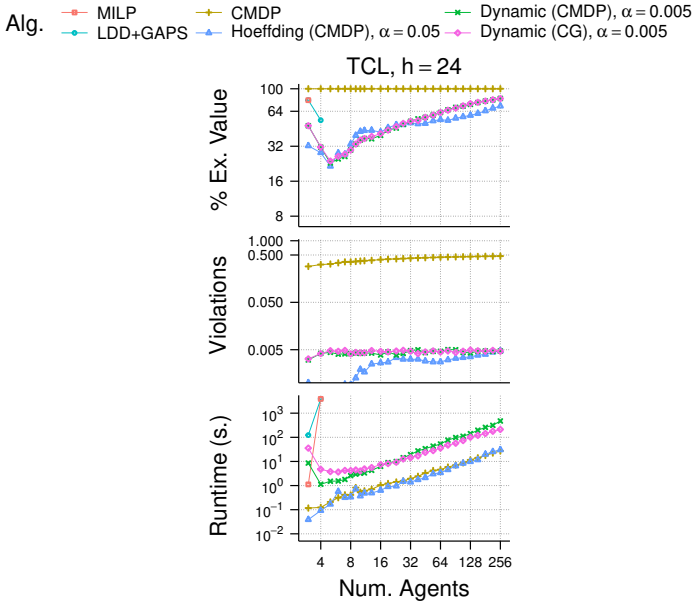


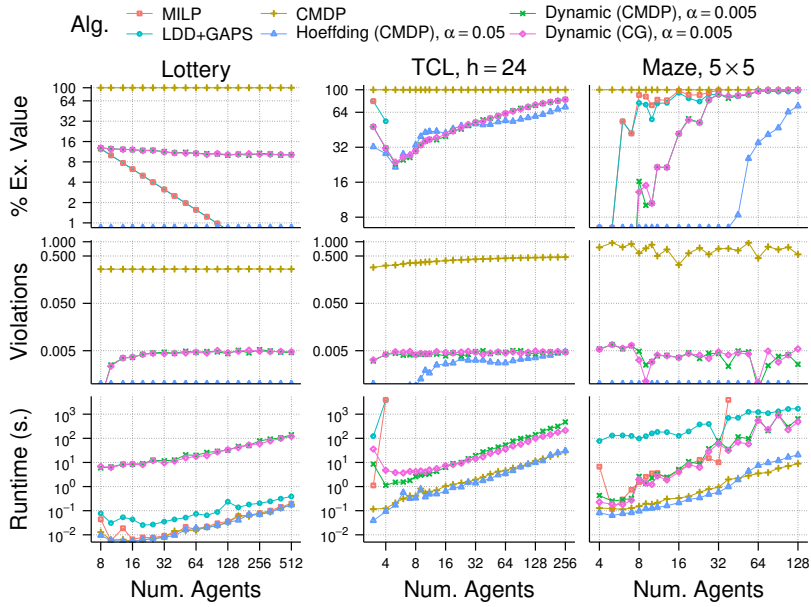
Satisfying Limits in Expectation (**CMDP**)

Reduced Limits with **Hoeffding**'s inequality given α

Dynamic Relaxation of Reduced Limits by Simulation




Alg.
MILP
LDD+GAPS
CMDP
Hoeffding (CMDP), α = 0.05
Dynamic (CMDP), α = 0.005
Dynamic (CG), α = 0.005
TCL, h = 24
% Ex. Value
100
64
32
16
8
Violations
1.000
0.500
0.050
0.005
Runtime (s.)
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
4
8
16
32
64
128
256
Num. Agents


Alg.
MILP
LDD+GAPS
CMDP
Hoeffding (CMDP), α = 0.05
Dynamic (CMDP), α = 0.005
Dynamic (CG), α = 0.005
Lottery
TCL, h = 24
Maze, 5 × 5
% Ex. Value
Violations
Runtime (s.)
Num. Agents
Num. Agents
Num. Agents

Discussion

- Static preallocation (MILP): worse expected value and poor scaling (exponential in number of agents)
- Expected consumption (CMDP): high expected value but also high likelihood of violations (near 50%)
- Directly applying Hoeffding bound underestimates violation likelihood by an order of magnitude
- Dynamic bounding ensures tightly bounded violation probability, outperforming other approaches

However,

- TCL owners can obtain more comfort by declaring a slightly higher desired temperature

Summary

- Data science and decision making are tightly interlinked, because of computational limits
- The Future Power System has significant Computational Challenges
- General techniques: dynamic programming, (mixed) integer programming, decision-theoretic planning, also for multiple agents
- Concrete successes in the context of smart grids: modeling wind scenarios, coordinating heat pumps

- There are many remaining challenges for computer science on the path to the future grid.

This is an open invitation to everyone to contribute to
the creation of a smart grid.

Please contact us at `m.t.j.spaan@tudelft.nl` and `m.m.deweerdt@tudelft.nl`

Big thank you to students and colleagues who contributed to this talk:

- Rens Philipsen, German Morales Espana, and Laurens de Vries
- Frits de Nijs and Erwin Walraven
- Vincent Conitzer and Michael Alberts

References

- Frits de Nijs, Matthijs T. J. Spaan, and Mathijs de Weerdt (2015). Best-Response Planning of Thermostatically Controlled Loads under Power Constraints. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 615–621. AAAI Press.
- Frits de Nijs, Erwin Walraven, Mathijs de Weerdt, and Matthijs T. J. Spaan (2017). Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pp. 3562-3568, San Francisco, CA, USA. AAAI.
- Rens Philipsen, Mathijs de Weerdt, and Laurens de Vries (2016). Auctions for Congestion Management in Distribution Grids. In *13th International Conference on the European Energy Market*.
- Rens Philipsen, German Morales-Espana, Mathijs de Weerdt, and Laurens de Vries (2016). Imperfect Unit Commitment Decisions with Perfect Information: a Real-time Comparison of Energy versus Power. In Proc. of the *Power Systems and Computation Conference*.
- Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nicholas R Jennings. Putting the 'Smarts' into the Smart Grid: A Grand Challenge for Artificial Intelligence. *Communications of the ACM* 55, no. 4 (2012): 86–97.
- Erwin Walravan and Matthijs T. J. Spaan (2015), Planning under Uncertainty with Weighted State Scenarios. In *Proc. of Uncertainty in Artificial Intelligence*.