

From Policy Learning to Autonomous Algorithm Design: AI for Network Optimization
Universidad Carlos III de Madrid

Decisions under constraints and dynamics: safe contextual bandits and reinforcement learning

2026-09-21

Jose A. Ayala Romero · NEC Laboratories Europe

Agenda

1. Constraints in networking problems
2. Lagrange-like utility functions
3. Bayesian online learning: the technique
4. Bayesian online learning: BP-vRAN and EdgeBOL
5. Distributional characterisation of constraints
6. Reinforcement learning: the technique
7. Reinforcement learning: AegisRAN
8. Summary of Day 1

Constraints in networking problems

Why “maximise the KPI” is rarely the whole story

Most network objectives come with a constraint

- Minimise **energy** subject to a minimum **throughput** per user
- Maximise the **sleep time** of a base station subject to a maximum **delay**
- Minimise **computing resources** subject to a minimum **quality of service**
- Minimise **bandwidth** subject to a **fidelity** target for the transmitted image

In all of them the constraint must hold **at every decision step**, not on average, and it must hold **while we are still learning**.

The constrained contextual bandit

One metric to maximise, others to keep below a bound, all of them unknown and noisy

$$\max_{\pi} \sum_{t=1}^T r_t(c_t, \pi(c_t)) \quad \text{subject to} \quad g_t^{(m)}(c_t, \pi(c_t)) \leq g_{\max}^{(m)} \quad \forall m, \forall t$$

- Reward r_t and constraint metrics $g_t^{(m)}$ are **random**: measurement noise plus intrinsic randomness of the system
- Their dependence on context and action is **unknown**: we learn it from the same samples we act on
- Violations **during learning count**: the network is in production while we explore

Three ways to handle the constraint

Penalise

Lagrange-like utility
 $u = r - \lambda [g - g_{\max}]^+$

any bandit algorithm
no guarantee

Model the uncertainty

Gaussian processes
safe set from
confidence bounds

data-efficient, guarantees
costly with many samples

Learn the distribution

distributional critics
bound a quantile
of each metric

neural, constant-time
risk level α as a knob

Agenda

1. Constraints in networking problems
2. **Lagrange-like utility functions**
3. Bayesian online learning: the technique
4. Bayesian online learning: BP-vRAN and EdgeBOL
5. Distributional characterisation of constraints
6. Reinforcement learning: the technique
7. Reinforcement learning: AegisRAN
8. Summary of Day 1

Lagrange-like utility functions

The quick fix, and what it does not give you

Turn the constraint into a penalty

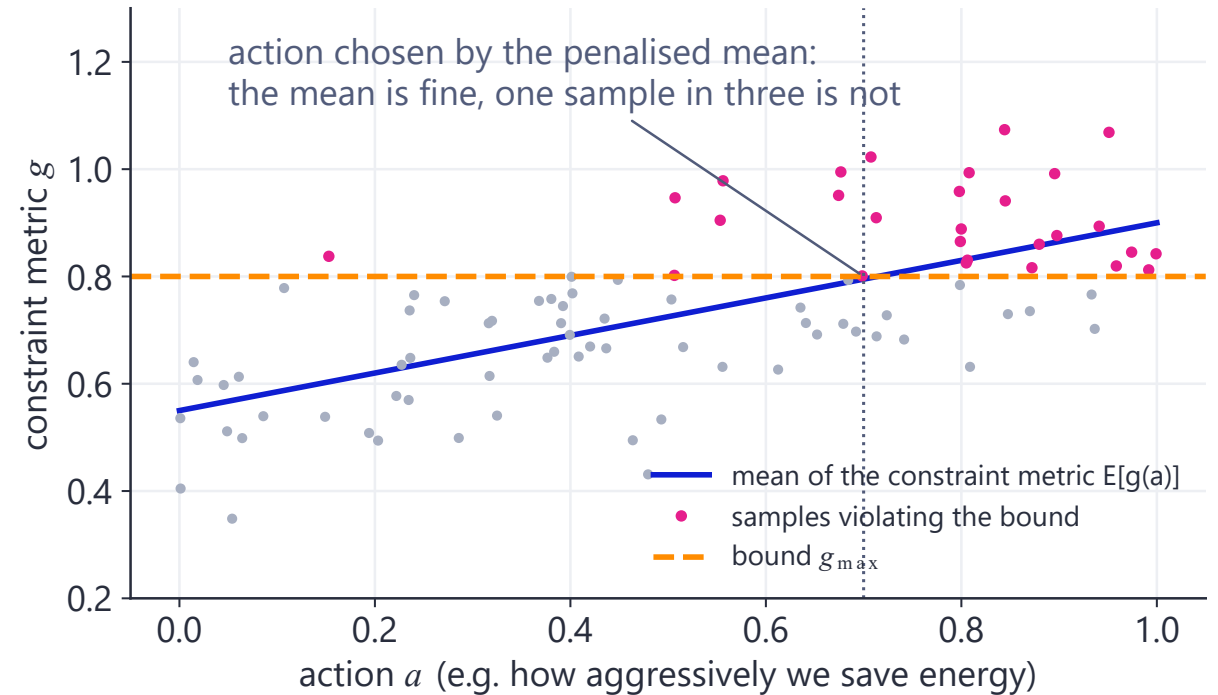
Fold the constraint into the reward and run any unconstrained algorithm

$$u(c, a) = r(c, a) - \lambda [g(c, a) - g_{\max}]^+ \quad \text{or, as in vrAIn,} \quad r = \sum_i (J_i - M \varepsilon_i - \lambda c_i)$$

- λ (or M) prices a violation; a large value makes the learner avoid it on average
- Everything from yesterday applies unchanged: LinUCB, neural bandits, actor-critic
- Inspired by Lagrangian relaxation, but λ is a **fixed design choice**, not a multiplier that is optimised

What can go wrong

- λ must be tuned per problem, and re-tuned when the noise changes
- No guarantee at any step, only a tendency
- The learner sees the **mean** of the penalised utility: an action whose mean satisfies the bound can still violate it in a third of the rounds
- One scalar hides *which* metric failed and by how much



When a penalty is enough

Acceptable when

- the requirement is soft and violations are cheap
- the noise is small and roughly the same everywhere
- a fixed trade-off between metrics is what you want anyway

Not enough when

- the requirement is hard (reliability, delay budgets, power supply)
- the noise depends on the action (heavy tails in some regions)
- violations during learning are unacceptable

A penalty encodes a preference, not a guarantee

Exercise 1: same reward, different outcome

Two actions A and B have the same mean reward, and their constraint metric g has the same mean. A's g is tightly concentrated while B's is widely spread.

1. How do their violations differ, and does the mean of g show it?
2. What does a larger λ do to A and B?

- A exceeds the bound by a little; B by a lot, compensated on average by values far below the bound. The mean of g cannot tell them apart
- The penalty prices the expected excess $\mathbb{E}[(g - g_{\max})^+]$, which is much larger for B: a larger λ penalises B more and pushes the learner away from it first
- λ trades violations against reward on average; it never bounds a single step. For that we need a model of the uncertainty (next section) or the distribution of g itself (the section after)

Agenda

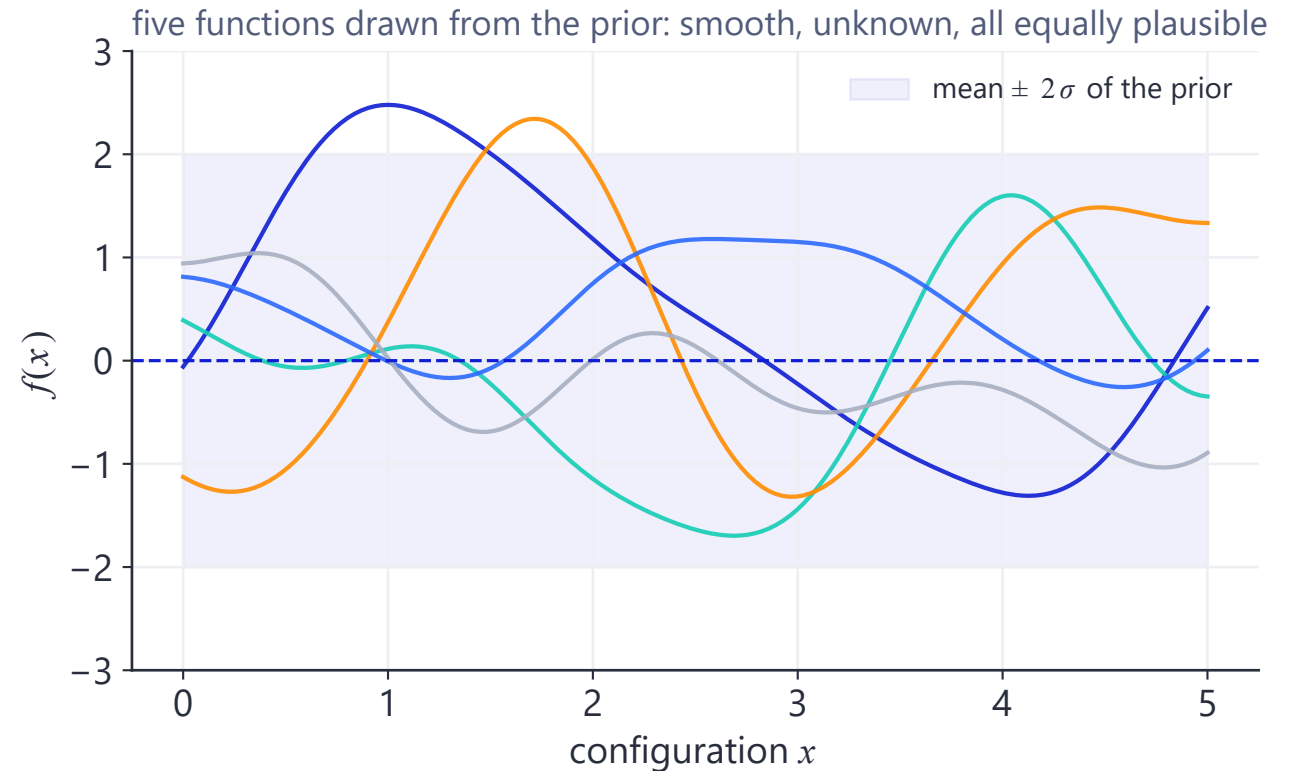
1. Constraints in networking problems
2. Lagrange-like utility functions
3. **Bayesian online learning: the technique**
4. Bayesian online learning: BP-vRAN and EdgeBOL
5. Distributional characterisation of constraints
6. Reinforcement learning: the technique
7. Reinforcement learning: AegisRAN
8. Summary of Day 1

Bayesian online learning: the technique

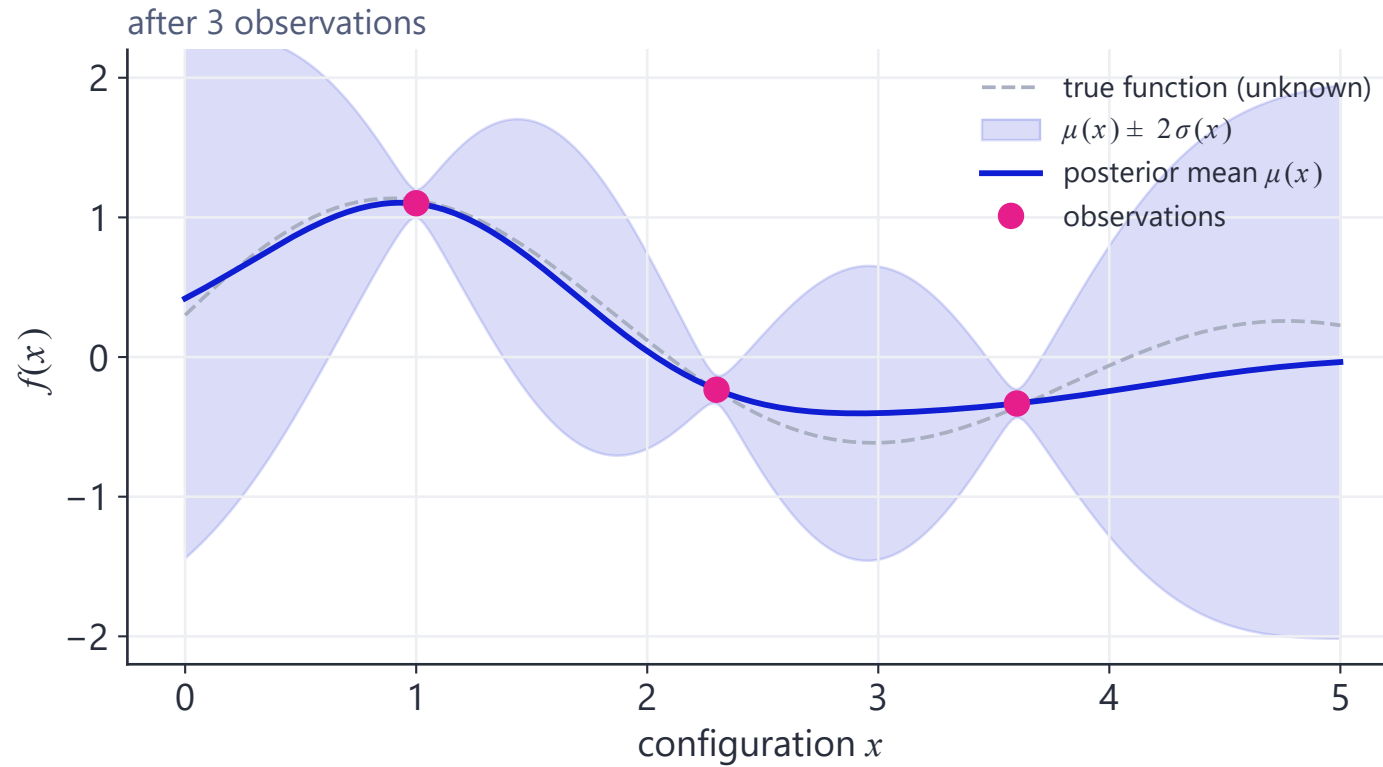
Gaussian processes, acquisition functions and safe sets

A Gaussian process is a distribution over functions

- A GP is specified by a mean function $m(x)$ and a covariance (kernel) function $k(x, x')$
- Ask it about any x and it returns a Gaussian: mean $\mu(x)$, standard deviation $\sigma(x)$
- Before any data, all smooth functions are equally plausible
- We take $m(x) = 0$; the kernel carries the prior knowledge ([Rasmussen and Williams 2006](#))

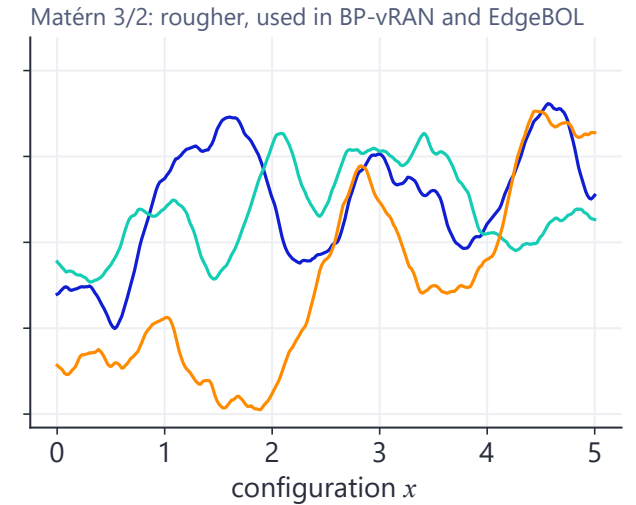
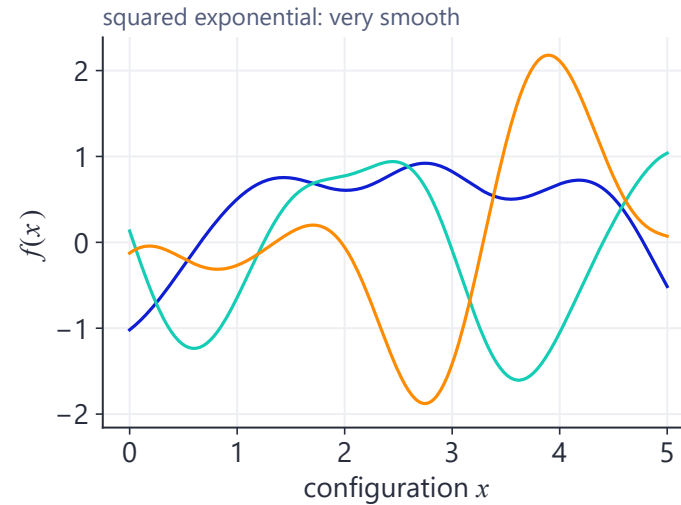


Three observations shrink the uncertainty



The kernel encodes smoothness

- $k(x, x')$ says how correlated two function values are as a function of their distance
- Squared exponential: very smooth functions
- **Matérn 3/2**: rougher, closer to what testbeds show; chosen in BP-vRAN and EdgeBOL
- One **length scale per dimension** (anisotropic): each knob has its own reach
- Hyperparameters are fitted on prior data and then **kept fixed**: re-fitting online can trap the optimiser



The posterior has a closed form

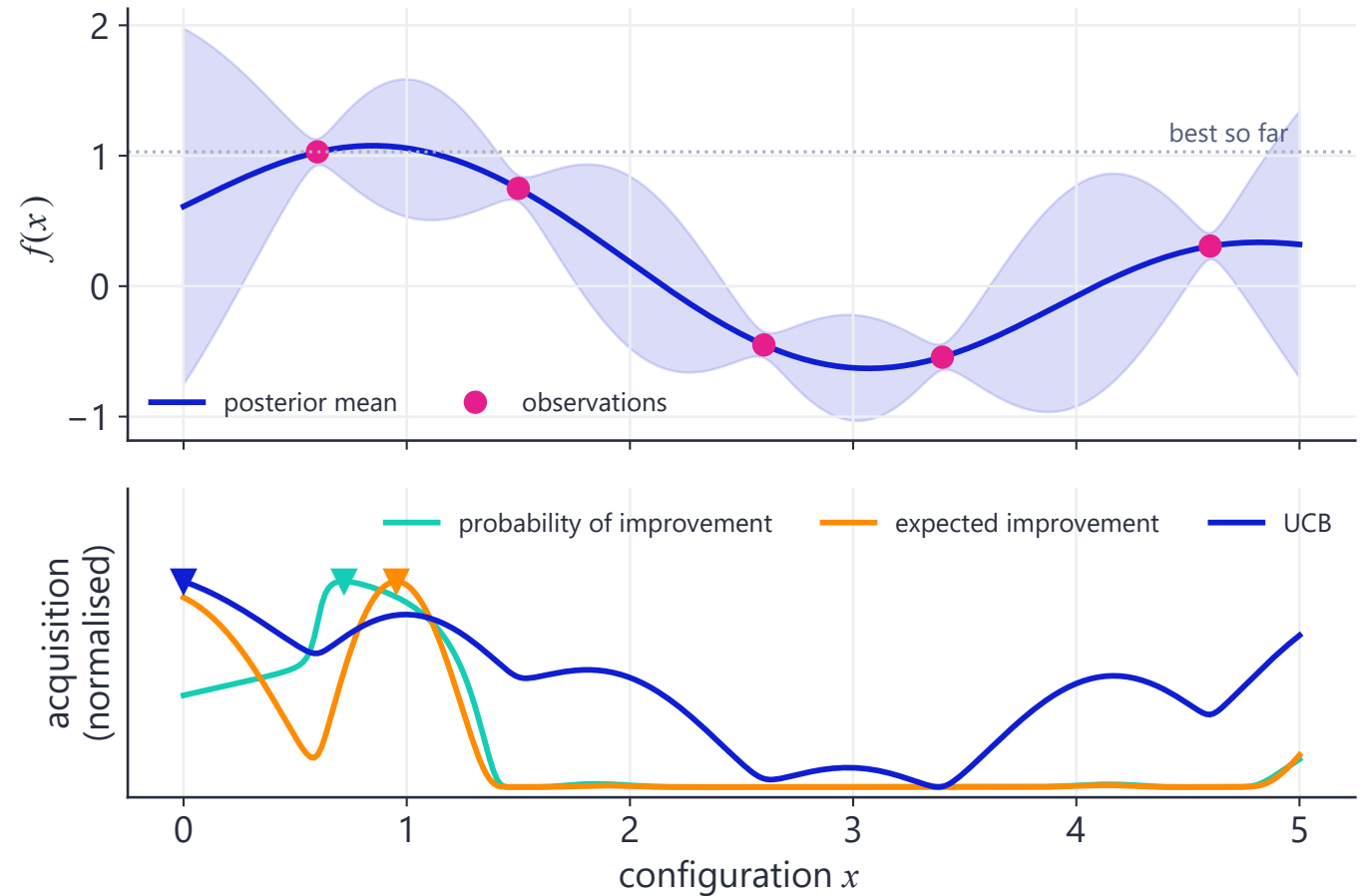
Observations $y_i = f(x_i) + \varepsilon_i$ at points X , with noise variance σ_n^2 . For a new point x :

$$\mu(x) = k(x, X) [K + \sigma_n^2 I]^{-1} y, \quad \sigma^2(x) = k(x, x) - k(x, X) [K + \sigma_n^2 I]^{-1} k(X, x)$$

- K is the kernel matrix of the observed points; the joint of observed and new values is Gaussian, and conditioning a Gaussian is linear algebra
- Cost $O(N^3)$ in the number of observations: fine for hundreds of decisions per hour, not for millions
- This is the entire learning step; no gradient descent, no epochs

Acquisition functions guide the search

- The posterior tells us where the function is high and where it is uncertain; an **acquisition function** turns both into a score
- Probability of improvement, expected improvement, and **UCB** $\mu(x) + \sqrt{\beta} \sigma(x)$
- UCB is yesterday's optimism principle applied to a continuous space ([Brochu et al. 2010](#))



Contextual GP-UCB

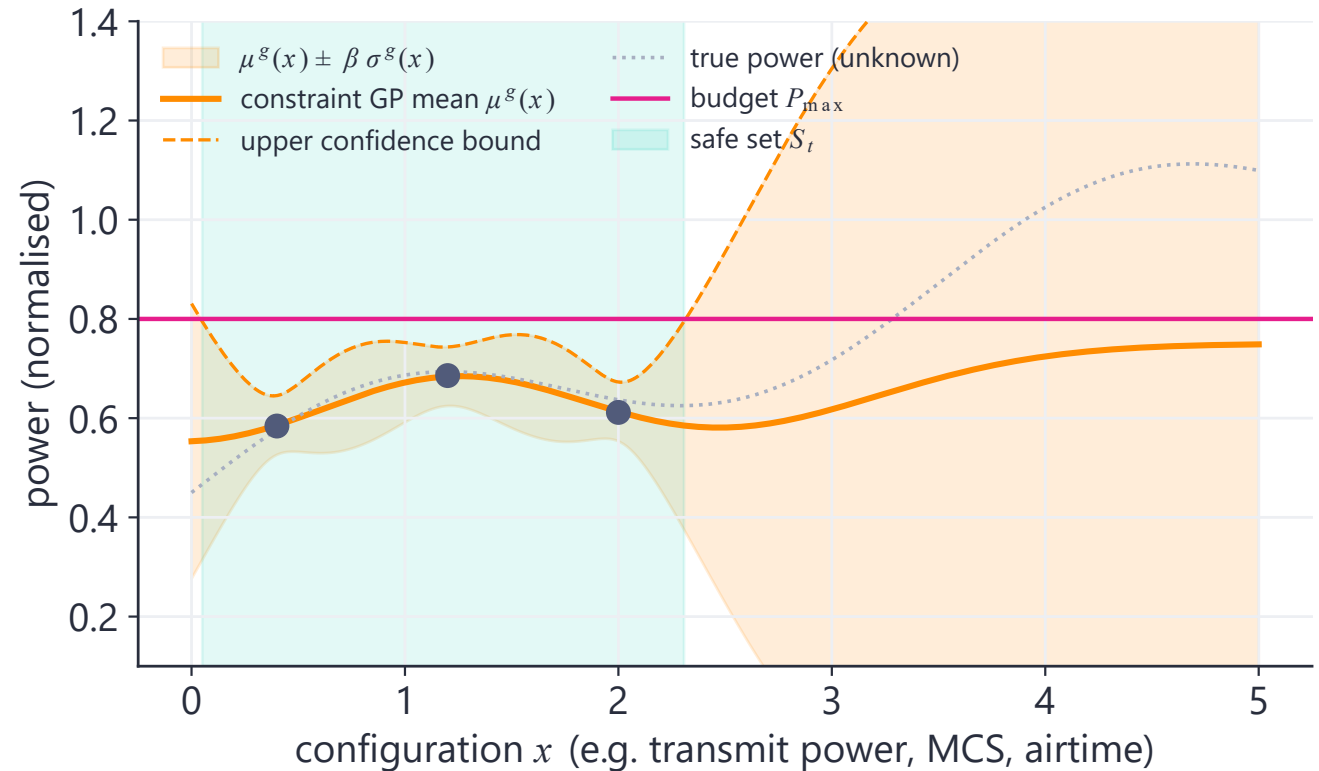
One GP over the joint (context, action) space; at every round, maximise over the action only

$$x_t = \arg \max_{x \in \mathcal{X}} \mu_{t-1}(c_t, x) + \sqrt{\beta_t} \sigma_{t-1}(c_t, x)$$

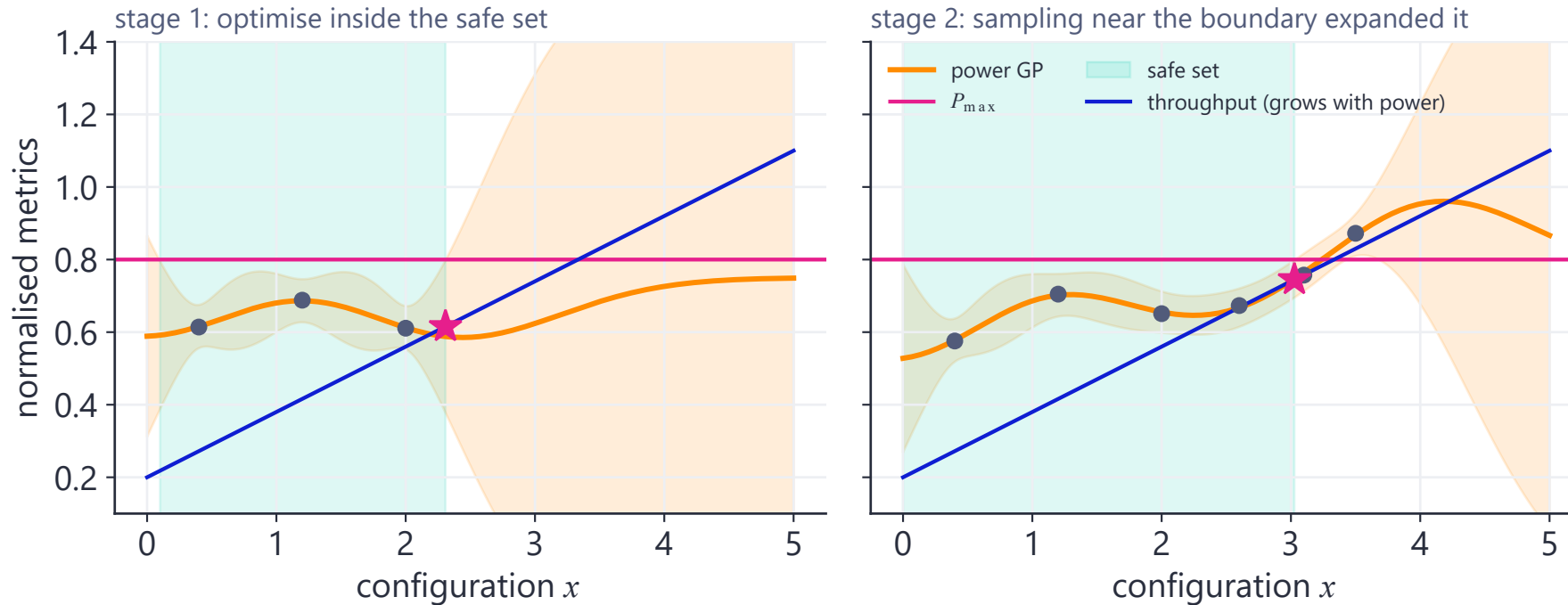
- The kernel over contexts transfers what was learned under similar conditions
- With the prescribed schedule for β_t , the contextual regret is sublinear (no-regret) ([Krause and Ong 2011](#))
- In practice β is a fixed constant: larger means more exploration

Safe optimisation: act only where the constraint is likely to hold

- Fit a second GP to the constraint metric g
- **Safe set** at round t :
$$S_t = S_0 \cup \{x : \mu^g(c_t, x) + \beta \sigma^g(c_t, x) \leq g_{\max}\}$$
- S_0 : a few conservative configurations known to be safe
- Maximise the reward acquisition **inside** S_t only
- Violations happen only when the confidence bound lies, i.e. with small probability



Expanding the safe set: explicit or implicit



- SafeOpt (Sui et al. 2015) spends samples explicitly on expanding the boundary; in our experiments that was too slow
- When reward and constraint are positively correlated (throughput and power), the best safe configurations sit **on the boundary**; optimising there also shrinks the uncertainty there, so the safe set grows for free

Agenda

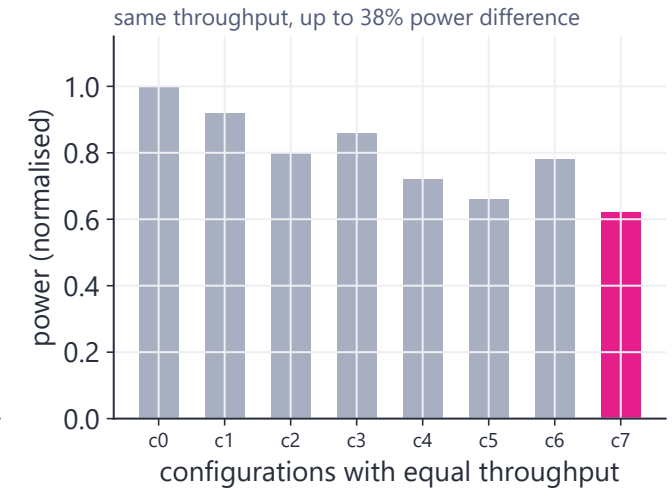
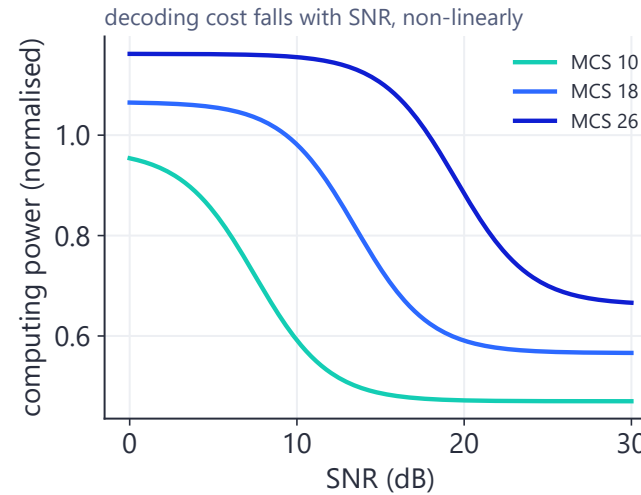
1. Constraints in networking problems
2. Lagrange-like utility functions
3. Bayesian online learning: the technique
4. **Bayesian online learning: BP-vRAN and EdgeBOL**
5. Distributional characterisation of constraints
6. Reinforcement learning: the technique
7. Reinforcement learning: AegisRAN
8. Summary of Day 1

Bayesian online learning: BP-vRAN and EdgeBOL

Energy-aware configuration of virtual base stations and edge AI services

Energy-aware vBS configuration: what the testbed showed

- Computing power of a software base station is comparable to, or larger than, the radio unit's power
- SNR and MCS affect decoding time, hence power, **non-linearly**; uplink is costlier and more volatile
- Many (MCS, airtime) configurations reach the **same throughput** with very different power
- Joint DL + UL power is not the sum of the parts: legacy linear power models do not hold ([Ayala-Romero et al. 2023b](#))



Context, action, reward

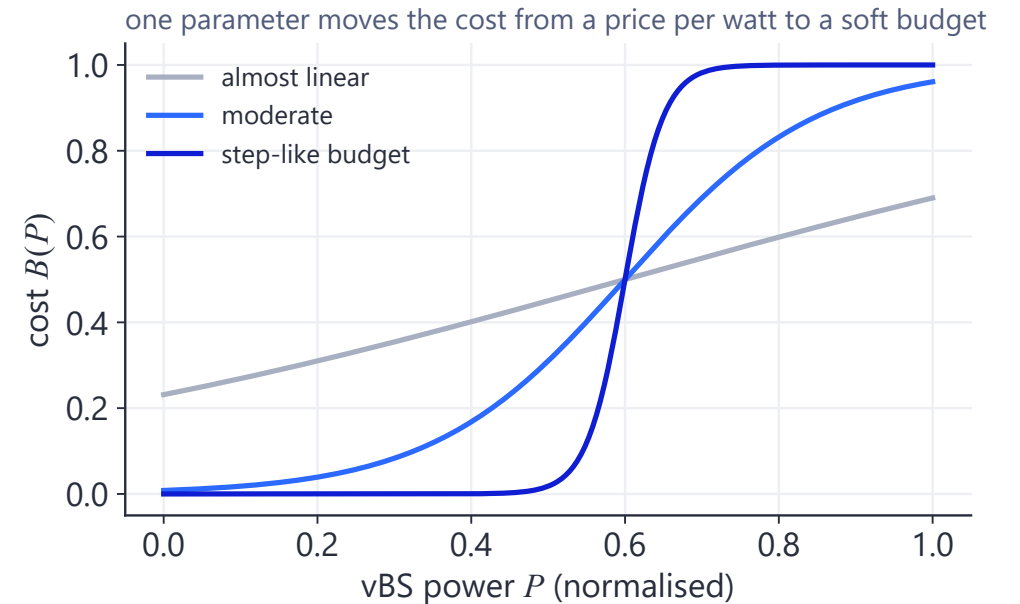
Context c_t : DL and UL mean and variance of the users' channel quality, plus new bit arrivals (exogenous)

Action x_t : DL transmit power, DL and UL maximum MCS, DL and UL maximum airtime

Reward: $r = \log(1 + R^{dl}/d^{dl}) + \log(1 + R^{ul}/d^{ul})$, throughput normalised by demand

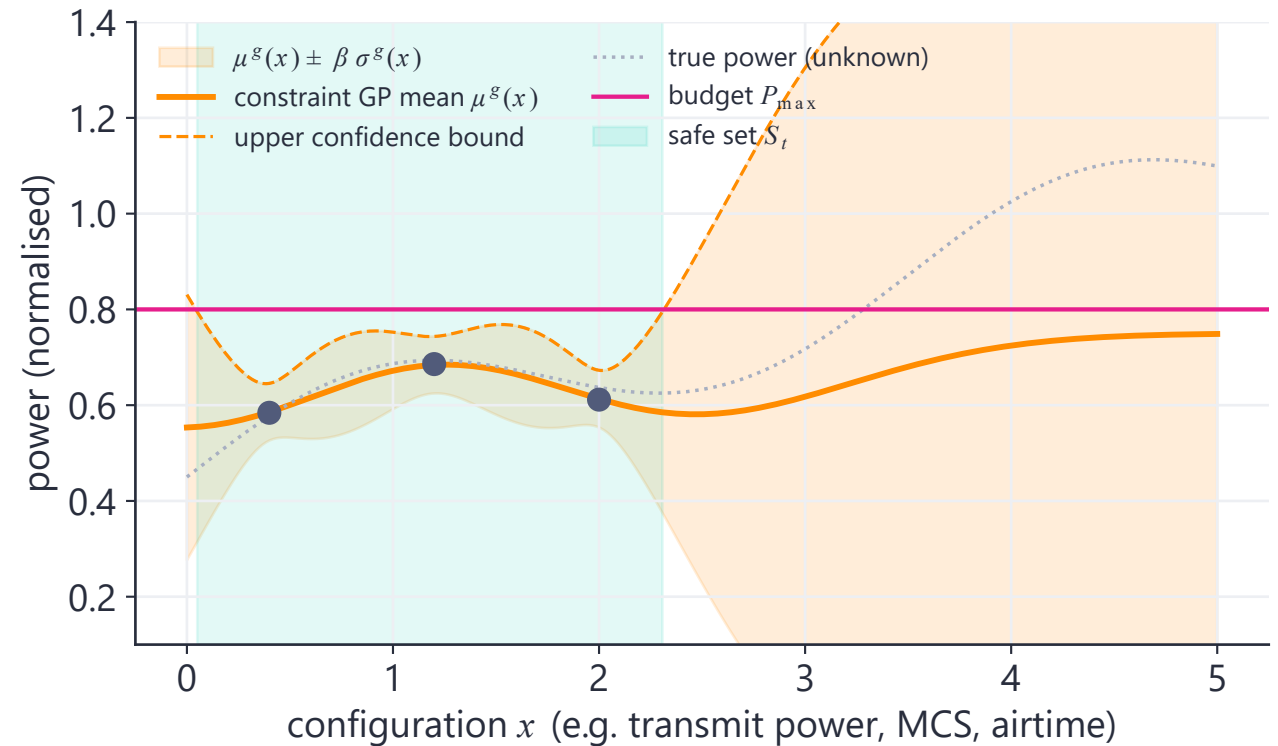
Use case 1: balance throughput and power (BP-vRAN)

- A tethered small cell with mains power: no hard limit, but energy has a price
- Utility $u = r - \delta B(P)$ with a parameterised sigmoid cost B : from a linear price per watt to a soft budget
- **One GP** over (context, action) for u ; contextual GP-UCB acquisition
- No-regret guarantee inherited from CGP-UCB



Use case 2: hard power budget (SBP-vRAN)

- Solar, battery or Power-over-Ethernet cells: exceeding $P_{\max}$ is not an option, not even while exploring
- **Two GPs:** throughput reward and power; the safe set from the power GP
- GP-UCB on the reward restricted to S_t ; the safe set expands implicitly along the boundary
- $P_{\max}$ can be changed at run time without retraining

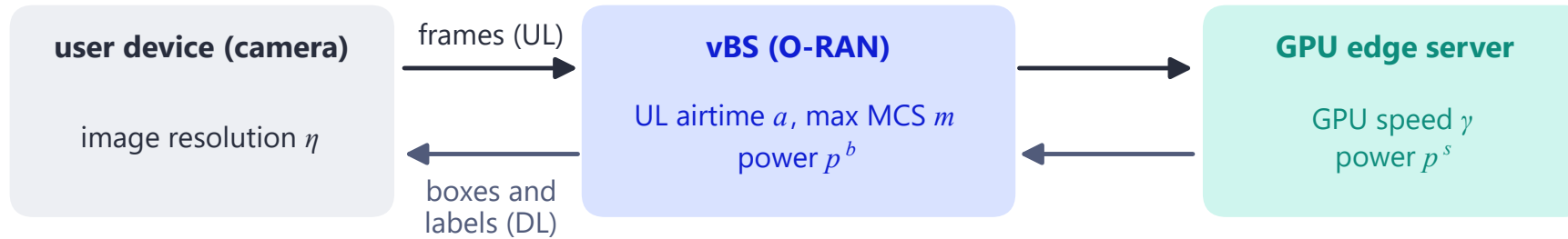


Data efficiency versus deep RL

- The journal version compares with a DDPG actor-critic adapted to the contextual bandit setting (yesterday's vrAIIn machinery)
- The Bayesian learner converges **an order of magnitude faster** and reaches the optimum
- Why: the kernel encodes smoothness, so every sample informs its neighbourhood; no network to train from scratch
- The price: cubic cost and the need for prior data to fit the kernel

EdgeBOL: jointly configuring the base station and the edge AI service

context: number of users, mean and variance of their UL channel quality



minimise $\delta_1 p^s + \delta_2 p^b$ s.t. delay $d \leq d_{\max}$ and accuracy $\rho \geq \rho_{\min}$, every period

- Mobile video analytics: users upload frames, the edge server returns detections
- Higher resolution: better accuracy, more delay; more airtime: less delay, more power at both ends; faster GPU: more power, less delay. All platform dependent
- Optimising one element alone can increase total energy ([Ayala-Romero et al. 2023a](#))

EdgeBOL: three GPs and a safe set with two constraints

How

- GPs for the cost, the delay and the accuracy (mAP), Matérn kernels
- Safe set: upper bound of delay $\leq d_{\max}$ **and** lower bound of mAP $\geq \rho_{\min}$
- Lower confidence bound on the cost, minimised inside the safe set
- Context = number of users and CQI statistics: fixed size for any number of users

What it gave

- Adapts to different hardware and service requirements at run time
- Outperforms neural-network-based RL benchmarks on the prototype with real traces
- Falls back to S_0 when the constraints are infeasible for the current context

One GP per metric, and the constraint becomes a region you can see

Exercise 2: draw the safe set

Look at the safe-set figure from the technique slide: a GP of the power metric, its confidence band, and the budget $P_{\max}$.

1. Where is the safe set?
2. What happens to it if β doubles?
3. Where should the next sample go if we want the safe set to grow?

- Where the **upper** edge of the band is below the budget, not where the mean is
- A larger β widens the band: the safe set shrinks, exploration becomes more conservative
- Just inside the boundary: sampling there shrinks the band where it matters, and that is also where the best safe reward usually is

Agenda

1. Constraints in networking problems
2. Lagrange-like utility functions
3. Bayesian online learning: the technique
4. Bayesian online learning: BP-vRAN and EdgeBOL
5. **Distributional characterisation of constraints**
6. Reinforcement learning: the technique
7. Reinforcement learning: AegisRAN
8. Summary of Day 1

Distributional characterisation of constraints

Learning the whole distribution of each metric, and choosing how much risk to accept

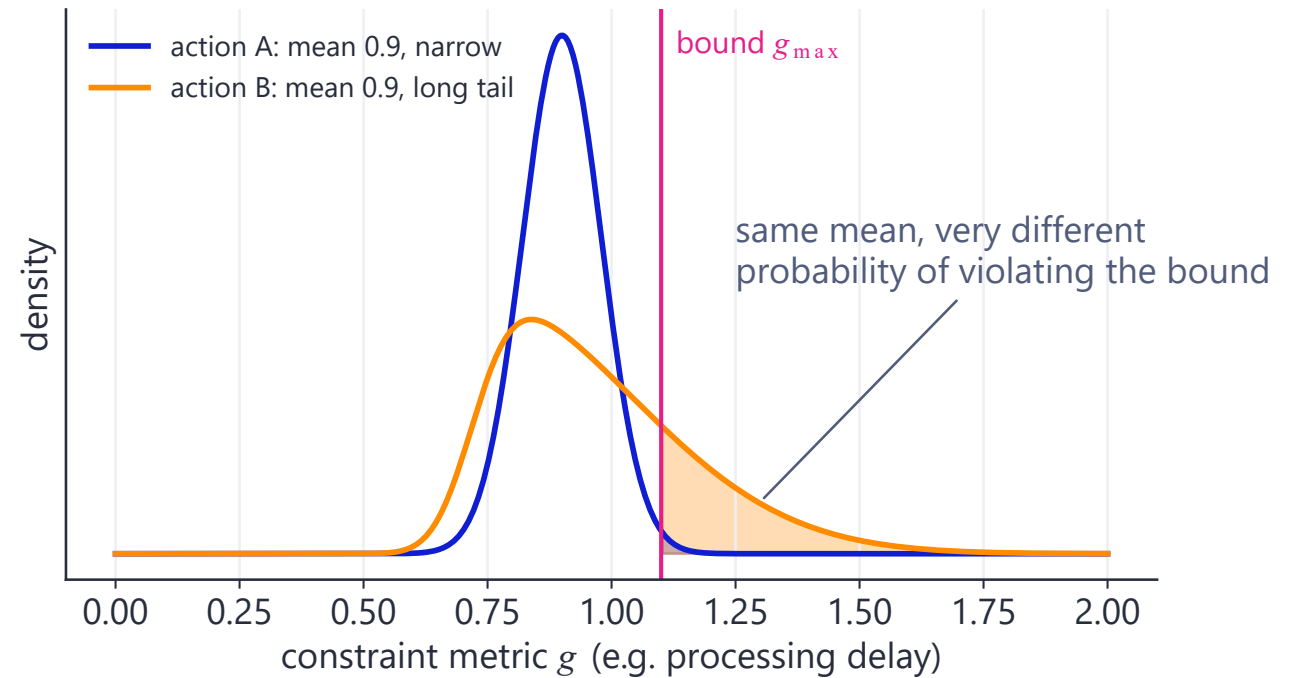
Limits of the GP approach

- $O(N^3)$ per update
- Kernel hyperparameters need prior data from the same platform
- An initial safe set S_0 must be known
- Dimensionality: a GP over many knobs and many context features needs many samples

Wanted: a neural learner with constant-time inference that still treats constraints as constraints, not as a penalty on the mean.

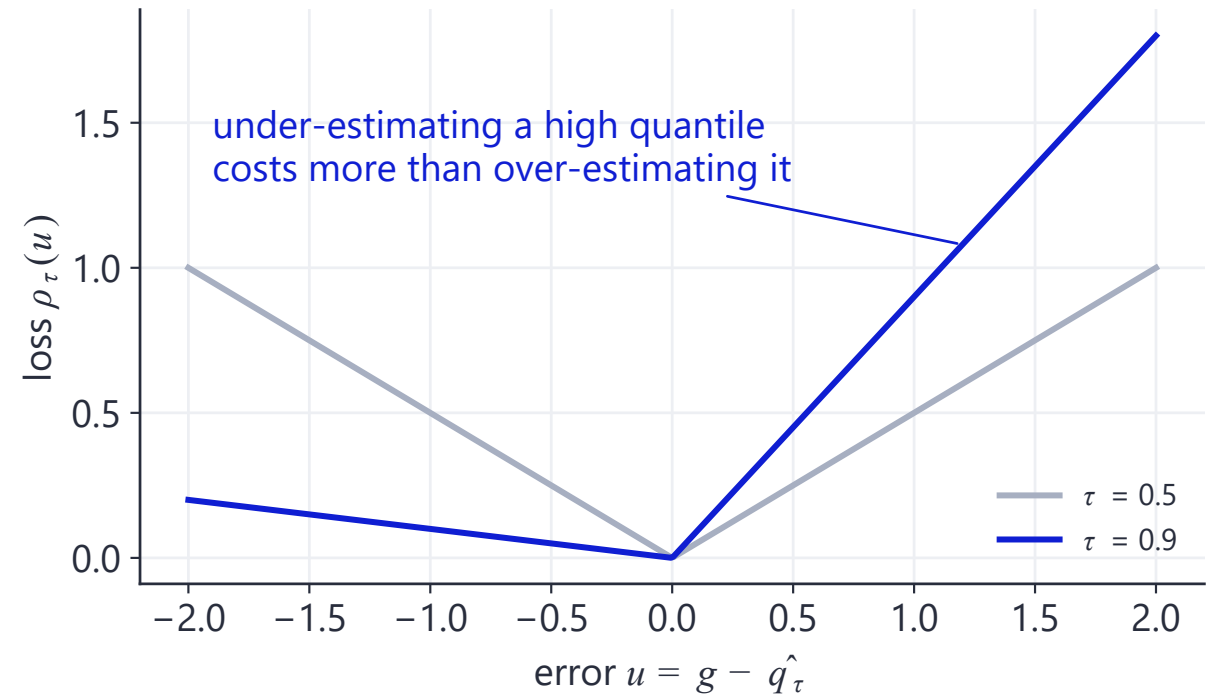
Noise is intrinsic: two actions with the same mean are not the same

- Constraint metrics are random even for a fixed context and action (**aleatoric** uncertainty): decoding times, queue delays, accuracy of a detector
- Per-step satisfaction depends on the **tail** of the distribution
- A method that learns only the mean (or a penalised mean) cannot tell A from B



Learning a distribution with quantile regression

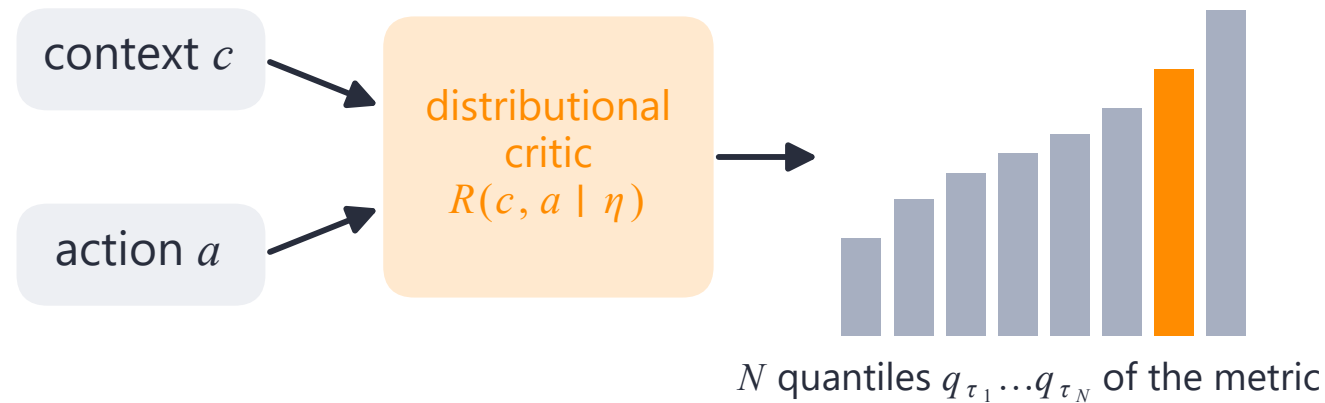
- The quantile function $q_\tau = F^{-1}(\tau)$ of a metric: the value below which a fraction τ of the samples fall
- Regress it directly with the **quantile loss**
 $\rho_\tau(u) = u(\tau - 1\{u < 0\})$, $u = g - \hat{q}_\tau$
- Asymmetric: for $\tau = 0.9$, under-estimating costs nine times more than over-estimating
- A smoothed version (quantile Huber loss) trains well with gradient descent



A distributional critic outputs N quantiles

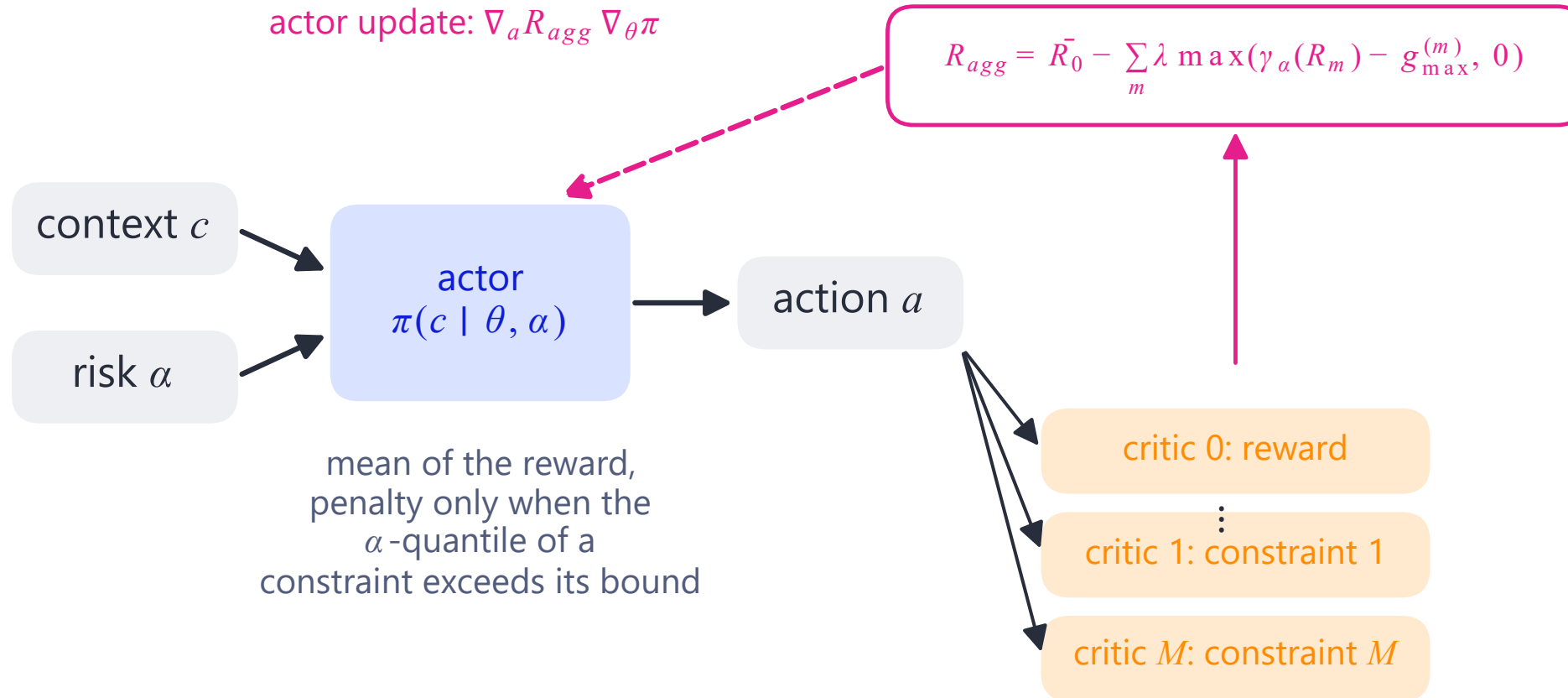
trained with the quantile Huber loss on every observed (c, a, g)

γ_α : the α -quantile (orange)



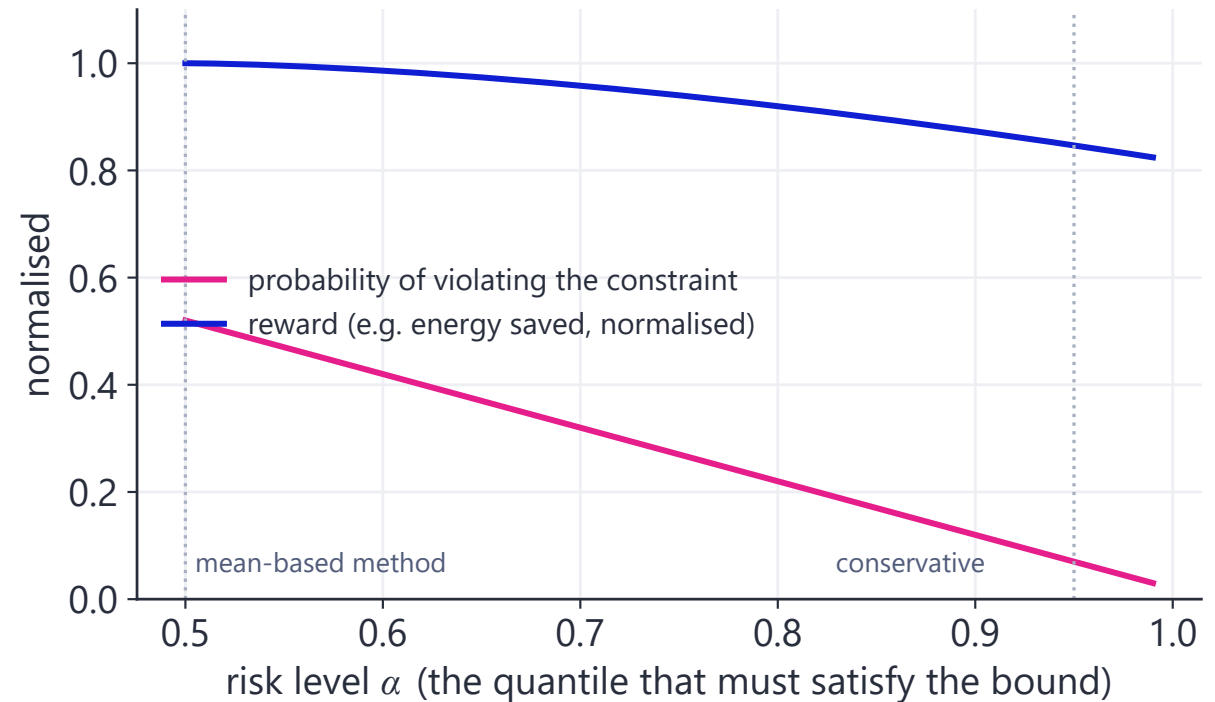
- One network from (context, action) to N quantile values, trained with the quantile Huber loss on every observed sample
- The α -quantile γ_α is read off directly: "the value the metric stays below with probability α "
- Same inference cost as an ordinary critic

RANCB: one actor, one critic per metric



The risk level α is an input of the actor

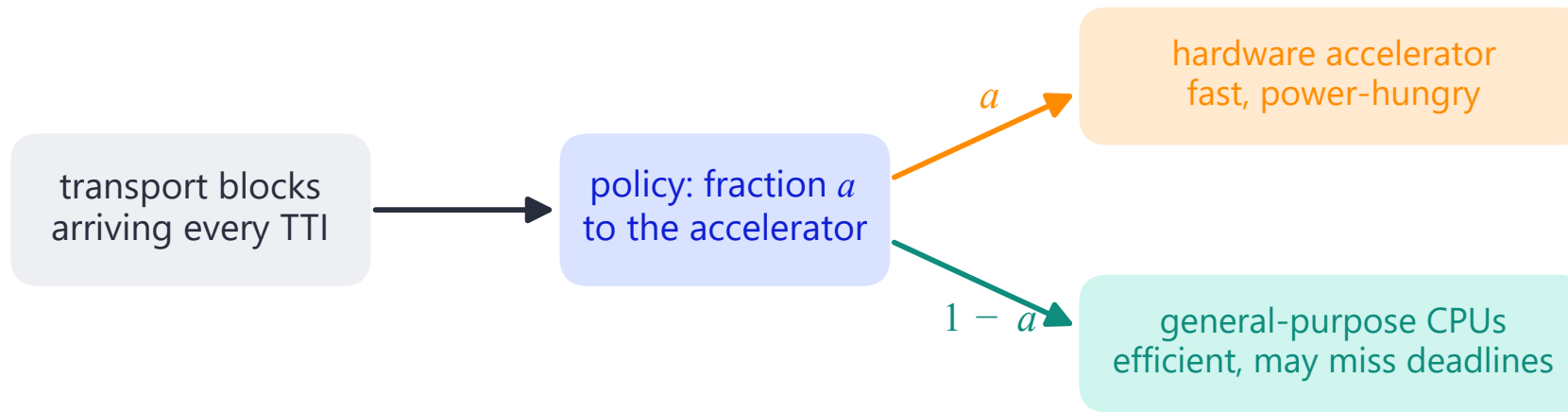
- Policy $\pi(c \mid \theta, \alpha)$: the actor sees α and learns one policy **per risk level**
- Training: for every mini-batch, compute R_{agg} for a whole set of α values; no extra data needed
- $\alpha \rightarrow 0.5$ reproduces the mean-based method; $\alpha \rightarrow 1$ makes the tail satisfy the bound
- Different α per constraint are possible



RANCB training loop

1. Observe context c_t ; act with $a_t = \pi(c_t) + \epsilon$ exploration noise
2. Observe reward r_t and constraint metrics $g_t^{(1)}, \dots, g_t^{(M)}$; store the transition in a replay buffer
3. Sample a mini-batch; update every critic with the quantile Huber loss on its metric
4. For each α in the training set, form R_{agg} from the critics' quantiles
5. Update the actor by the deterministic policy gradient through R_{agg}
6. Repeat; exploration noise can be switched off in production

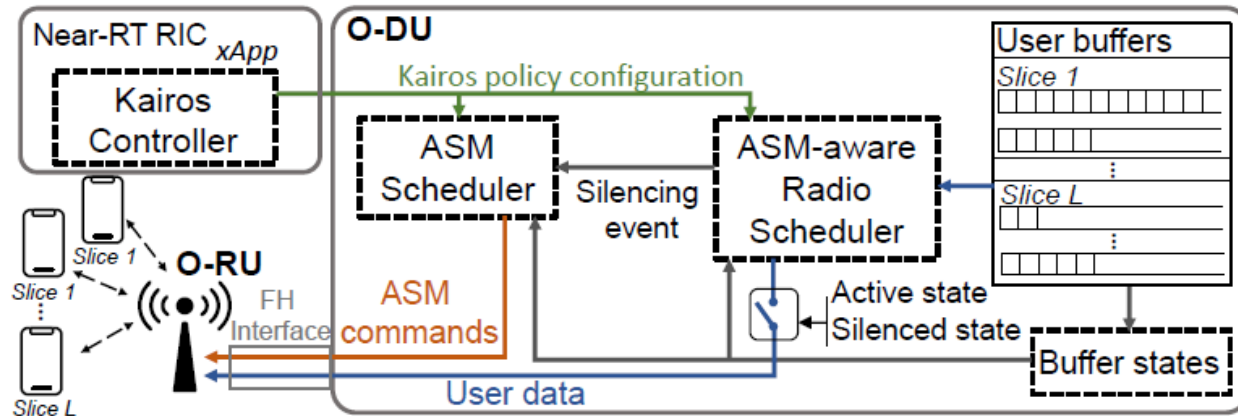
Application: CPUs versus hardware accelerators in 5G



context: load and channel quality · reward: energy · constraint: fraction of blocks decoded in time $\geq$ target

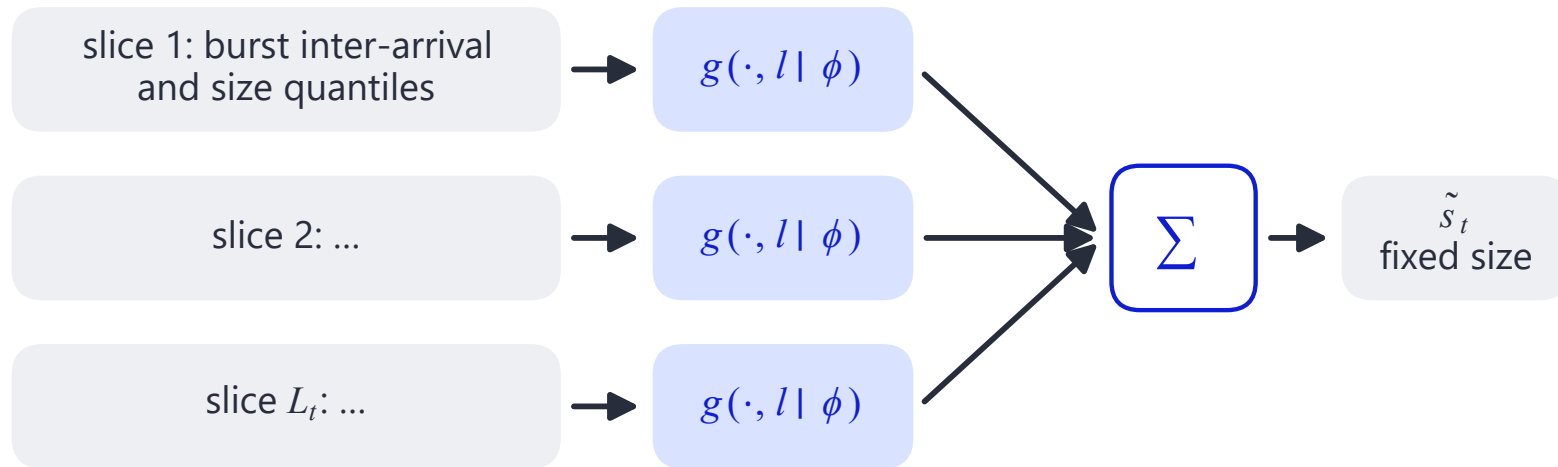
- Every 100 ms decide which fraction of the signal-processing load goes to the energy-hungry accelerator
- Minimise energy subject to a **reliability** target: the fraction of blocks processed within their deadline
- On the real platform only the risk-aware method meets the target consistently, at a modest energy price ([Ayala-Romero et al. 2024](#))

Kairos: sleep modes with per-slice delay guarantees



- The radio unit is about 90% of a base station's energy, and cells are idle most of the time at millisecond scale
- **Advanced sleep modes**: deeper modes save more but take longer to wake up
- Real-time part in the DU: pack data into few symbols, silence when the buffer empties, wake when the oldest burst exceeds a delay budget d_t
- Learning part in an $xApp$: choose d_t from the traffic context (Salvat Lozano et al. 2025)

Kairos: a variable number of constraints



the same network g embeds every slice with its id; summing makes the input size independent of L_t

- Context per slice: quantiles of burst inter-arrival times and burst sizes; the set of active slices changes over time
- One distributional **critic per active slice** for its delay, plus one for energy; penalties flow only through active critics
- Same actor, same aggregated cost, same deterministic policy gradient

SPIFF: the actor conditioned on the requirements

The problem

- Images over a satellite link: keep the perceptually important regions intact, compress the rest, reconstruct with generative AI at the receiver
- Action: bounding-box size and fraction of preserved patches (continuous)
- Bits to minimise; fidelity minima per relevance rank (soft); an end-to-end delay deadline (hard)

The learner

- One critic per term: bits, each fidelity, delay
- Low quantile for the fidelities, high quantile for the delay
- The actor takes the requirement weights λ as input, so changing the requirements needs **no retraining** ([Palena et al. 2026](#))

Condition the policy on what may change: risk level, weights, number of slices

Exercise 3: the operator changes a bound

Two slices share a radio unit: a URLLC slice with a 1 ms delay bound and an eMBB slice with a 10 ms bound. At run time, the operator lowers the eMBB bound to 5 ms.

1. What has to be retrained in Kairos, and what can be kept?
2. How could the design avoid the retraining?
3. What has to be retrained in the GP approach?

- The **actor**: its policy was optimised for the old bound. The critics can be kept: they model the delay distribution of each context and action, which does not depend on the bound
- Make the bounds an **input of the actor** and train it over a range of bounds, as SPIFF does with its requirement weights; a new bound then needs no retraining
- Nothing: the GPs model the metric itself, and the bound only enters the safe-set test, which is recomputed at every round

Agenda

1. Constraints in networking problems
2. Lagrange-like utility functions
3. Bayesian online learning: the technique
4. Bayesian online learning: BP-vRAN and EdgeBOL
5. Distributional characterisation of constraints
6. **Reinforcement learning: the technique**
7. Reinforcement learning: AegisRAN
8. Summary of Day 1

Reinforcement learning: the technique

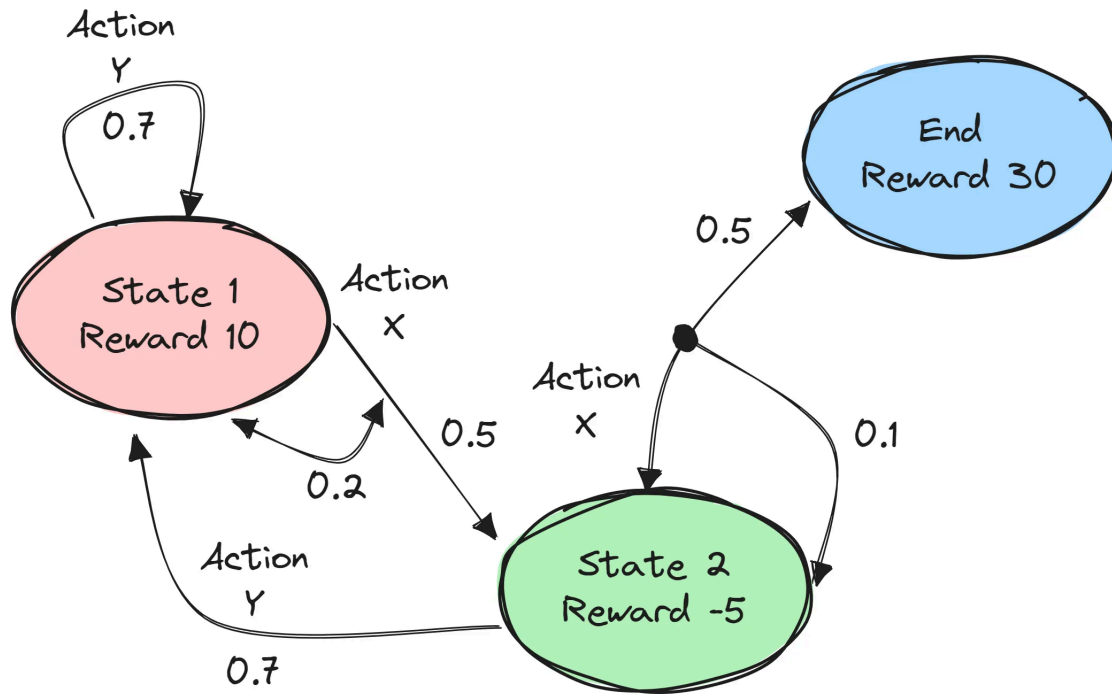
When actions change the future

When the bandit assumptions fail

The reward is delayed, and today's action changes tomorrow's state

- **Delayed, accumulated reward:** what we optimise is the return over a trajectory, not the reward of one step
- **Actions shape the future:** the next state depends on the current state and action
- Classic examples: chess, robot control, navigation
- Networking examples: a resource assignment made in several dependent steps; a sleep decision that lets queues grow; a routing choice that changes the loads we will observe

The Markov decision process



- States $s \in \mathcal{S}$, actions $a \in \mathcal{A}$
- Transition probabilities $\mathbb{P}[s_{t+1} = s' \mid s_t = s, a_t = a]$
- Reward $\mathbb{E}[r_{t+1} \mid s_t, a_t]$
- **Markov property:** the state summarises the history,
 $\mathbb{P}[s_{t+1} \mid s_t, a_t] = \mathbb{P}[s_{t+1} \mid s_1, a_1, \dots, s_t, a_t]$

Return, policy, value functions

- **Return:** the discounted sum of future rewards, $G_t = \sum_{k \geq 0} \gamma^k r_{t+k+1}$, with $\gamma \in [0, 1)$
- **Policy:** $\pi(a \mid s) = \mathbb{P}[a_t = a \mid s_t = s]$
- **State value:** $v_\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s]$; **action value:** $q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a]$
- **Bellman idea:** the value of a state is the immediate reward plus the discounted value of where you land; every RL algorithm is a way of solving that recursion from samples ([Sutton and Barto 2018](#))

Families of algorithms

Family	Learns	Examples	Actions
Value-based	$q(s, a)$, policy = arg max	Q-learning, DQN	discrete
Policy gradient	$\pi_{\theta}(a \mid s)$ directly	REINFORCE, TRPO, PPO	any
Actor-critic	both: a critic guides the actor	DDPG, TD3, SAC	any

- Yesterday's actor-critic for contextual bandits is the special case with a one-step horizon: the critic estimates r instead of q
- With $\gamma > 0$, the critic must bootstrap from its own estimate of the next state's value

On-policy versus off-policy

On-policy (SARSA, PPO, TRPO)

- the policy that collects data is the one being improved
- fresh data after every update
- simpler and stable, but data-hungry

Off-policy (Q-learning, DDPG, TD3, SAC)

- learn a target policy from data collected by a **behaviour** policy
- reuse old trajectories from a replay buffer (sample efficiency)
- the behaviour policy can be the network's current operating policy

In networks the cost of collecting new data decides the choice: off-policy methods can learn from logs and from a digital twin.

Soft actor-critic: keep the policy random on purpose

$$J(\pi) = \mathbb{E} \left[\sum_t r_t + \eta \mathcal{H}(\pi(\cdot \mid s_t)) \right]$$

- Maximise return **plus the entropy** of the policy: exploration is part of the objective, not an add-on
- Off-policy, twin critics against over-estimation, robust to hyperparameters ([Haarnoja et al. 2018](#))
- **Discrete SAC** ([Christodoulou 2019](#)): with a finite action set the state value is computed exactly over the policy's probabilities, no sampling of actions
- Convergence guarantees under standard conditions

The practitioner's toolkit for RL in networks

- [Digital twins](#): learned or simulated environments to generate trajectories without touching production
- [Replay buffers](#): reuse every transition many times (off-policy)
- [Invalid-action masking](#): give zero probability to actions that violate hard constraints, instead of penalising them ([Huang and Ontañón 2020](#))
- [Dense reward shaping](#): a signal at every step instead of one at the end of an episode
- [Placement](#): as an rApp in the Non-RT RIC, reconfiguring through O2 and reading telemetry through E2 and O2

Every item returns in the next section.

Agenda

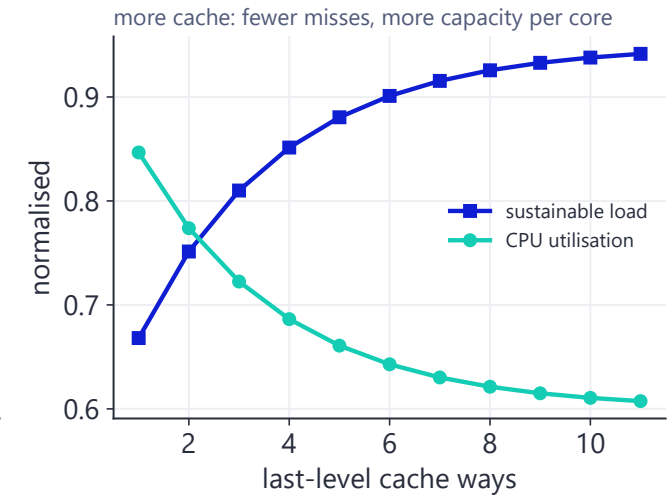
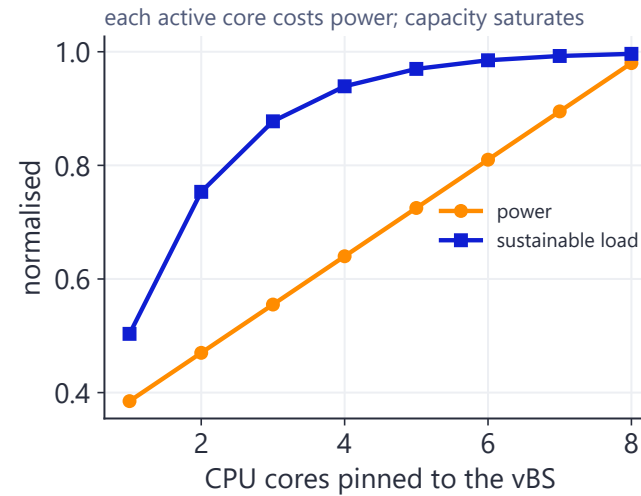
1. Constraints in networking problems
2. Lagrange-like utility functions
3. Bayesian online learning: the technique
4. Bayesian online learning: BP-vRAN and EdgeBOL
5. Distributional characterisation of constraints
6. Reinforcement learning: the technique
7. **Reinforcement learning: AegisRAN**
8. Summary of Day 1

Reinforcement learning: AegisRAN

Fair and energy-efficient computing allocation for many virtual base stations

Cores and cache decide energy and reliability in vRAN

- More pinned cores raise power linearly, while capacity saturates
- More last-level cache ways reduce misses: more capacity per core, better energy efficiency
- Exceeding the compute capacity loses user synchronisation: throughput drops to **zero**
- The optimum depends on the traffic and MCS of each base station ([Sanchez Hidalgo et al. 2025](#))

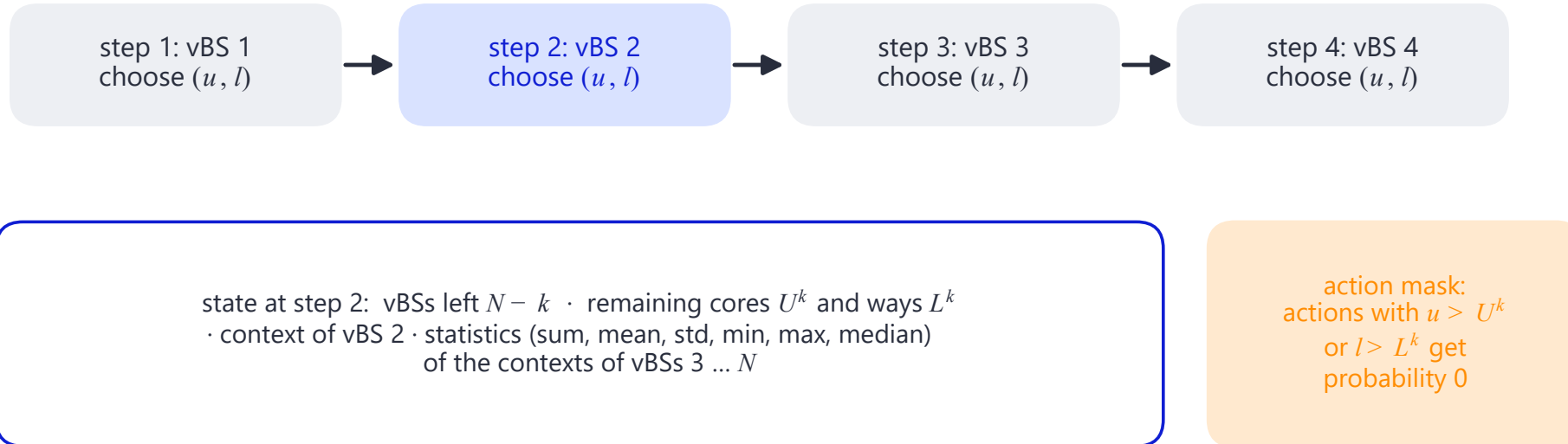


The problem: energy, reliability and fairness with 10^{36} actions

- N virtual base stations share a node with U cores and L cache ways, allocated in a dedicated way: $x^n = (u^n, l^n)$
- Context per base station: uplink and downlink demand and MCS; power $p^n(c, x)$ and bit-loss ratio $b^n(c, x)$ are unknown functions
- **Not saturated**: minimise total power s.t. bit loss $\leq \epsilon$ and capacity
- **Overbooked**: minimise $\sum_n f_\alpha(b^n)$, an α -fair function of the losses ($\alpha = 1$ proportional, $\alpha \rightarrow \infty$ max-min)
- One cannot tell in advance which regime applies, and the joint action space has $\binom{N+U}{N} \binom{N+L}{N}$ elements: about 10^{36} for $N = 20, U = 40, L = 20$

Turn one huge decision into a trajectory of N small ones

one epoch = one allocation round: N steps, action space $(U+1)(L+1)$ instead of $\binom{N+U}{N}\binom{N+L}{N}$



fixed-size state for any N ; the statistics let the agent leave room for the heavy vBSs still to come

A dense reward that covers both regimes

One signal per step, energy when there are no losses, fairness when there are

$$r_k = -p_k - \lambda f_\alpha(b_k)$$

- p_k : power of the base station served at step k ; b_k : its bit loss
- Without losses the reward is pure energy minimisation; with a large λ , losses dominate and the α -fair term spreads them
- A reward at every step avoids the sparse end-of-epoch signal that makes RL slow to learn
- Discrete SAC at the core: entropy-driven exploration, little tuning

Training on a digital twin, deployed as an rApp

Training

- One learned twin per base-station *type*, predicting power and bit loss from context and allocation
- Twins are stacked to emulate large multi-cell systems: on-policy data at scale with no risk to the network

Deployment and results

- rApp in the Non-RT RIC: reallocates container resources through O2, reads context through E2, energy through O2
- Near-optimal across contexts and numbers of base stations; up to 25% energy savings in medium-scale scenarios

RL earned its place here because the decision itself is sequential

Exercise 4: bandit or RL

vrAI and AegisRAN both allocate computing resources to virtual base stations from their traffic context.

Why is one a contextual bandit and the other a Markov decision process?

- The **environment** is the same in both: contexts are exogenous, the reward of an allocation is immediate
- AegisRAN *creates* the MDP by splitting one allocation into N dependent steps: what is left after step k is the state of step $k + 1$
- The trick buys a small action space and a fixed-size state; it costs an RL algorithm instead of a bandit
- Lesson: the formulation is a design choice, not a property of the network

Agenda

1. Constraints in networking problems
2. Lagrange-like utility functions
3. Bayesian online learning: the technique
4. Bayesian online learning: BP-vRAN and EdgeBOL
5. Distributional characterisation of constraints
6. Reinforcement learning: the technique
7. Reinforcement learning: AegisRAN
8. **Summary of Day 1**

Summary of Day 1

Choosing a technique

Question	If yes	If no
Is the reward immediate and are rounds independent?	bandit family	reinforcement learning
Is there observable side information?	contextual bandit	multi-armed bandit
Are there constraints? soft	penalised reward	plain reward
Are there constraints? hard, few knobs, prior data	GP with a safe set	
Are there constraints? hard, many knobs, fast decisions	distributional critics, risk level α	
Is the action continuous?	actor-critic	

Tomorrow: when a learned policy is not what you want to deploy

- Everything today produced a **policy**: a NN or a GP that maps context to action
- Operators ask: can I read it, can I audit it, does it run in microseconds, does it survive a new deployment?
- Tomorrow: **generating heuristics** with large language models instead of learning policies, from FunSearch to AlphaEvolve, and where the two days meet

References

- Ayala-Romero, Jose A., Andres Garcia-Saavedra, and Xavier Costa-Perez. 2024. "Risk-Aware Continuous Control with Neural Contextual Bandits." *Proceedings of the AAAI Conference on Artificial Intelligence* 38.
- Ayala-Romero, Jose A., Andres Garcia-Saavedra, Xavier Costa-Perez, and George Iosifidis. 2023a. "EdgeBOL: A Bayesian Learning Approach for the Joint Orchestration of vRANs and Mobile Edge AI." *IEEE/ACM Transactions on Networking* 31 (6): 2978–93.
- Ayala-Romero, Jose A., Andres Garcia-Saavedra, Xavier Costa-Perez, and George Iosifidis. 2023b. "Orchestrating Energy-Efficient vRANs: Bayesian Learning and Experimental Results." *IEEE Transactions on Mobile Computing* 22 (5): 2910–24.
- Brochu, Eric, Vlad M. Cora, and Nando de Freitas. 2010. "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning." *arXiv Preprint arXiv:1012.2599*.
- Christodoulou, Petros. 2019. "Soft Actor-Critic for Discrete Action Settings." *arXiv Preprint arXiv:1910.07207*.
- Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." *International Conference on Machine Learning (ICML)*.
- Huang, Shengyi, and Santiago Ontañón. 2020. "A Closer Look at Invalid Action Masking in Policy Gradient Algorithms." *arXiv Preprint arXiv:2006.14171*.
- Krause, Andreas, and Cheng Soon Ong. 2011. "Contextual Gaussian Process Bandit Optimization." *Advances in Neural Information Processing Systems (NeurIPS)*.
- Palena, Marco, Jose A. Ayala-Romero, Andres Garcia-Saavedra, and Carla Fabiana Chiasserini. 2026. "SPIFF: Selective Preservation of Image Fidelity for Bandwidth-Constrained Heterogeneous Networks." *IEEE INFOCOM*.
- Rasmussen, Carl Edward, and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press.
- Salvat Lozano, J. Xavier, Jose A. Ayala-Romero, Andres Garcia-Saavedra, and Xavier Costa-Perez. 2025. "Kairos: Energy-Efficient Radio Unit Control for O-RAN via Advanced Sleep Modes." *IEEE INFOCOM*.
- Sanchez Hidalgo, Ethan, Jose A. Ayala-Romero, J. Xavier Salvat Lozano, Andres Garcia-Saavedra, and Xavier Costa-Perez. 2025. "AegisRAN: A Fair and Energy-Efficient Computing Resource Allocation Framework for vRANs." *IEEE Transactions on Mobile Computing*.
- Sui, Yanan, Alkis Gotovos, Joel Burdick, and Andreas Krause. 2015. "Safe Exploration for Optimization with Gaussian Processes." *International Conference on Machine Learning (ICML)*.
- Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. 2nd ed. MIT Press.

NEC

\Orchestrating a brighter world