NETSIM FILE EXCHANGE · AI/ML FOR COMMUNICATION NETWORKS

AI/ML-based Mobility Load Balancing in 5G

Learning Cell Individual Offsets with Bayesian optimisation

Applicable Release: NetSim v15.1

Applicable Version(s): NetSim Standard

Project download link:

https://github.com/NetSim-TETCOS/AI-ML-CIO-Load-Balancing-v15.1/archive/refs/heads/main.zip

The link has the exported NetSim scenario and the Python scripts used in this document.

Please refer to slide 21 for the steps to run the project.

INTRODUCTION

Mobility load balancing in multi-band 5G

In a multi-band cell site, each UE attaches to the cell it hears best. The lowest band usually wins, so it saturates while the higher-band carriers sit idle.

PROBLEM

One cell carries the load

An overloaded cell gives every UE it serves a small share of resource blocks: lower throughput and more delay, with spare capacity unused next door.

LEVER

Cell Individual Offset

A CIO makes a cell look better or worse in the association and handover decision, steering UEs towards under-loaded cells.

THIS PROJECT

From example to best CIOs

The built-in NetSim example shows that CIO moves load. This project extends it: an agent searches for the CIOs that balance the load best.

The result is one CIO pair for one scenario and one load, found by a search over NetSim runs, not a trained policy. Load balancing is one of the use cases of the 3GPP Rel-17 study on AI/ML for NG-RAN (TR 37.817).

THE PROBLEM

Why the low band attracts most users

The three gNBs are co-located and identical except for frequency: 55 dBm, 4T4R sector antennas, 40 MHz each.

Path loss grows with frequency (the 20·log10(4π·d0/λ) term), so every UE hears gNB2 4.9 dB and gNB3 7.6 dB weaker than gNB1.

Cell	Band	Frequency	Extra path loss vs gNB1
gNB1	n50	1475 MHz	0 dB
gNB2	n38	2595 MHz	4.9 dB
gNB3	n78	3550 MHz	7.6 dB

GNB1 · 1.5 GHZ

80%

PRB utilisation, saturated; serves 374 of 500 UEs

GNB2 · 2.6 GHZ

28%

PRB utilisation; serves 94 UEs

GNB3 · 3.5 GHZ

9%

PRB utilisation; serves 32 UEs

Without CIO, NetSim v15.1 · PRB utilisation after the 1 s warm-up

3GPP BACKGROUND

Cell Individual Offset and the A3 event

```
Event A3 entering condition: Mn + Ofn + Ocn - Hys > Mp + Ofp + Ocp + Off
```

```
NetSim: SINR_effective = SINR_measured + CIO
```

NR RRC, TS 38.331 §5.5.4.4. With equal frequency offsets (Ofn = Ofp) it reduces to Mn + Ocn – Hys > Mp + Ocp + Off. Ocn and Ocp are the cell individual offsets (cellIndividualOffset) of the neighbour and serving cells, defined since LTE RRC (TS 36.331 Rel-8, §5.5.4.4, where the serving cell is written Ms, Ocs) and in NR RRC from Rel-15. When the condition holds for the time-to-trigger, the UE sends a measurement report and the serving cell decides the handover.

Negative CIO

The cell looks worse, so it hands UEs over to its neighbours.

Positive CIO

The cell looks better, so it attracts UEs from its neighbours.

Only differences matter

Adding the same offset to every cell changes nothing, so one cell can stay at 0 dB as the reference.

In NetSim: gNB › Interface (5G RAN) › Datalink Layer › Handover › Cell Individual Offset

3GPP BACKGROUND

AI/ML for NG-RAN: the functional framework

The 3GPP Rel-17 study on AI/ML for NG-RAN (TR 37.817) defines four functions: data collection, model training, model inference and the actor, with feedback from the network back to data collection. This project's NetSim workflow maps onto them:

1
Data collection
PRB utilisation per cell and total throughput, from NetSim’s logs

2
Model training
Gaussian-process regression, refitted to all (CIO pair, reward) results after every run

3
Model inference
The GP predicts reward and uncertainty; Expected Improvement selects the next CIOs

4
Actor
The CIOs are written into the gNB configuration

↺
Feedback
Return path: the next simulation measures the effect and feeds data collection

The Rel-17 study names three use cases: network energy saving, load balancing and mobility optimisation. Rel-18 then specified supporting data collection and Xn signalling (TS 38.423).

NETSIM MODEL

Scenario and system parameters

Parameter	Value
gNBs	3 co-located sector carriers: n50 (1.5 GHz), n38 (2.6 GHz), n78 (3.5 GHz)
UEs and area	500 static UEs (75 near, 125 mid, 300 cell edge) in 600 m × 500 m
Numerology, bandwidth	1, 40 MHz per carrier
Tx power, antennas	55 dBm, 4T4R sector antennas
Channel	Log-distance path loss ($\eta = 3.8$), log-normal shadowing ($\sigma = 5$ dB)
Scheduler	Round Robin
Traffic per UE	Custom: 1460 B packets, exponential inter-arrival, mean 25 ms (467 kbps)
Handover	Handover margin 3 dB, time-to-trigger 0
Simulation time	10 s (3 s for each learning run)

Based on the NetSim v15.1 example “Load balancing in 5G using Cell Individual Offset (CIO)”, 5G-NR manual §4.5

BUILT-IN EXAMPLE

The example’s CIOs: –4.77 / 0 / +4.77 dB

The built-in NetSim example sets –4.77 dB on gNB1 and +4.77 dB on gNB3, with gNB2 at 0 dB.

The idea is intuitive. A negative CIO on gNB1 releases UEs from the loaded cell. A positive CIO on gNB3 attracts UEs to the lightly loaded cell.

The example shows that CIO moves load. It does not optimise the values. For reference, cancelling the path-loss gap of the previous slides gives about **–4.9 / 0 / +2.7 dB**.

EXAMPLE CIOS · NETSIM V15.1, 10 S

Cell	PRB utilisation	UEs served
gNB1 1.5 GHz	54%	214
gNB2 2.6 GHz	30%	101
gNB3 3.5 GHz	64%	185

Jain fairness 0.923 · throughput 229.2 Mbps. Load moves off gNB1; gNB3 becomes the busiest cell and gNB2 stays under-used.

UEs served: final association of the 3 s runs

AI MODE

The optimisation problem

ACTION

Two CIOs

CIO of gNB1 in −9…+3 dB and of gNB3 in −3…+9 dB, in 0.25 dB steps. gNB2 stays at 0 dB as the reference.

OBSERVATION

Load per cell

Mean PRB utilisation of each gNB after the 1 s warm-up, and the total application throughput.

REWARD

Balance, not at any cost

Jain’s fairness of the three PRB utilisations, minus a penalty for any throughput lost against the baseline.

$$J = (u1 + u2 + u3)^2 / (3 \cdot (u1^2 + u2^2 + u3^2))$$

```
reward = J - max(0, (T0 - T) / T0)
```

J = 1 is perfect balance; J = 1/3 means one cell carries everything. T0 is the throughput with all CIOs at 0 dB. Without CIO J = 0.630; with the example CIOs J = 0.923.

AI MODE

Learning with Bayesian optimisation

ML PART

Gaussian-process regression

Trains on the (CIO pair, reward) results of the runs so far and fits its kernel parameters to them. Predicts the reward and an uncertainty for pairs not yet simulated.

OPTIMISATION PART

Expected Improvement

Uses the prediction and the uncertainty to select the next CIO pair: a high predicted reward (exploit) or a high uncertainty (explore).

WHY NOT DEEP RL

Few, costly runs

Each 3 s NetSim run takes from about 15 s to a minute. Bayesian optimisation needs about 10 runs; deep RL would need thousands.

Output: one CIO pair for one scenario and one load, not a trained policy. A new load needs a new search. Iteration 0 is the network as deployed (all CIO = 0 dB), iterations 1–2 are exploration probes (−6 / +6 and −2 / +2 dB), and from iteration 3 Expected Improvement chooses every setting.

AI MODE · THE EQUATIONS

The model and the decision

ML · GAUSSIAN-PROCESS REGRESSION, KERNEL MATÉRN N = 2.5

$$k(x, x') = s^2 \left(1 + \sqrt{5}\, r + 5r^2/3\right) \exp(-\sqrt{5}\, r) + n^2\, [x = x']$$
$$r = \sqrt{\left(x1 - x1'\right)^2/L1^2 + \left(x2 - x2'\right)^2/L2^2}$$

$$\text{Prediction } \mu(x*) = k*^{T}\, K^{-1}\, y \qquad \text{Uncertainty } \sigma^2(x*) = k(x*, x*) - k*^{T}\, K^{-1}\, k*$$

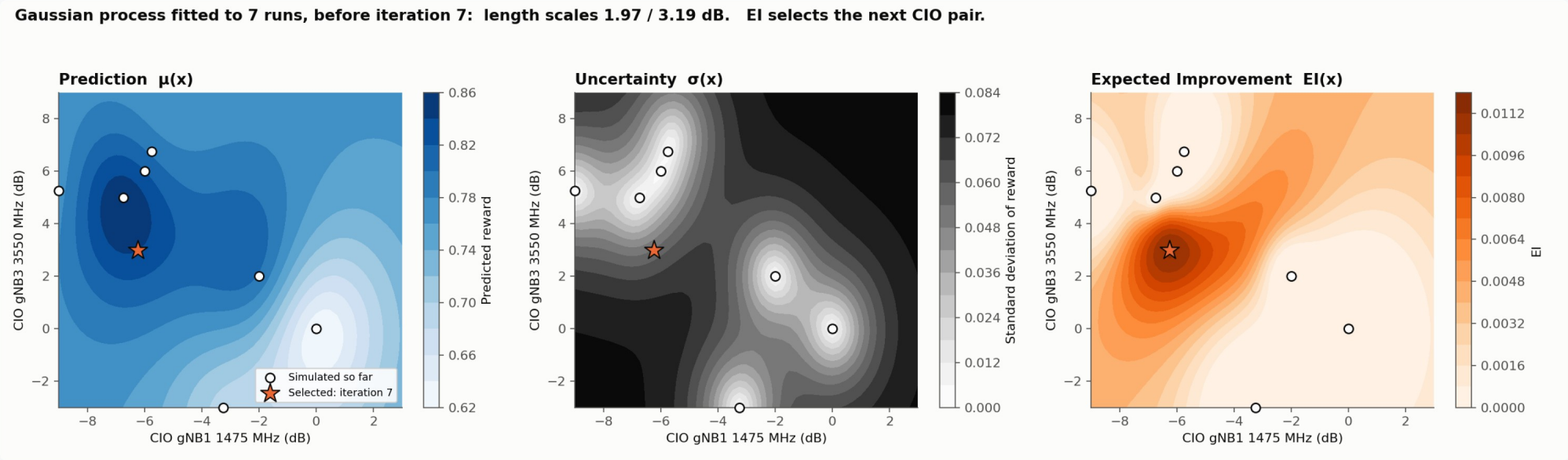
OPTIMISATION · EXPECTED IMPROVEMENT

$$\text{gain} = \mu(x) - f_best - 0.01 \qquad EI(x) = \text{gain} \cdot \Phi(\text{gain}/\sigma) + \sigma \cdot \varphi(\text{gain}/\sigma)$$

x = (CIO of gNB1, CIO of gNB3). K is the kernel matrix of the simulated pairs, y their rewards, k* the kernel between x* and each of them. s^2, L1, L2 and n^2 are fitted to the data by maximising the marginal likelihood after every run. f_best is the best reward so far; Φ and φ are the standard normal CDF and PDF. EI is evaluated on the 0.25 dB grid and its maximum is the next pair.

AI MODE · INSIDE ONE ITERATION

Explore first, then follow the model



The GP fitted to the 7 runs before iteration 7: prediction μ, uncertainty σ (small near simulated pairs) and EI, whose maximum (star) is the pair run next. Until iteration 6, one fitted length scale sits at its limit (10^{-5} or 10^{5} dB), so those iterations explore. In this run, at iteration 7 the length scales are 1.97 and 3.19 dB, inside the search range, and the model starts to guide the search. They keep changing as runs are added (5.70 and 5.45 dB at iteration 9); gp_fit.csv lists them for every step.

AI MODE

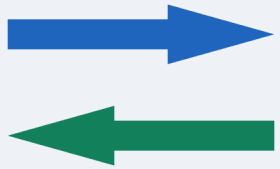
Agent–environment interaction

Python agent

`cio_agent.py`

Gaussian-process model, Expected Improvement, reward, logs and figures

Writes the CIOs into a copy of Configuration.netsim, runs NetSimCore.exe from the command line



Reads LTENR_PRB_Utilization.csv and Metrics.xml

NetSim v15.1

`NetSimCore.exe`

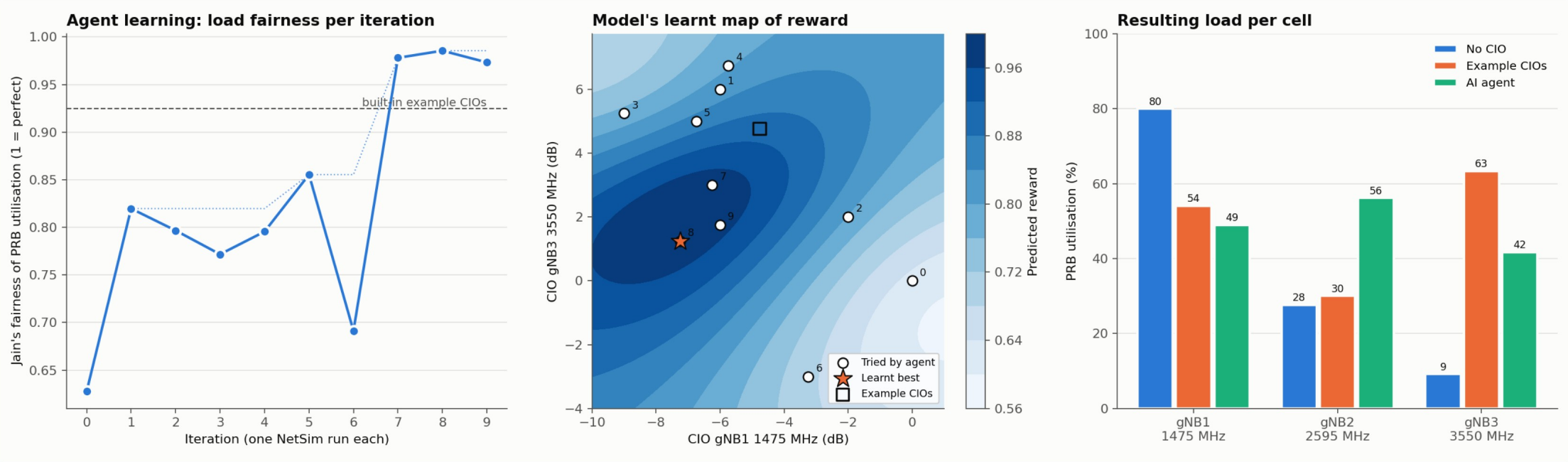
The CIO example scenario, 3 gNBs and 500 UEs, simulated for 3 s per learning run

One iteration = one NetSim run. After the last iteration the best CIOs are validated at the full 10 s and saved as the scenario AI_CIO, which opens in the NetSim GUI.

The agent uses NetSim’s command-line interface; no change to the NetSim source code is needed

RESULTS

Fairness passes the example CIOs by iteration 7



Left: fairness per iteration; the dotted line is the best so far. Middle: the model's learnt reward map; points are the settings tried, the star is the best. Right: PRB utilisation per cell. The dips at iterations 3, 4 and 6 are exploration: until iteration 6 the model's length scales sit at their limits; from iteration 7 the model guides the search.

RESULTS

No CIO vs example CIOs vs AI agent

Case	CIO gNB1 / 2 / 3 (dB)	PRB util. gNB1 / 2 / 3	Busiest cell	Fairness J	Throughput
No CIO	0 / 0 / 0	80.0 / 27.6 / 9.4 %	80.0 %	0.630	208.2 Mbps
Example CIOs	−4.77 / 0 / +4.77	53.6 / 30.0 / 64.0 %	64.0 %	0.923	229.2 Mbps
AI agent	−7.25 / 0 / +1.25	48.7 / 56.4 / 41.8 %	56.4 %	0.985	229.2 Mbps

BALANCE

0.985

Jain fairness, against 0.923 for the example CIOs

BUSIEST CELL

56%

PRB utilisation, against 64% with the example CIOs and 80% without CIO

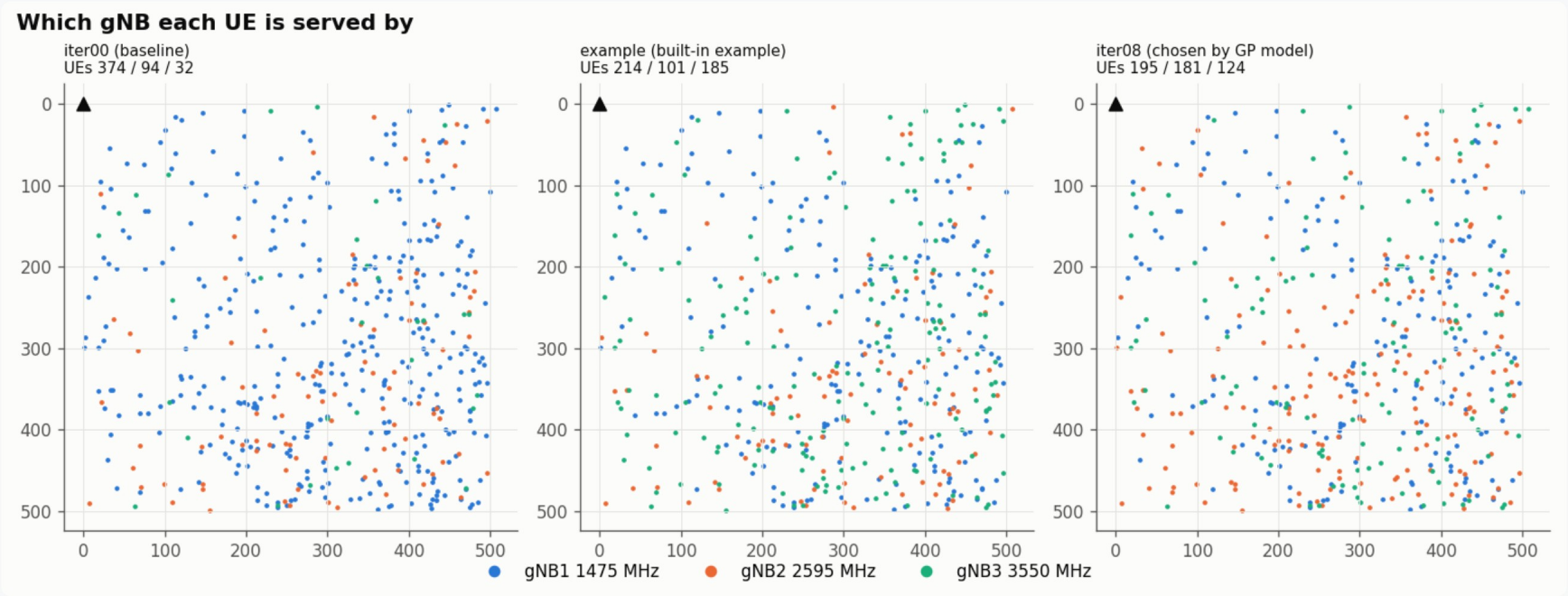
THROUGHPUT

229.2

Mbps, the same as the example CIOs and 21 Mbps above no CIO

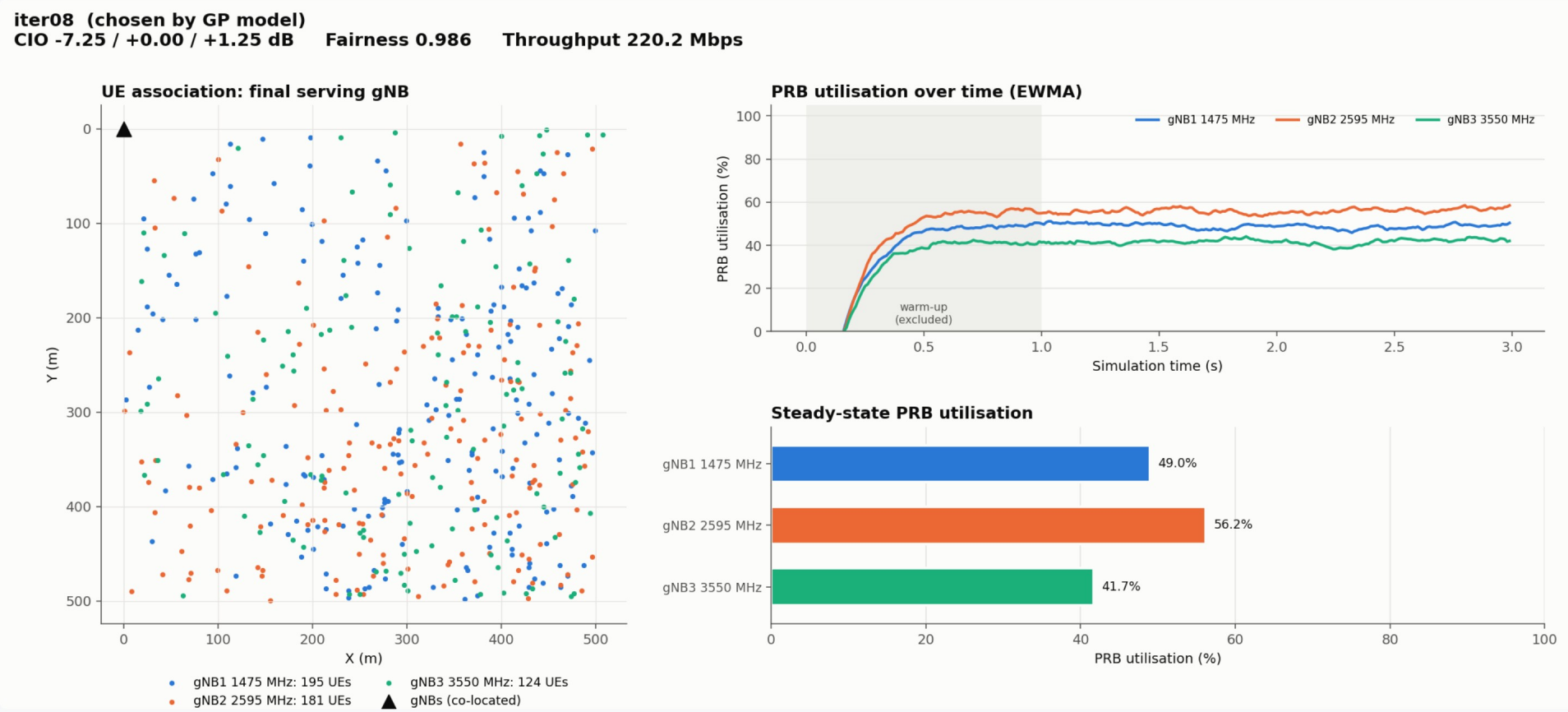
RESULTS

Which gNB serves each UE



UEs on gNB1 / 2 / 3: 374 / 94 / 32 without CIO, 214 / 101 / 185 with the example CIOs, 195 / 181 / 124 AI. The colours mix because 5 dB shadowing, not only distance, decides each UE's best cell.

The learnt setting: association and load over time



DISCUSSION

What the agent learnt

A different strategy

The example CIOs push load mainly onto gNB3 (64%). The agent offloads gNB1 to both gNB2 and gNB3, so no cell rises above 57%.

Consistent with the physics

It pushes gNB1 down by 7.25 dB, more than the 4.9 dB path-loss gap alone. The extra offsets the 3 dB handover margin that keeps UEs on gNB1.

No physics needed

The agent knows nothing about path loss, bands or the scheduler. It learns only from the PRB utilisation and throughput it measures.

Exploration is part of learning

Low points in the learning curve are deliberate: the model tests uncertain settings, and each one sharpens its map of the reward.

RESULTS · MORE TRAFFIC PER UE

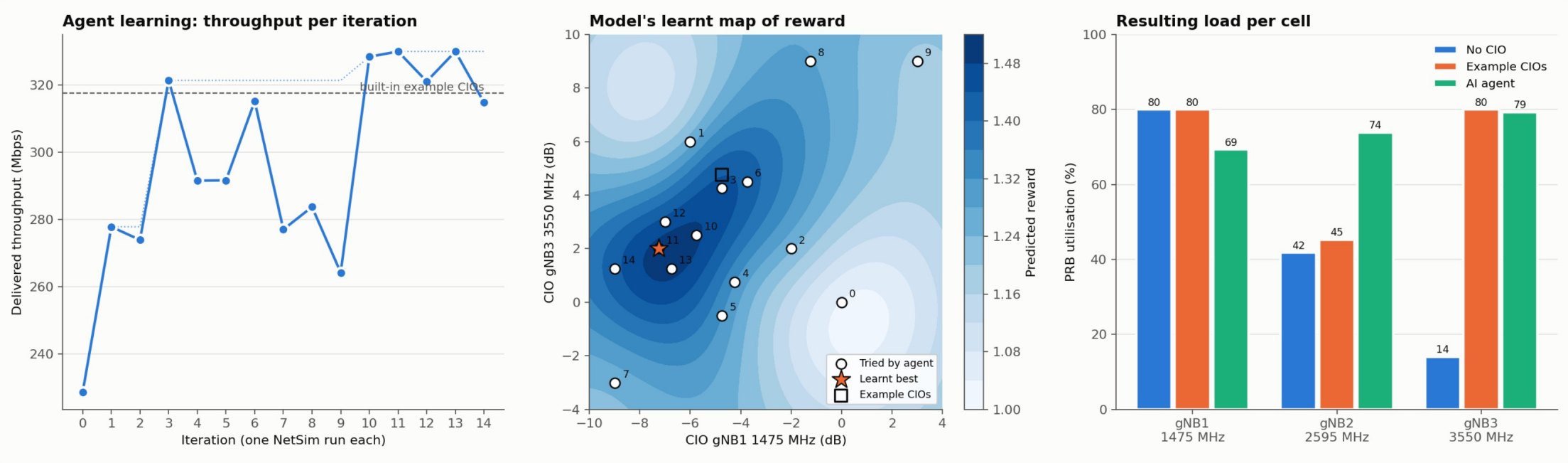
When balance turns into throughput

At the example's traffic, the example CIOs and the AI agent both deliver 98% of it, so throughput is capped by demand. Here the traffic per UE is increased 1.5× (700.8 kbps per UE, 350.4 Mbps offered); everything else is unchanged. The agent searches again for this load, maximising throughput, with the example CIOs as a starting data point.

Case	CIO gNB1 / 2 / 3 (dB)	PRB util. gNB1 / 2 / 3	Fairness J	Throughput	Delivered
No CIO	0 / 0 / 0	80.0 / 41.5 / 14.2 %	0.738	238.7 Mbps	68%
Example CIOs	−4.77 / 0 / +4.77	80.0 / 45.3 / 80.0 %	0.946	331.5 Mbps	95%
AI agent	−7.25 / 0 / +2.00	69.9 / 74.3 / 79.2 %	0.997	344.8 Mbps	98%

+13.3 Mbps (+4.0%) over the example CIOs, with no cell saturated. The example CIOs saturate gNB1 and gNB3 at 80% while gNB2 sits at 45%; the agent moves load onto gNB2.

Learning for throughput with 1.5× traffic per UE



A new search for the new load (the first model does not carry over). 15 runs of 3 s: with the example CIOs as a starting data point, iteration 3 already passes them, the model guides the search from iteration 4, and it settles on a broad plateau near −7 / +1 to +2 dB (iterations 10, 11, 13). The previous slide's table is the 10 s validation.

DISCUSSION

Limitations and extensions

LIMITATIONS

- Offline: the agent tunes between simulations, with NetSim as a digital twin, not inside a live gNB.
- One CIO pair for one scenario and one load, not a policy: the model does not carry over to a new load, so the 1.5× case needed its own search.
- Two tuned parameters on three co-located cells.
- The reward balances PRB utilisation; other goals need other rewards.

EXTENSIONS

- Online control through NetSim’s Python socket interface, changing parameters while the simulation runs.
- Load-aware models that map measured cell load to CIOs.
- Reinforcement learning for traffic that changes over time.
- More cells, or tuning handover margin and time-to-trigger together with CIO.
- The other Rel-17 use cases, such as network energy saving.

APPENDIX

How-to guide

Download and set up the project, run the agent, choose its options, read its outputs, and check the system requirements.

Download and set up the project

1. Download the project from the GitHub link on slide 1 and extract the zip file.
2. The folder has everything needed to run: Python files, results and a Scenarios folder with Without CIO (the base the agent tunes), With CIO (the built-in example's CIOs) and AI CIO (learnt), each also with 1.5× traffic per UE. No import is needed to run the agent.
3. NetSim Standard v15.1 in its default install folder is found automatically. For another location, note its bin_x64 folder, which holds NetSimCore.exe.
4. Note your licence: the licence server as 5053@<server IP address>, or the path of the licence file.
5. To view the scenarios in the NetSim GUI, import AI-ML-CIO-Load-Balancing.netsimexp into a v15.1 workspace, following the support article "Downloading and setting up NetSim File Exchange projects": [link to be added].

HOW-TO GUIDE · 2 OF 5

Set up the Python environment

Open a command prompt in the project's Python files folder and create a virtual environment:

```
cd "<project folder>\Python files"
python -m venv venv
venv\Scripts\activate
pip install -r requirements.txt
```

Python

Version 3.10 or later (tested with 3.12, 3.13 and 3.14) on the same computer as NetSim.

Packages

numpy, pandas, matplotlib, scipy and scikit-learn, installed from requirements.txt.

Afterwards

Type deactivate to leave the environment; activate it again before the next run.

Run the agent

```
python cio_agent.py --license <licence>
```

Run it from the Python files folder. It uses the bundled Scenarios\Without CIO and the default NetSim v15.1 install, and takes about 5–20 minutes depending on the computer; run.log ends with the total time. Give the licence in one of two forms:

Licence type	--license value
Licence server	5053@<licence server IP>, for example --license 5053@10.0.0.25
Licence file	"<full path of the licence file>" (in quotes if the path has spaces)

With 1.5× traffic per UE, maximising throughput (the scenario path is relative to the Python files folder):

```
python cio_agent.py --license <licence> --iopath "..\Scenarios\Without CIO - 1.5x traffic" --objective throughput --learn-from-example --iterations 15
```

Command-line options

Option (default)	Meaning
--license (required)	Licence server 5053@<IP> or the path of the licence file
--app-path (standard install)	NetSim's bin_x64 folder, if NetSim is not in C:\Program Files\NetSim\Standard_v15_1
--iopath (Scenarios\Without CIO)	The scenario to tune; it must have three gNBs
--objective (fairness)	fairness: balance the load; throughput: maximise delivered throughput
--learn-from-example	Give the example's CIOs to the model as a starting data point
--iterations (10), --seed (0)	Learning runs; the same options and seed give the same results
--association (none)	key or all: also plot UE association (adds a few minutes for key)
--run-timeout (900)	Stop if one NetSim run takes longer (s). NetSim waits indefinitely when the licence server is unreachable.

More options, such as --sim-time, --bounds, --example-cio, --no-validate and --out: python cio_agent.py --help

Read the outputs

Each run writes a folder runs\<date_time>\ next to the scripts:

File or folder	Content
run.log	The console output: each iteration’s CIOs, PRB utilisation, fairness and reward; ends with the total run time
ai_log.csv	One row per NetSim run (baseline, example CIOs, iterations)
compare.csv	No CIO, example CIOs and AI at the full 10 s
config.json	The settings of the run (no licence details are stored)
figures\ai_convergence.png	Learning curve, learnt reward map, load per cell
figures\gp_step.png, gp_fit.csv	GP prediction, uncertainty and EI at one iteration; fitted kernel parameters at every step
figures\prb\, figures\combined\	PRB utilisation per run; association and PRB per case (with --association)
AI_CIO\	The learnt scenario at 10 s, to open in the NetSim GUI
iterations\, validation\	The NetSim run folders, about 180 MB in total; safe to delete

HOW-TO GUIDE

System requirements and execution guidelines

SYSTEM

- 16 GB RAM, Intel i5 / AMD Ryzen 5 or above
- Windows 10 or Windows 11, 64-bit
- NetSim Standard v15.1 with a valid licence
- Python 3.10 or later (tested with 3.12, 3.13 and 3.14)
- About 200 MB of free disk per run; with --association, up to 2 GB more while each extra run is processed
- Run time depends mainly on the processor: a 3 s learning run took 16–20 s on an Intel i5-13400 and about 55 s on an i5-4460

GUIDELINES

- Run from a command prompt with the virtual environment active.
- If the licence server is unreachable, NetSim waits; the agent stops that run after --run-timeout (900 s) with a licence message. Other failed runs are retried up to three times.
- The same options and --seed give the same learning path and results.
- To try another load or scenario, export it from the GUI and pass its folder as --iopath; it must have three gNBs.