Lessons Learnt from Developing & Maintaining the World’s Largest* CP Model Using MiniZinc

Presented by Erik Cervin-Edin

Content

- Quick background on Ericsson and RAN networks
- Using combinatorial optimization in product configuration
- Developing, executing & maintaining very large CP models

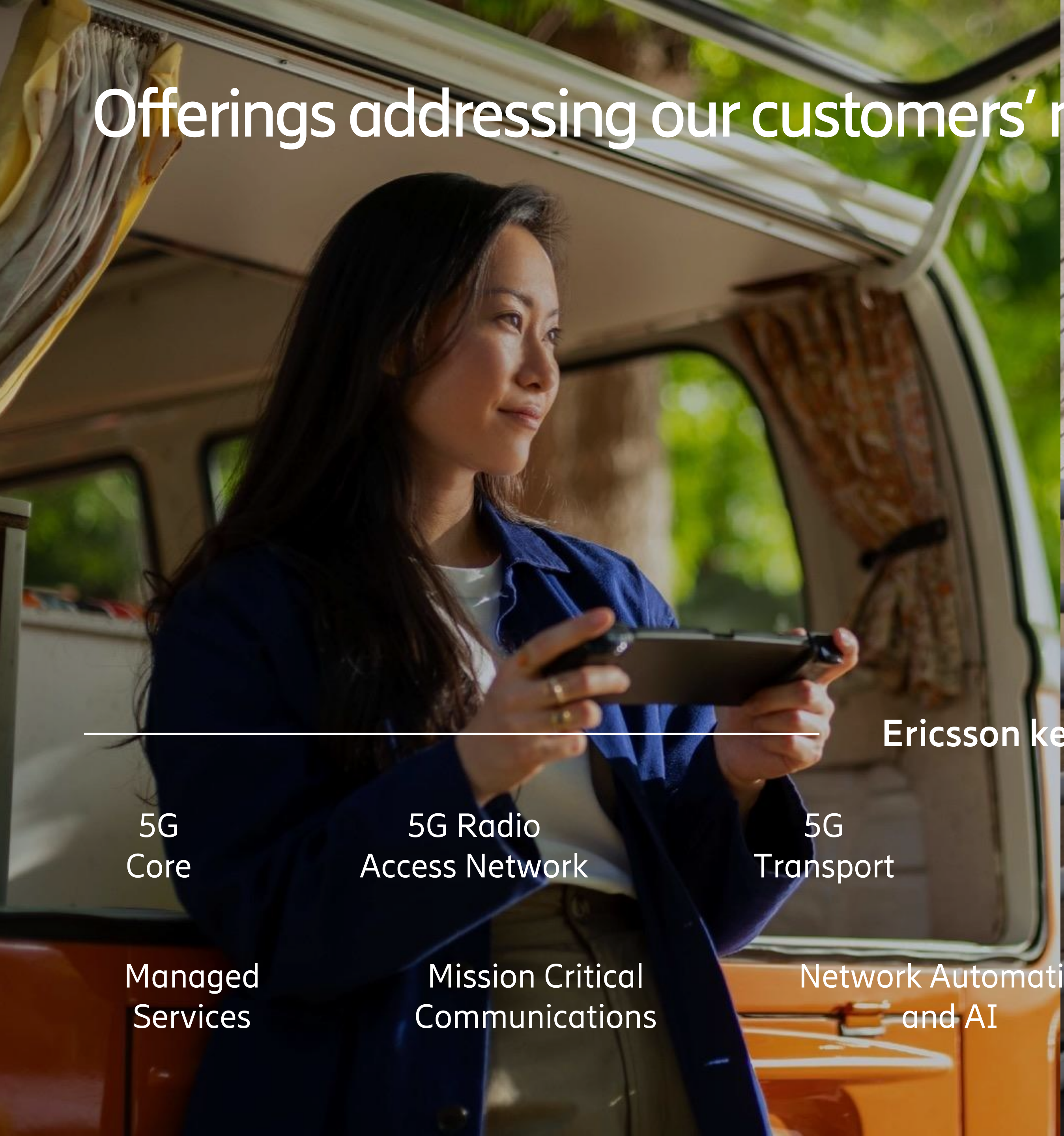
Erik @ Ericsson

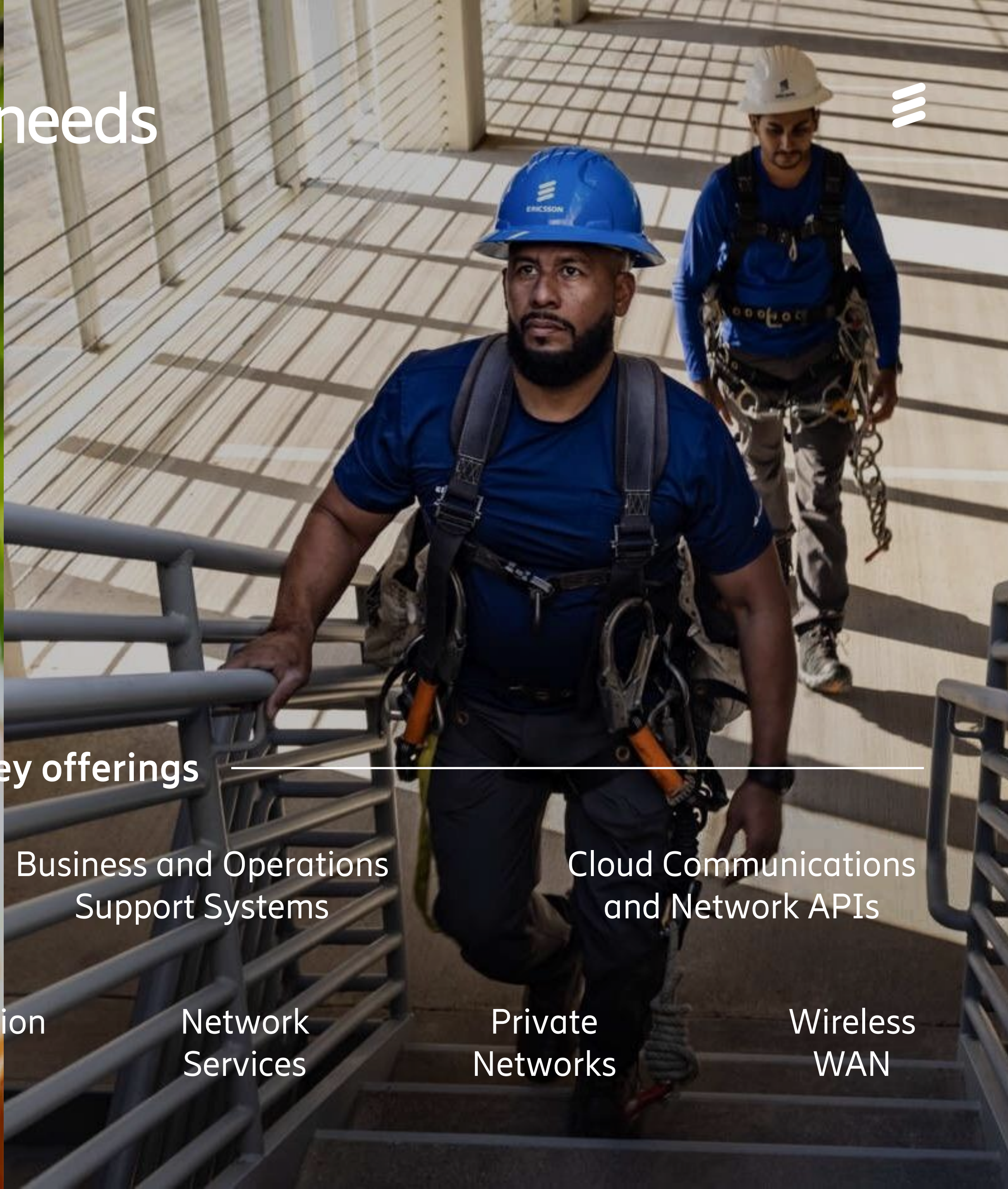
- Erik Cervin Edin
- Software Developer @ Ericsson
- Computer Science, Uppsala University
- Ericsson since Feb 2023

https://github.com/CervEdin

https://linkedin.com/in/erikcervinedin

Offerings addressing our customers' needs

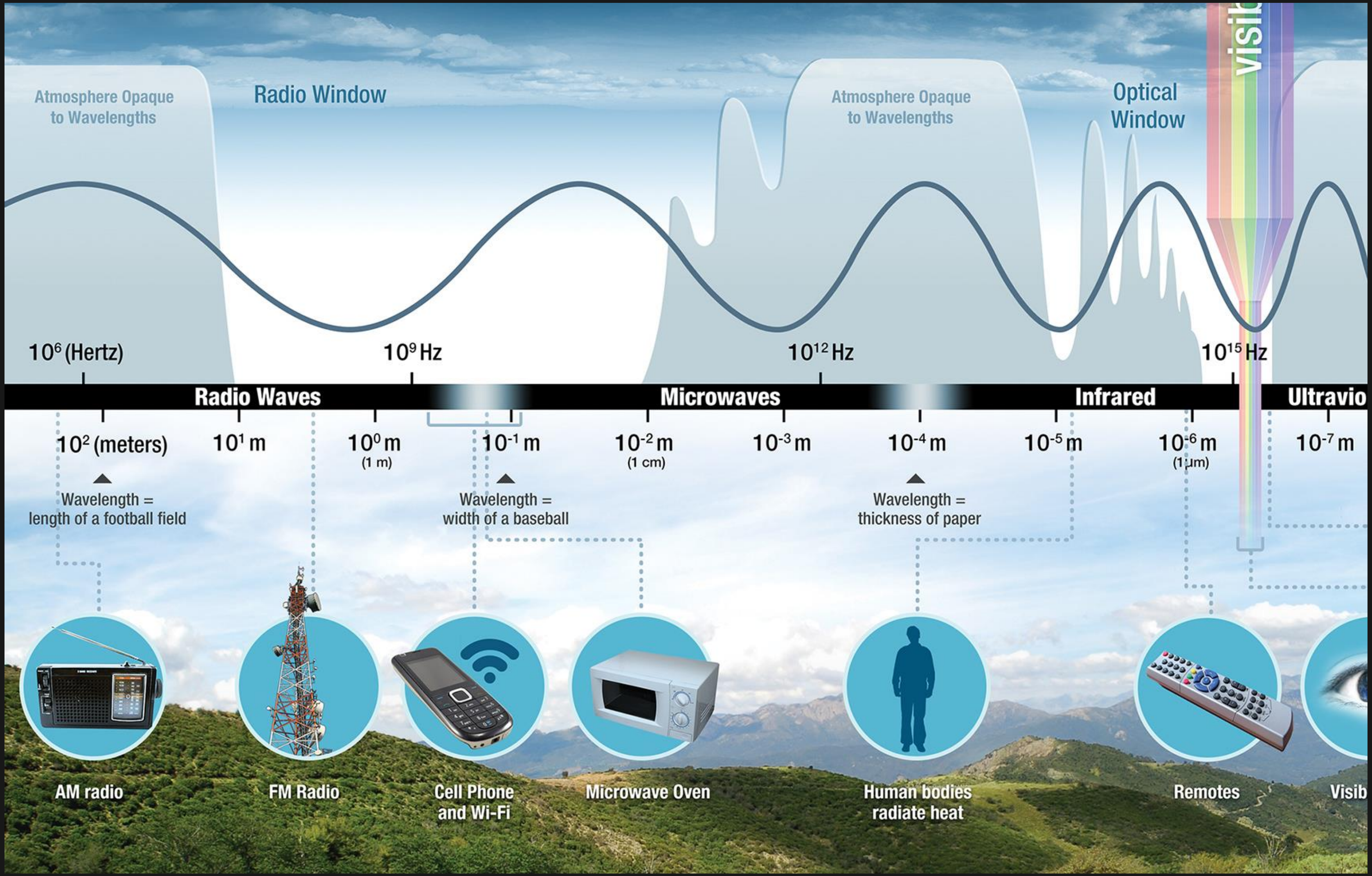




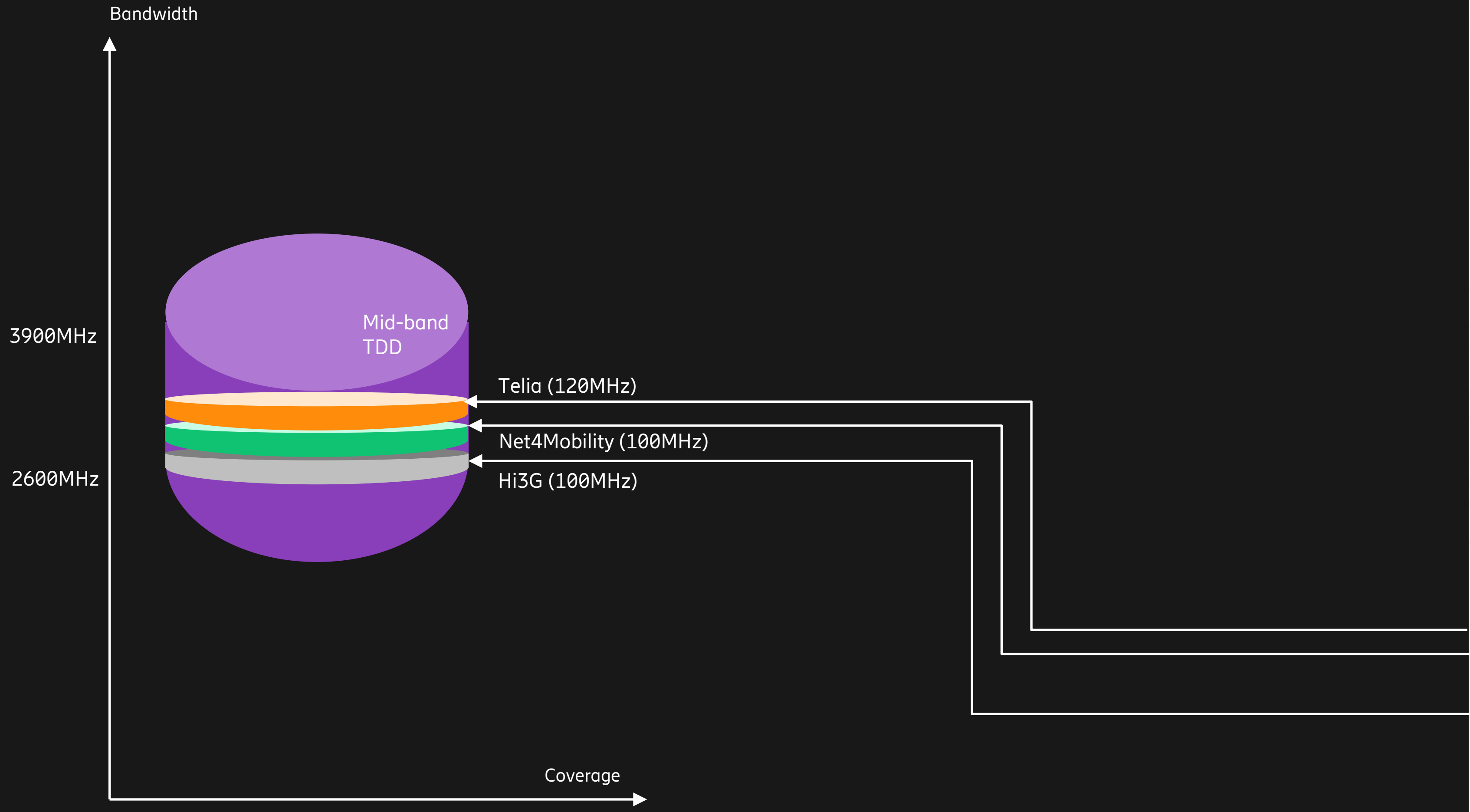
Ericsson key offerings

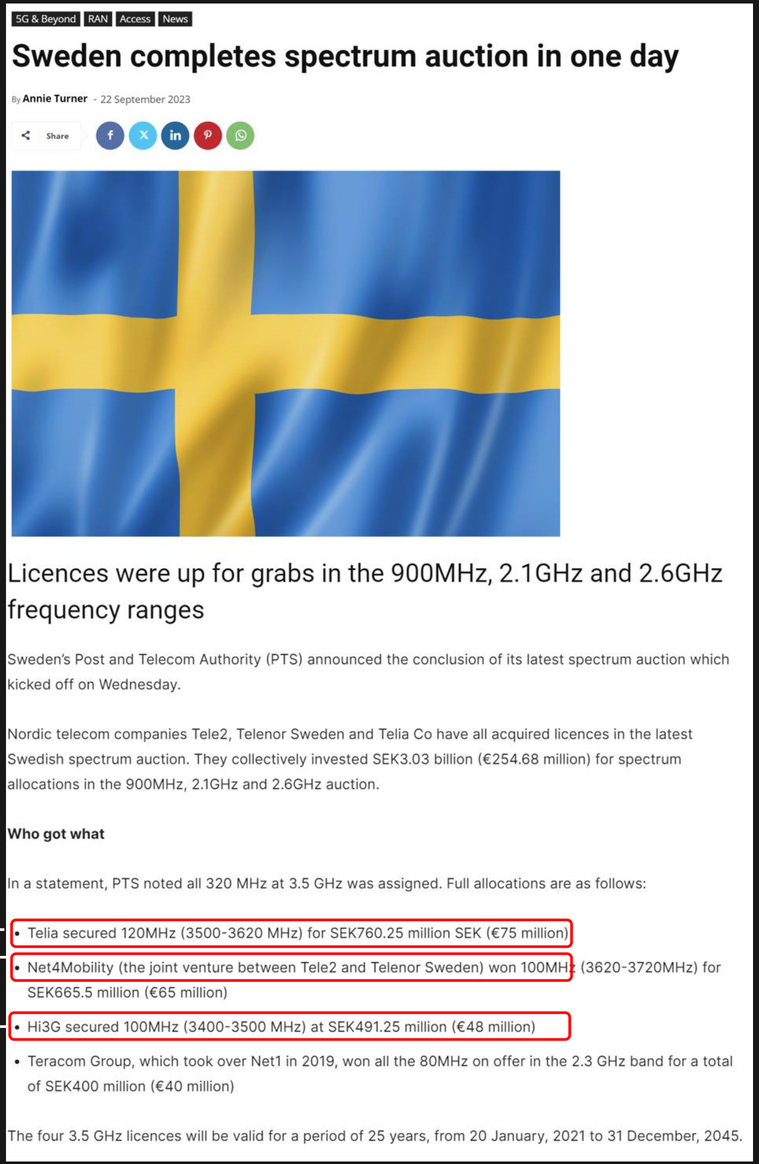
- 5G Core
- 5G Radio Access Network
- 5G Transport
- Business and Operations Support Systems
- Cloud Communications and Network APIs
- Managed Services
- Mission Critical Communications
- Network Automation and AI
- Network Services
- Private Networks
- Wireless WAN

Electromagnetic spectrum



Spectrum allocation





Sweden completes spectrum auction in one day

By Annie Turner - 22 September 2023

Licences were up for grabs in the 900MHz, 2.1GHz and 2.6GHz frequency ranges

Sweden's Post and Telecom Authority (PTS) announced the conclusion of its latest spectrum auction which kicked off on Wednesday.

Nordic telecom companies Tele2, Telenor Sweden and Telia Co have all acquired licences in the latest Swedish spectrum auction. They collectively invested SEK3.03 billion (€254.68 million) for spectrum allocations in the 900MHz, 2.1GHz and 2.6GHz auction.

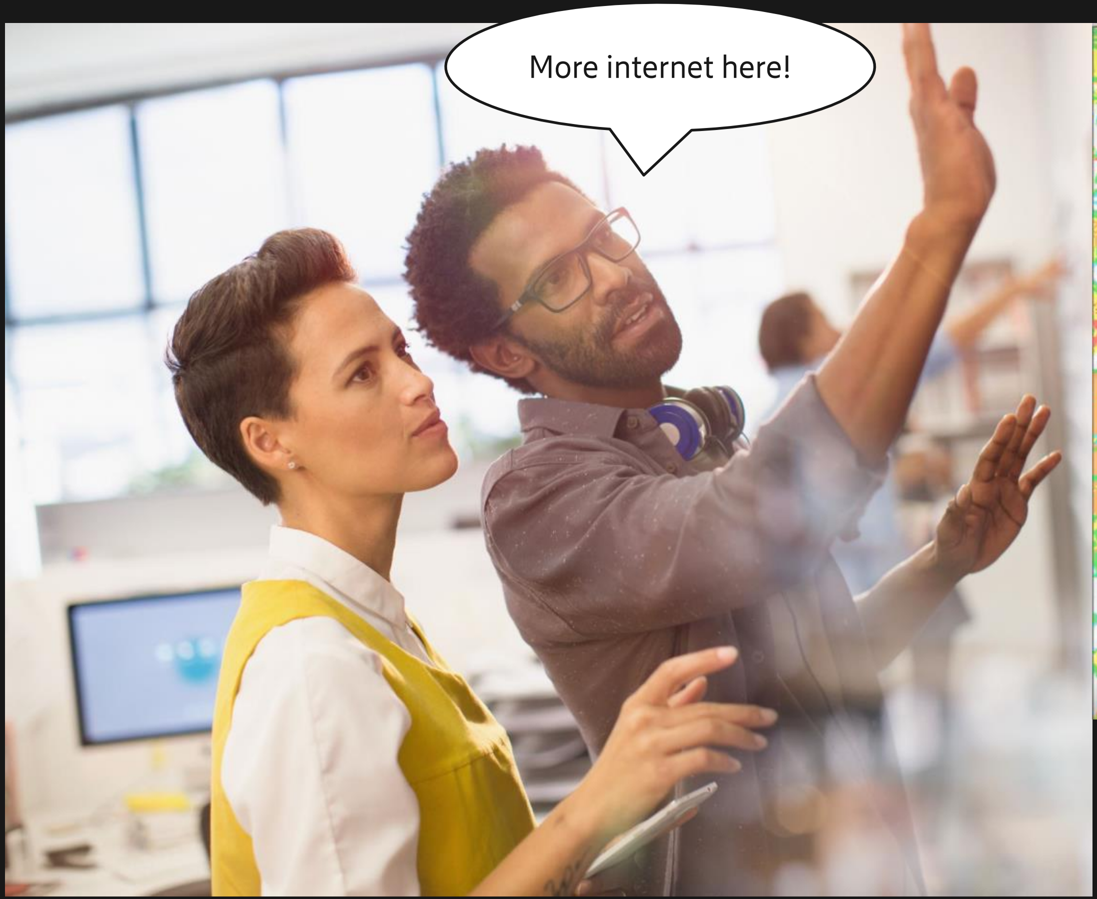
Who got what

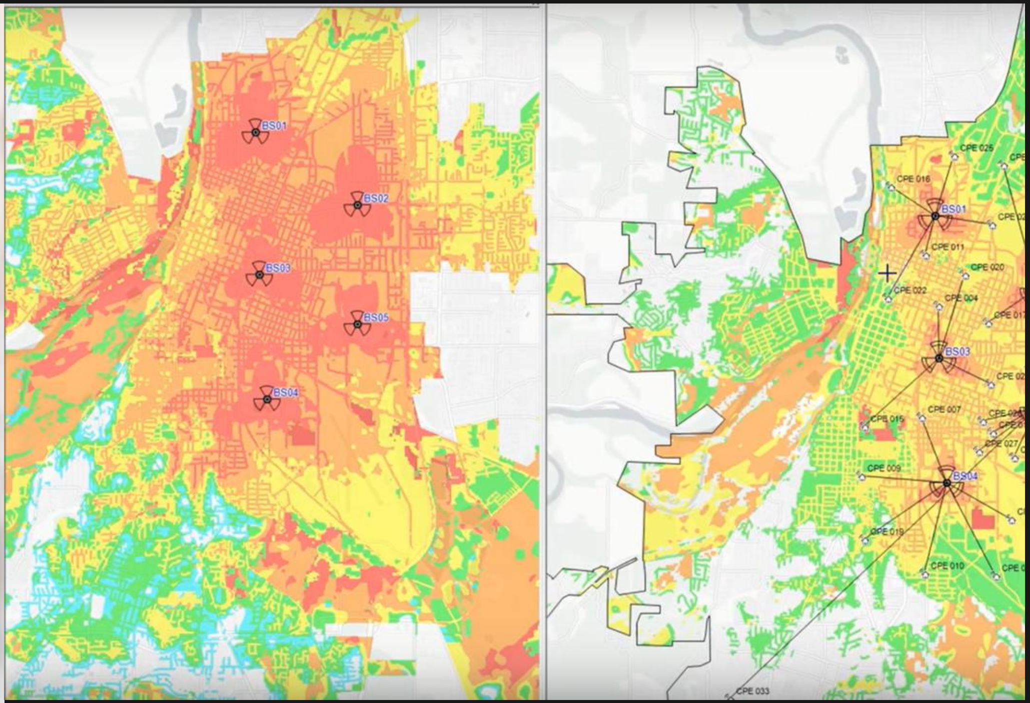
In a statement, PTS noted all 320 MHz at 3.5 GHz was assigned. Full allocations are as follows:

- Telia secured 120MHz (3500-3620 MHz) for SEK760.25 million SEK (€75 million)
- Net4Mobility (the joint venture between Tele2 and Telenor Sweden) won 100MHz (3620-3720MHz) for SEK665.5 million (€65 million)
- Hi3G secured 100MHz (3400-3500 MHz) at SEK491.25 million (€48 million)
- Teracom Group, which took over Net1 in 2019, won all the 80MHz on offer in the 2.3 GHz band for a total of SEK400 million (€40 million)

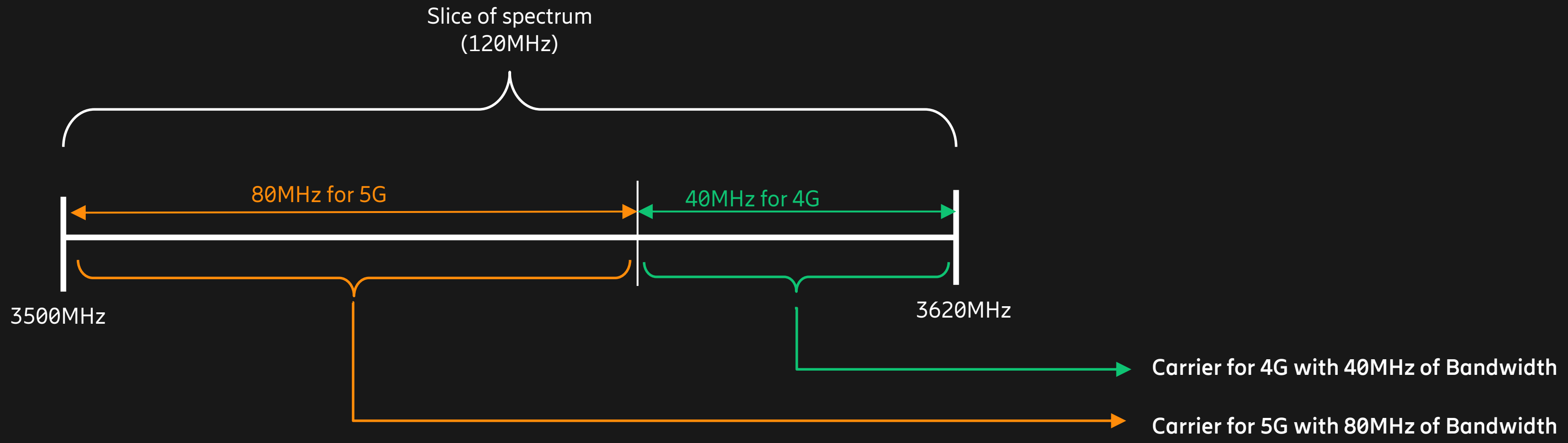
The four 3.5 GHz licences will be valid for a period of 25 years, from 20 January, 2021 to 31 December, 2045.

The network plan





Output of Planning Activity



Slice of spectrum
(120MHz)

80MHz for 5G

40MHz for 4G

3500MHz

3620MHz

Carrier for 4G with 40MHz of Bandwidth

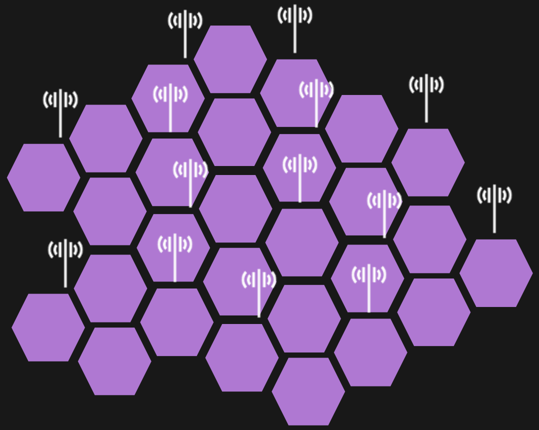
Carrier for 5G with 80MHz of Bandwidth

Option-1
- Full 120MHz for LTE (4G)

Option-2
- Full 120MHz for NR (5G)

Option-3
- Mix of 4G & 5G
  - 60MHz for each
  - 40MHz for LTE & 80MHz for NR
  - And so on.....

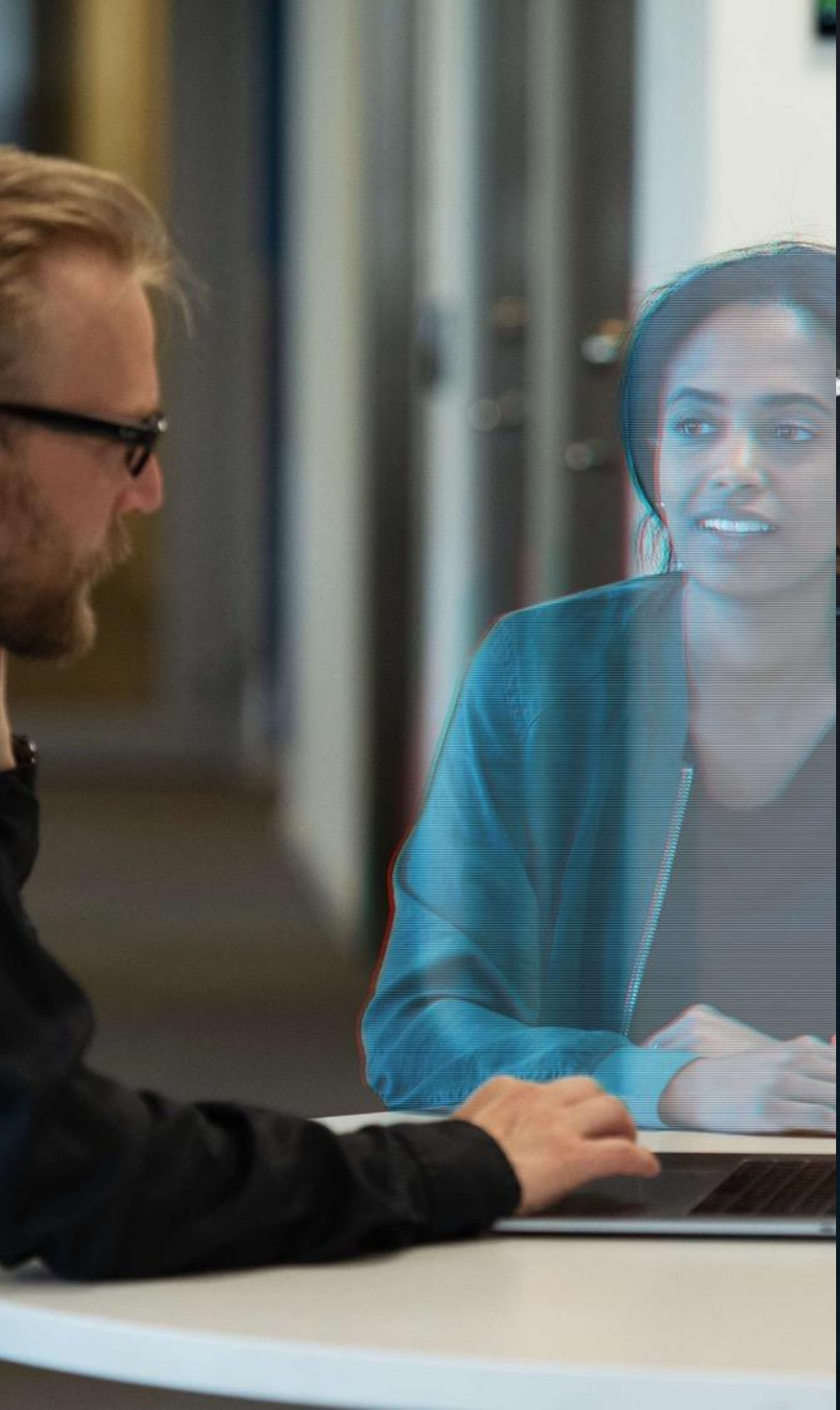
Number of Sites

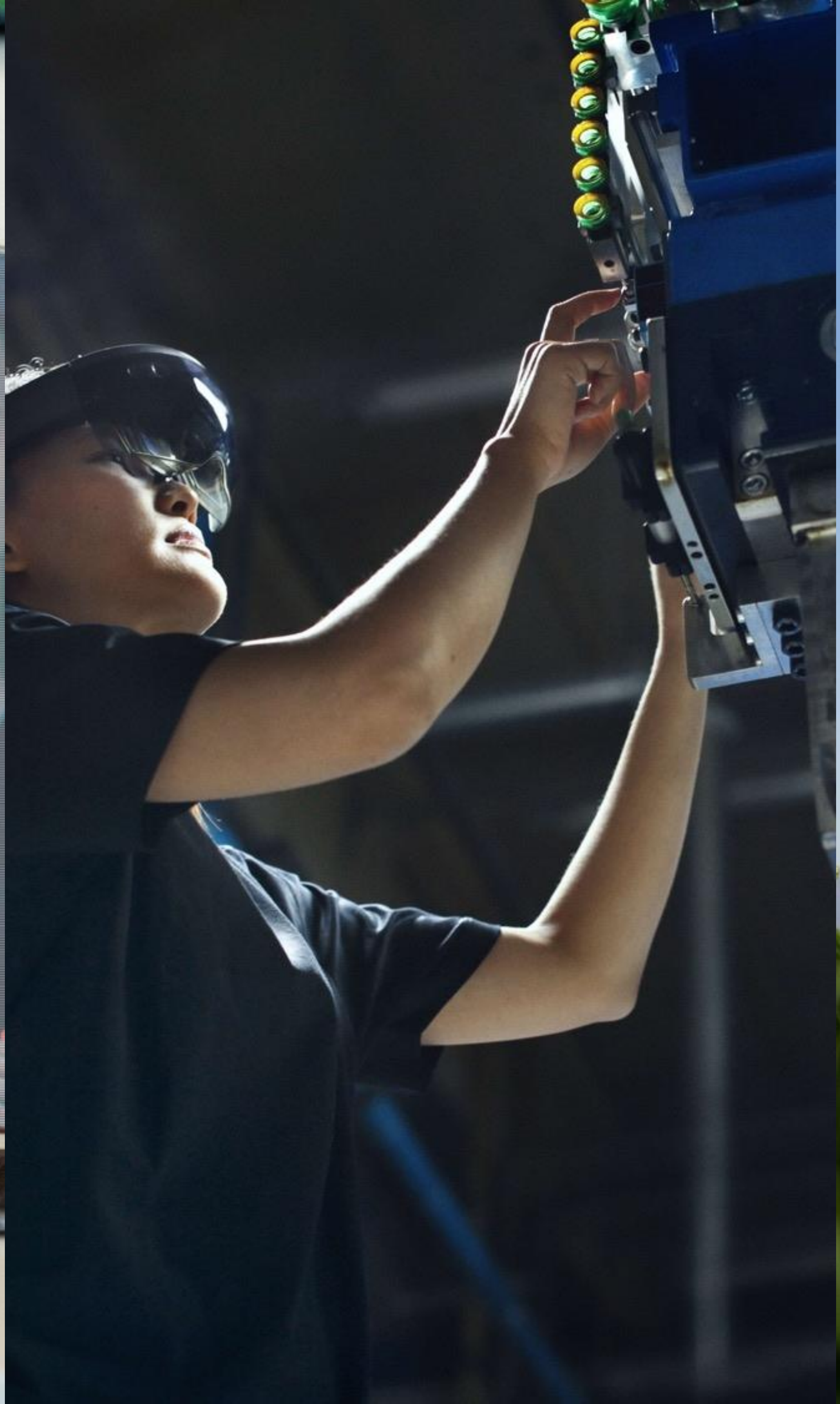


Radio ?? (3500-3700MHz)

Radio ?? (3410-3800MHz)

Radio ?? (3450-3800MHz)

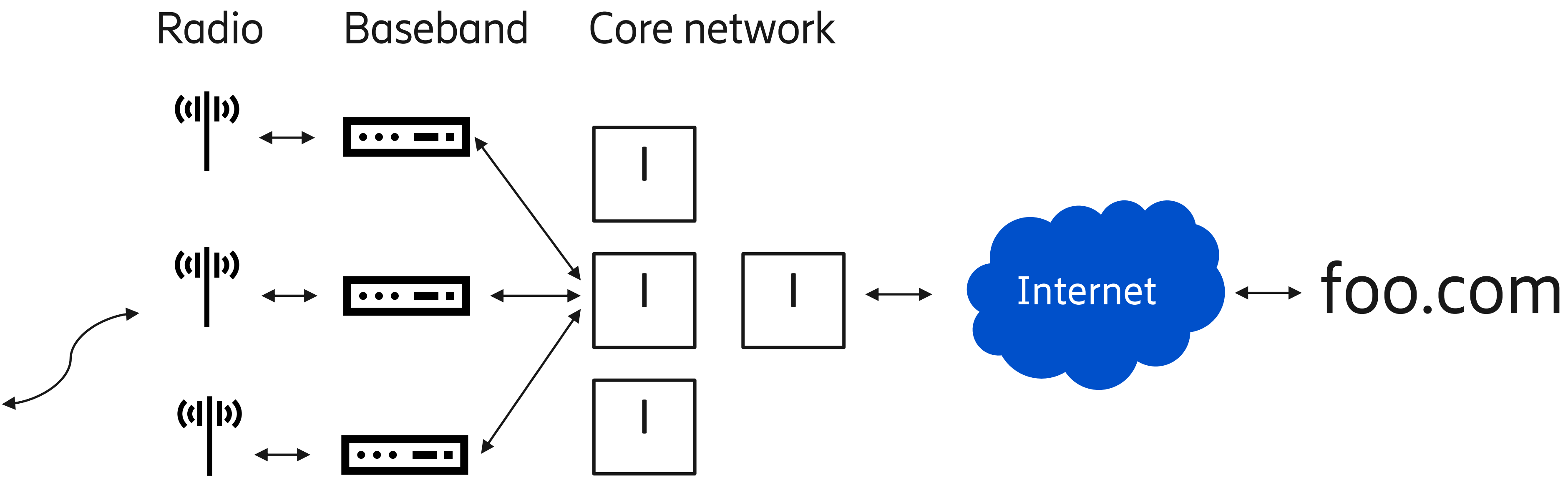




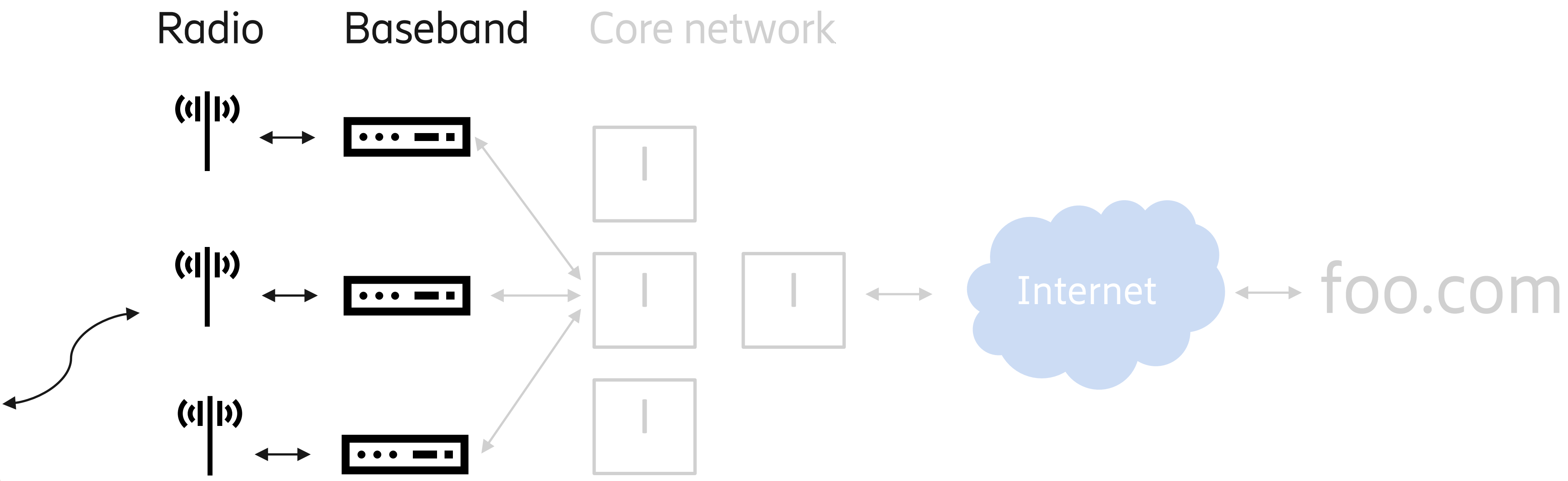
Product Configuration @ Ericsson

- Aid sales & support
- Configuration engines customize products to meet needs – like buying a couch

Radio Access Network (RAN) Overview

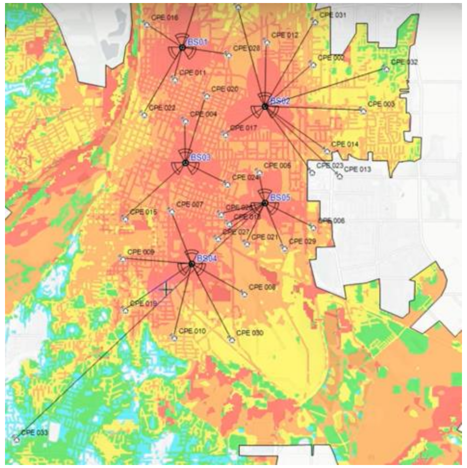


Radio Access Network (RAN) Overview



Product configuration – The requirements

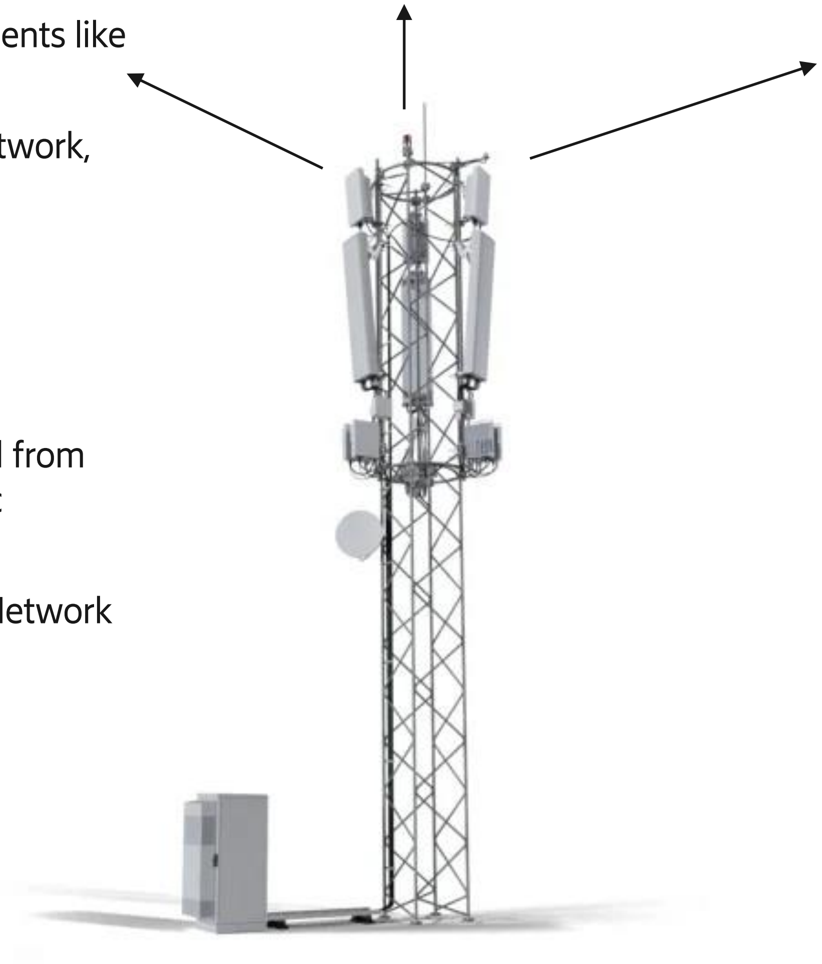
- Site 1
  - GSM carrier
  - 3G carrier
  - 2 x 5G carrier
- Site 2
  - 2 x 4G carrier
  - 5G carrier
- Site 3
  - GSM carrier
  - LTE carrier
- …



Product configuration – The site

1+ radio solutions

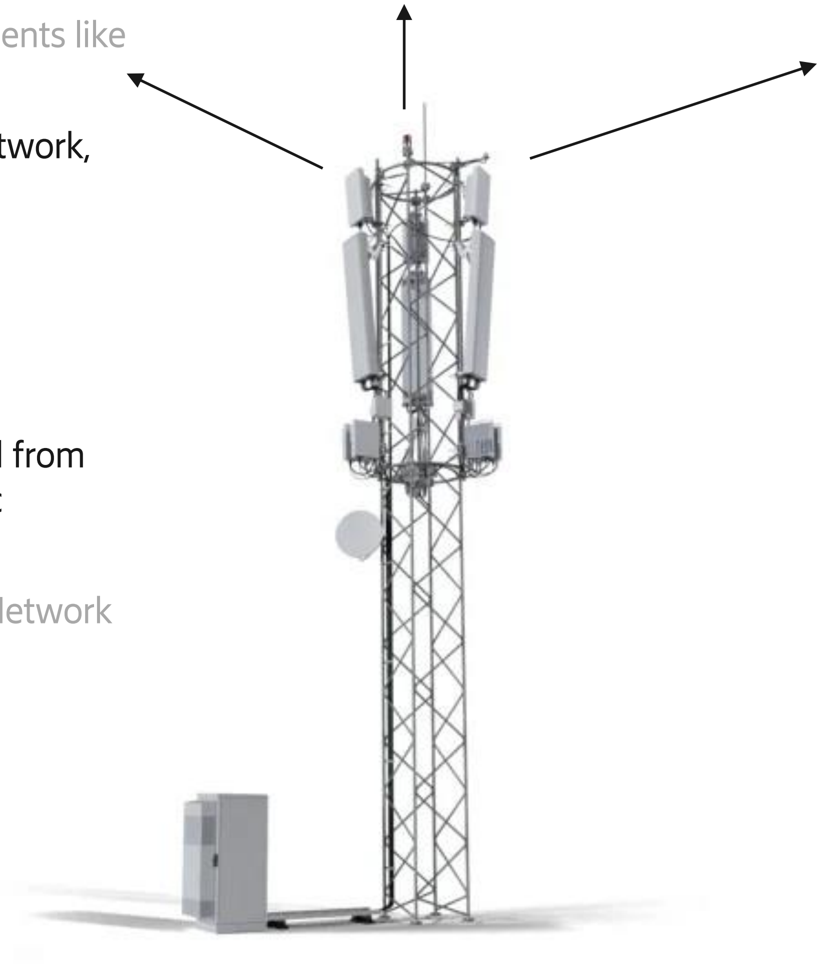
- **Antenna System**: responsible for transmitting and receiving radio signals. It includes components like antennas, cables, and connectors.
- **Carrier:** range of frequencies allocated for transmitting and receiving signals on a wireless network, typically defined by its center frequency and bandwidth. (ephemeral)
- **RF Port:** interface that connects radio to antennas, split RX⬇/TX⬆ traffic.
- **Radio:** wireless communication component that transmits and receives radio signals.
- **CPRI Ports:** interface that connects radio to baseband, ⬇⬆ traffic.
- **Baseband:** network that handles the lower frequency signals, after they have been converted from radio frequencies (RF) by an antenna and receiver. It performs tasks such as switching, traffic management, timing, baseband processing, and radio interfacing.
- **RAN Compute**: This refers to the computing resources required to support the Radio Access Network (RAN), such as baseband units or virtualized RAN functions.
- **Enclosure**: physical housing that protects the radio equipment
- **Power Supply**: provides the electrical power to operate the radio solution
- …



Product configuration – The site

1+ radio solutions

- **Antenna System**: responsible for transmitting and receiving radio signals. It includes components like antennas, cables, and connectors.
- **Carrier:** range of frequencies allocated for transmitting and receiving signals on a wireless network, typically defined by its center frequency and bandwidth. (ephemeral)
- **RF Port:** interface that connects radio to antennas, split RX⬇/TX⬆ traffic.
- **Radio:** wireless communication component that transmits and receives radio signals.
- **CPRI Ports:** interface that connects radio to baseband, ⬇⬆ traffic.
- **Baseband:** network that handles the lower frequency signals, after they have been converted from radio frequencies (RF) by an antenna and receiver. It performs tasks such as switching, traffic management, timing, baseband processing, and radio interfacing.
- **RAN Compute**: This refers to the computing resources required to support the Radio Access Network (RAN), such as baseband units or virtualized RAN functions.
- **Enclosure**: physical housing that protects the radio equipment
- **Power Supply**: provides the electrical power to operate the radio solution
- …



Product configuration – Which is best?

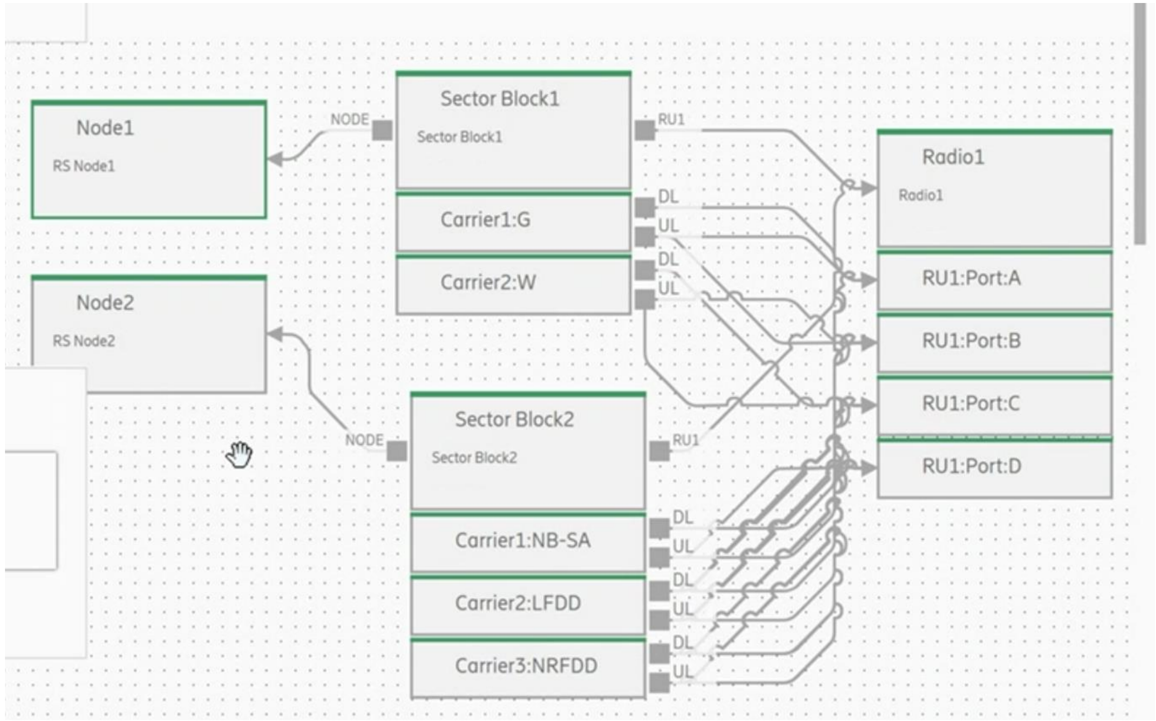
Lexicographic Optimization

- Minimize radio equipment
  - number of radios
  - …
- Optimize
  - Output power
  - Weight
  - Size
  - Other customer desires?
- The objective function is subjective!

Product configuration – Radio solution

Resource allocation - a combinatorial optimization problem

- Radio Solution:
  - Carriers ⇆ Radios ↔ Basebands
- Challenge:
  - CSP: Allocating components
  - COP: Minimize waste
- Solution:
  - Bin-packing (e.g. connecting cables)
  - ~50 table constraints
    - channeling/side-constraints (e.g. HW specific capabilities)
  - Linear constraints (capacities)

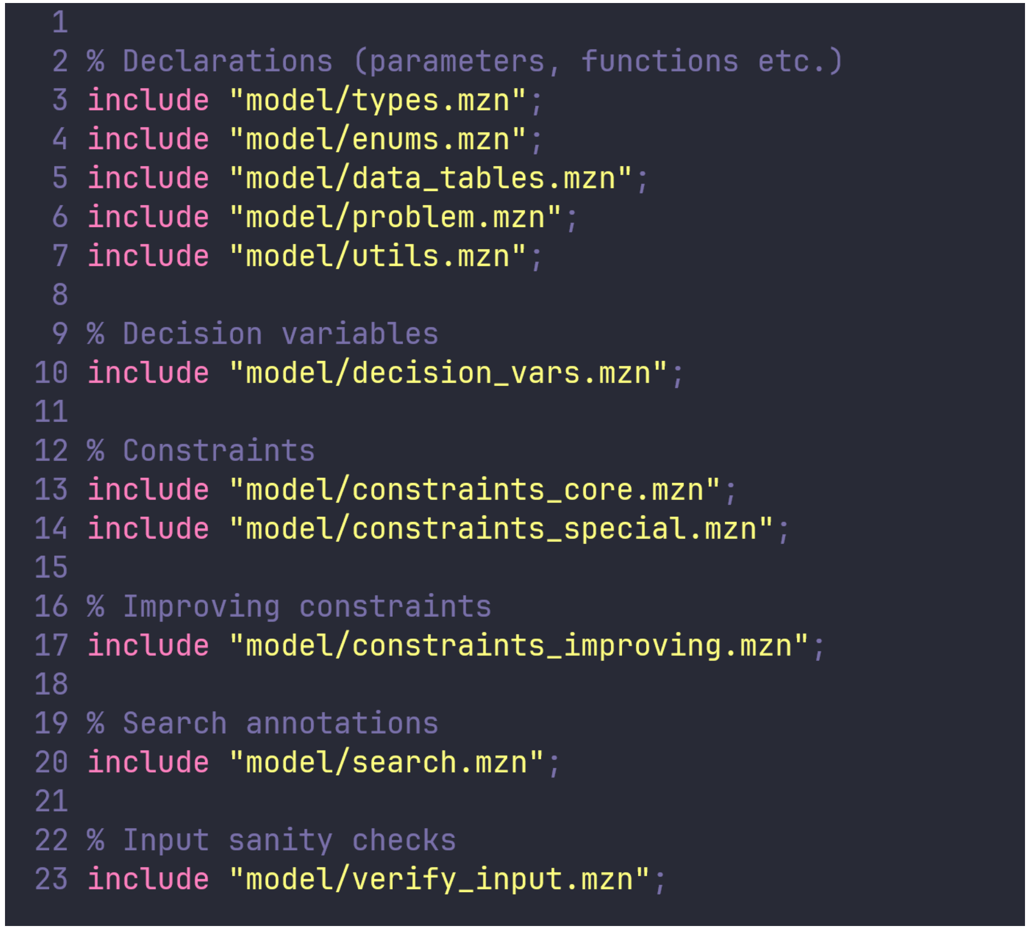


Given a set C of carriers, select a set R of radio units and map every carrier $c \in C$ to a radio $r \in R$ such that r meets all demands of c mapped to r and the capacities of r are not exceeded.

Structuring a large CP model

Modularizing a MiniZinc model

The core

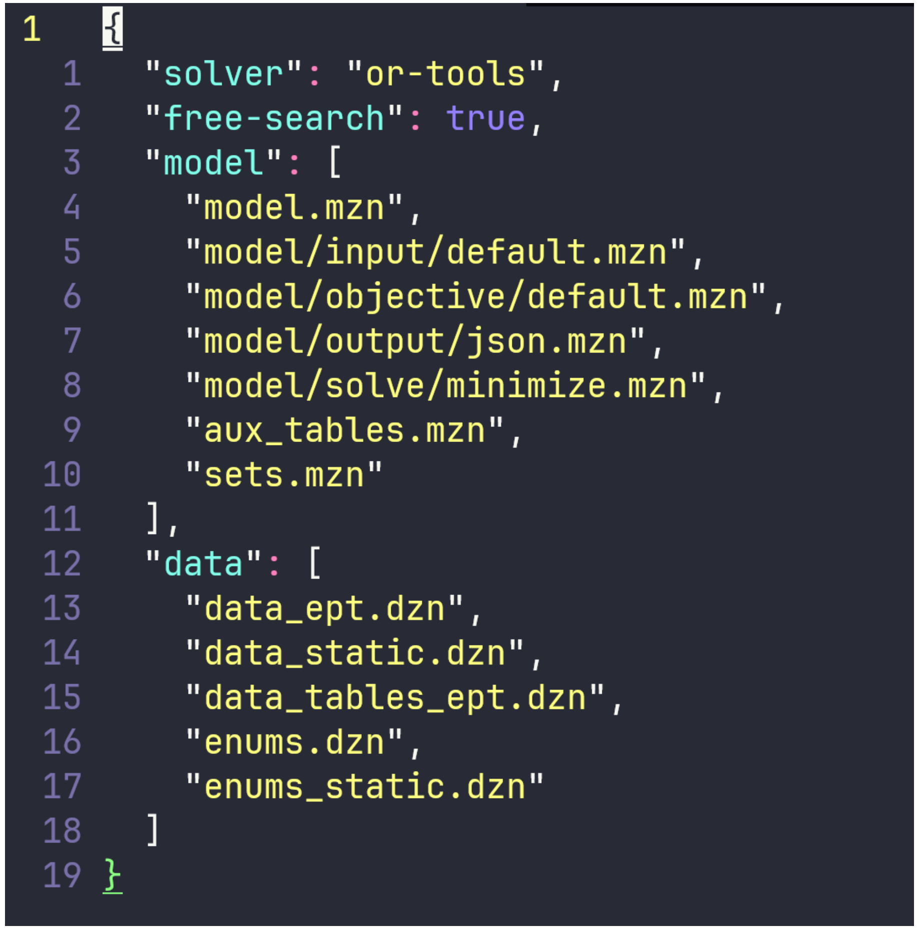


```

% Declarations (parameters, functions etc.)
include "model/types.mzn";
include "model/enums.mzn";
include "model/data_tables.mzn";
include "model/problem.mzn";
include "model/utils.mzn";

% Decision variables
include "model/decision_vars.mzn";

% Constraints
include "model/constraints_core.mzn";
include "model/constraints_special.mzn";

% Improving constraints
include "model/constraints_improving.mzn";

% Search annotations
include "model/search.mzn";

% Input sanity checks
include "model/verify_input.mzn";
```

- Declarations
  - Variables
    - Constraints
      - Improving constraints
      - Search annotations
      - Sanity checks (assertions)
- Easier to debug!
- Easier to maintain!

Modular configurations

The default configuration



```
{
  "solver": "or-tools",
  "free-search": true,
  "model": [
    "model.mzn",
    "model/input/default.mzn",
    "model/objective/default.mzn",
    "model/output/json.mzn",
    "model/solve/minimize.mzn",
    "aux_tables.mzn",
    "sets.mzn"
  ],
  "data": [
    "data_ept.dzn",
    "data_static.dzn",
    "data_tables_ept.dzn",
    "enums.dzn",
    "enums_static.dzn"
  ]
}
```

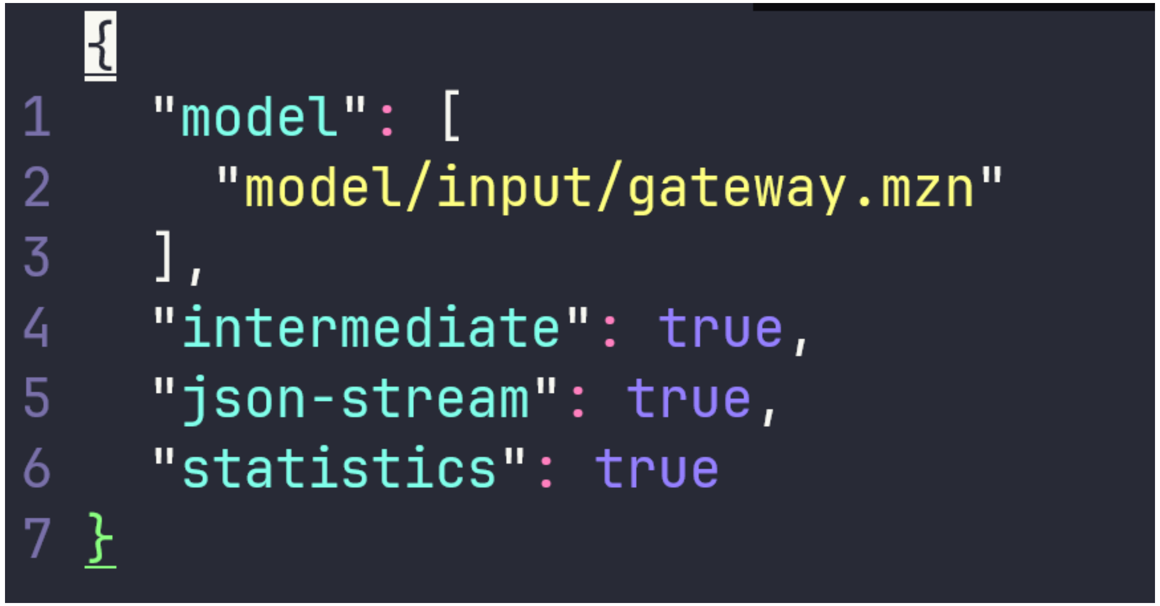
- All files included in model.mzn
- Modularized input
  - multiple input formats possible!
- Modularized objective
- Modularized output

```
minizinc default.mpc instance.dzn
```

- Other configurations can derive from this!

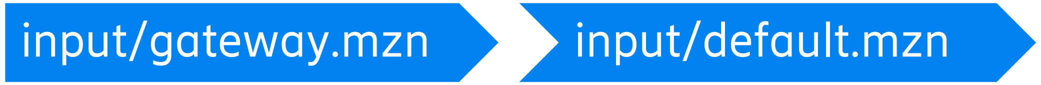
Using sub-configurations

Dedicated configuration for EC2 service



```
{
  "model": [
    "model/input/gateway.mzn"
  ],
  "intermediate": true,
  "json-stream": true,
  "statistics": true
}
```

- The *gateway* input file is a “*function*” to the *default* input file



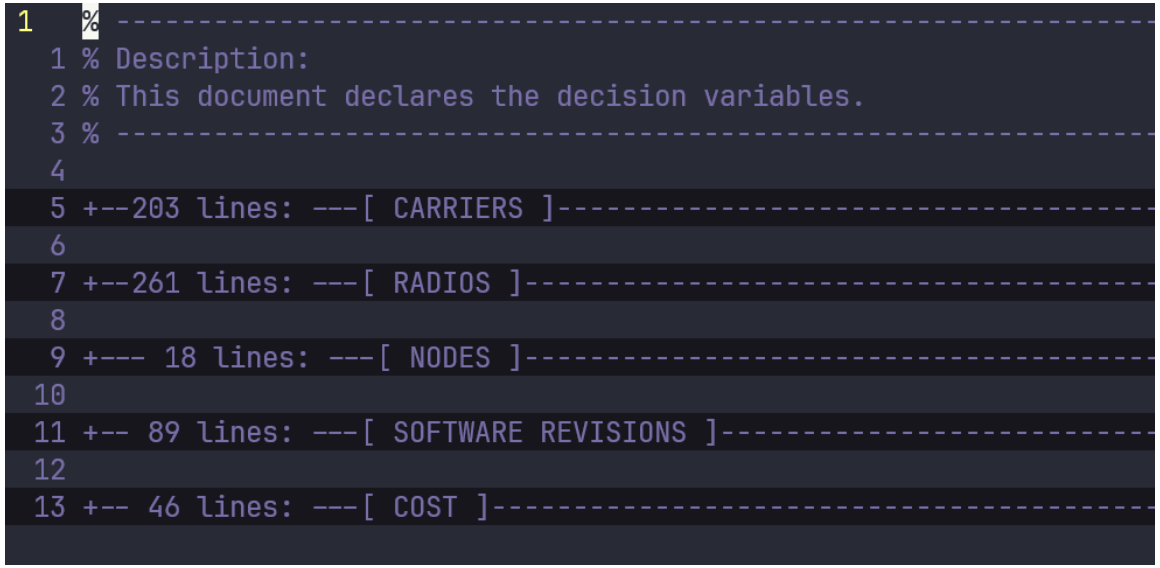
input/gateway.mzn

input/default.mzn

- Can activate additional flags (Json output)
- Allows non-breaking input updates!

```
minizinc default.mpc gateway.mpc instance.dzn
```

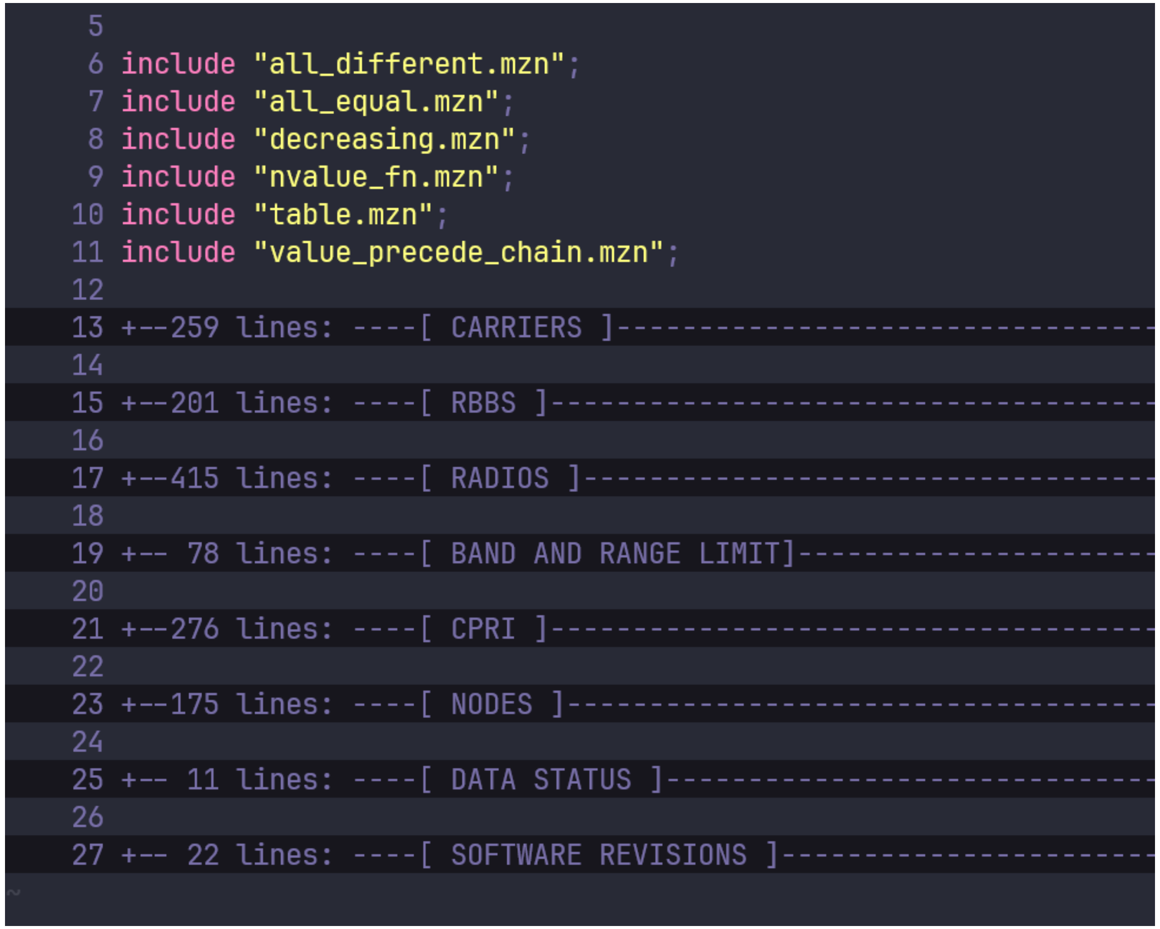
Decision Variables



```
% ------------------------------------------------------------
% Description:
% This document declares the decision variables.
% ------------------------------------------------------------

+--203 lines: ---[ CARRIERS ]-------------------------------------

+--261 lines: ---[ RADIOS ]---------------------------------------

+--- 18 lines: ---[ NODES ]---------------------------------------

+-- 89 lines: ---[ SOFTWARE REVISIONS ]---------------------------

+-- 46 lines: ---[ COST ]-----------------------------------------
```

- 109 decision variable declarations
- Categorized by RAN component

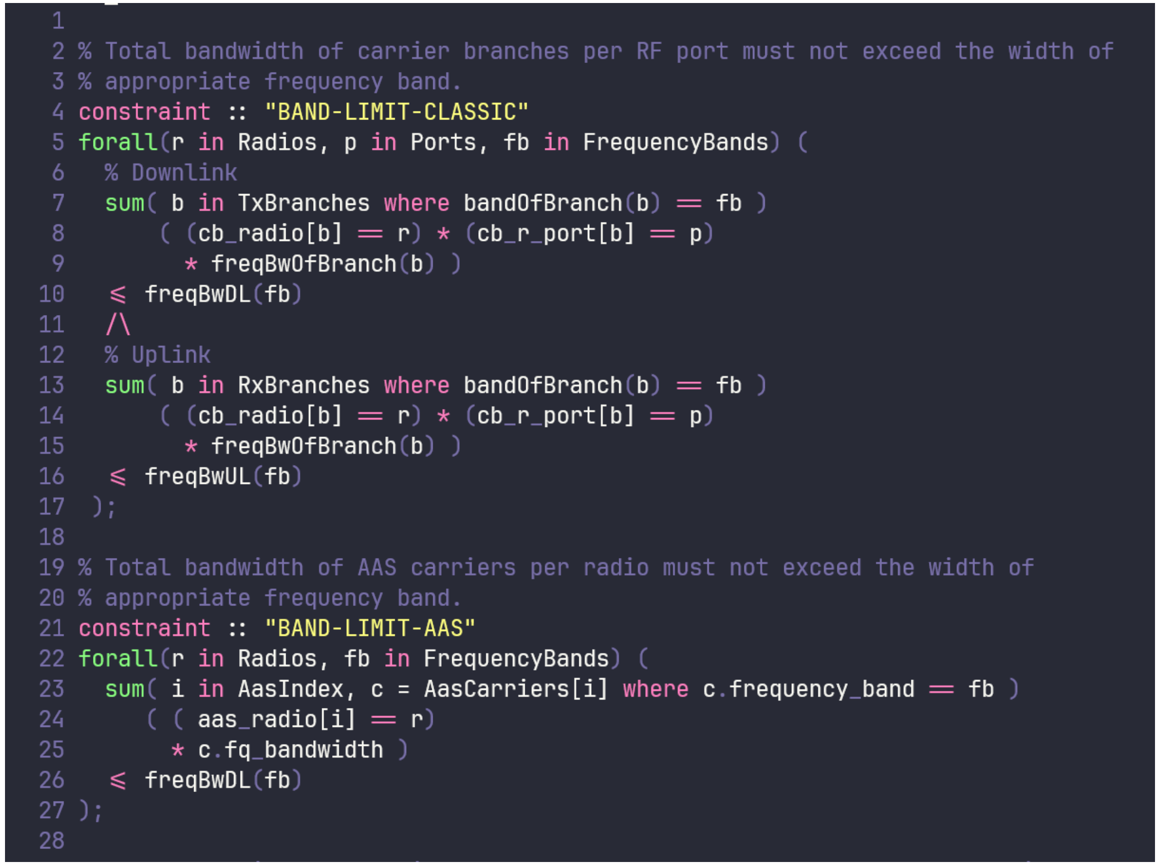
Constraints — Core



```
include "all_different.mzn";
include "all_equal.mzn";
include "decreasing.mzn";
include "nvalue_fn.mzn";
include "table.mzn";
include "value_precede_chain.mzn";

+--259 lines: ----[ CARRIERS ]--------------------------------

+--201 lines: ----[ RBBS ]------------------------------------

+--415 lines: ----[ RADIOS ]----------------------------------

+-- 78 lines: ----[ BAND AND RANGE LIMIT]---------------------

+--276 lines: ----[ CPRI ]------------------------------------

+--175 lines: ----[ NODES ]-----------------------------------

+-- 11 lines: ----[ DATA STATUS ]-----------------------------

+-- 22 lines: ----[ SOFTWARE REVISIONS ]----------------------
~
```

- 83 constraints
  - Table is the most common global constraint
- Categorized according to RAN

A typical constraint
2 out of 83

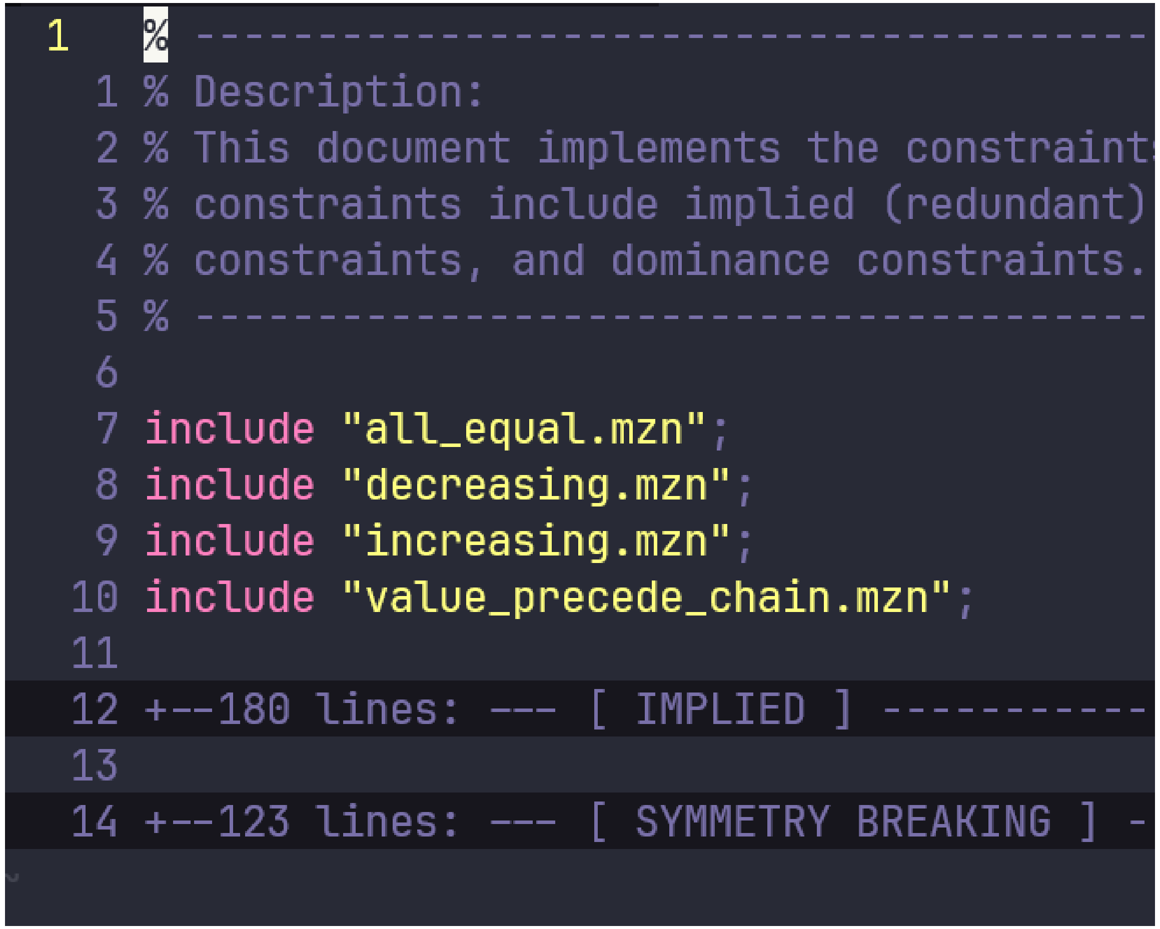


```
% Total bandwidth of carrier branches per RF port must not exceed the width of
% appropriate frequency band.
constraint :: "BAND-LIMIT-CLASSIC"
forall(r in Radios, p in Ports, fb in FrequencyBands) (
  % Downlink
  sum( b in TxBranches where bandOfBranch(b) == fb )
      ( (cb_radio[b] == r) * (cb_r_port[b] == p)
        * freqBwOfBranch(b) )
  <= freqBwDL(fb)
  /\
  % Uplink
  sum( b in RxBranches where bandOfBranch(b) == fb )
      ( (cb_radio[b] == r) * (cb_r_port[b] == p)
        * freqBwOfBranch(b) )
  <= freqBwUL(fb)
 );

% Total bandwidth of AAS carriers per radio must not exceed the width of
% appropriate frequency band.
constraint :: "BAND-LIMIT-AAS"
forall(r in Radios, fb in FrequencyBands) (
  sum( i in AasIndex, c = AasCarriers[i] where c.frequency_band == fb )
     ( ( aas_radio[i] == r)
       * c.fq_bandwidth )
  <= freqBwDL(fb)
);
```

- Constraints are annotated (flatzinc, findMus)
- Multiply with bool var, avoids reification

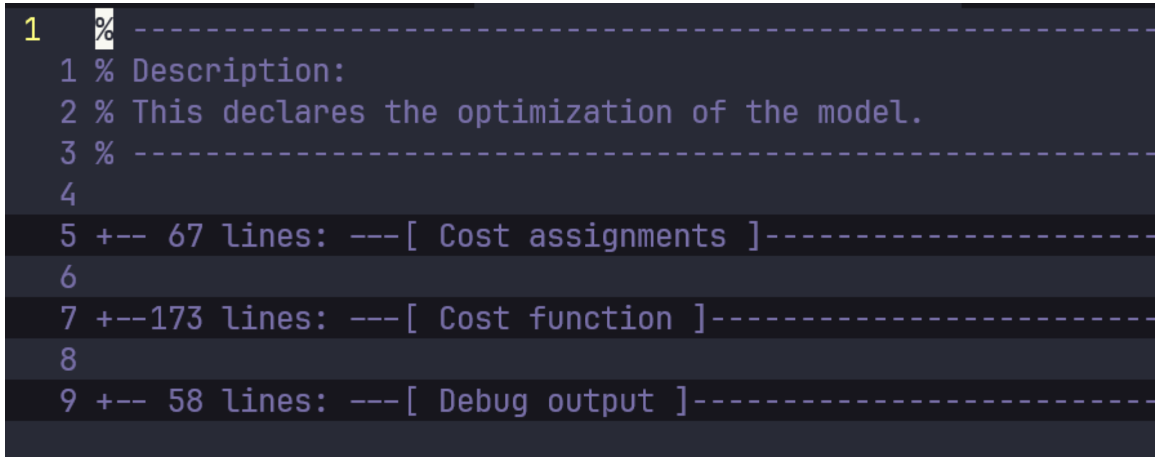
Constrains — Improving

- 24 improving constraints
- Mainly implied/symmetry breaking
  - Fixing dummy values etc.



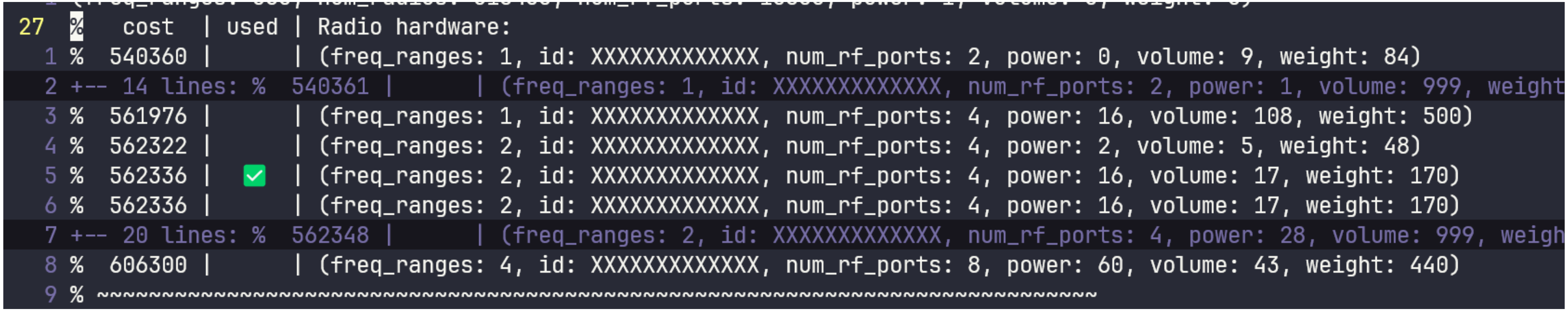
```
% ----------------------------------------
% Description:
% This document implements the constraint
% constraints include implied (redundant)
% constraints, and dominance constraints.
% ----------------------------------------

include "all_equal.mzn";
include "decreasing.mzn";
include "increasing.mzn";
include "value_precede_chain.mzn";

+--180 lines: --- [ IMPLIED ] -----------

+--123 lines: --- [ SYMMETRY BREAKING ] -
```

Optimization Function



```
% -----------------------------------------------------------
% Description:
% This declares the optimization of the model.
% -----------------------------------------------------------

+-- 67 lines: ---[ Cost assignments ]----------------------

+--173 lines: ---[ Cost function ]-------------------------

+-- 58 lines: ---[ Debug output ]--------------------------
```

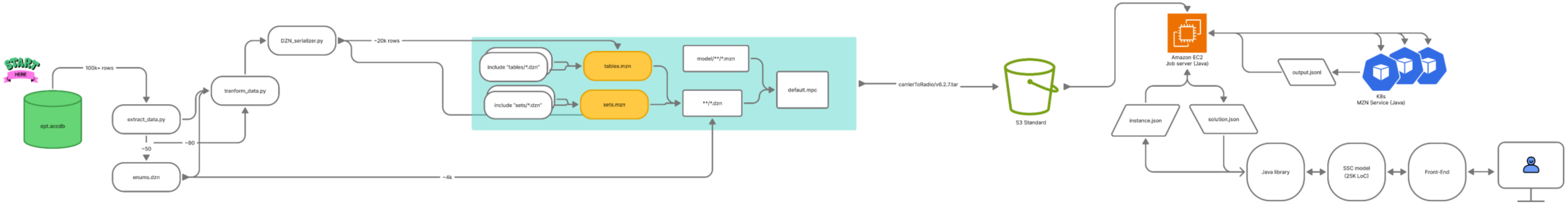
- Lexicographic Optimization
- Different Optimization Scenarios
  - Prioritize less weight, power etc.



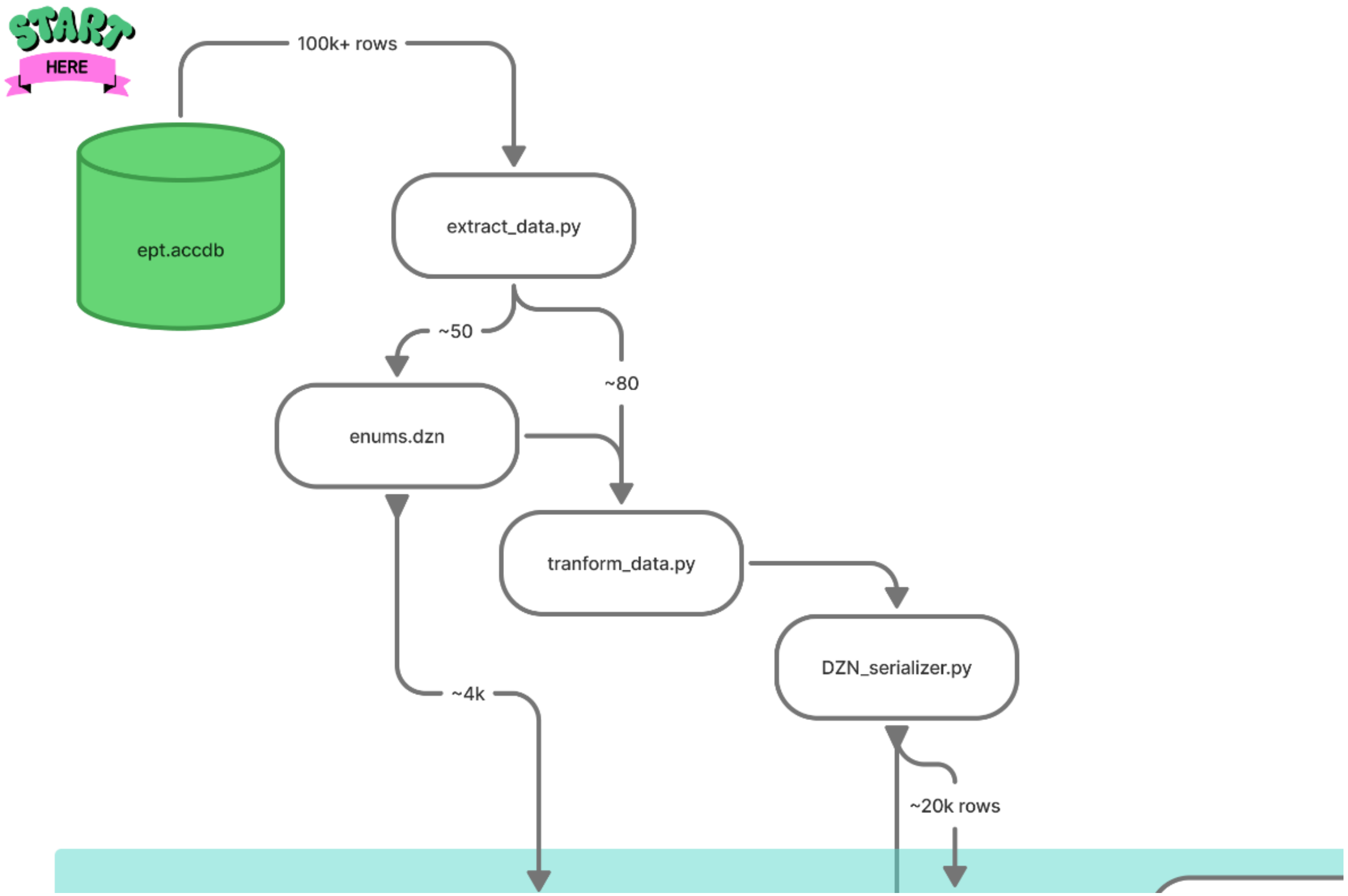
```
%    cost   | used | Radio hardware:
%  540360 |      | (freq_ranges: 1, id: XXXXXXXXXXXXX, num_rf_ports: 2, power: 0, volume: 9, weight: 84)
+-- 14 lines: %  540361 |      | (freq_ranges: 1, id: XXXXXXXXXXXXX, num_rf_ports: 2, power: 1, volume: 999, weight
%  561976 |      | (freq_ranges: 1, id: XXXXXXXXXXXXX, num_rf_ports: 4, power: 16, volume: 108, weight: 500)
%  562322 |      | (freq_ranges: 2, id: XXXXXXXXXXXXX, num_rf_ports: 4, power: 2, volume: 5, weight: 48)
%  562336 |  ✓   | (freq_ranges: 2, id: XXXXXXXXXXXXX, num_rf_ports: 4, power: 16, volume: 17, weight: 170)
%  562336 |      | (freq_ranges: 2, id: XXXXXXXXXXXXX, num_rf_ports: 4, power: 16, volume: 17, weight: 170)
+-- 20 lines: %  562348 |      | (freq_ranges: 2, id: XXXXXXXXXXXXX, num_rf_ports: 4, power: 28, volume: 999, weigh
%  606300 |      | (freq_ranges: 4, id: XXXXXXXXXXXXX, num_rf_ports: 8, power: 60, volume: 43, weight: 440)
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Executing our CP model

From data to user

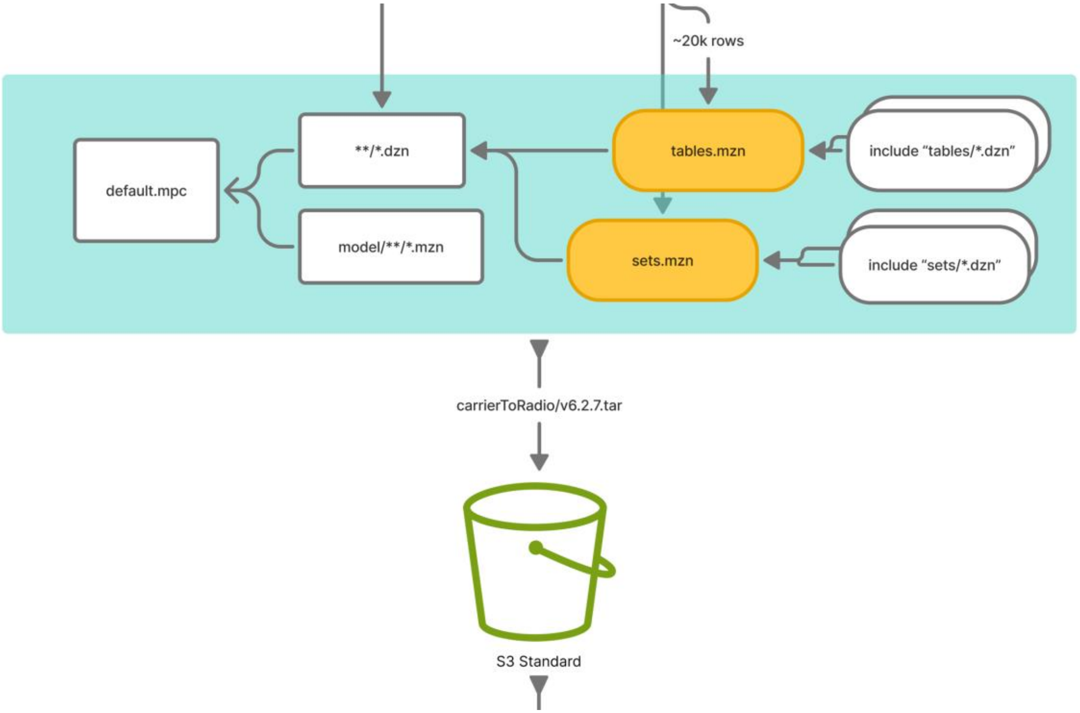


ETL — The Data Pipeline



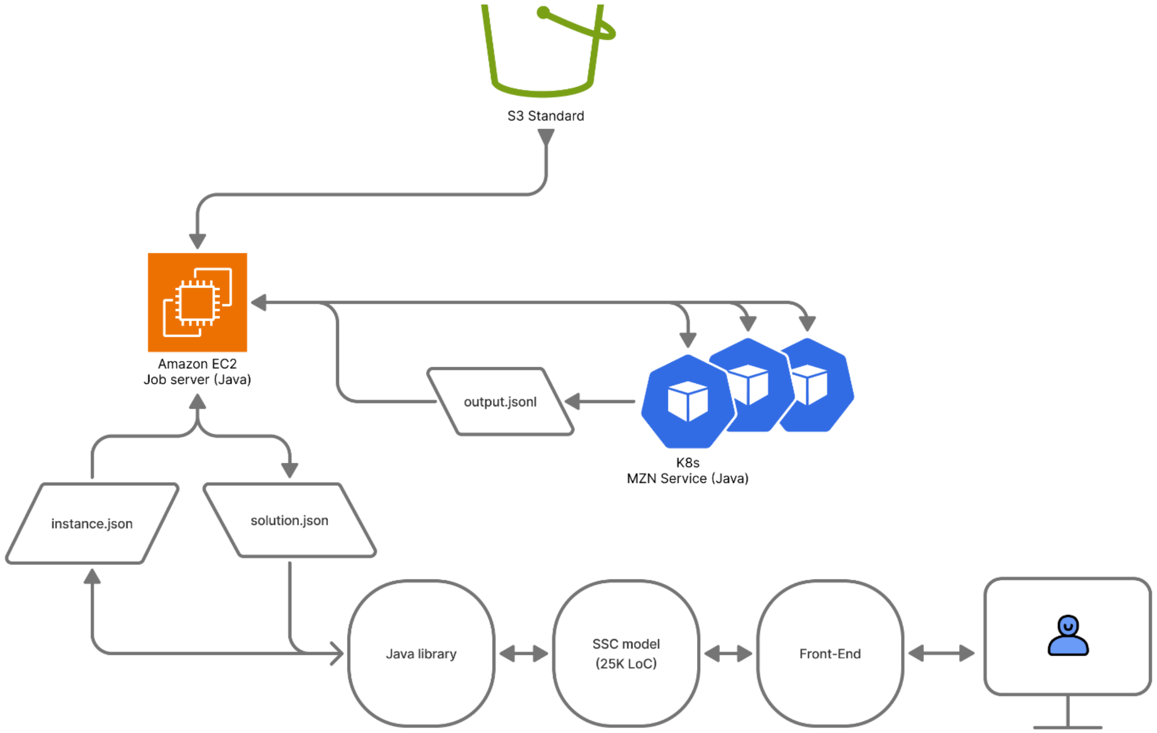
- ~8K python LOC
- Largest enum, 118 chars
- Largest table, ~20k rows
- Challenges:
  - Enums are in the global namespace
  - Special characters in unquoted enums
  - Creating “Null” enums
  - Serializing complex DZN types

Packaging — The model artifact



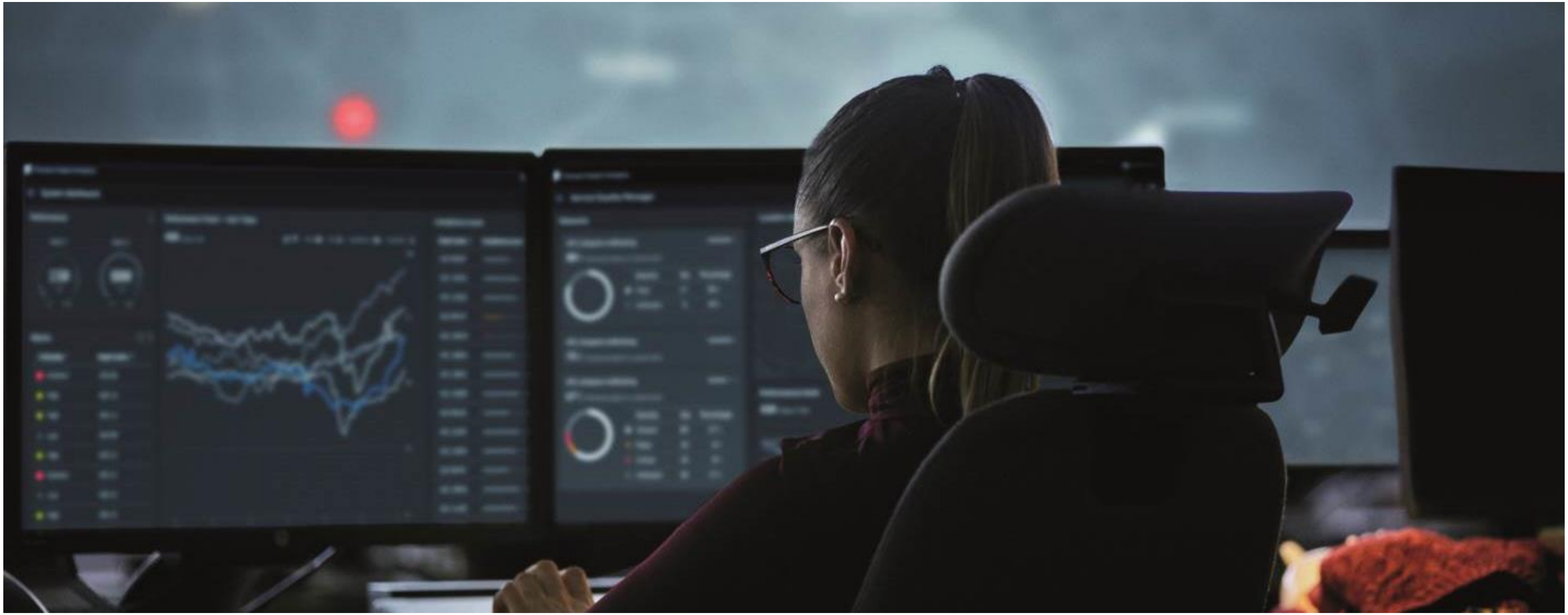
- Simple TAR archive (*.mzn + *.dzn)
- Tagged in Git with SemVer
- Name = model/Semver
- Regression/Integration tested
  - Uploaded to AWS S3 for distribution

Execution — The runtime



- Continuous delivery
  - SemVer (always get newest)
  - Tiered environments (dev, test, prod)
- Scalability
  - Job queue
  - Parallel processing (K8 cluster)

Maintaining our CP model

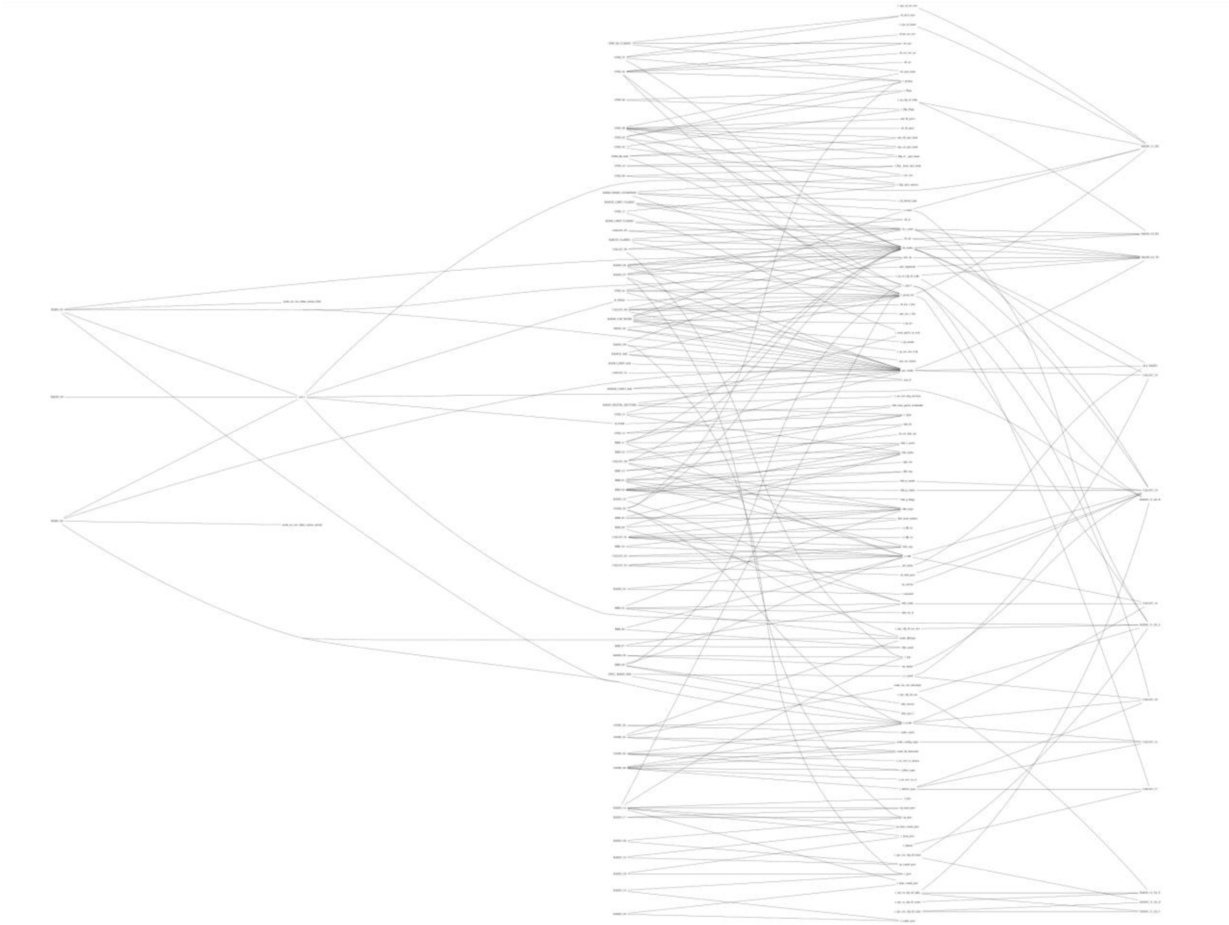


Ever increasing complexity
2x over the last 2 years

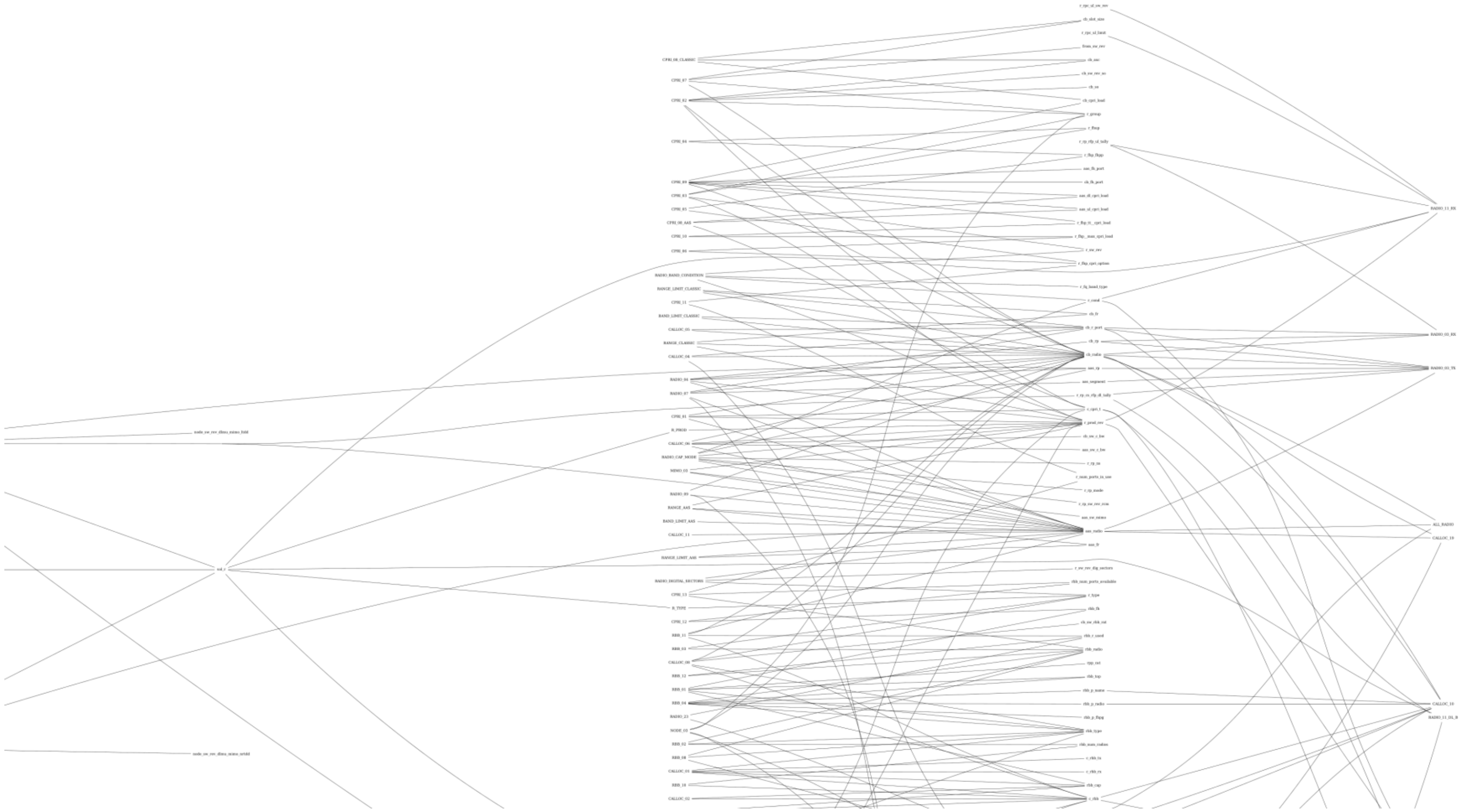
- +4K LoC *.py → +8K LoC *.py
- +4K LoC *.mzn → +8K LoC * .mzn
- +7K LoC *.rst → +10K LoC *.rst
- +60K LoC *.dzn → +120K LoC *.dzn
- 30 testcases → ~140 testcases

- New products
- New rules
- New language features

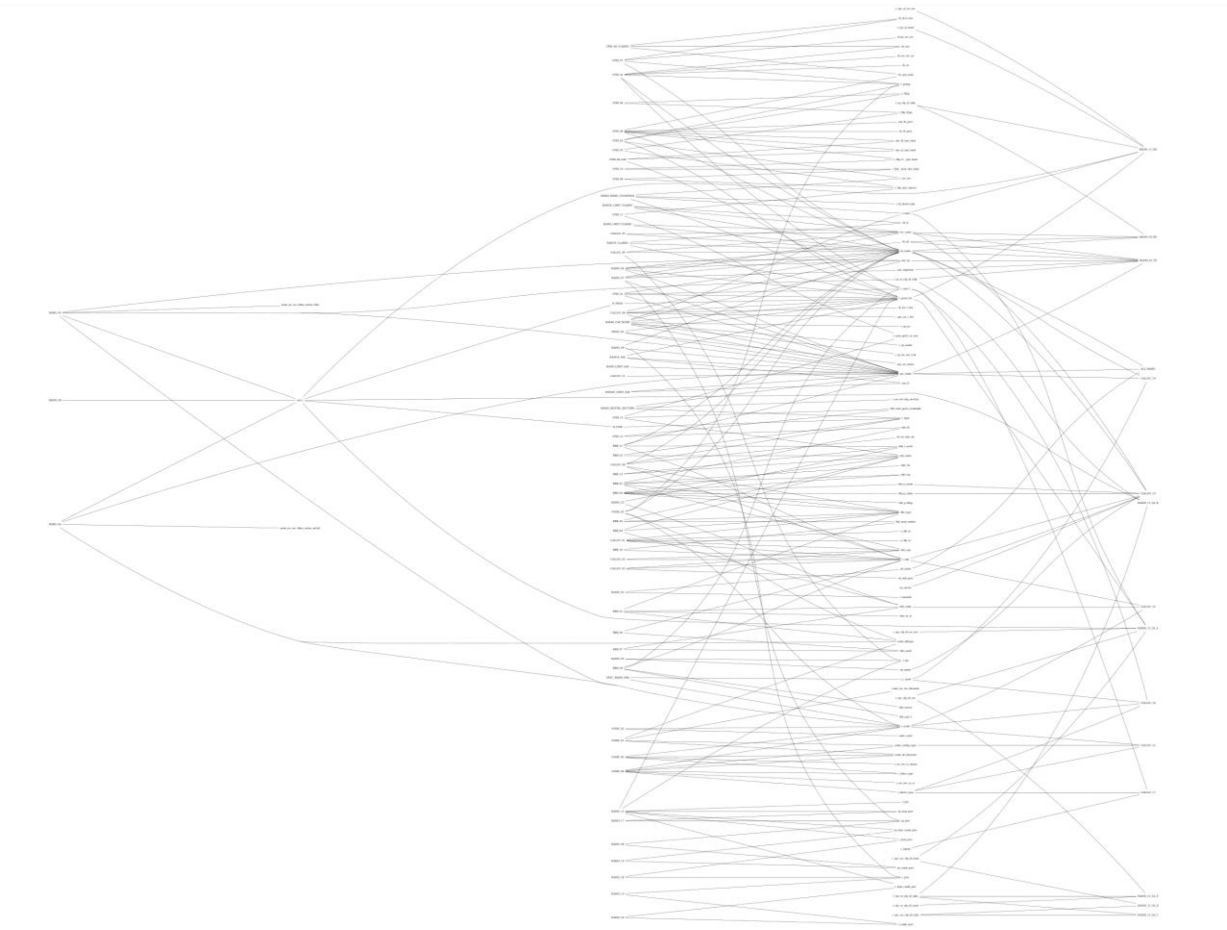
All constraints & decision variables



All constraints & decision variables

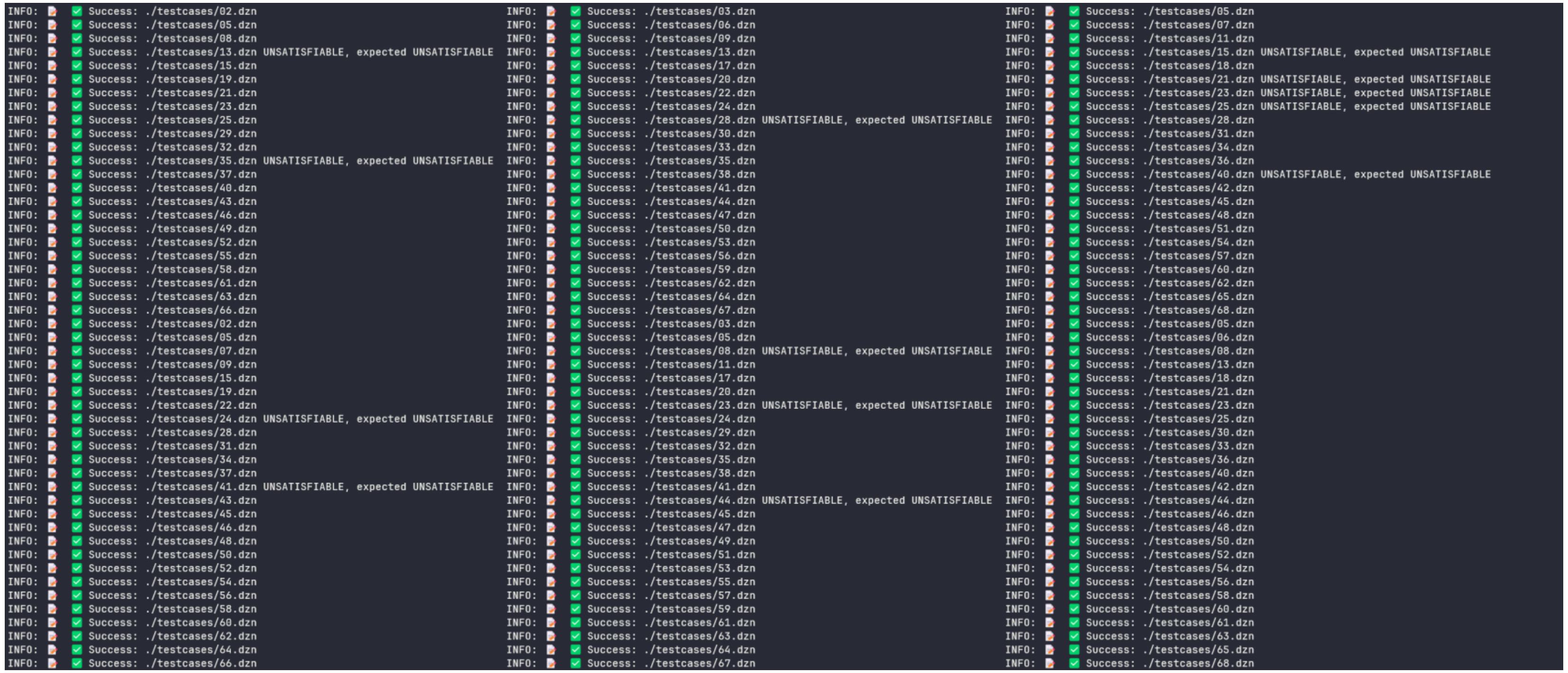


How do we deal with this complexity?



More test

Better test coverage, less regressions & bugs 👏 👏 👏

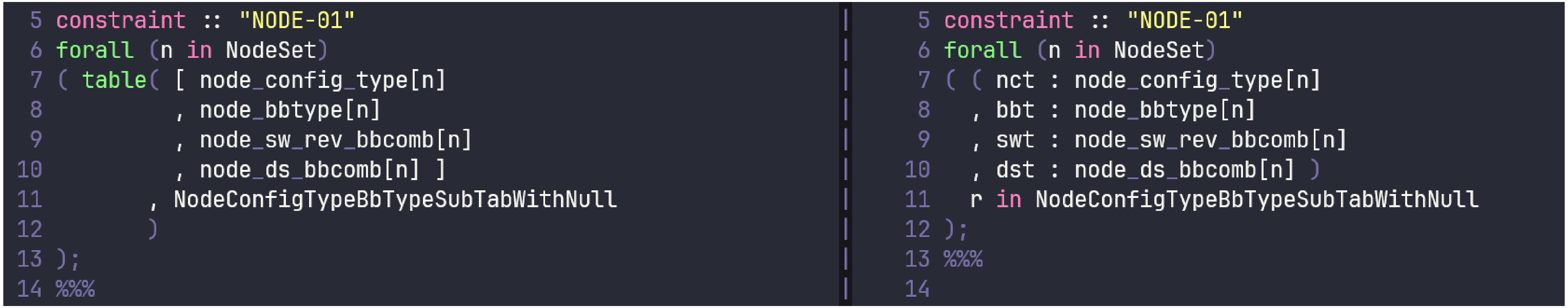


```
INFO: 📄 ✅ Success: ./testcases/02.dzn                                        INFO: 📄 ✅ Success: ./testcases/03.dzn                                        INFO: 📄 ✅ Success: ./testcases/05.dzn
INFO: 📄 ✅ Success: ./testcases/05.dzn                                        INFO: 📄 ✅ Success: ./testcases/06.dzn                                        INFO: 📄 ✅ Success: ./testcases/07.dzn
INFO: 📄 ✅ Success: ./testcases/08.dzn                                        INFO: 📄 ✅ Success: ./testcases/09.dzn                                        INFO: 📄 ✅ Success: ./testcases/11.dzn
INFO: 📄 ✅ Success: ./testcases/13.dzn UNSATISFIABLE, expected UNSATISFIABLE  INFO: 📄 ✅ Success: ./testcases/13.dzn                                        INFO: 📄 ✅ Success: ./testcases/15.dzn UNSATISFIABLE, expected UNSATISFIABLE
INFO: 📄 ✅ Success: ./testcases/15.dzn                                        INFO: 📄 ✅ Success: ./testcases/17.dzn                                        INFO: 📄 ✅ Success: ./testcases/18.dzn
INFO: 📄 ✅ Success: ./testcases/19.dzn                                        INFO: 📄 ✅ Success: ./testcases/20.dzn                                        INFO: 📄 ✅ Success: ./testcases/21.dzn UNSATISFIABLE, expected UNSATISFIABLE
INFO: 📄 ✅ Success: ./testcases/21.dzn                                        INFO: 📄 ✅ Success: ./testcases/22.dzn                                        INFO: 📄 ✅ Success: ./testcases/23.dzn UNSATISFIABLE, expected UNSATISFIABLE
INFO: 📄 ✅ Success: ./testcases/23.dzn                                        INFO: 📄 ✅ Success: ./testcases/24.dzn                                        INFO: 📄 ✅ Success: ./testcases/25.dzn UNSATISFIABLE, expected UNSATISFIABLE
INFO: 📄 ✅ Success: ./testcases/25.dzn                                        INFO: 📄 ✅ Success: ./testcases/28.dzn UNSATISFIABLE, expected UNSATISFIABLE  INFO: 📄 ✅ Success: ./testcases/28.dzn
INFO: 📄 ✅ Success: ./testcases/29.dzn                                        INFO: 📄 ✅ Success: ./testcases/30.dzn                                        INFO: 📄 ✅ Success: ./testcases/31.dzn
INFO: 📄 ✅ Success: ./testcases/32.dzn                                        INFO: 📄 ✅ Success: ./testcases/33.dzn                                        INFO: 📄 ✅ Success: ./testcases/34.dzn
INFO: 📄 ✅ Success: ./testcases/35.dzn UNSATISFIABLE, expected UNSATISFIABLE  INFO: 📄 ✅ Success: ./testcases/35.dzn                                        INFO: 📄 ✅ Success: ./testcases/36.dzn
INFO: 📄 ✅ Success: ./testcases/37.dzn                                        INFO: 📄 ✅ Success: ./testcases/38.dzn                                        INFO: 📄 ✅ Success: ./testcases/40.dzn UNSATISFIABLE, expected UNSATISFIABLE
INFO: 📄 ✅ Success: ./testcases/40.dzn                                        INFO: 📄 ✅ Success: ./testcases/41.dzn                                        INFO: 📄 ✅ Success: ./testcases/42.dzn
INFO: 📄 ✅ Success: ./testcases/43.dzn                                        INFO: 📄 ✅ Success: ./testcases/44.dzn                                        INFO: 📄 ✅ Success: ./testcases/45.dzn
INFO: 📄 ✅ Success: ./testcases/46.dzn                                        INFO: 📄 ✅ Success: ./testcases/47.dzn                                        INFO: 📄 ✅ Success: ./testcases/48.dzn
INFO: 📄 ✅ Success: ./testcases/49.dzn                                        INFO: 📄 ✅ Success: ./testcases/50.dzn                                        INFO: 📄 ✅ Success: ./testcases/51.dzn
INFO: 📄 ✅ Success: ./testcases/52.dzn                                        INFO: 📄 ✅ Success: ./testcases/53.dzn                                        INFO: 📄 ✅ Success: ./testcases/54.dzn
INFO: 📄 ✅ Success: ./testcases/55.dzn                                        INFO: 📄 ✅ Success: ./testcases/56.dzn                                        INFO: 📄 ✅ Success: ./testcases/57.dzn
INFO: 📄 ✅ Success: ./testcases/58.dzn                                        INFO: 📄 ✅ Success: ./testcases/59.dzn                                        INFO: 📄 ✅ Success: ./testcases/60.dzn
INFO: 📄 ✅ Success: ./testcases/61.dzn                                        INFO: 📄 ✅ Success: ./testcases/62.dzn                                        INFO: 📄 ✅ Success: ./testcases/62.dzn
INFO: 📄 ✅ Success: ./testcases/63.dzn                                        INFO: 📄 ✅ Success: ./testcases/64.dzn                                        INFO: 📄 ✅ Success: ./testcases/65.dzn
INFO: 📄 ✅ Success: ./testcases/66.dzn                                        INFO: 📄 ✅ Success: ./testcases/67.dzn                                        INFO: 📄 ✅ Success: ./testcases/68.dzn
INFO: 📄 ✅ Success: ./testcases/02.dzn                                        INFO: 📄 ✅ Success: ./testcases/03.dzn                                        INFO: 📄 ✅ Success: ./testcases/05.dzn
INFO: 📄 ✅ Success: ./testcases/05.dzn                                        INFO: 📄 ✅ Success: ./testcases/05.dzn                                        INFO: 📄 ✅ Success: ./testcases/06.dzn
INFO: 📄 ✅ Success: ./testcases/07.dzn                                        INFO: 📄 ✅ Success: ./testcases/08.dzn UNSATISFIABLE, expected UNSATISFIABLE  INFO: 📄 ✅ Success: ./testcases/08.dzn
INFO: 📄 ✅ Success: ./testcases/09.dzn                                        INFO: 📄 ✅ Success: ./testcases/11.dzn                                        INFO: 📄 ✅ Success: ./testcases/13.dzn
INFO: 📄 ✅ Success: ./testcases/15.dzn                                        INFO: 📄 ✅ Success: ./testcases/17.dzn                                        INFO: 📄 ✅ Success: ./testcases/18.dzn
INFO: 📄 ✅ Success: ./testcases/19.dzn                                        INFO: 📄 ✅ Success: ./testcases/20.dzn                                        INFO: 📄 ✅ Success: ./testcases/21.dzn
INFO: 📄 ✅ Success: ./testcases/22.dzn                                        INFO: 📄 ✅ Success: ./testcases/23.dzn UNSATISFIABLE, expected UNSATISFIABLE  INFO: 📄 ✅ Success: ./testcases/23.dzn
INFO: 📄 ✅ Success: ./testcases/24.dzn UNSATISFIABLE, expected UNSATISFIABLE  INFO: 📄 ✅ Success: ./testcases/24.dzn                                        INFO: 📄 ✅ Success: ./testcases/25.dzn
INFO: 📄 ✅ Success: ./testcases/28.dzn                                        INFO: 📄 ✅ Success: ./testcases/29.dzn                                        INFO: 📄 ✅ Success: ./testcases/30.dzn
INFO: 📄 ✅ Success: ./testcases/31.dzn                                        INFO: 📄 ✅ Success: ./testcases/32.dzn                                        INFO: 📄 ✅ Success: ./testcases/33.dzn
INFO: 📄 ✅ Success: ./testcases/34.dzn                                        INFO: 📄 ✅ Success: ./testcases/35.dzn                                        INFO: 📄 ✅ Success: ./testcases/36.dzn
INFO: 📄 ✅ Success: ./testcases/37.dzn                                        INFO: 📄 ✅ Success: ./testcases/38.dzn                                        INFO: 📄 ✅ Success: ./testcases/40.dzn
INFO: 📄 ✅ Success: ./testcases/41.dzn UNSATISFIABLE, expected UNSATISFIABLE  INFO: 📄 ✅ Success: ./testcases/41.dzn                                        INFO: 📄 ✅ Success: ./testcases/42.dzn
INFO: 📄 ✅ Success: ./testcases/43.dzn                                        INFO: 📄 ✅ Success: ./testcases/44.dzn UNSATISFIABLE, expected UNSATISFIABLE  INFO: 📄 ✅ Success: ./testcases/44.dzn
INFO: 📄 ✅ Success: ./testcases/45.dzn                                        INFO: 📄 ✅ Success: ./testcases/45.dzn                                        INFO: 📄 ✅ Success: ./testcases/46.dzn
INFO: 📄 ✅ Success: ./testcases/46.dzn                                        INFO: 📄 ✅ Success: ./testcases/47.dzn                                        INFO: 📄 ✅ Success: ./testcases/48.dzn
INFO: 📄 ✅ Success: ./testcases/48.dzn                                        INFO: 📄 ✅ Success: ./testcases/49.dzn                                        INFO: 📄 ✅ Success: ./testcases/50.dzn
INFO: 📄 ✅ Success: ./testcases/50.dzn                                        INFO: 📄 ✅ Success: ./testcases/51.dzn                                        INFO: 📄 ✅ Success: ./testcases/52.dzn
INFO: 📄 ✅ Success: ./testcases/52.dzn                                        INFO: 📄 ✅ Success: ./testcases/53.dzn                                        INFO: 📄 ✅ Success: ./testcases/54.dzn
INFO: 📄 ✅ Success: ./testcases/54.dzn                                        INFO: 📄 ✅ Success: ./testcases/55.dzn                                        INFO: 📄 ✅ Success: ./testcases/56.dzn
INFO: 📄 ✅ Success: ./testcases/56.dzn                                        INFO: 📄 ✅ Success: ./testcases/57.dzn                                        INFO: 📄 ✅ Success: ./testcases/58.dzn
INFO: 📄 ✅ Success: ./testcases/58.dzn                                        INFO: 📄 ✅ Success: ./testcases/59.dzn                                        INFO: 📄 ✅ Success: ./testcases/60.dzn
INFO: 📄 ✅ Success: ./testcases/60.dzn                                        INFO: 📄 ✅ Success: ./testcases/61.dzn                                        INFO: 📄 ✅ Success: ./testcases/61.dzn
INFO: 📄 ✅ Success: ./testcases/62.dzn                                        INFO: 📄 ✅ Success: ./testcases/63.dzn                                        INFO: 📄 ✅ Success: ./testcases/63.dzn
INFO: 📄 ✅ Success: ./testcases/64.dzn                                        INFO: 📄 ✅ Success: ./testcases/64.dzn                                        INFO: 📄 ✅ Success: ./testcases/65.dzn
INFO: 📄 ✅ Success: ./testcases/66.dzn                                        INFO: 📄 ✅ Success: ./testcases/67.dzn                                        INFO: 📄 ✅ Success: ./testcases/68.dzn
```

- Testing both SAT/UNSAT
- New rules
  ➡️ new test
- New bug/regression
  ➡️ new test

Stronger types

Records instead of 2d array of int 👏 👏 👏



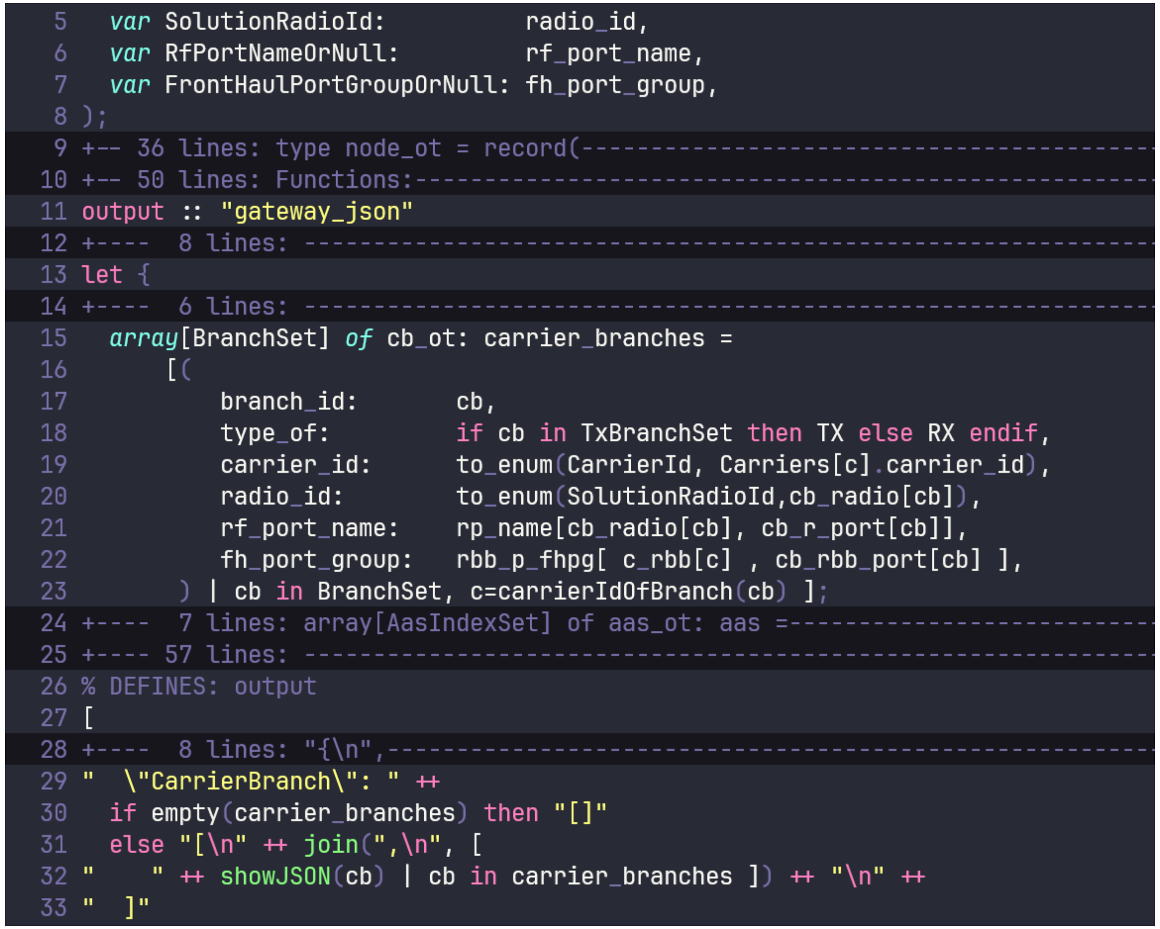
```
constraint :: "NODE-01"
forall (n in NodeSet)
( table( [ node_config_type[n]
         , node_bbtype[n]
         , node_sw_rev_bbcomb[n]
         , node_ds_bbcomb[n] ]
       , NodeConfigTypeBbTypeSubTabWithNull
       )
);
%%%
```

```
constraint :: "NODE-01"
forall (n in NodeSet)
( ( nct : node_config_type[n]
  , bbt : node_bbtype[n]
  , swt : node_sw_rev_bbcomb[n]
  , dst : node_ds_bbcomb[n] )
  r in NodeConfigTypeBbTypeSubTabWithNull
);
%%%

```

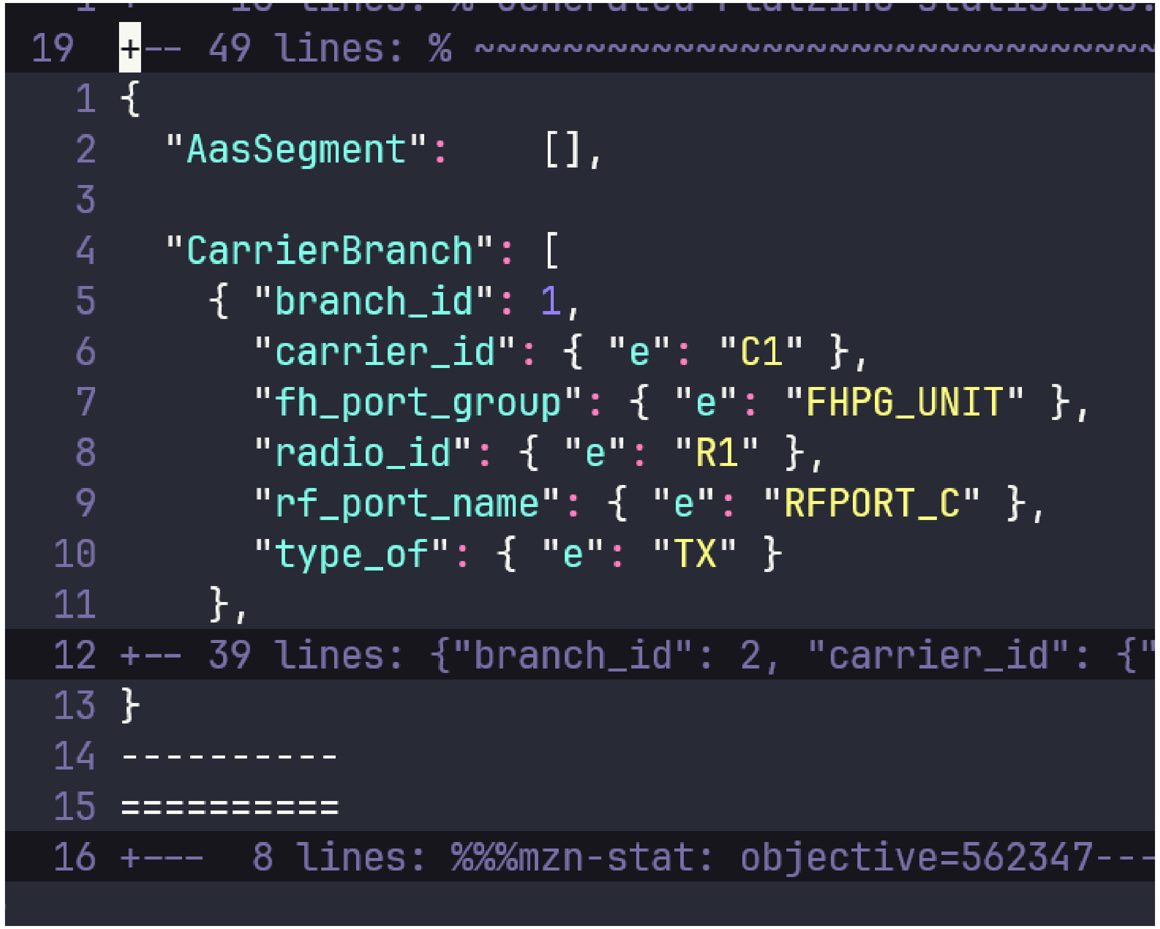
- Explicit declaration (no documentation rot)
- No accidental column mismatch
- No accidental type coercions

Records for input/output

Safe & robust object serialization 👏 👏 👏



```
  var SolutionRadioId:          radio_id,
  var RfPortNameOrNull:         rf_port_name,
  var FrontHaulPortGroupOrNull: fh_port_group,
);
+-- 36 lines: type node_ot = record(---------------------------------------------
+-- 50 lines: Functions:-----------------------------------------------------------
output :: "gateway_json"
+----  8 lines: -------------------------------------------------------------------
let {
+----  6 lines: -------------------------------------------------------------------
  array[BranchSet] of cb_ot: carrier_branches =
      [(
          branch_id:      cb,
          type_of:        if cb in TxBranchSet then TX else RX endif,
          carrier_id:     to_enum(CarrierId, Carriers[c].carrier_id),
          radio_id:       to_enum(SolutionRadioId,cb_radio[cb]),
          rf_port_name:   rp_name[cb_radio[cb], cb_r_port[cb]],
          fh_port_group:  rbb_p_fhpg[ c_rbb[c] , cb_rbb_port[cb] ],
        ) | cb in BranchSet, c=carrierIdOfBranch(cb) ];
+----  7 lines: array[AasIndexSet] of aas_ot: aas =-------------------------------
+---- 57 lines: -------------------------------------------------------------------
% DEFINES: output
[
+----  8 lines: "{\n",-------------------------------------------------------------
"  \"CarrierBranch\": " ++
  if empty(carrier_branches) then "[]"
  else "[\n" ++ join(",\n", [
"     " ++ showJSON(cb) | cb in carrier_branches ]) ++ "\n" ++
"  ]"
```

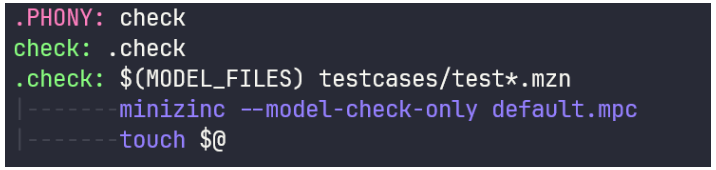


```
+---- 16 lines: % Generated FlatZinc statistics:
+-- 49 lines: % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
{
  "AasSegment":    [],

  "CarrierBranch": [
    { "branch_id": 1,
      "carrier_id": { "e": "C1" },
      "fh_port_group": { "e": "FHPG_UNIT" },
      "radio_id": { "e": "R1" },
      "rf_port_name": { "e": "RFPORT_C" },
      "type_of": { "e": "TX" }
    },
+-- 39 lines: {"branch_id": 2, "carrier_id": {"
}
----------
==========
+---  8 lines: %%%mzn-stat: objective=562347---
```

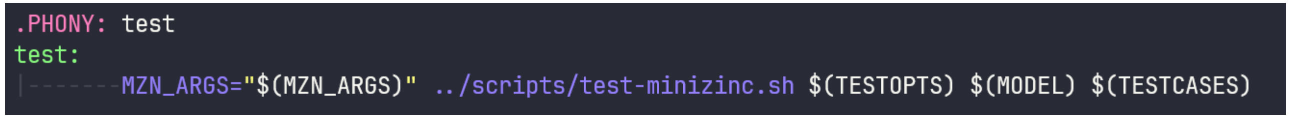
Exterminating bugs!

🐛 🐞 🦗

- Compiler errors
  - Stronger types (enums, records etc.)
- Consistency errors
  - Automatic test coverage (~120 testcases) + git bisect
  - Oracle model (SAP SSC)
- Interface errors
  - Strong contracts (types/SemVer)
  - Integration testing
- Solver bugs
  - Compare solvers
- Compiler bugs
  - Segfault/Strange behavior
    - Divide & conquer

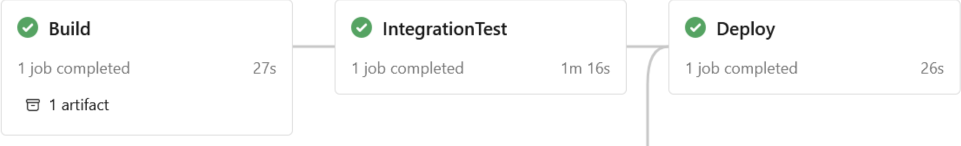


```
.PHONY: check
check: .check
.check: $(MODEL_FILES) testcases/test*.mzn
	minizinc --model-check-only default.mpc
	touch $@
```



```
.PHONY: test
test:
	MZN_ARGS="$(MZN_ARGS)" ../scripts/test-minizinc.sh $(TESTOPTS) $(MODEL) $(TESTCASES)
```

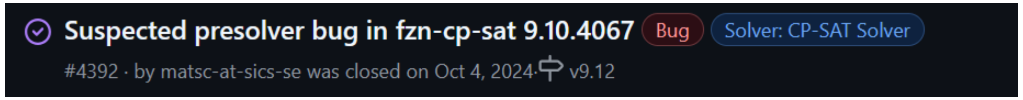
4.2.1
MAJOR
Minor
patch



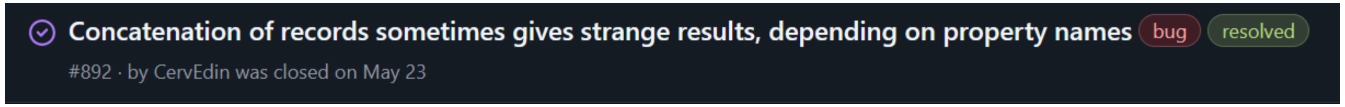
✓ Build
1 job completed 27s
1 artifact

✓ IntegrationTest
1 job completed 1m 16s

✓ Deploy
1 job completed 26s



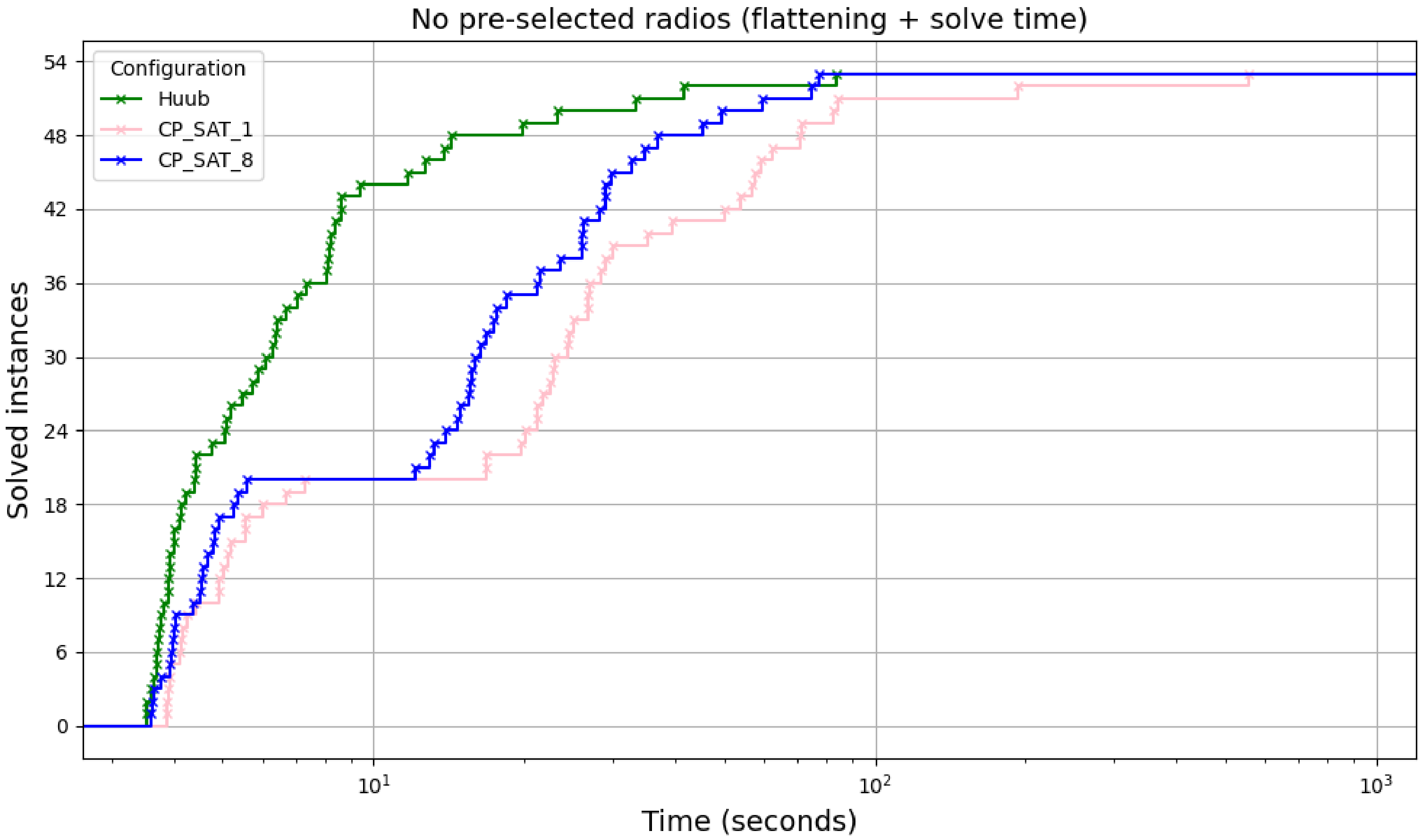
Suspected presolver bug in fzn-cp-sat 9.10.4067 Bug Solver: CP-SAT Solver
#4392 · by matsc-at-sics-se was closed on Oct 4, 2024 v9.12



Concatenation of records sometimes gives strange results, depending on property names bug resolved
#892 · by CervEdin was closed on May 23

Improving performance 🚀

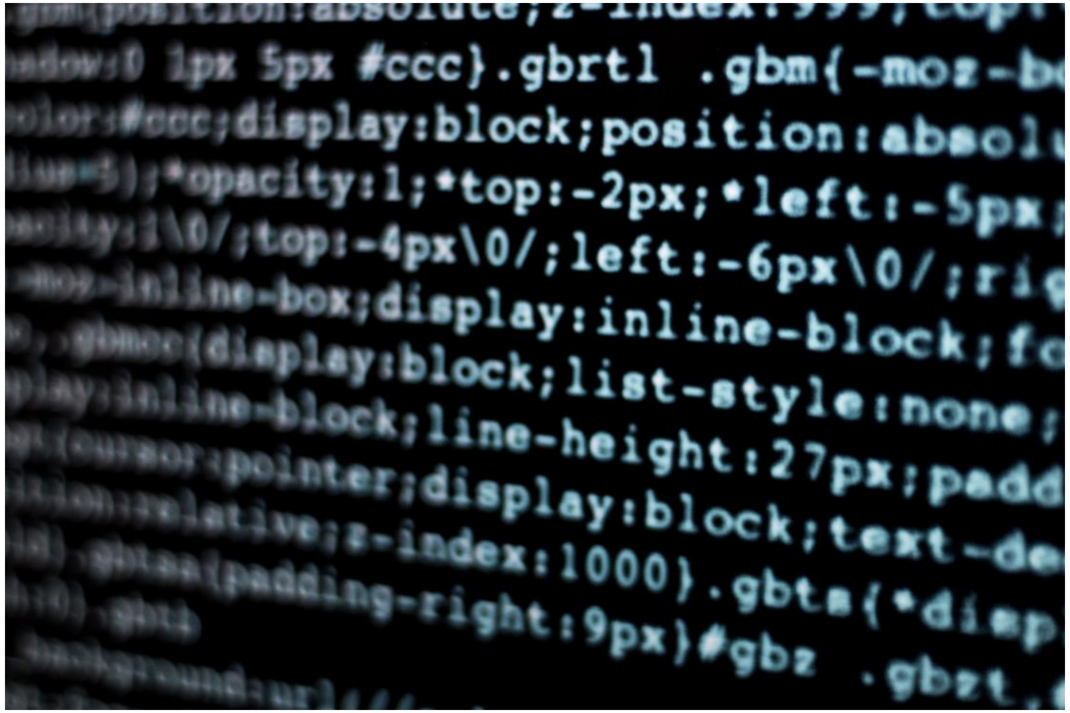
Experimenting with other solvers



Challenges – Pre-filtering

Less code, more constraints

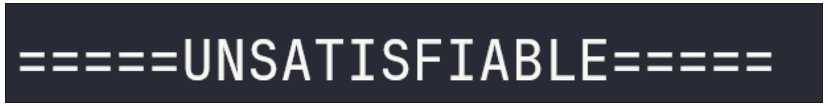
- Large models, ~8k+ LoC of MiniZinc
  - 1/4th is constraints
  - A substantial part is “pre-filtering” or “massaging”
    - Challenging in a DSL
    - Better data-types (caching, indexing etc.)
    - Less complexity, better debugging



```
position:absolute;z-index:999;top:
adow:0 1px 5px #ccc}.gbrtl .gbm{-moz-b
olor:#ccc;display:block;position:absol
ius=5);*opacity:1;*top:-2px;*left:-5px;
acity:1\0/;top:-4px\0/;left:-6px\0/;ri
-moz-inline-box;display:inline-block;fo
.gbmc{display:block;list-style:none;
play:inline-block;line-height:27px;padd
{cursor:pointer;display:block;text-de
ition:relative;z-index:1000}.gbts{*disp
.gbtsa{padding-right:9px}#gbz .gbzt,
.gbt
background:url(//
```

Challenges – Explainability

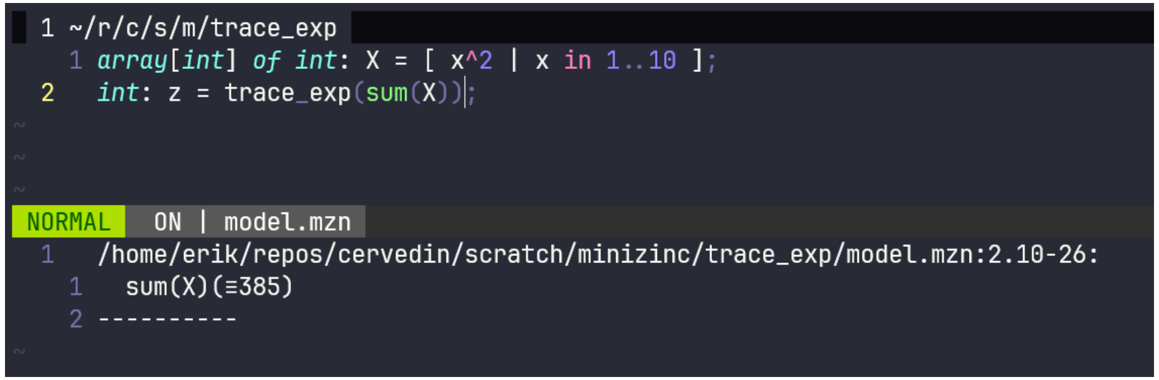
- Users want explanations, not just **no**
  - Explanations also help debugging
- Soft constraints
  - Challenging for CP
  - Suited for preferences, not explaining
- FindMUS
  - Expensive
  - Cryptic
  - Can be MANY



```
=====UNSATISFIABLE=====
```

Challenges – Debugging

- Conditional debug output
- `trace_exp`
  - Prints an expression and value
- Black box
  - Poke the box and see what happens!
    - Manual assignments
    - Manually bisect constraints (delete/comment)
      until UNSAT becomes SAT

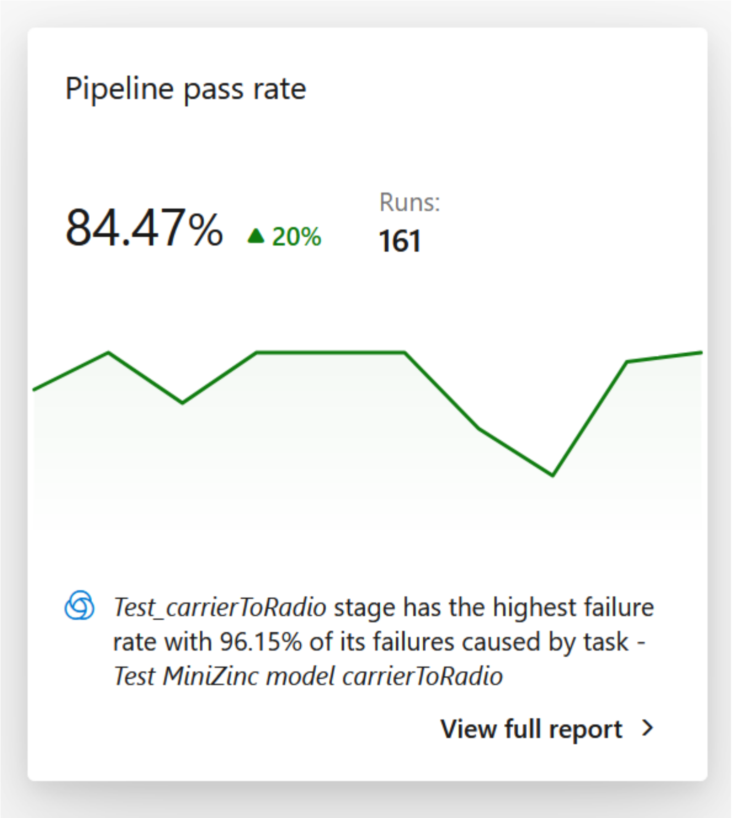


```
array[int] of int: X = [ x^2 | x in 1..10 ];
int: z = trace_exp(sum(X));
```

```
/home/erik/repos/cervedin/scratch/minizinc/trace_exp/model.mzn:2.10-26:
  sum(X)(≡385)
----------
```

Challenges – Testing

- Hard to test specific parts of the model
  - Need to test the whole model
  - Test using partial assignments
  - Test both for SAT/UNSAT
  - Indirect tests
- Rigorous testing
  - Automated testing (CI/CD/MLOPS)
  - Improving coverage (removing constraint should fail)
  - Large changes difficult (testcase might be symmetrical)

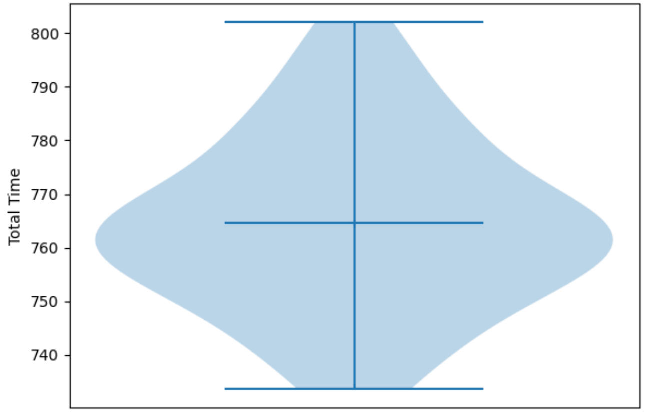


Test_carrierToRadio stage has the highest failure rate with 96.15% of its failures caused by task - *Test MiniZinc model carrierToRadio*

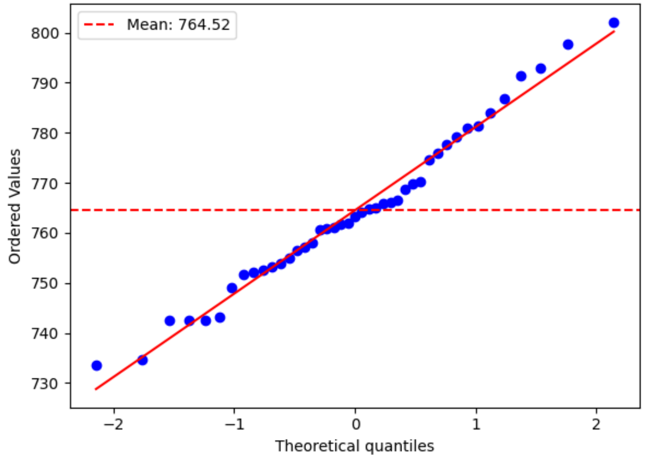
Challenges – Benchmarking Performance

- Small impact, hard to measure
- Parallel solving → high variance
  - Times are normally distributed
  - Benchmark ~5 times
  - Compare flatzinc statistics
- Instance dependent variance
  - Benchmark several different instances

Violin Plot of Total Times per Run



Q-Q Plot for Total Times per Run



Conclusion

How to build a large CP model & live to tell the tale

- Test, test & test
- Stronger types 👍
- Make incremental changes
- Flexibility & reliability before performance
  - It's easier to make a **correct** program fast than a **fast** program correct.
- Have fun!

Q&A

Acknowledgements

- Mats Carlsson 🧑‍🚀
- Co-Workers
  - Fredrik, Sverker, Marko, Samuel, Marko, Zdravko, José, Mariia, Nils, Olle, Danyal
- MiniZinc team (Jip, Guido, Jason, Peter & friends)
- OR-Tools team (Laurent & friends)
- GeCode (Christian, Guido again, Mikael)
- Uppsala Optimization group (Mats, Pierre, Justin, Maria, Ramiz & friends)

Imagine Possible

ericsson.com/careers