# Exercise cpc-09: Example 5: Optimize Configuration

## Exercise-09b: Optimize the configuration of track indicators

**This exercise is an alternative to 09a!**
**You need not implement both, but only one …**

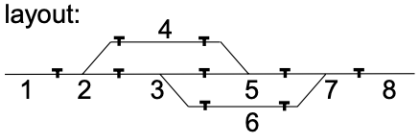
Change your MiniZinc implementation of the track indicator configuration from exercise-08b to solve the optimization problems specified in **example 5** in cpc-examples.pdf.

**Required result**: PDF file named surname(s)-09b.pdf containing following information

- objective function for minimal cost (delta to MiniZinc of exercise-08b)
- objective function for equal distribution (delta to MiniZinc of exercise-08b)
- Pareto front of both (for the given example)

Based on a (much more complicated) example from the railroad domain:

- Example track layout:




- T-separators divide tracks into sections (e.g. 1-8)
- Each section is monitored by a track indicator
- Each track indicator has a type (A, B, C, D)
- Neighboring indicators are subject to restrictions:
  - A allows only B as neighbors
  - B allows only A or C as neighbors
  - C allows only B or D as neighbors
  - D allows only C as neighbors

**Tasks**:

- Model - specific for the example track layout, e.g. single mzn file with variables, domains, constraints, test with all solutions
- Generic model for arbitrary track layouts, e.g. array of variables for sections in an mzn file, table for neighborship in a dzn file (for each layout)
- Test with various layouts and compare solving times e.g. duplicate example layout (and optionally connect the two)
- Define an unsatisfiable layout
- Implement **optimization** alternatives:
  - Minimize cost: A=1, B=2, C=3, D=4
  - Ensure equal distribution: All 4 types shall be selected as equally as possible
  - Pareto front of both alternatives (criteria)

Objective Function for minimal cost:

```
array[Type] of int: costs = [1, 2, 3, 4];
var int: c = sum(i in Sections)(costs[indicators[i]]);
solve minimize c;
output ["indicators=\(indicators) c=\(c)\n"];
```

Output:

```
Running tracks_opt.mzn, sample8_1.dzn                                  233msec
indicators=[A, B, A, A, B, B, A, B] c=12
----------
==========
%%%mzn-stat: failures=4
%%%mzn-stat: initTime=0.013798
%%%mzn-stat: nodes=9
%%%mzn-stat: peakDepth=4
%%%mzn-stat: propagations=262
%%%mzn-stat: propagators=24
```

```
%%%mzn-stat: restarts=0
%%%mzn-stat: solutions=1
%%%mzn-stat: solveTime=0.004617
%%%mzn-stat: variables=23
%%%mzn-stat-end
Finished in 233msec.
```

Objective Function for equal distribution:

```
var int: d = max(i in Type)(count(indicators, i)) - min(i in Type)(count(indicators, i));
solve minimize d;
output ["indicators=\(indicators) d=\(d)\n"];
```

Output:

```
Running tracks_opt.mzn, sample8_1.dzn                                    189msec
indicators=[A, B, A, A, B, B, A, B] d=4
----------
indicators=[C, D, C, C, B, B, A, B] d=2
----------
indicators=[D, C, B, D, C, A, B, A] d=0
----------
==========
%%%mzn-stat: failures=8
%%%mzn-stat: initTime=0.000472
%%%mzn-stat: nodes=22
%%%mzn-stat: peakDepth=8
%%%mzn-stat: propagations=438
%%%mzn-stat: propagators=16
%%%mzn-stat: restarts=0
%%%mzn-stat: solutions=3
%%%mzn-stat: solveTime=0.000232
%%%mzn-stat: variables=15
%%%mzn-stat-end
Finished in 189msec.
```

Pareto Front for both:

```
solve satisfy;
output ["c=\(c) d=\(d)\n"];
```

| C | D |
|---|---|
| 12 | 4 |
| 14 | 4 |
| 16 | 4 |
| 18 | 4 |
| 20 | 2 |
| 16 | 2 |
| 18 | 2 |
| 20 | 0 |
| 22 | 2 |
| 20 | 4 |
| 22 | 4 |
| 24 | 4 |
| 24 | 2 |
| 26 | 4 |
| 28 | 4 |