The Work Task Variation Problem

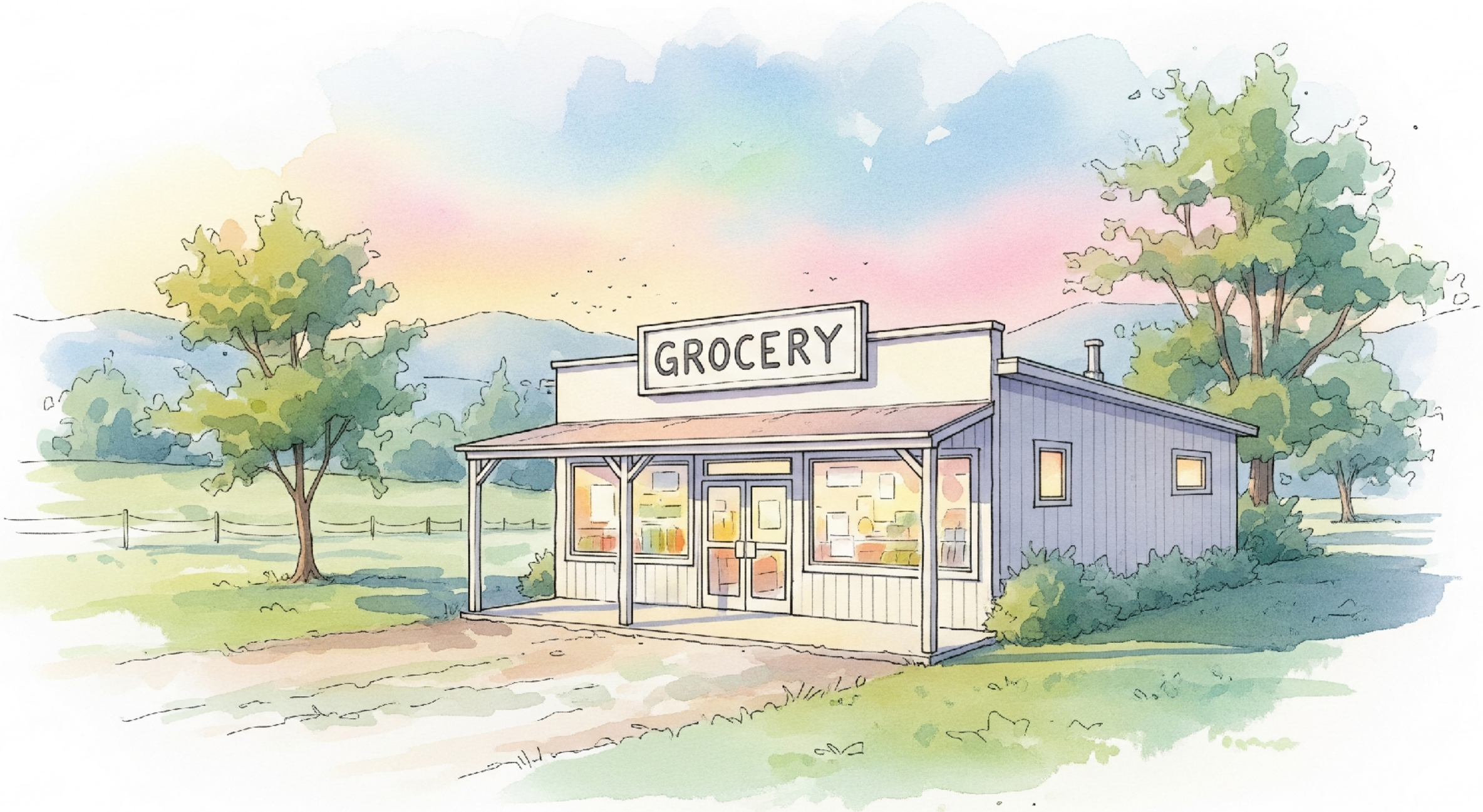
Implementing a solver for a single problem

Mikael Zayenz Lagerkvist – Optischedule / sambanova

Magnus Rattfeldt – Optischedule / Jeppesen

Optischedule

NordConsNet 2025 Uppsala


GROCERY

Shift plan for one person

W	T	S	W	T	S	W	T	S	W	T	S	W	T	S	W

S	S	S	S	S	T	T	T	T	T	W	W	W	W	W	W

S	S	T	T	T	W	W	W	W	W	W	S	S	S	T	T

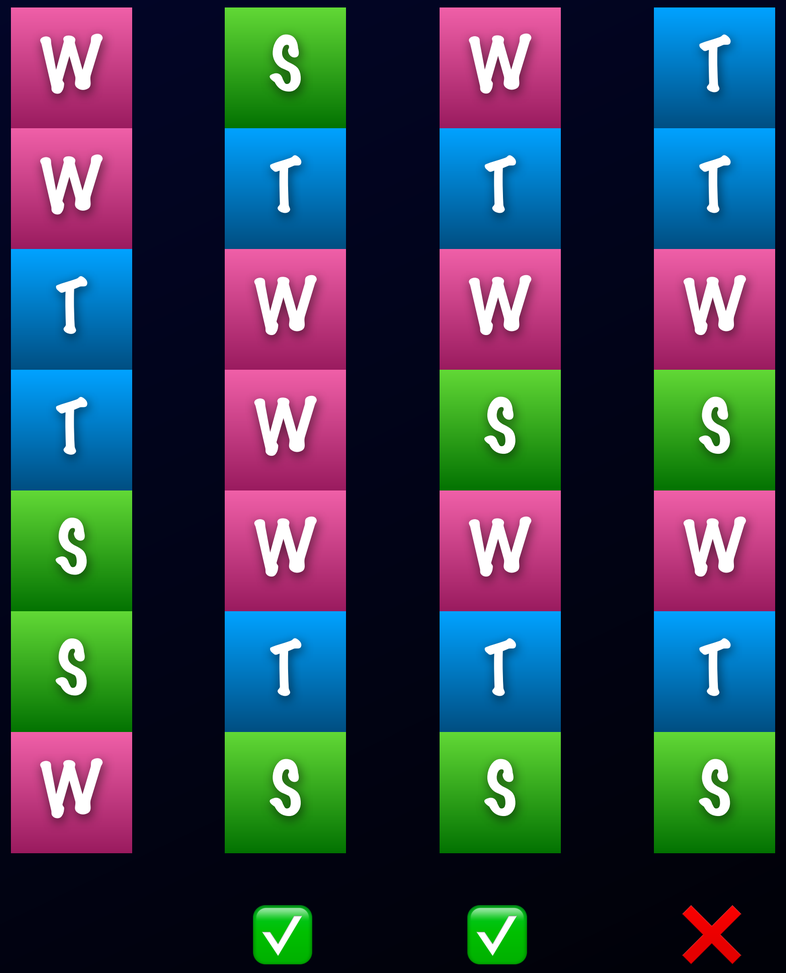
Shift plan for store

Shift plan for store

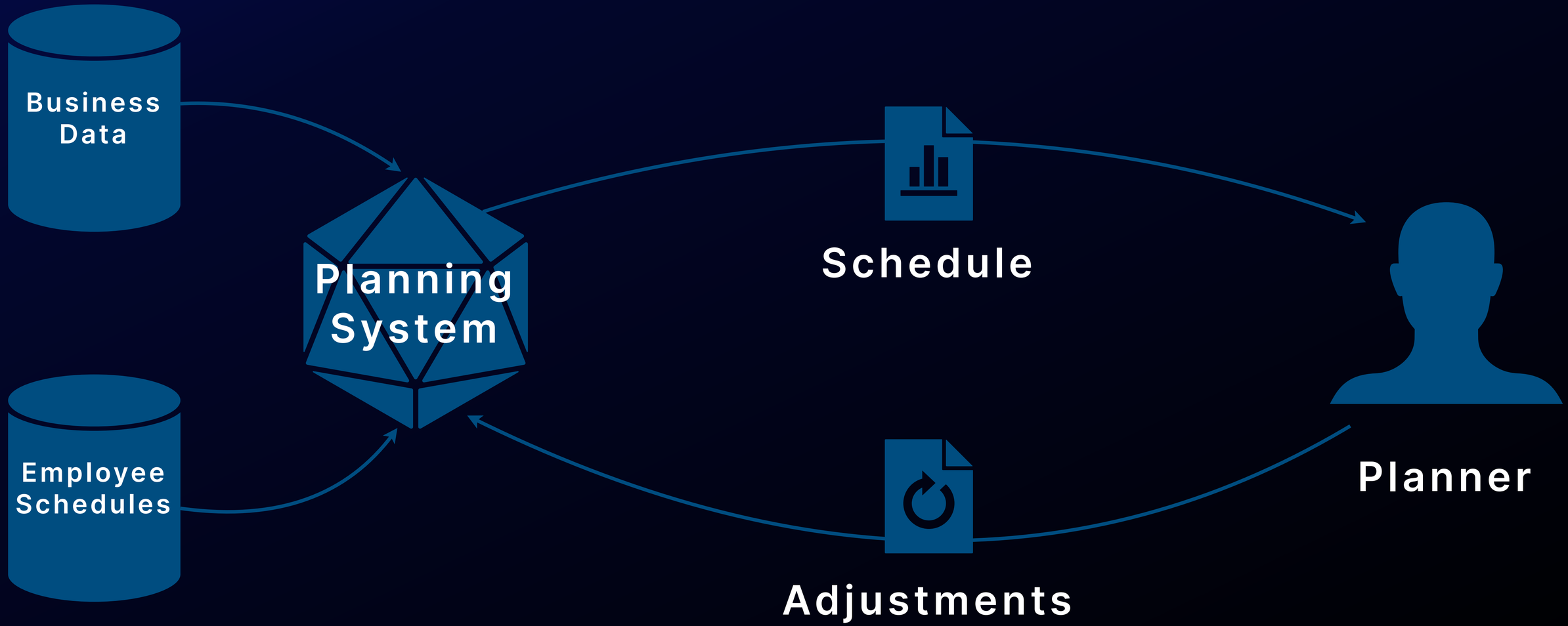
Shift plan for store

Requirements for time slot

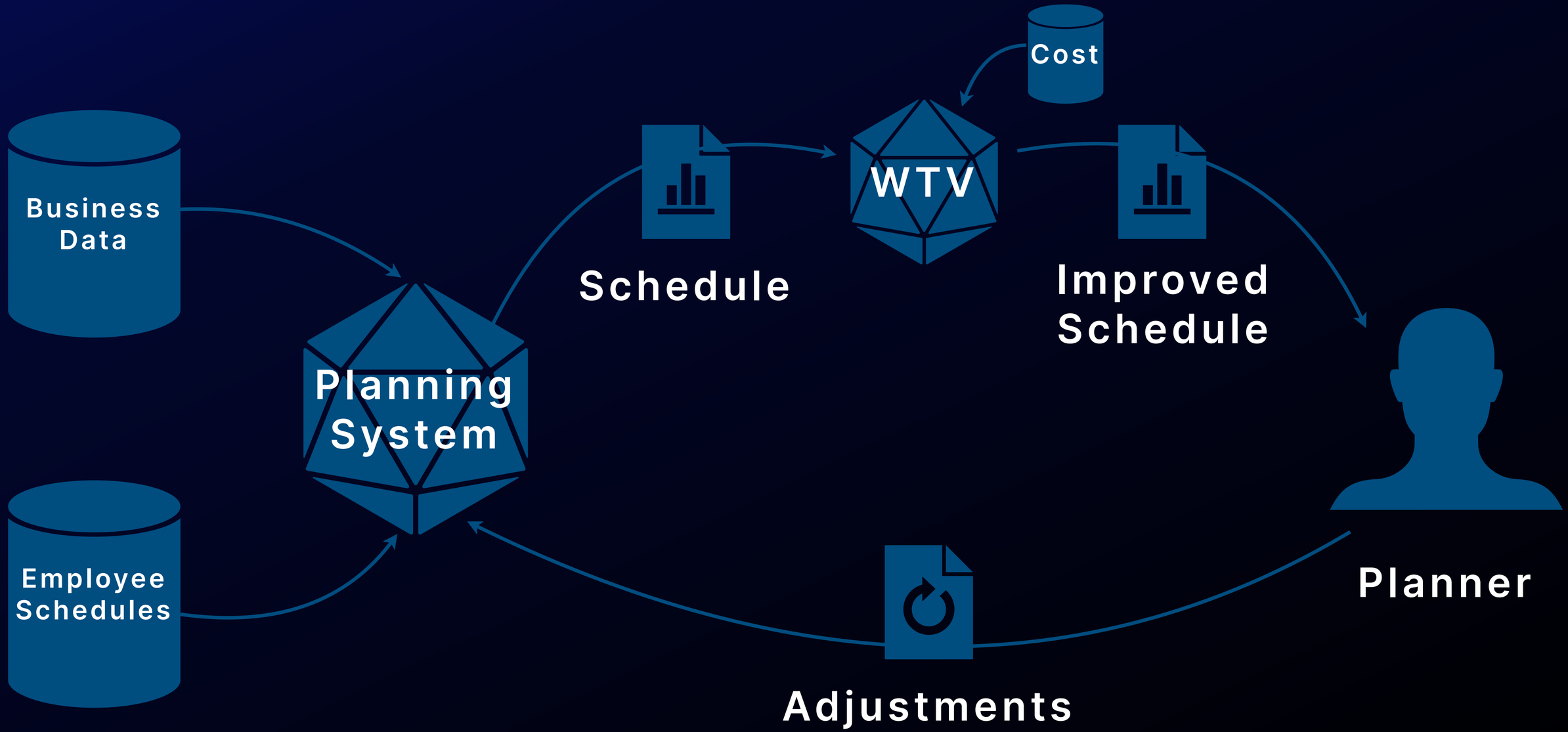
- 3 people on Warehouse tasks
- 2 people at Tills
- 2 people in Store



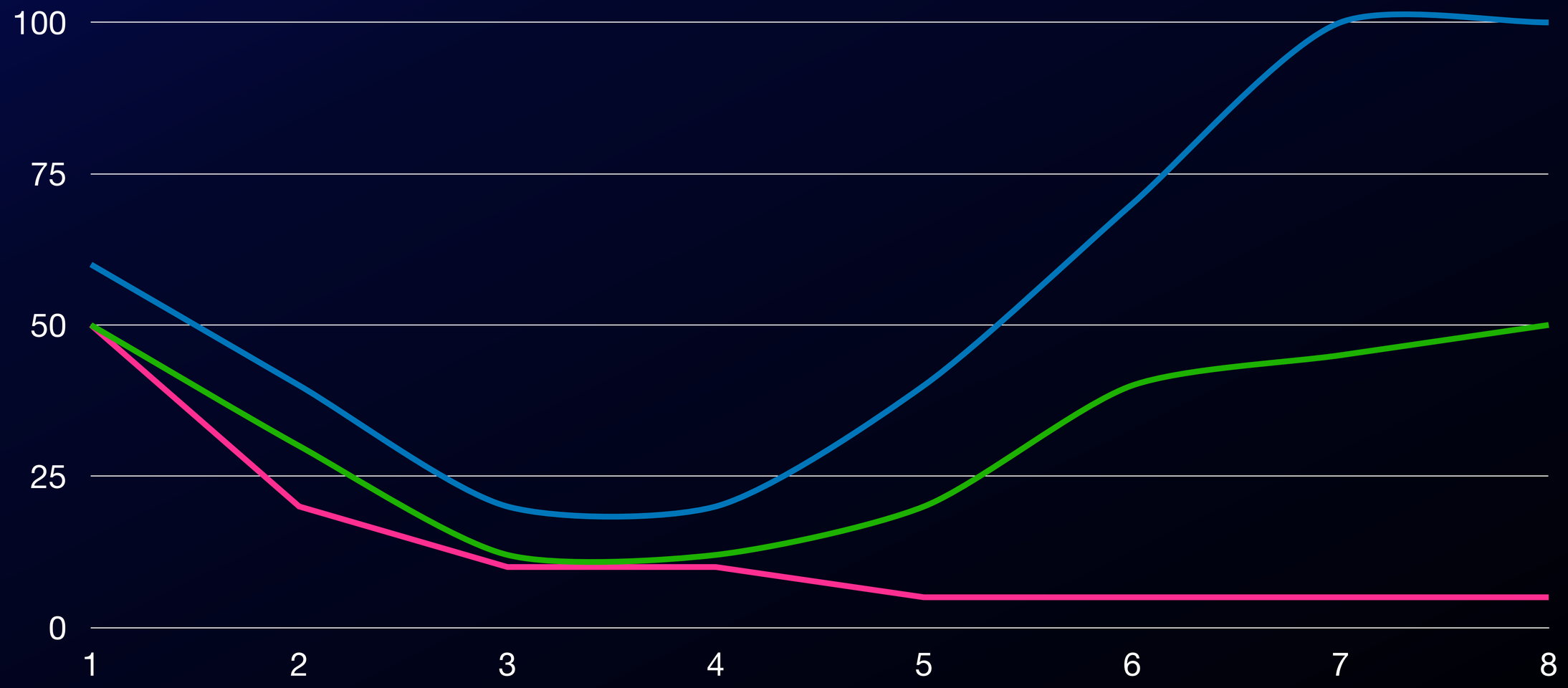
Planning context



Planning context



Cost curve example



Real world infrastructure

What to do first

- Reading input
- Data model
- Plotting
- Logging
- Producing output
- ...

It's a dangerous business, Frodo, going out your door. You step onto the road, and if you don't keep your feet, there's no knowing where you'll be swept off to.

RosterLogic Variation

A CBLS inspired solver

- Small and compact data-structures
- Runs are invariants, moves evaluated using simulation
  - Moves preserve hard constraints - no need for violations
- Simple moves of blocks and groups of blocks
- Pattern moves for structure
- All the standard searches

RosterLogic Variation

Pattern swaps

Search for potential patterns in row, swapping in from other rows.

Here, pattern is **SSSTT**

Slot	1	2	3	4	5	6	7	8
Shift 0	S	T	S	T	S	O	O	O
Shift 1	S	S	T	S	T	T	T	T
Shift 2	T	S	S	S	S	O	O	O
Shift 3	T	T	T	S	T	S	S	S

(a) Initial Schedule.

Slot	1	2	3	4	5	6	7	8
Shift 0	S	**S**	S	T	**T**	O	O	O
Shift 1	S	S	T	S	T	T	T	T
Shift 2	T	**T**	S	S	S	O	O	O
Shift 3	T	T	T	S	**S**	S	S	S

(b) Schedule After Pattern Swaps.

RosterLogic Variation

Local search algorithm configuration

Base algorithm

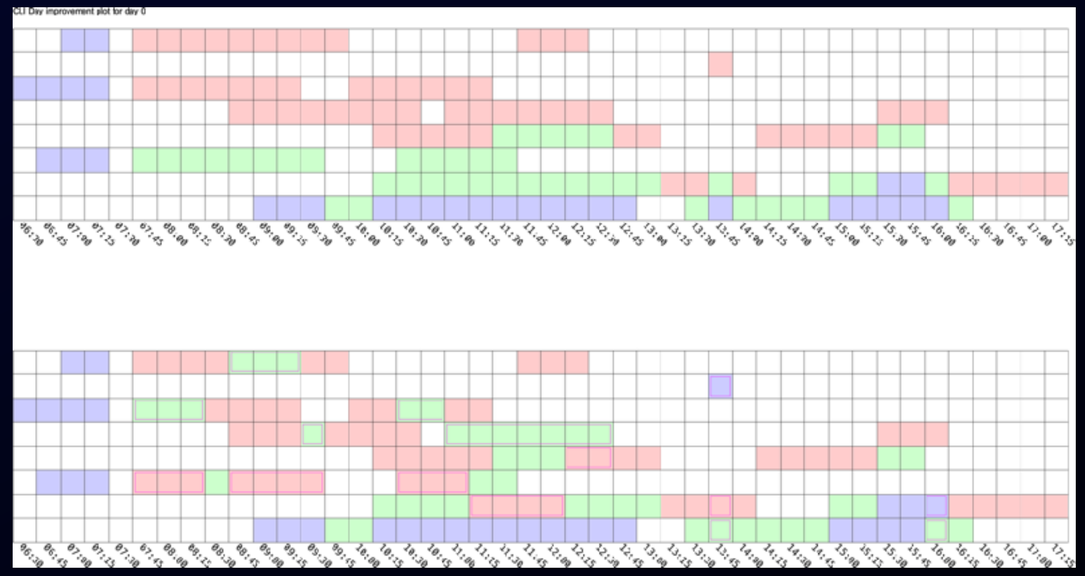
- Parallel restarts
  - Parallel portfolio
    - Steepest ascent
    - Simulated annealing
    - Tabu search (x3)
    - Scrambled steepest ascent

Usage

- Base algorithm
  - Swap Till ↔ Store
- Base algorithm
  - Swap Warehouse ↔ Till/Store
- Repeat twice
  - Steepest ascent, Till ↔ Store
  - Steepest ascent, Patterns swaps

RosterLogic Variation
Pragmatics

- Mostly developed during 2019
- Java and Kotlin
- Development on Mac and Linux,
  - Deployment as Windows CLI binary
  - Deployment for demo as AWS Lamba
- CSV (schedule) and Json (cost) as input formats
- Plotting invaluable



Building a Custom Solver for one problem?

Experience guides design

- We know what we are doing (hopefully)
- Customization critically important
  - Example: Filtering moves for custom rules
- Feedback using progress logging
- Fast iteration, full control
- Full IP rights, few dependencies

Is RosterLogic Variation good?
Comparing with MiniZinc model

- Full MiniZinc model in paper
  - MiniZinc Challenge 2025 model
- RosterLogic Variation used in practice
  - Speed usable
  - Results usable
- RosterLogic Variation from 2019, comparison to 2025 solvers

WTV Instances

Customer data is secret 😢

Generated WTV Instances to the rescue 😀

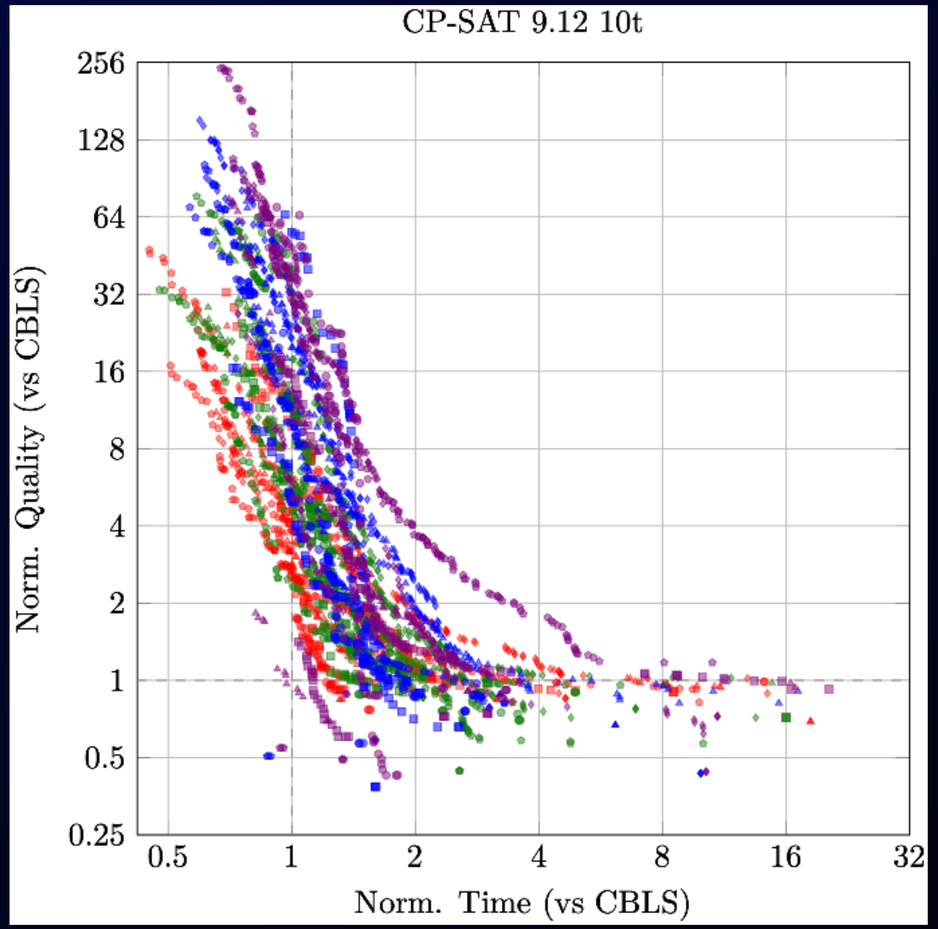
Looks similar to real data

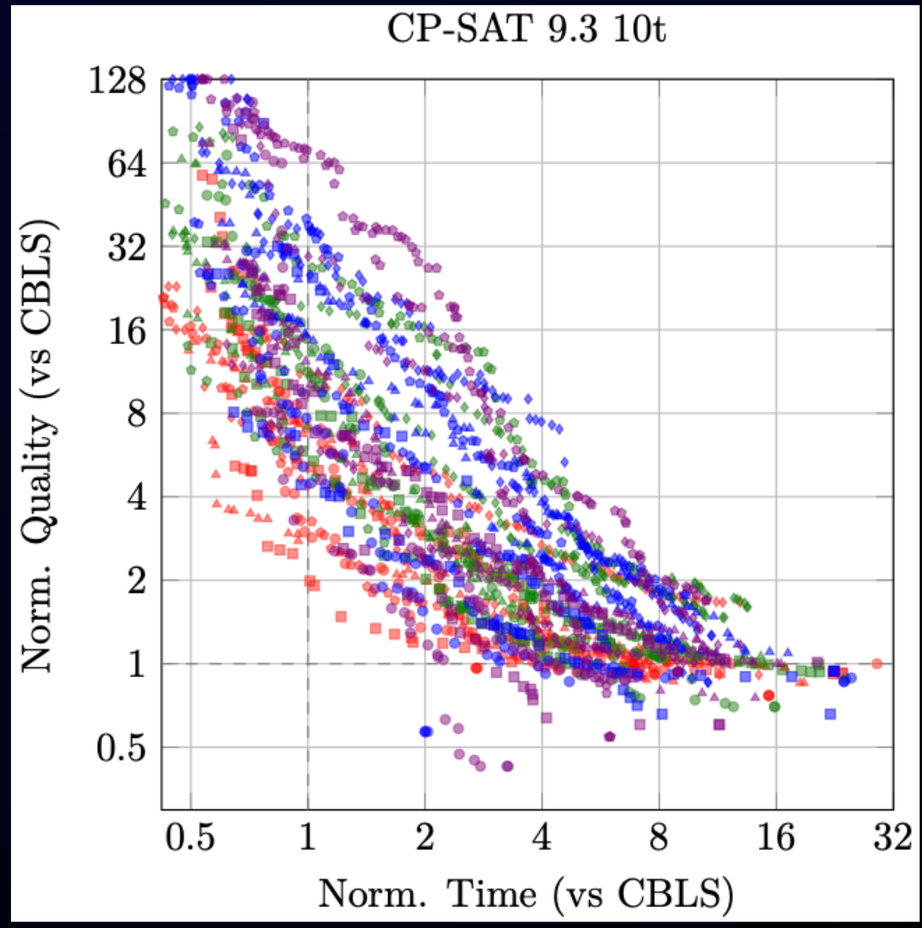
- 8 to 16 workers
- 10 to 16 hours store opening hours
- 5 or 15 minutes block length
- Three tasks to optimise
- One task is most constrained
- Includes lunch, breaks, ...

`https://github.com/optischedule/work-task-variation-instances`

CP-SAT

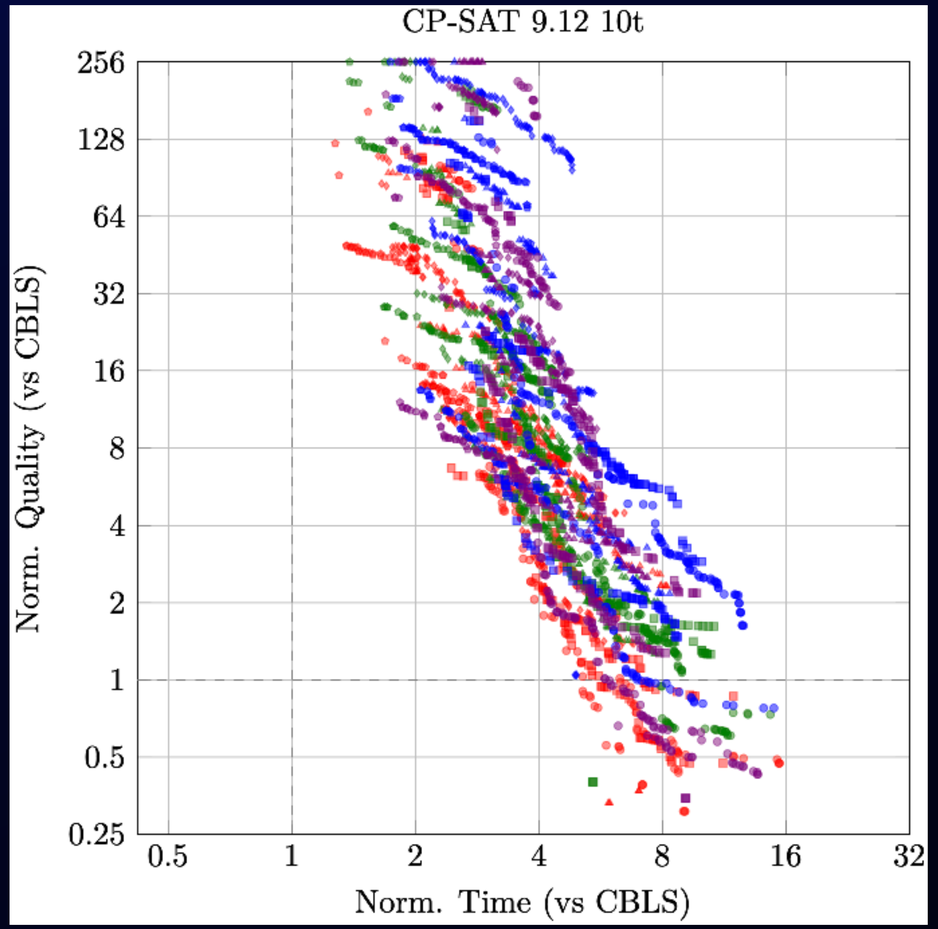
15 minute block size

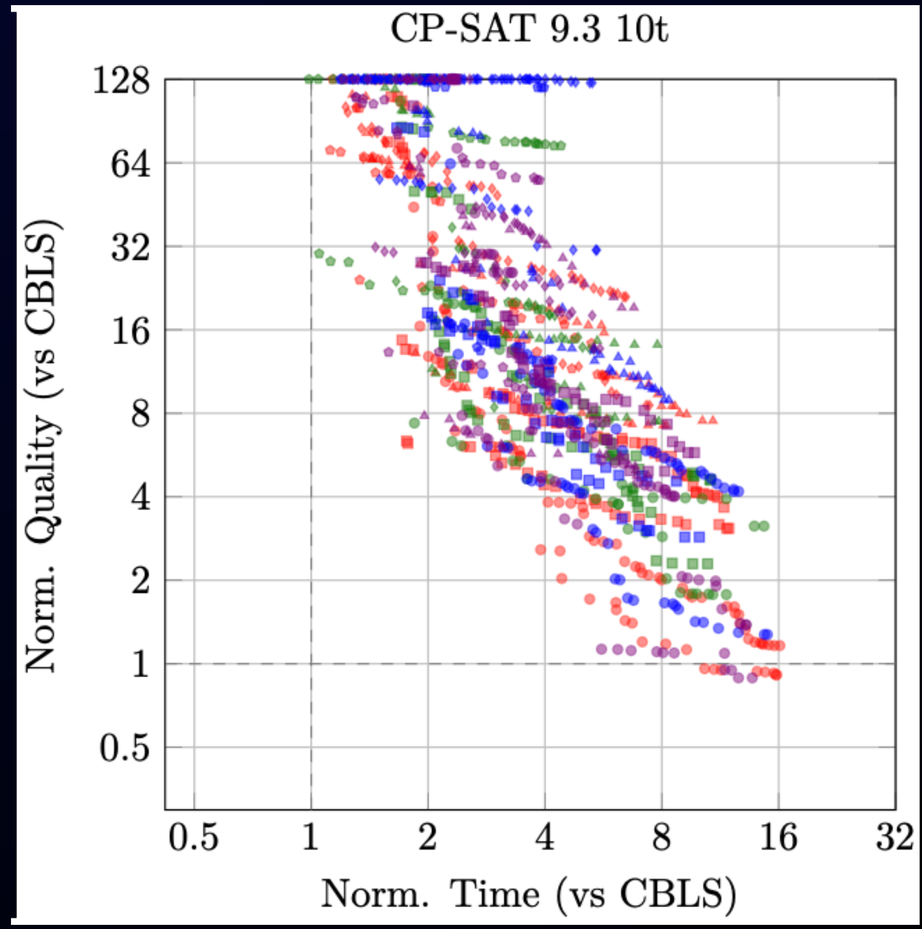




CP-SAT

5 minute block size





The Work Task Variation Problem

What have we learned?

- CP technology has improved rapidly
  - Still, lots to do for plug-and-play usability
- Writing you own solver is fun, and sometimes useful
  - Full control and customisation key features
- WTV useful problem for better work-days
- Should be more common in planning systems
- Fun new benchmark to play with

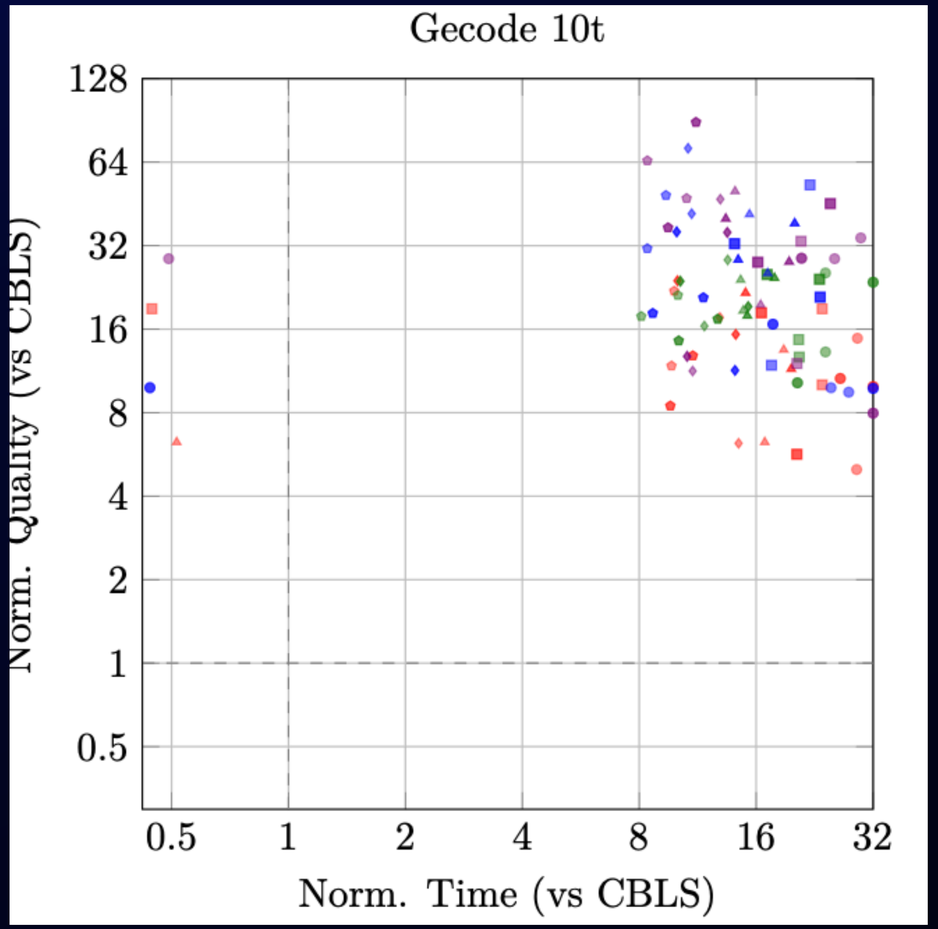
MiniZinc Model

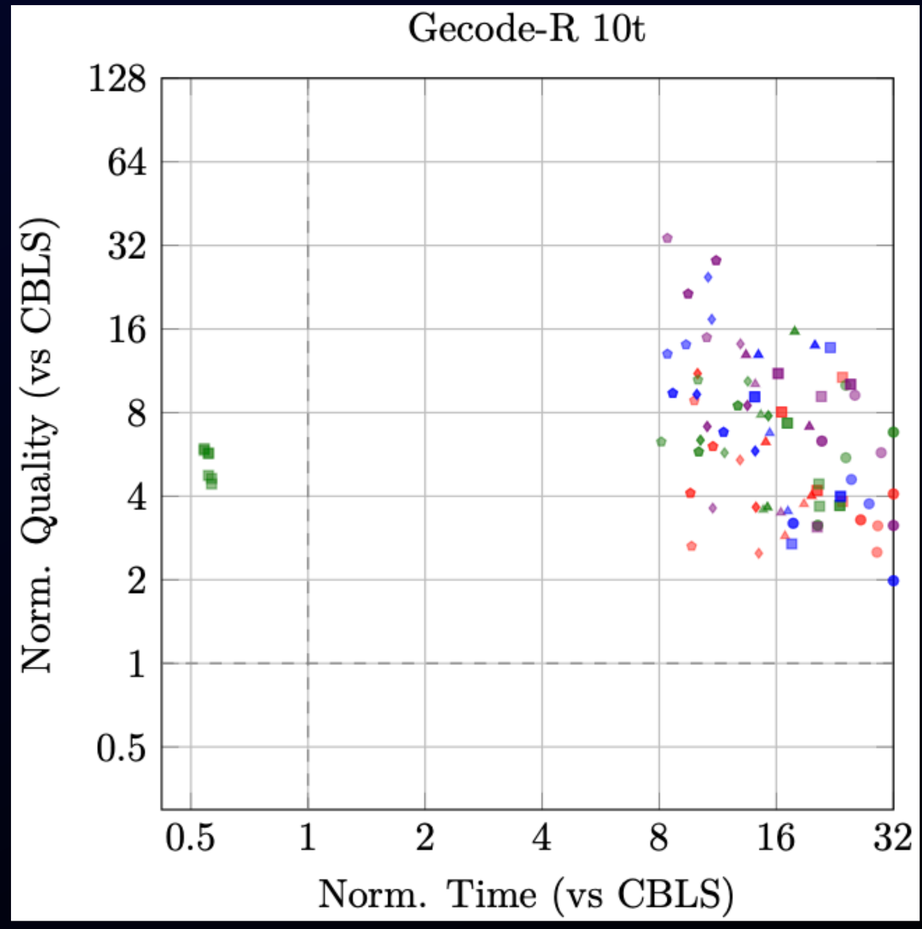
Specifying and Solving using Constraint Programming

- Full model in paper
- Planning block structure gives nice matrix schedule
- Requirements are `global cardinality` constraints
- Cost based on runs of is kind of messy
- Testing different systems over time

Gecode

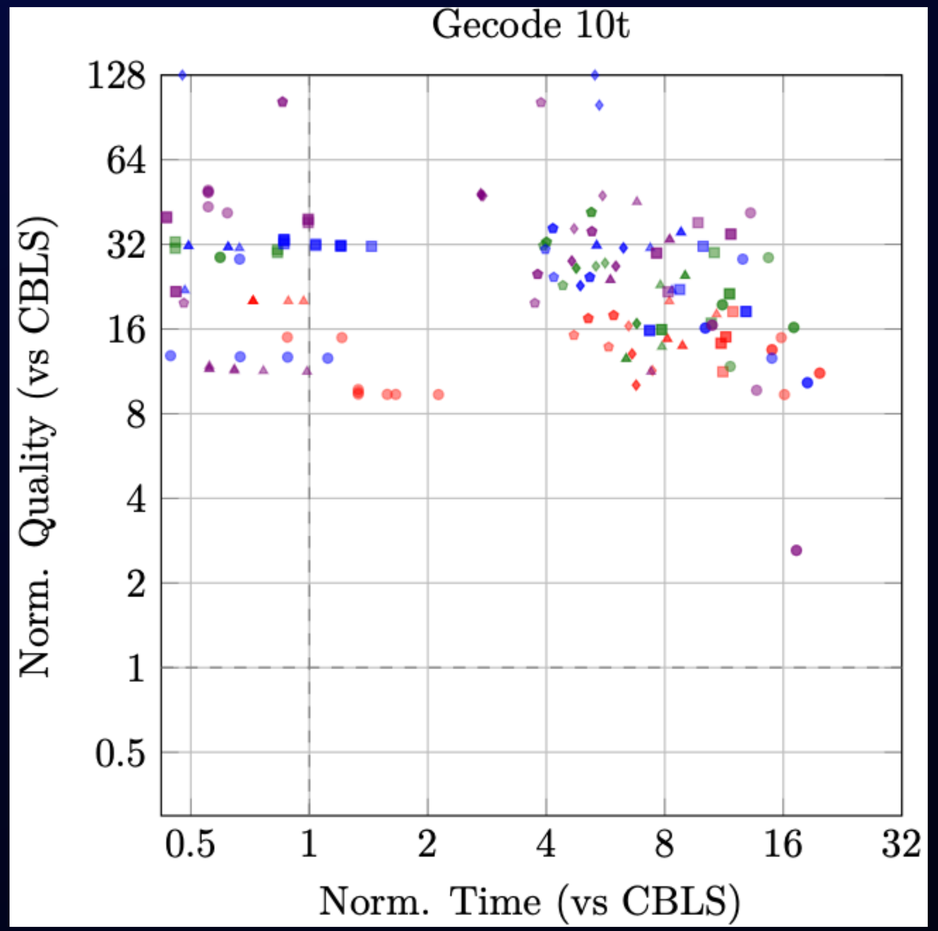
15 minute block size

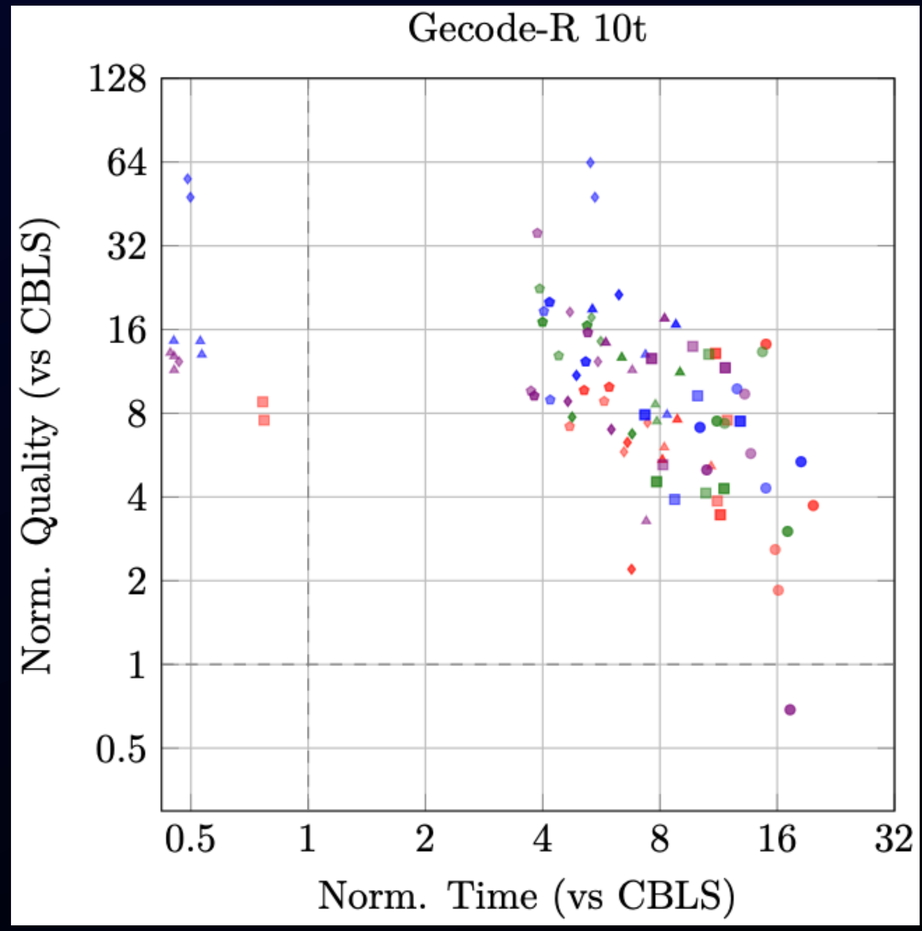




Gecode

5 minute block size





Constraint Programming News

14 Aug 2025

Breaking: MiniZinc Challenge Results!

The results of the MiniZinc Challenge 2025 were releaseed! New breaking results results on the famous Work Task Variation problem problem , in, will this change everything? OR Tools CP SAT dominates, but Chuffed is close by and Gecode does well.

MiniZinc Challenge Results

Score area ranking

- OR Tools wins
  - Both LCG and LS!
- Chuffed does well
- Gecode ok
- Many solvers crashed 🤔
  - Atlantis, CBC, CP Optimizer, CPLEX, Gurobi, HiGHS, Huub, iZPlus, Pumpkin, Scip, yuck

Solver	Score	Score Incomplete	Score Area
TOTAL	285	313	27773483.299
or-tools_cp-sat-par	44.64	43.50	395191.68
or-tools_cp-sat_ls-par	35.00	39.00	1985256.15
chuffed-free	32.60	35.50	2299944.69
or-tools_cp-sat_ls-free	30.50	34.50	2344757.53
gecode-par	29.50	33.50	2646583.75
jacop-free	25.50	29.50	3019617.32
fzn_picat_sat-free	31.76	34.00	3037304.39
choco-solver__cp-sat_-par	19.00	22.00	3433271.94
choco-solver__cp_-par	15.00	16.00	4287235.58
sicstus_prolog-free	21.00	25.00	4324320.27

MiniZinc Model

Data model

```
enum Resources;
enum Activities;

enum ActivitiesOrNone = A(Activities) ++ { None };

int: slots;
set of int: Slots = 1..slots;
set of int: SlotsAndZero = 0..slots;

array[Activities, Slots] of 0..card(Resources): requirements;

array[Resources, Slots] of opt ActivitiesOrNone: fixed;

array[Activities, SlotsAndZero] of int: activity_run_cost;
array[Activities, SlotsAndZero] of int: activity_frequency_cost;
```

MiniZinc Model

Variables

```
% The actual schedule, what activities are done when for each resource
array[Resources, Slots] of var ActivitiesOrNone: schedule;

% Markers for when runs end
array[Resources, Slots] of var bool: run_end;

% Length for each run at the current slot from the currents runs start
array[Resources, Slots] of var SlotsAndZero: run_length;

% Cost for each run at the end of a run with zero cost in the middle of runs
array[Resources, Slots] of var int: run_cost;

% Cost for number of runs of each activity per resource
array[Resources, Activities] of var int: frequency_cost;

% The total cost of runs
var int: cost = sum(run_cost) + sum(frequency_cost);
```

MiniZinc Model

Requirement constraints

```
% All shifts are only Activities (that is, not None) and surrounded with None
constraint forall (r in Resources) (
    regular(schedule[r, ..], "None* [^None]* None*")
);

% Always respect the requirements for each slot (column in the schedule)
constraint forall (s in Slots) (
    global_cardinality(schedule[.., s], ActivitiesOrNone, extended_requirements[.., s])
);

% Always respect the fixed requirements
constraint forall (r in Resources, s in Slots where occurs(fixed[r, s])) (
    schedule[r, s] = deopt(fixed[r, s])
);
```

MiniZinc Model

Cost constraints

```
% Mark when runs end
constraint forall (r in Resources, s in Slots) (
    if s = slots then
        run_end[r, s] = true
    else
        run_end[r, s] = (schedule[r, s] != schedule[r, s+1])
    endif
);

% Count length of runs
constraint forall (r in Resources, s in Slots) (
    if s = 1 \/ run_end[r, s-1] then
        run_length[r, s] = 1
    else
        run_length[r, s] = run_length[r, s-1] + 1
    endif
);
```

MiniZinc Model

Run length constraints

```
% Mark when runs end
constraint forall (r in Resources, s in Slots) (
    if s = slots then
        run_end[r, s] = true
    else
        run_end[r, s] = (schedule[r, s] != schedule[r, s+1])
    endif
);

% Count length of runs
constraint forall (r in Resources, s in Slots) (
    if s = 1 \/ run_end[r, s-1] then
        run_length[r, s] = 1
    else
        run_length[r, s] = run_length[r, s-1] + 1
    endif
);
```

MiniZinc Model

Cost computation

```
% Count run costs
constraint forall (r in Resources, s in Slots) (
    if run_end[r, s] then
        run_cost[r, s] = extended_activity_run_cost[schedule[r, s], run_length[r, s]]
    else
        run_cost[r, s] = 0
    endif
);

%Count frequency costs
constraint forall (r in Resources, a in Activities) (
    let {
        var int: activity_run_count = count(s in Slots) (
            run_end[r, s] /\ schedule[r, s] = A(a)
        )
    } in
    frequency_cost[r, a] = activity_frequency_cost[a, activity_run_count]
);
```

RosterLogic Variation

Plot