No More Awkward Silences with Table Talk Tuning

celebrating ideas

Martin Butler · Mikael Z. Lagerkvist

ModRef 2026

What will guests talk about?

○ STRUCTURALLY FEASIBLE

The room works

- capacities hold
- together/apart requests hold
- table balance stays within bounds

□ SOCIALLY ACCEPTABLE

The table works

- each guest has shared ground
- a strict majority can sustain the topic
- protect the weakest-served guest first

Strict majorities at Table 19

TABLE 19 · INTEREST SCORES

bold = personal favourite

	JAZZ	HIKING	FOOD	GARDEN	TRAVEL	GAMES
Mina	**9**	3	5	4	1	2
Theo	**8**	6	5	4	2	1
Leila	**7**	5	4	3	1	2
Oskar	1	4	**9**	2	2	1
Petra	2	2	1	**8**	1	1
TABLE FLOOR	**7**	**4**	**5**	**4**	**1**	**1**

Different topics for different guests

JAZZ
7
Mina · Theo · Leila

FOOD
5
Oskar

GARDEN
4
Petra

Each guest keeps their best supported topic

Structure first; conversation breaks ties

later levels break ties only

1. **Minimise total slack**
2. **Minimise maximum table slack** — STRUCTURE
3. **Minimise total imbalance**
4. **Maximise minimum guest interest**
5. **Maximise total guest interest** — CONVERSATION

The model stays close to that definition

$$b(t,k) = \text{strict-majority order statistic of the seated guests' scores}$$

$$q(g,t) = \max_{k} \min\{b(t,k), s(g,k)\}$$

1. Take the strict-majority floor for each topic.
2. Clip it by the guest's own interest.
3. Keep the best supported topic.

Detail: one scalar preserves all five levels

```
int: total_interest_weight = 1;
int: min_interest_weight = total_max_possible_interest + 1;
int: imbalance_weight =
  (max_possible_interest + 1) * min_interest_weight;
int: max_slack_weight =
  (max_possible_imbalance + 1) * imbalance_weight;
int: total_slack_weight =
  (max_max_slack + 1) * max_slack_weight;

var int: goal =
    total_slack * total_slack_weight
  + max_slack * max_slack_weight
  + total_imbalance * imbalance_weight
  + min_interest_penalty * min_interest_weight
  + total_interest_penalty;
```

Each base is one larger than the complete range of all lower-priority digits.

MiniZinc Model Results

ANYTIME RESULTS

Gecode + LNS

strongest seating quality

BASE MODEL

Huub best configuration

PROOF COVERAGE

10 / 45

instances closed by any configuration

PROOF SPECIALISTS

Huub · CP-SAT

Portable MiniZinc model · 60-second budget

Sequential optimisation in native Gecode?

> **Optimise one component. Prove it. Fix it. Move to the next.**
>
> The objective is lexicographic; why not make the native search lexicographic too?

Sequential search makes the conversation terms wait

○ LEVELS 1–3

Optimise · prove · fix

- total slack
- maximum slack
- imbalance

□ LEVELS 4–5

Still waiting

- minimum interest
- total interest

A 60-second budget can disappear in the first three proofs.

Scalar and direct lexicographic usually win

SAME NATIVE GECODE MODEL • SAME CONSTRAINTS • SAME FIVE OBJECTIVES

1

Scalar

one mixed-radix value

priority encoded in weights

2

Direct lexicographic

one five-value vector

priority remains visible

3

Sequential

one level at a time

prove before moving on

Portfolio scalar and direct lexicographic are strongest most often

Model shared ground; search decides how far we get

- Model shared conversational ground without pretending to predict conversation.
- Use Gecode LNS for strong seatings; Huub and CP-SAT help prove them on the standard model.
- Native Gecode shows that sequential optimisation is natural —but not automatically best.