**Step 6 – Parameter tuning with OR-Tools**

**What we did (method)**

1. **Built a small tuning grid** to explore solver behavior:
   * Time limits: **10 / 30 / 60 s**
   * Metaheuristics: **GLS** (Guided Local Search) and **TABU**
   * Vehicle penalty **V**: **0** (minimize distance only) and **10 000** (strongly discourage extra vehicles)
   * Tag format: TL{time}\_M{meta}\_V{penalty} (e.g., TL60\_MGLS\_V0).
2. **Ran every configuration on all Solomon VRPTW instances** using scripts/sweep\_ortools.py.  
   For each tag the script:
   * Called vrptw\_ortools.py --all …
   * Saved raw logs (run\_stdout.txt, run\_stderr.txt), a summary.csv, and JSON route files in data/benchmarks/<TAG>/.
3. **Aggregated and analyzed outputs** with scripts/aggregate\_benchmarks.py:
   * Concatenated all summary.csv files into data/reports/benchmarks\_all\_runs.csv.
   * Added columns (tag, time\_limit, meta, vehicle\_cost, family).
   * Defined a **unified objective** for selecting winners:  
     objective\_ort = dist\_ort + vehicle\_cost \* vehicles\_ort.
   * For each instance chose the **best tag** (minimal unified objective).
   * Compared this “best-of-sweep” against the greedy **baseline** (Step 4).
   * Wrote tables + plots into data/reports and data/figures.

**Why we did it**

* To **choose a default configuration** for our solver with evidence (time limit, metaheuristic, and how strongly to penalize vehicles).
* To **quantify gains** over the greedy baseline on the same 56 Solomon instances.
* To understand **trade-offs**: distance vs number of vehicles, and how results vary by instance family (**C**, **R**, **RC**).

**What we got (key results)**

**Feasibility / coverage**

* All 12 configurations show **100% success rate** on the runs captured in the OR-Tools summaries; each tag has **50 feasible instances recorded** (the count column in the means table).  
  *Note:* our copies of summary.csv contain 50 rows per tag; if you want success rate out of all 56, we’ll extend the aggregator to treat “missing from summary” as infeasible and join against the full manifest.

**Average performance by tag** (feasible only; lower distance is better)

* Best two by mean distance:  
  TL60\_MTABU\_V0 ≈ **1022.1** dist, **10.08** vehicles (count 50)  
  TL60\_MGLS\_V0 ≈ **1022.3** dist, **10.08** vehicles (count 50)
* Adding a **vehicle penalty (V=10000)** slightly **reduces vehicles** (~9.78–9.86 avg) but **increases distance** (~1030–1054 avg).  
  (See data/reports/means\_by\_tag.csv.)

**Best-of-sweep vs baseline (percent distance improvement, + is better)**

* The distribution by family shows **large gains**:
  + **R** and **RC** families center around **≈30%** improvement.
  + **C** family centers a bit lower (≈ **25%**), but still strong.  
    (See data/figures/best\_vs\_base\_box\_by\_family.png.)

**Top individual improvements** (best configuration chosen per instance)

* RC206: **+38.0%** with TL60\_MGLS\_V0 (60s, GLS, V=0)
* R206: **+35.7%** with TL30\_MGLS\_V0
* R203: **+35.6%** with TL60\_MGLS\_V0
* C108: **+34.7%** with TL10\_MGLS\_V0
* R103: **+33.8%** with TL60\_MGLS\_V0  
  …and more in data/reports/step6\_summary.txt (top-10 list).

**What the figures/tables show**

* data/figures/success\_rate\_by\_tag.png and feasible\_count\_by\_tag.png: every tag solved all recorded instances (bars at **100%** and **50** respectively).
* data/figures/best\_vs\_base\_box\_by\_family.png: best configuration per instance vs baseline, grouped by C / R / RC—**consistent 20–35%** savings, with **R/RC slightly higher** than **C**.
* Full CSVs are in data/reports/… for reproducible stats:
  + benchmarks\_all\_runs.csv (all raw runs)
  + success\_rate\_by\_tag.csv, success\_rate\_by\_family\_tag.csv
  + means\_by\_tag.csv
  + best\_method\_per\_instance.csv
  + best\_vs\_baseline.csv
  + Narrative summary: step6\_summary.txt

**Interpretation & decision**

* **Time** helps: 60 s tags (GLS or TABU) give the best mean distance.
* **Vehicle penalty** (**V=10000**) trades distance for fewer vehicles; if the business objective is **min vehicles**, keep a penalized tag; if it’s **min distance**, prefer **V=0**.
* **Recommended default for distance**: **TL60\_MGLS\_V0** (stable top-tier mean, appears in many top improvements).  
  **Recommended default when fleet size is critical**: **TL60\_MGLS\_V10000**.

**Files created in Step 6**

* Per-tag folders: data/benchmarks/<TAG>/ with logs, summary.csv, and JSON solutions.
* Reports:  
  data/reports/benchmarks\_all\_runs.csv, success\_rate\_by\_tag.csv, success\_rate\_by\_family\_tag.csv, means\_by\_tag.csv, best\_method\_per\_instance.csv, best\_vs\_baseline.csv, step6\_summary.txt
* Figures:  
  data/figures/success\_rate\_by\_tag.png, feasible\_count\_by\_tag.png, best\_vs\_base\_box\_by\_family.png

**Small caveat (to note in the report)**

Our sweep copies the solver’s **summary.csv** after each run. In your runs that file had **50 rows** per tag. That’s why the success plots show **100%** and **50 feasible** per tag. If you want the success rate against **all 56** instances, we can adjust the aggregator to join against the manifest and count any **missing** instances as **infeasible**.

**Step 6 — Configuration sweep & aggregation (abstract)**

We performed a controlled sweep over key OR-Tools settings—time limits {10,30,60}s, metaheuristics {GLS,TABU}, and vehicle-penalty {0,10000}—using sweep\_ortools.py. Each run wrote a full summary.csv plus JSON solutions under data/benchmarks/<TAG>/. We then aggregated everything with aggregate\_benchmarks.py, producing combined tables (benchmarks\_all\_runs.csv, means\_by\_tag.csv, best\_method\_per\_instance.csv, best\_vs\_baseline.csv) and figures (success\_rate\_by\_tag.png, feasible\_count\_by\_tag.png, best\_vs\_base\_box\_by\_family.png). This step identifies, per instance, the best-performing tag (objective or distance trade-offs) and shows global trends (longer time limits improve feasibility and distance; vehicle penalty trades distance vs fleet), setting us up to freeze a single best configuration in Step 7 (which we then used to generate the final deterministic reference plans).