BCPC Pipeline

A reproducible workflow to sketch city–to–city rail corridors

Miguel Ibrahim et al.
https://github.com/MiguelIbrahimE/Train_Scheduler

Last built: July 4, 2025

Abstract

BCPC ("*Bring Cities Back to the People, not the Cars*") is a PythonPoetry project that assembles open data, lightweight demand models and mixed-integer optimisation to test the *first-order* feasibility of new rail links. The codebase is intentionally compact (~1k LOC) yet modular, allowing one to swap in richer sub-models—or run the whole workflow from a Jupyter notebook—with minimal glue code.

1 Motivation

Motorisation rates in many middle-income countries are rising faster than road capacity, causing congestion, pollution and social inequity. While full railway master plans require years of engineering, governments and NGOs still need *provisional* figures for *"how much would a 150 km line cost and carry?"*. BCPC fills that gap with an open, scriptable tool-chain.

2 Source code layout

3 Data sources

- **OpenStreetMap** via `osmnx` – city admin boundaries, rail/road graphs.
- **OpenTopography GlobalDEM API** – SRTMGL1_E 30 m and ASTER v3 fallback.
- **Wikipedia** – scraped catalogues of high-speed lines and trainsets.
- User-provided CSV with *population*, *tourism_index*, optional *daily_commuters*, *terrain_ruggedness*, etc.

All downloaded artefacts (boundaries, GeoTIFFs, catalogs) are cached under `data/_cache/` for deterministic reruns offline.

File	**Responsibility**
`scenario_io.py`	CSV validation, light NLP (Unicode → ASCII), budget forward-fill.
`city_pipeline.py`	End-to-end execution for *one* city row – boundary fetch, DEM, routing, costs.
`terrain.py`	DEM download (OpenTopography API), caching, slope / ruggedness utilities.
`routing.py`	Terrain-aware A* on rail network, road fallback.
`demand.py`	Quick commuter + discretionary demand estimator.
`optimise.py`	OR-Tools MIP (with greedy fallback) – choose track gauge, rolling stock, fleet size.
`catalog_fetch.py`	Scrape WIKIPEDIA lists to build rolling-stock & track catalogues in YAML.

Table 1: Core modules (~350 LOC, excluding comments and tests).

4 Pipeline walk-through

1. **Read scenario.** The CLI accepts `-csv <file>` pointing at a UTF-8 table. Missing commuters or ruggedness values are imputed on-the-fly.

2. **Boundary + DEM.** `enrich.get_city_boundary()` calls Nominatim once, stores the polygon as GeoPackage. The bounding-box is enlarged by 0.05° and passed to `terrain.load_dem()`, which either returns a 30 m raster or a 1×1 "flat" DEM if OpenTopography is offline.

3. **Routing.** Two extreme points on the city ring are chosen via a vectorised farthest-pair heuristic. `routing.trace_route()` snaps them to OSM rail; if no rail path exists it switches to `network_type = "drive"`.

4. **Demand & optimisation.** The daily boardings (per direction) feed `optimise.optimise_design()`, a mixed-integer model that decides: (i) metric vs standard gauge, (ii) diesel vs EMU, (iii) how many trainsets. Output is a `NetworkDesign` dataclass.

5. **Cost breakdown.** Track km, stations and fleet go through `cost.estimate_cost()`—producing both a human-friendly log line and JSON-serialisable object.

6. **Export.** Edges are converted to WGS84 GeoJSON for quick drag-and-drop into QGIS or Leaflet. If the budget is too small a plain text file explains why.

5 Command-line usage

```
# 1. Install (inside conda or python -m venv)
poetry install

# 2. Run the Lebanon demo (8 cities, global budget
   ffilled down the CSV)
poetry run bcpc --csv input/lebanon_cities_2024.csv

# 3. Inspect artefacts
open output/*.geojson   # macOS; on Linux use 'xdg-open'
```

6 Extending BCPC

- **Better demand.** Swap `demand.py` with a mobile-phone OD matrix; the optimiser API takes only a scalar *ppd* so the coupling is loose.
- **Real corridor geometry.** Replace the “1 km star” stub in `_demo_star_graph()` with a multi-segment line once the route finder matures.
- **Country-wide allocation.** The forward-fill on `budget_total_eur` already lets you enforce a single cap-ex pot. A future step can solve an outer-level knapsack: *which subset of cities should be funded to maximise pax-km?*

7 Limitations

1. Only curvature and slope checks are applied; tunnels, bridges and land acquisition are *not* modelled.
2. Cost coefficients are EU 2019 averages—please localise.
3. Wikipedia catalogs occasionally change column headers; the scraper may need tweaks.

8 Conclusion

BCPC is *not* a replacement for a full feasibility study, but it bridges the gap between Excel guesswork and $500 k consultancy reports. Its entire dependency stack is open-source; every run is reproducible end-to-end in <2 minutes on a laptop.