Branch-and-Price for the Traveling Tournament Problem (TTP):

Implementation Notes for `BNP_NLX.py`

September 28, 2025

1 Problem statement and notation

Let the set of teams be $T = \{1, \dots, n\}$ and the set of time slots for a double round-robin be $S = \{1, \dots, 2(n-1)\}$. Distances are given by a matrix $D = (d_{ij})_{i,j \in T}$. Each team plays every other team twice (home/away), subject to *break constraints* limiting consecutive home/away streaks to the interval $[L, U]$. Optional *non-repeater constraints* (NRCs) forbid immediate home-and-away reversals across consecutive slots.

A *tour* for team t is a feasible assignment of opponents and home/away decisions over all slots. In the branch-and-price (B&P) scheme, tours are the *columns*.

2 Dantzig–Wolfe decomposition and the the RMP

Let $\lambda_{t,p} \in [0,1]$ denote whether tour p (from the current pool) is selected for team $t \in T$. The Restricted Master Problem (RMP) is a set-partitioning model:

$$\min_{\lambda} \quad \sum_{t \in T} \sum_{p \in \mathcal{P}_t} c_{t,p}\, \lambda_{t,p} \tag{1}$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}_t} \lambda_{t,p} \;=\; 1, \qquad \forall t \in T \qquad \text{(one tour per team)} \tag{2}$$

$$\sum_{p \in \mathcal{P}_t} a_{t,p,s}\, \lambda_{t,p} \;=\; 1, \qquad \forall (t,s) \in T \times S \qquad \text{(slot coupling)} \tag{3}$$

$$\sum_{t \in T} \sum_{p \in \mathcal{P}_t} b_{t,p,i,s,j}\, \lambda_{t,p} \;\le\; 1, \qquad \forall (i,s,j) \qquad \text{(NRCs, separated)} \tag{4}$$

$$0 \le \lambda_{t,p} \le 1 \qquad \text{(LP; IP solves for UB).}$$

Here $a_{t,p,s} \in \{0,1\}$ indicates that tour p for team t plays in slot s, and $b_{t,p,i,s,j} \in \{0,1\}$ aggregates the NRC pattern "i hosts j at s and j hosts i at $s{+}1$" on a tour p.

Code map. RMP construction/solve is handled by `MasterLP`. Dynamic separation of NRCs (4) is implemented by `separate_nrc_cuts`. Columns are inserted when pricing returns negative reduced-cost tours.

3 Duals and reduced costs

Let π_t be the dual of (2) for team t, let $\mu_{t,s}$ be the dual of (3) for (t, s), and let $\beta_{i,s,j}$ be the dual of an active NRC inequality for (i, s, j). The reduced cost of a tour $p \in \mathcal{P}_t$ is

$$\tilde{c}_{t,p} \; = \; c_{t,p} - \pi_t - \sum_{s \in S} \mu_{t,s}\, y_{t,p}(s) - \sum_{(i,s,j)} \beta_{i,s,j}\, z_{t,p}(i, s, j), \tag{5}$$

where $y_{t,p}(s) \in \{0, 1\}$ captures the slot usage of p, and $z_{t,p}(i, s, j) \in \{0, 1\}$ captures NRC patterns induced by p. A negative $\tilde{c}_{t,p}$ indicates a profitable column to add.

Arc-based accounting. Rather than compute (5) at the tour level, the code pushes dual terms onto *arcs* of a time-expanded network for team t and solves a shortest path with resource constraints (SPPRC). For an arc used at slot s from state u to v with travel d_{uv}, the arc's reduced cost is

$$\tilde{c}^{\mathrm{arc}}_{t,s}(u \to v) \; = \; d_{uv} \; - \; \mu_{t,s} \; - \; (\beta\text{-term if an NRC pattern is triggered}). \tag{6}$$

The implementation uses integer-scaled duals (e.g., `SCALE`$= 10^6$ via `to_int_dual`) to avoid numerical tolerance issues.

Code map. Arc costs with (μ, β) are built in `build_rc_edges_with_beta_int`; forward/backward DP potentials in `rc_forward_dp_int` and `rc_backward_dp_int`; labeling-based pricing in `pricing_single_best_`

4 Pricing network and resource feasibility

The time-expanded network for a fixed team t includes nodes encoding slot index, venue/opponent, break-streak state, and a bitmask of visited opponents. The SPPRC enforces:

- **Elementarity:** each opponent appears at most once from t's away perspective (bitmask resource).
- **Break limits:** $L \leq$ consecutive home/away streak $\leq U$; streak resource is updated along arcs.
- **Branching filters:** forced home/away and include/exclude specific games are implemented by removing or forcing arcs in the relevant slots.
- **Symmetry reduction:** at the middle boundary between halves (slot $n-1$), away opponents are restricted (e.g., "opponent index $< t$"), removing mirror-symmetric solutions; see the `symmetry_middle` option in `build_rc_edges_with_beta_int`.

5 Dynamic NRC separation

After each RMP solve, the fractional λ-solution is projected to slot-level quantities to form $y(i, s, j)$ values. Any violated NRC of type

$$y(i, s, j) + y(j, s+1, i) \; \leq \; 1 \tag{7}$$

is added to the RMP, and the LP is re-solved. Duals β from these cuts then feed into arc reduced costs via (6).

Code map. `separate_nrc_cuts`.

6 Exact pruning in pricing via bidirectional DP

Let $fw[(s,u)]$ denote the best reduced cost from the source to node (s,u), and $bw[(s,u)]$ the best reduced cost from (s,u) to the sink, computed on the *arc-reduced-cost* network ignoring resource conflicts. For an arc $(s, u \to v)$ with cost $r = \tilde{c}^{\text{arc}}_{t,s}(u \to v)$, if

$$fw[(s,u)] \;+\; r \;+\; bw[(s{+}1,v)] \;-\; \pi_t \;\geq\; 0, \tag{8}$$

then no completion through this arc can produce a negative reduced-cost tour and the arc may be safely eliminated *before* labeling. The code computes integer fw, bw via `rc_forward_dp_int`/`rc_backward_dp_int` and prunes arcs accordingly; the remaining graph is then searched by labeling.

7 Column generation loop

The overall CG loop (see `column_generation_with_branch`) is:

1. Solve the RMP (LP) and extract duals (π, μ, β).
2. For each team $t \in T$, run `pricing_single_best_int` to search for a tour with $\tilde{c}_{t,p} < 0$.
3. If any negative columns are found, add them to the RMP and return to step 1.
4. If none are found, solve the RMP as an IP to get a feasible schedule (upper bound, UB), and then apply reduced-cost fixing (Proposition A) to shrink the RMP.

8 Reduced-cost fixing (Proposition A)

Let LB be the current RMP value and UB the best incumbent. Any column with integer reduced cost $\tilde{c} \geq (\text{UB} - \text{LB})$ can be fixed at $\lambda = 0$ without losing optimality. The implementation computes integer $\tilde{c}$ for active columns and tightens variable bounds; see `apply_proposition_A`.

9 Branching and strong branching

If the LP solution remains fractional after CG and cut separation, the algorithm performs best-bound B&P search (see `branch_and_price_bestbound`). The branching rules are compatible with pricing:

- **Include/Exclude event:** force or forbid a specific game (home i vs j at slot s).
- **Force H/A at slot:** for a given (t, s), force home or away.

These are enforced in the RMP by equalities/zero-sums over λ columns and respected in pricing by filtering arcs in the affected slots (see `ensure_branch_feasibility` and `build_rc_edges_with_beta_int`). Candidate selection uses event strong branching (`rank_fractional_events`) with fallback to home/away (`pick_fractional_HA`).

10 Initialization (seeding) and symmetry

To start CG with a feasible RMP, the code builds canonical round-robin schedules (Berger tables; `generate_single_rr`, `generate_double_rr`), constructs team tours obeying break limits (`build_seed_tours`), and applies light local improvements (2-opt within halves; `improve_team_tour_by_2opt`). The option `symmetry_middle` restricts the opponent set at the half boundary to kill mirrored solutions early.

11 Bounds, search, and termination

At each node, the LP objective provides a lower bound (LB). Solving the RMP integrally yields feasible schedules and updates the global UB. Nodes with LB $\geq$ UB are fathomed. The search terminates when the open-node list is empty or when LB = UB. Final schedules and travel statistics are reported via `print_solution`.

12 Compact mathematical formulation (self-contained)

For a *compact*, cross-reference-free summary (avoids `??` if you compile once), we give a re-stated master and pricing with local tags:

$$\min_{\lambda} \quad \sum_{t \in T} \sum_{p \in \mathcal{P}_t} c_{t,p} \, \lambda_{t,p} \tag{M0}$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}_t} \lambda_{t,p} \; = \; 1, \qquad \forall t \in T \qquad \text{(one tour)} \tag{M1}$$

$$\sum_{p \in \mathcal{P}_t} a_{t,p,s} \, \lambda_{t,p} \; = \; 1, \qquad \forall (t,s) \in T \times S \qquad \text{(coupling)} \tag{M2}$$

$$\sum_{t \in T} \sum_{p \in \mathcal{P}_t} b_{t,p,i,s,j} \, \lambda_{t,p} \; \leq \; 1, \qquad \forall (i,s,j) \qquad \text{(NRC)} \tag{M3}$$

$$0 \leq \lambda_{t,p} \leq 1. \tag{M4}$$

Tour reduced costs and arc reduced costs for pricing (team t) are:

$$\tilde{c}_{t,p} = c_{t,p} - \pi_t - \sum_{s \in S} \mu_{t,s} \, y_{t,p}(s) - \sum_{(i,s,j)} \beta_{i,s,j} \, z_{t,p}(i,s,j), \tag{P1}$$

$$\tilde{c}^{\text{arc}}_{t,s}(u \to v) = d_{uv} - \mu_{t,s} - \big(\beta\text{-term if NRC is triggered}\big). \tag{P2}$$

Bidirectional DP pruning condition:

$$fw[(s,u)] + \tilde{c}^{\text{arc}}_{t,s}(u \to v) + bw[(s{+}1,v)] - \pi_t \;\geq\; 0 \implies \text{prune arc } (s, u \to v). \tag{P3}$$

Reduced-cost fixing threshold (Proposition A): fix columns with $\tilde{c} \geq (\text{UB} - \text{LB})$.