POLICY-GUIDED MONTE CARLO TREE SEARCH FOR CAD-GROUNDED PCB ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learned printed circuit board (PCB) routing policies can produce suboptimal solutions on unseen boards. We propose PW-MCTS, a test-time search method that refines a pretrained routing policy through Monte Carlo tree search over native CAD operations. Board-wise iterated Bellman calibration aligns critic outputs with target-board returns. Engine determinization enables reproducible branch evaluation; incremental checkpointing, memoized branch-packed inference, and invalid-action filtering reduce search overhead. We evaluate PW-MCTS on a subset of PCBWorld-Bench, an open-source PCB dataset, using a policy trained only on synthetic boards. At equal draws, PW-MCTS raises the fraction of attempts that complete a board without error-severity design-rule violations from 42.2% to 63.2%, and mean routability from 0.742 to 0.885. Under matched rollout-time budgets after calibration, search raises board coverage from 84.7% to 89.3% with eight simulations per decision and roughly one-quarter as many trajectories as policy sampling. These results support test-time refinement by balancing search effort and repeated sampling, with attainable performance still shaped by the routing MDP and prior quality.

1 INTRODUCTION

In printed circuit board (PCB) design, *routing* determines conductive paths that connect components while satisfying geometric and electrical constraints. Under strict design-rule constraints, the objective is to minimize total wirelength within layers and the number of vias connecting different layers. Conventional routers combine heuristic search with local modifications, such as rip-up and reroute (RRR), to optimize routing solutions under hand-designed costs (Dees & Karger, 1982; Kahng et al., 2021; Lin et al., 2021). Because connections compete for shared routing resources, constructing or revising one route can alter the feasibility and cost of others. The challenge is therefore to assess how individual routing decisions affect the quality of the complete solution. Such effects depend on the board geometry and the evolving routing configuration, making them difficult to capture with fixed heuristic rules.

Learning-based routers seek to address this challenge by learning routing strategies, but differ in which routing decisions they control and at what spatial resolution. Learned RRR uses trained policies to select nets or regions for revision based on features such as the spatial distribution of design-rule violations (Gandhi et al., 2023; 2024; Keng et al., 2025), while net-ordering policies provide higher-level control over a detailed router's sequential, net-by-net routing by learning the net order (Qu et al., 2021; Lin et al., 2022; Zhou et al., 2025). Grid-based policies construct routes through local moves, requiring longer action sequences at finer spatial resolutions (Liao et al., 2020; He et al., 2022). The CAD-engine abstraction in PCBWorld (Song et al., 2026) has enabled policies to construct connections through native routing operations while considering geometric and design-rule constraints. In practice, however, learned policies can yield suboptimal routing solutions, especially on out-of-distribution boards, motivating additional search at test time. Our objective is to refine an RL prior learned in this CAD-grounded environment at test time, improving its routing decisions without retraining the policy. We evaluate both the gains over the same prior and their computational cost relative to repeated policy sampling.

Monte Carlo tree search (MCTS), guided by a pretrained policy and value function, can reassess routing decisions at test time by comparing how alternative actions affect subsequent connections

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

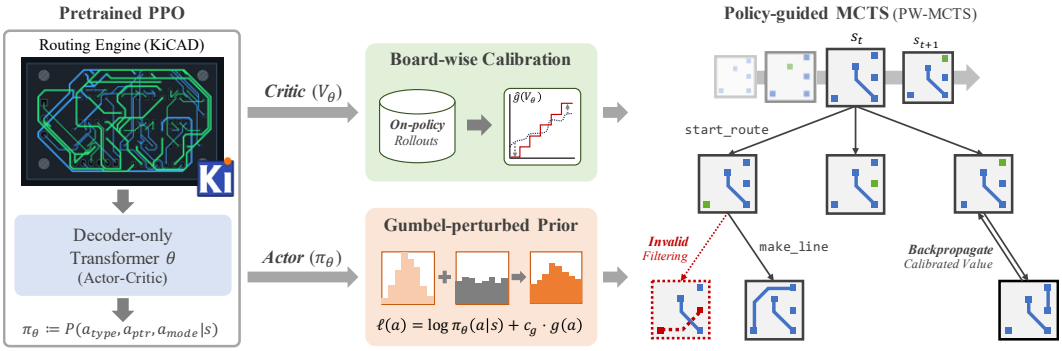


Figure 1: Overview of PW-MCTS. A pretrained policy and board-calibrated critic guide Gumbel MCTS over native CAD actions. At each routing state, search compares alternative continuations and executes the action selected at the root.

before committing to an action (Silver et al., 2017; Schrittwieser et al., 2020; Danihelka et al., 2022). Its interpretation as approximate regularized policy optimization in the AlphaZero family (Grill et al., 2020) provides a framework for instance-wise refinement without retraining the policy. Its effectiveness, however, depends on the search budget and the accuracy of value estimates (Hamrick et al., 2021). Prior work has applied MCTS to routing on grid-based representations (He et al., 2022). Searching through native CAD operations instead allows alternatives to be evaluated directly under the board's geometry and design rules. Realizing this approach requires managing unreliable leaf estimates under board and search-state distribution shifts, consistent restoration of routing states across search branches, and manageable costs for engine simulation, state restoration, and neural inference.

We propose PW-MCTS, a framework that refines pretrained routing policies at test time through search over PCBWorld's native CAD operations (Figure 1). The pretrained actor–critic guides the search, allowing alternative routing decisions to be evaluated in the context of the target board. PW-MCTS uses target-board policy rollouts to calibrate critic outputs to raw return units and anchors terminal values. We also determinize the CAD engine so that revisiting a search branch reproduces the same routing state. To make repeated simulations practical, incremental checkpointing reduces state-restoration overhead, memoized branch-packed inference reduces neural computation, and invalid-action filtering avoids further search along rejected branches.

We evaluate a policy trained only on procedurally generated boards on 50 open-source PCBs from PCBWorld-Bench (Song et al., 2026). At equal draws, PW-MCTS raises clean pass from 42.2% to 63.2% and routability from 0.742 to 0.885. Under matched-time budgets, smaller simulation budgets permit more candidates and improve coverage, while route quality varies with allocation and board difficulty. Checkpoint and training-seed variation show that refinement remains dependent on the learned prior. Search also operates within the actions, reward, and horizon of the routing MDP. These results motivate improving the underlying RL prior and the realism of its synthetic training data alongside the allocation of test-time computation.

The contributions of this work are as follows.

- **Test-time policy refinement for routing.** We formulate PW-MCTS over native CAD operations to refine a fixed pretrained routing policy. Board-wise iterated Bellman calibration and terminal anchoring adapt critic estimates for search on each target board.
- **Reproducible and efficient CAD-engine search.** We determinize PCBWorld to reproduce routing states across search branches and reduce search overhead through incremental checkpointing, memoized branch-packed inference, and invalid-action filtering.
- **Cost–benefit analysis on real boards.** On 50 open-source PCBs, paired equal-draw and matched-time evaluations characterize trajectory quality and post-calibration rollout cost. Analyses of search budgets, training checkpoints, invalid actions, and execution costs characterize the conditions under which search improves coverage and route quality.

2 Related Work

Learning-Based Routing. Learned routers differ in decision scope and action resolution. Grid-based methods extend connectivity cell by cell, requiring longer action sequences at finer spatial resolutions (Liao et al., 2020; He et al., 2022). Net-ordering policies learn routing sequences for a detailed router (Qu et al., 2021; Lin et al., 2022; Zhou et al., 2025). PCBWorld exposes native CAD operations for constructing connections under engine-level geometry and design-rule handling (Song et al., 2026), while world-model RL with Freerouting optimizes net ordering through imagined trajectories (Liao et al., 2026). Learned RRR selects nets or windows for revision based on routing-state features (Gandhi et al., 2023; 2024; Keng et al., 2025). Related strategic RRR evaluates potential routing changes before applying them (Devereux-Smith & Madden, 2025). Our focus is test-time refinement of pretrained policies through direct CAD-engine search.

Policy-Guided Tree Search. MCTS trades exhaustive enumeration against greedy selection within a fixed budget (Kocsis & Szepesvári, 2006; Browne et al., 2012), often guided by neural priors (Silver et al., 2016; 2017; Schrittwieser et al., 2020). Expert Iteration couples such search with policy learning (Anthony et al., 2017). Grill et al. (2020) interpret AlphaZero search as approximate regularized policy optimization, while Danihelka et al. (2022) use Gumbel-based selection and sequential halving for policy improvement with limited simulations. Planning does not always pay (Hamrick et al., 2021; Bapst et al., 2019), motivating work on safe improvement and inference cost (Castellini et al., 2023; Frankston & Howard, 2026; Fickinger et al., 2021). Tree search has also been applied to language model agents (Zhou et al., 2024; Yu et al., 2025; Li et al., 2025). In neural combinatorial optimization, efficient active search adapts a subset of model parameters at test time (Hottung et al., 2022). PW-MCTS keeps network weights fixed and evaluates CAD-engine search against policy sampling in trajectory quality and computational cost.

OOD-Aware Value Calibration. Dyna-style planning studies highlight value errors propagated through model-generated experience (Aminmansour et al., 2024; Abbas et al., 2020). Value-network overfitting also limits generalization across environments (Moon et al., 2022). CAD-grounded search uses engine transitions but can still reach states outside the critic's training distribution. Adaptive value normalization handles changing reward scales during learning (van Hasselt et al., 2016), but alone does not address target-board distribution shift. Iterated Bellman calibration maps value predictions to Bellman targets (van der Laan & Kallus, 2026). We apply calibration to target-board policy rollouts to recover raw return units.

3 Routing as Policy-Guided Search

3.1 MDP Formulation

Policy learning and MCTS share a finite-horizon MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ built on PCBWorld (Song et al., 2026). We retain its engine-backed state and transition interface while adapting the action space, potential, and discount factor. Appendix A collects the notation.

The input is a netlist $\mathsf{NL} = \{N_1, \ldots, N_{|\mathsf{NL}|}\}$ and immutable board data $\mathcal{B}$, including the board outline, copper layers, fixed obstacles, and design rules. Each net $N = \{p_1, \ldots, p_{|N|}\}$ comprises pads, with their geometry and layer information, that must be connected at the same electrical potential. A state $s \in \mathcal{S}$ is the PCBWorld board snapshot containing NL and $\mathcal{B}$ together with the mutable routing geometry, connectivity and design-rule feedback, and the active routing context, including the selected net and route head.

State quality is measured by the remaining connection count $\mathsf{U}(s)$, normalized wirelength $\overline{\mathsf{WL}}(s)$, total via count $\mathsf{NV}(s)$, and design-rule violations $\mathsf{DRV}(s)$. We normalize total wirelength by the larger of the board height and width. A clean pass requires $\mathsf{U}(s) = 0$ and $\mathrm{DRV}_{\mathrm{err}}(s) = 0$, where $\mathrm{DRV}_{\mathrm{err}}$ counts error-severity violations. We combine these quantities in the potential

$$\Phi(s) \;=\; \beta \cdot \mathbf{1}\big[\,\mathsf{U}(s) = 0\,\big] \;-\; \lambda_{\mathrm{u}}\,\mathsf{U}(s) \;-\; \lambda_{\mathrm{wl}}\,\overline{\mathsf{WL}}(s) \;-\; \lambda_{\mathrm{via}}\,\mathsf{NV}(s) \;-\; D(s) \tag{1}$$

Here β is a completion bonus that scales logarithmically with the number of nets, and $D(s)$ penalizes both the spread of violations across nets and their count within each net. We use potential-based

rewards, with search accumulating $\Delta\Phi_t = \Phi(s_{t+1}) - \Phi(s_t)$ and discount $\gamma = 0.995$. Invalid search actions are removed without backup by the `pop` rule (§4.3). Appendix B defines these terms.

Actions and Dynamics. Actions are drawn from the CAD-engine operations exposed through PCBWorld rather than grid cells. Our action space contains `net_select`, `start_route`, `make_line`, `make_via`, and `net_end`, but excludes PCBWorld's `finish` action so that connections are routed through explicit agent decisions. A candidate is the tuple $a = (a_{\text{type}}, a_{\text{ptr}}, a_{\text{mode}})$ of action type, pointer, and routing mode, with a pointer into the board-entity and directional candidate pool (Appendix B). The fields depend on the action type: `net_select` and `start_route` use the pointer, `make_line` and `make_via` use both pointer and mode, and `net_end` uses neither. The engine resolves the resulting geometry. Deny-by-default masking defines the admissible candidate set $\mathcal{A}(s) \subseteq \mathcal{A}$; rejection during CAD execution is handled in Section 4.3. Executing $a \in \mathcal{A}(s)$ through the CAD engine induces the transition $P(s' \mid s, a)$, which becomes deterministic under the engine controls of Section 4.2. An episode ends when every net is closed, at the horizon T, or when $\mathcal{A}(s) = \emptyset$, with $T = 512$ decisions.

3.2 Calibrated Leaf Values and Action Selection

A simulation ending at depth d combines the discounted path reward R_{path} with a critic estimate of the remaining return. These quantities initially have different units: PPO trains the critic on rewards $\tilde{r}_t = \text{clip}(r_t/\sigma, -10, 10)$, normalized by a running standard deviation without mean subtraction (van Hasselt et al., 2016), whereas search accumulates raw potential differences. Away from clipping, $V_\theta(s_t) \approx \mathbb{E}[G_t \mid s_t]/\sigma$, where G_t is the raw discounted return. Variation in σ during training and return shifts on the target board motivate board-wise calibration.

Board-wise iterated Bellman calibration. We fit an isotonic map $\hat{g}$ from critic outputs to target-board returns. Stochastic policy rollouts without early stopping are split by episode into disjoint fitting and hold-out halves, $\mathcal{D}_{\text{fit}}$ and $\mathcal{D}_{\text{hold}}$. The fitting set is $\mathcal{D}_{\text{fit}} = \{(V_\theta(s_i), r_i, V_\theta(s'_i), \delta_i)\}_{i=1}^{M}$, where δ_i indicates whether s'_i is terminal. On the fitting set, we alternate Bellman target updates with isotonic regression:

$$y_i^{(j)} \;=\; r_i \;+\; \gamma\,(1-\delta_i)\,\hat{g}_j\big(V_\theta(s'_i)\big), \tag{2}$$

$$\hat{g}_{j+1} \;=\; \operatorname*{arg\,min}_{g \in \mathcal{G}_\uparrow} \textstyle\sum_{i=1}^{M} \big(g(V_\theta(s_i)) - y_i^{(j)}\big)^2. \tag{3}$$

Here $\mathcal{G}_\uparrow$ is the class of non-decreasing scalar functions. The pool-adjacent-violators algorithm (PAVA) solves each isotonic regression exactly (Barlow & Brunk, 1972; Best & Chakravarti, 1990). The resulting map adjusts scale, offset, and monotone nonlinear distortion. Appendix B provides implementation details, the stopping criterion, and the relation to offline Bellman calibration (van der Laan & Kallus, 2026).

Backup of the calibrated leaf value. Let ρ_{cal} be the held-out Pearson correlation between fitted values and a discounted-return proxy (Appendix B). The implementation uses only the path reward when $\rho_{\text{cal}} \le 0$. The value that a simulation backs up is defined as

$$G_{\text{sim}} \;=\; \underbrace{\textstyle\sum_{k=0}^{d-1} \gamma^k\,\Delta\Phi_k}_{R_{\text{path}}} \;+\; \gamma^d\,(1-\delta_d)\,\mathbf{1}\big[\,\rho_{\text{cal}} > 0\,\big]\,\big(\,\hat{g}(V_\theta(s_d)) - \hat{g}(V_{\text{done}})\,\big) \tag{4}$$

Here δ_d indicates a terminal leaf, and $\hat{g}(V_{\text{done}})$ denotes the median calibrated value at complete-routing states ($\mathsf{U} = 0$) reached by calibration rollouts, or zero if none are reached. The search map includes the knot compression and monotonicity adjustment in Appendix B. Subtracting this anchor removes the offset carried by non-terminal values when they are compared with terminal children, whose bootstrap is zero. Calibration places the leaf estimate in the same units as the path reward; the subsequent node-wise completed-Q transformation controls its scale in action selection.

Gumbel action selection. We use Gumbel search for its policy-improvement formulation under limited simulation budgets (Danihelka et al., 2022). At the root, independent noise $g(a) \sim \text{Gumbel}(0, 1)$ is drawn once per action. The top m perturbed logits form $\mathcal{C}(s)$, and sequential halving (Karnin et al.,

2013) allocates n_{sim} simulations among them. The final action is selected by

$$\ell(a) \;=\; \log \pi_\theta(a \mid s) + c_g\, g(a), \quad \mathcal{C}(s) \;=\; \underset{a \in \mathcal{A}(s)}{\text{Top-}m}\; \ell(a), \quad a_{\mathrm{root}} \;=\; \underset{a \in \mathcal{C}_{\mathrm{final}}(s)}{\arg\max} \left[\, \ell(a) + \hat{q}(s,a) \,\right] \tag{5}$$

where $\hat{q}$ completes unvisited action values, normalizes them over the children of s, and rescales them according to visit counts. The coefficient c_g controls root noise; c_v controls the normalized value contribution in $\hat{q}$. Appendix B defines this transformation, $\mathcal{C}_{\mathrm{final}}(s)$, interior-node selection, and the PUCT baselines.

4 Model Architecture and Search Optimization

4.1 Policy and Value Network

Learning and search share a decoder-only Transformer actor–critic with attention-based pointer outputs (Vaswani et al., 2017; Vinyals et al., 2015). The observation is tokenized with one token for each board entity, including nets, pads, traces, vias, ratsnest edges, the route head, and candidates tagged by class; the training configuration omits design-rule-violation tokens. No sequence-position encoding is applied to the state tokens, making the state encoder permutation-equivariant. Net membership is represented by a learned same-net attention bias, which distinguishes entities on the same net without requiring a hand-built adjacency. State tokens attend bidirectionally, while action tokens attend to the state and preceding action tokens. State tokens never attend to action tokens, allowing their per-layer keys and values to be reused during action decoding.

The action is factorized autoregressively as $a = (a_{\mathrm{type}}, a_{\mathrm{ptr}}, a_{\mathrm{mode}})$. The state prefix supplies the action-type logits. Decoding a token for a given type yields the query used to score net or geometric-candidate pointers against the encoded state. For `make_line` and `make_via`, a further pointer token yields all routing-mode logits conditioned on that type and candidate; no separate mode token is needed. Thus, the three action factors are scored using the prefix and at most two appended tokens. The resulting joint scores are normalized over the legal set $\mathcal{A}(s)$ used by the tree, so no action remapping is required at search time. The state prefix is encoded once per node; its hidden representation produces V_θ and is reused to decode all action factors. Appendix C details the architecture and training configuration.

4.2 Deterministic Routing Engine

The transition P of §3.1 is implemented by the KiCad engine (KiCad). Our search restores a saved state before exploring another branch, so replaying an action must not depend on the branches explored previously. Memory-address-based tie-breaking, UUID generation, and wall-clock termination can otherwise change the resulting geometry and reward. We use address-independent ordering, seed and restore the UUID-generator state, and replace the shove loop's time cutoff with an iteration limit. Both restore paths also canonicalize joint-link order and rebuild the spatial index in a deterministic item order, preventing insertion history from affecting subsequent routing. Together with the saved routing session and environment state, these controls support reproducible branch evaluation. Appendix D describes the engine fixes.

4.3 Search Efficiency

A simulation costs three things: restoring the board from a checkpoint, evaluating the policy and the value, and executing actions in the CAD engine. We reduce these costs through incremental checkpointing and the architecture- and search-level mechanisms in Figure 2: policy-value memoization, branch-packed decoding, action masking, and invalid-action filtering. The environment- and architecture-level techniques reduce repeated state reconstruction and network evaluation. Restoration is checked against a full-restore reference, and packed inference preserves the action factorization up to floating-point roundoff. The search-level techniques exclude targets with no connectivity gain before expansion and remove failed or ineffective actions after execution. See Appendix E for details.

Environment level. A full restore replaces all tracks and vias and rebuilds the routing world and connectivity, including obstacles derived from unchanged pads, footprints, zones, and the outline.

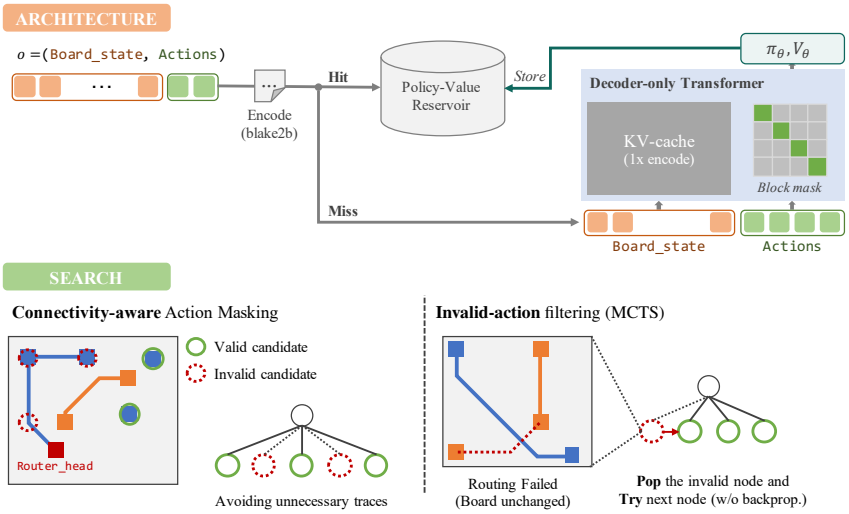


Figure 2: Architecture- and search-level optimizations in PW-MCTS. ARCHITECTURE: policy-value memoization reuses cached outputs; on a miss, branch-packed decoding shares the state K/V cache across action branches with a block attention mask. SEARCH: connectivity-aware masking excludes redundant targets, while invalid-action filtering removes failed branches after execution and retries without backup.

Incremental checkpointing retains a snapshot of the routing state but restores only tracks and vias that differ in identifiers or fields. It updates the routing world and ratsnest while preserving static obstacles, and restores the routing session and DRC state. Each node also stores the environment and wrapper state, including the observation and pointer tables, to avoid rebuilding these on every restore. The full-restore path serves as the reference in tests of board geometry and connectivity.

Architecture (model) level. Each decision builds a new search tree, but states explored in earlier decisions may recur. **Policy-value memoization** reuses their priors and provider leaf values through an episode-scoped cache keyed by the observation fingerprint and legal-action set. After each committed action, entries not accessed during the preceding decision are discarded; tree statistics are not cached. On a miss, **branch-packed decoding** encodes the state once and reuses its per-layer keys and values in a single packed decode of action-type and pointer-conditioned branches. A block attention mask gives each branch access only to the shared prefix and its own action context. The joint prior is assembled from the resulting autoregressive factors, and the critic is evaluated once from the shared state representation. This avoids a separate state encoding for every candidate action.

Search level. **Action masking** defines admissible candidates before execution: while routing, candidate generation removes existing-copper targets in the route head's connected cluster, avoiding redundant connections. Directional proposals bypass this connectivity filter. Enumeration applies type, net, mode, and pointer masks, including the type-specific layer restriction, to form $\mathcal{A}(s)$. Its cardinality determines the available branches; prior-computation cost also depends on the state and candidate-token counts. Some failures can only be detected by the engine, such as placing a via at an occupied point or routing from an unreachable location. The environment classifies failed or ineffective actions after execution. For nonterminal outcomes, **invalid-action filtering** removes the provisional child without backup and releases its checkpoint; the same simulation then selects another child. The removed edge contributes no visit or value backup; if removal empties the child set, the parent becomes terminal and its path return is backed up. If the root has only one admissible action, search uses one simulation to check its outcome instead of spending the full budget on a forced decision.

Table 1: Equal-draw comparison on the same 16 rollout seeds, mean (sample sd). All metrics include incomplete and non-clean outcomes. Method parentheses give simulations per decision; bold marks the best mean. Difficulty definitions are in Appendix G.

Subset	Metric	PCBWorld (PPO)*	PW-MCTS (4)	PW-MCTS (8)	PW-MCTS (16)	PW-MCTS (32)
All	CP	0.422 (0.494)	0.558 (0.497)	0.568 (0.496)	0.609 (0.488)	**0.632** (0.482)
	Rout.	0.742 (0.333)	0.849 (0.268)	0.859 (0.255)	0.875 (0.247)	**0.885** (0.235)
	FP (Φ)	−19.312 (23.601)	−12.646 (19.649)	−11.979 (18.803)	−10.584 (17.866)	**−9.860** (17.154)
Easy	CP	0.827 (0.378)	0.938 (0.242)	0.928 (0.259)	0.940 (0.238)	**0.952** (0.213)
	Rout.	0.974 (0.090)	0.992 (0.045)	0.994 (0.040)	0.990 (0.057)	**0.996** (0.038)
Medium	CP	0.388 (0.488)	0.536 (0.499)	0.556 (0.497)	0.638 (0.481)	**0.667** (0.472)
	Rout.	0.834 (0.238)	0.913 (0.173)	0.914 (0.175)	0.942 (0.148)	**0.952** (0.129)
Hard	CP	0.027 (0.163)	0.180 (0.384)	0.197 (0.398)	0.225 (0.418)	**0.255** (0.436)
	Rout.	0.397 (0.306)	0.628 (0.339)	0.656 (0.325)	0.680 (0.327)	**0.696** (0.316)

5 EXPERIMENTS

5.1 SETUP

Protocol. We evaluate the policy of §4.1, trained on synthetic boards (Appendix F), on 50 held-out boards from PCBWorld-Bench (Song et al., 2026). PW-MCTS and the stochastic PPO baseline use the same pretrained policy. PCBWorld (PPO)* is abbreviated PPO in prose and PPO* in compact plots. Asterisks mark PCBWorld baselines using this work's implementation optimizations. Experimental settings are given in Appendix H (Table 8); board difficulty groups are defined in Appendix G.

Comparison. We evaluate PW-MCTS with $n_{\text{sim}} \in \{4, 8, 16, 32\}$ simulations per decision. **Equal draws** pairs each setting with the first PPO rollout on the same 16 rollout seeds. **Matched time** selects best-of-N (BoN) outcomes completed within each board–seed pair's 16-trajectory PW-MCTS (32) time budget. Search pools contain up to 128/64/32/16 candidates, respectively; exhausted pools retain observed outcomes. Time excludes initialization and calibration; Appendix H describes shared calibration accounting. In Table 2, N is the mean completed candidate count; PCBWorld (Astra)* uses GPT-6-astra-medium for one attempt per board and is not time-matched. Replay and baseline comparisons are detailed in Appendices H and J.

Metrics. We report clean pass (CP), routability (Rout.), final potential (FP, Φ), wirelength (WL), and via count (Via). Clean pass requires complete routing with no error-severity violations ($\text{DRV}_{\text{err}} = 0$). Table 1 reports CP, routability, and FP over all outcomes. Table 2 uses all outcomes for CP and routability, and jointly clean seeds for FP, WL, and Via. Pass denotes boards with at least one seed where all five PPO/PW-MCTS settings pass clean; Other denotes the remainder. These quality comparisons are conditional on success. Astra quality is shown only where it passes the entire cohort; dashes denote differing success cohorts. Reference uses the corresponding original boards. Appendix J separates all-board baseline performance from method-specific clean-route characteristics. Evaluation and aggregation rules are specified in Appendix H; FP leads below 0.01 are not ranked in bold.

5.2 RESULTS

Equal draws. Search improves individual attempts from the same learned prior: every simulation budget raises CP and routability in every difficulty group (Table 1). Aggregate per-draw performance improves with more simulations; at $n_{\text{sim}} = 32$, CP rises from 42.2% to 63.2% and routability from 0.742 to 0.885. Mean FP over all outcomes also improves, consistent with the gains in routing progress and completion. At this budget, the CP advantage persists across checkpoints, while its magnitude varies with the prior (Appendix I).

Matched time. A common time budget exposes the tradeoff between improving each attempt and generating more candidates (Table 2; Figure 3). Within the recorded pools, search with $n_{\text{sim}} \in \{4, 8\}$ achieves the highest CP at 89.3%, compared with 84.7% for PPO. This coverage gain comes from hard boards, where CP rises from 52.1% to 66.7%. On easy and medium boards, repeated PPO

Table 2: BoN within the PW-MCTS (32) 16-trajectory time budget, with a separate single-attempt Astra baseline. Entries are mean (sample sd) across boards after averaging PPO/PW-MCTS seeds. Pass and Other give board counts; bold marks the best PPO/PW-MCTS mean. Full results are in Table 14. *Implementation optimizations from this work.

Subset	Metric	PCBWorld (PPO)*	PW-MCTS (4)	PW-MCTS (8)	PW-MCTS (16)	PW-MCTS (32)	PCBWorld (Astra)*	Reference
All	CP	0.847 (0.324)	**0.893** (0.281)	**0.893** (0.289)	0.880 (0.292)	0.867 (0.301)	0.580 (0.499)	
	Rout.	0.937 (0.184)	**0.952** (0.155)	0.951 (0.168)	0.950 (0.169)	0.949 (0.163)	0.830 (0.278)	
Easy	N	287.2	110.5	59.7	30.4	16.0		
	CP	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	1.000 (0.000)	
	FP (Φ)	0.520 (0.952)	0.521 (0.941)	0.500 (0.955)	0.483 (0.942)	0.419 (0.968)	0.783 (0.959)	0.804 (0.786)
	WL (mm)	**545.8** (303.1)	548.6 (307.1)	551.3 (309.4)	555.3 (305.2)	567.0 (314.5)	468.4 (275.6)	483.2 (275.6)
	Via	1.451 (2.017)	**1.373** (1.683)	1.412 (1.801)	**1.373** (1.863)	1.471 (1.990)	2.529 (3.826)	1.471 (2.601)
Medium	N	211.9	87.2	54.5	29.2	16.0		
	CP	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	0.588 (0.507)	
	FP (Φ)	−0.243 (0.928)	**−0.227** (0.859)	−0.296 (0.859)	−0.292 (0.858)	−0.359 (0.877)	—	0.098 (0.679)
	WL (mm)	752.3 (492.6)	**741.2** (479.7)	746.5 (481.3)	759.5 (503.7)	763.0 (499.0)	—	611.4 (403.1)
	Via	**2.804** (4.165)	3.098 (4.395)	3.451 (4.587)	3.235 (4.690)	3.510 (4.898)	—	5.176 (5.582)
Hard	N	238.7	93.5	54.4	29.5	16.0		
	CP	0.521 (0.421)	**0.667** (0.422)	**0.667** (0.439)	0.625 (0.419)	0.583 (0.413)	0.125 (0.342)	
Pass (10)	FP (Φ)	−2.881 (1.896)	**−2.560** (1.990)	−2.737 (2.048)	−2.573 (2.131)	−2.618 (2.065)	—	−1.578 (1.880)
	WL (mm)	830.1 (464.0)	**785.9** (431.1)	794.2 (440.0)	789.7 (462.8)	806.5 (461.1)	—	606.8 (386.6)
	Via	8.583 (6.827)	8.083 (5.279)	9.533 (7.493)	8.150 (5.152)	**7.867** (5.705)	—	9.700 (7.646)
Other (6)	Rout.	0.528 (0.297)	**0.605** (0.269)	0.596 (0.324)	0.591 (0.323)	0.604 (0.310)	0.473 (0.372)	

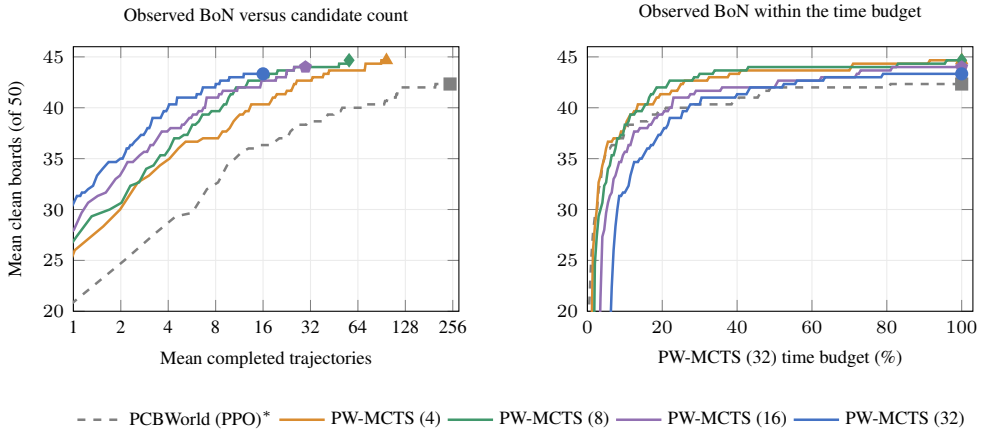


Figure 3: BoN tradeoff for PPO and PW-MCTS. Both panels show matched-time replay, averaged over three training seeds on 50 boards. Left: mean completed trajectories; counts vary across boards and seeds, so this is not an equal-draw comparison. Right: the fraction of each board–seed pair's PW-MCTS (32) 16-trajectory time budget. Endpoint markers use the full budget. Exhausted pools retain their observed result without extrapolation.

sampling already achieves full coverage, while generating more candidates and achieving higher BoN FP than search with eight or more simulations; the smallest search budget remains competitive. Thus, stronger individual attempts do not guarantee better BoN outcomes under a time budget, and the useful allocation of search and repetition depends on board difficulty. On harder boards, reference layouts use more vias and shorter wires, consistent with the learned policy's via avoidance (Figure 11).

Ablations. The ablations support both efficient branch evaluation and board-wise value adaptation. System optimizations reduce total search time by 7.95× and lower mean peak RSS (Table 3). Memoization and packing preserve recorded trajectory metrics. Removing invalid actions improves completion relative to penalization, at greater execution cost. The calibrated, anchored critic raises CP by 5.1 percentage points and shortens episodes relative to checkpoint-wide scaling (Table 13). These comparisons assess combined configurations; they do not isolate the calibrator, gate, anchor, or root-candidate count (Appendix K).

Table 3: Iteration-400 ablations, training seed 42. *Top*: cumulative implementation changes on 50 boards, rollout seed 0. *Bottom*: invalid-action handling on 800 paired instances per setting. Φ uses the aligned evaluation score; $\Delta\Phi$ is relative to `pop`. Profiling is enabled in the bottom block, so its times are not directly comparable with the unprofiled campaign. Time per decision uses searched decisions; speedup uses total search time. RSS is the mean per-board peak resident memory.

Setting	Sims/sec	Search time/ decision	Mean peak RSS	Speedup
Base (determinized engine)	1.28	24.936 s	9,899 MB	1.00×
+ incremental checkpointing	1.33	24.038 s	9,832 MB	1.04×
+ policy-value memoization	1.50	21.291 s	9,878 MB	1.17×
+ branch-packed decoding	**10.19**	**3.140** s	**671** MB	**7.95**×

Setting	Invalid ratio	Eff. sims	CP	Φ	s/decision	$\Delta\Phi$
`keep`	19.8%	22.2	51.9%	−12.57	1.135	−3.43
`penalize`	22.8%	21.4	64.5%	−9.43	1.013	−0.29
`pop`	52.2%	27.4	67.2%	−9.14	1.987	—

6 Discussion

Search–sampling tradeoff. Practical test-time refinement balances per-draw quality against candidate count. PW-MCTS (32) yields the highest per-draw CP, whereas PW-MCTS (4) and (8) tie for highest coverage within the common time budget (Figure 3). More search per decision can improve an attempt but leaves time for fewer repetitions. These results motivate jointly allocating search and repetitions according to board difficulty and the balance between completion and route quality.

Prior and MDP dependence. Checkpoint and training-seed variation show sensitivity to the starting policy (Appendices I and M). Search can improve its decisions, but the action space, reward, and horizon still determine which solutions are accessible and preferred. Stronger RL algorithms and synthetic training boards that better reflect real geometry and congestion could therefore complement test-time search.

Scope of comparison. Our controlled comparison measures improvement over the same pretrained PPO policy. Freerouting and PCBWorld powered by GPT-6-astra provide external routing references (Appendix J). Under their respective protocols, CP is 86.7–89.3% for PW-MCTS and 92% for Freerouting. These reference comparisons are not time-matched and contextualize completion and route-quality gaps beyond the learned prior.

7 Conclusion

PW-MCTS refines a pretrained RL routing prior at test time through native CAD-engine search. Board-wise value calibration and reproducible, efficient branch evaluation make this refinement practical to study on real boards. Search improves completion and routability at equal draws; under matched-time budgets, its benefit depends on the balance between candidate count and search effort. The resulting router remains shaped by its MDP and learned prior. Combining this refinement with stronger PPO training and more realistic synthetic data offers a direction for further improving learned routing policies.

AI use statement

We used generative AI tools to assist with coding, result analysis, figure preparation, and manuscript revision. The authors reviewed the AI-assisted work and take responsibility for the final text, claims, and artifacts. The LLM routing baseline is documented in Appendix J.

Ethics statement

This study uses synthetic and publicly available PCB designs and involves no human participants or personal data. The CAD engine, environment, and benchmark are credited to their sources (KiCad; Song et al., 2026). The authors declare no conflicts of interest.

Reproducibility statement

Appendices B–E describe the search, calibration, network architecture, and CAD-engine implementation. Training data and evaluation boards are documented in Appendices F and G. Appendix H specifies the checkpoints, seeds, hyperparameters, and evaluation protocols. Code is not included in this submission; we plan to make it available upon request or through a future public release.

References

Zaheer Abbas, Samuel Sokota, Erin Talvitie, and Martha White. Selective Dyna-style planning under limited model capacity. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1–10. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/abbas20a.html.

Farzane Aminmansour, Taher Jafferjee, Ehsan Imani, Erin J. Talvitie, Michael Bowling, and Martha White. Mitigating value hallucination in Dyna-style planning via multistep predecessor models. *Journal of Artificial Intelligence Research*, 80:441–473, June 2024. ISSN 1076-9757. doi: 10.1613/jair.1.15155. URL http://dx.doi.org/10.1613/jair.1.15155.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5360–5370, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d8e1344e27a5b08cdfd5d027d9b8d6de-Abstract.html.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. ReZero is all you need: Fast convergence at large depth. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 2021. URL https://proceedings.mlr.press/v161/bachlechner21a.html.

Victor Bapst, Alvaro Sanchez-Gonzalez, Carl Doersch, Kimberly Stachenfeld, Pushmeet Kohli, Peter Battaglia, and Jessica Hamrick. Structured agents for physical construction. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 464–474. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/bapst19a.html.

R. E. Barlow and H. D. Brunk. The isotonic regression problem and its dual. *Journal of the American Statistical Association*, 67(337):140–147, March 1972. ISSN 1537-274X. doi: 10.1080/01621459.1972.10481216. URL http://dx.doi.org/10.1080/01621459.1972.10481216.

Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, August 1952. ISSN 1091-6490. doi: 10.1073/pnas.38.8.716. URL http://dx.doi.org/10.1073/pnas.38.8.716.

Michael J. Best and Nilotpal Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1-3):425–439, May 1990. ISSN 1436-4646. doi: 10.1007/bf01580873. URL http://dx.doi.org/10.1007/BF01580873.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

Alberto Castellini, Federico Bianchi, Edoardo Zorzi, Thiago D. Simão, Alessandro Farinelli, and Matthijs T. J. Spaan. Scalable safe policy improvement via Monte Carlo tree search. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3732–3756. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/castellini23a.html.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with Gumbel. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=bERaNdoegnO.

William A. Dees and Patrick G. Karger. Automated rip-up and reroute techniques. In *Proceedings of the nineteenth design automation conference on - DAC '82*, DAC '82, pp. 432–439. ACM Press, 1982. doi: 10.1145/800263.809241. URL http://dx.doi.org/10.1145/800263.809241.

Morris H. DeGroot and Stephen E. Fienberg. The comparison and evaluation of forecasters. *The Statistician*, 32(1/2):12–22, March 1983. ISSN 0039-0526. doi: 10.2307/2987588. URL http://dx.doi.org/10.2307/2987588.

Rowan Devereux-Smith and Patrick H. Madden. Strategic rip-up and reroute. In *Proceedings of the Great Lakes Symposium on VLSI 2025*, GLSVLSI '25, pp. 770–776. ACM, June 2025. doi: 10.1145/3716368.3735162. URL http://dx.doi.org/10.1145/3716368.3735162.

Arnaud Fickinger, Hengyuan Hu, Brandon Amos, Stuart J. Russell, and Noam Brown. Scalable online planning via reinforcement learning fine-tuning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 16951–16963, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/8ce8b102d40392a688f8c04b3cd6cae0-Abstract.html.

Keith Frankston and Benjamin Howard. Accelerating Monte-Carlo tree search with optimized posterior policies. *arXiv preprint arXiv:2601.01301*, 2026. URL https://arxiv.org/abs/2601.01301.

Upma Gandhi, Erfan Aghaeekiasaraee, Ismail S. K. Bustany, Payam Mousavi, Matthew E. Taylor, and Laleh Behjat. RL-Ripper: A framework for global routing using reinforcement learning and smart net ripping techniques. In *Proceedings of the Great Lakes Symposium on VLSI 2023*, GLSVLSI '23, pp. 197–201. ACM, June 2023. doi: 10.1145/3583781.3590312. URL http://dx.doi.org/10.1145/3583781.3590312.

Upma Gandhi, Erfan Aghaeekiasaraee, Sahir, Payam Mousavi, Ismail S. K. Bustany, Mathew E. Taylor, and Laleh Behjat. Applying reinforcement learning to learn best net to rip and re-route in global routing. *ACM Transactions on Design Automation of Electronic Systems*, 29(4):1–21, July 2024. ISSN 1557-7309. doi: 10.1145/3664286. URL http://dx.doi.org/10.1145/3664286.

Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Remi Munos. Monte-Carlo tree search as regularized policy optimization. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3769–3778. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/grill20a.html.

Jessica B. Hamrick, Abram L. Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Buesing, Petar Veličković, and Théophane Weber. On the role of planning in model-based deep reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=IrM64DGB21.

Youbiao He, Hebi Li, Tian Jin, and Forrest Sheng Bao. Circuit routing using Monte Carlo tree search and deep reinforcement learning. In *2022 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–5. IEEE, April 2022. doi: 10.1109/vlsi-dat54769.2022.9768074. URL http://dx.doi.org/10.1109/VLSI-DAT54769.2022.9768074.

André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nO5caZwFwYu.

Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. In *The International FLAIRS Conference Proceedings*, volume 35, 2022. doi: 10.32473/flairs.v35i.130584. URL https://doi.org/10.32473/flairs.v35i.130584.

Andrew B. Kahng, Lutong Wang, and Bangqi Xu. TritonRoute: The open-source detailed router. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 40(3):547–559, 2021. doi: 10.1109/TCAD.2020.3003234. URL https://doi.org/10.1109/TCAD.2020.3003234.

Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1238–1246. PMLR, 2013. URL https://proceedings.mlr.press/v28/karnin13.html.

Yu-Chan Keng, Yu-Chun Pai, Wen-Hao Liu, Haoxing Ren, Danny Liu, Rongjian Liang, Mark Ho, Anthony Agnesina, and Yih-Lang Li. Reinforcement learning-driven window selection for enhanced window-based rip-up and reroute in chip detailed routing. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7. IEEE, June 2025. doi: 10.1109/dac63849.2025.11132699. URL http://dx.doi.org/10.1109/DAC63849.2025.11132699.

KiCad. KiCad. https://www.kicad.org/, n.d. Open-source electronics design automation suite. Source code primarily licensed under GNU GPL v3 or later. Accessed September 23, 2026.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*, pp. 282–293, 2006.

Dong Li, Xujiang Zhao, Linlin Yu, Yanchi Liu, Wei Cheng, Zhengzhang Chen, Zhong Chen, Feng Chen, Chen Zhao, and Haifeng Chen. SolverLLM: Leveraging test-time scaling for optimization problem via LLM-guided search. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 100028–100058. Curran Associates, Inc., 2025. doi: 10.52202/085713-3343. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/903475a4d33d21bfd75d4fa0d124abfe-Paper-Conference.pdf.

Haiguang Liao, Wentai Zhang, Xuliang Dong, Barnabas Poczos, Kenji Shimada, and Levent Burak Kara. A deep reinforcement learning approach for global routing. *Journal of Mechanical Design*, 142(6):061701, June 2020. ISSN 1528-9001. doi: 10.1115/1.4045044. URL http://dx.doi.org/10.1115/1.4045044.

Yin-Chi Liao, Sheng-Xin Pan, and Po-Jui Chiang. Automation of PCB autorouting via world-model reinforcement learning and freerouting integration. *Expert Systems with Applications*, 311:131424, May 2026. ISSN 0957-4174. doi: 10.1016/j.eswa.2026.131424. URL http://dx.doi.org/10.1016/j.eswa.2026.131424.

Shih-Ting Lin, Hung-Hsiao Wang, Chia-Yu Kuo, Yolo Chen, and Yih-Lang Li. A complete PCB routing methodology with concurrent hierarchical routing. In *58th ACM/IEEE Design Automation Conference, DAC 2021, San Francisco, CA, USA, December 5-9, 2021*, pp. 1141–1146. IEEE, 2021. doi: 10.1109/DAC18074.2021.9586143. URL https://doi.org/10.1109/DAC18074.2021.9586143.

Yibo Lin, Tong Qu, Zongqing Lu, Yajuan Su, and Yayi Wei. Asynchronous reinforcement learning framework and knowledge transfer for net-order exploration in detailed routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(9):3132–3142, September 2022. ISSN 1937-4151. doi: 10.1109/tcad.2021.3117505. URL http://dx.doi.org/10.1109/TCAD.2021.3117505.

Seungyong Moon, JunYeong Lee, and Hyun Oh Song. Rethinking value function learning for generalization in reinforcement learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/e19ab2dde2e60cf68d1ded18c38938f4-Abstract-Conference.html.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(27):815–857, 2008. URL http://jmlr.org/papers/v9/munos08a.html.

Tong Qu, Yibo Lin, Zongqing Lu, Yajuan Su, and Yayi Wei. Asynchronous reinforcement learning framework for net order exploration in detailed routing. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1815–1820. IEEE, February 2021. doi: 10.23919/date51398.2021.9474007. URL http://dx.doi.org/10.23919/DATE51398.2021.9474007.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, December 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL http://dx.doi.org/10.1038/s41586-020-03051-4.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017. ISSN 1476-4687. doi: 10.1038/nature24270. URL http://dx.doi.org/10.1038/nature24270.

Hyungseok Song, Junseok Park, Won-Seok Choi, Seohui Bae, Han-Seul Jeong, Youngjoon Park, and Soonyoung Lee. PCBWorld: A benchmark environment for engine-grounded PCB design automation, 2026. URL https://arxiv.org/abs/2607.05915.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/55053683268957697aa39fba6f231c68-Abstract.html.

Lars van der Laan and Nathan Kallus. Bellman calibration for V-learning in offline reinforcement learning, 2026. URL https://arxiv.org/abs/2512.23694.

Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29, pp. 4287–4295. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/5227b6aaf294f5f027273aebf16015f2-Paper.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, pp. 2692–2700, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. ExACT: Teaching AI agents to explore with Reflective-MCTS and exploratory learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=GBIUbwW9D8.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

Zhanwen Zhou, Hankz Hankui Zhuo, Jinghua Zhou, and Wushao Wen. Transformer-based reinforcement learning for net ordering in detailed routing. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI-2025, pp. 9492–9500. International Joint Conferences on Artificial Intelligence Organization, September 2025. doi: 10.24963/ijcai.2025/1055. URL http://dx.doi.org/10.24963/ijcai.2025/1055.

APPENDIX

Table of contents

A Notation 16

B Details of the Problem Formulation 16

C Network Architecture and PPO Pretraining 19

D Determinization of the CAD Engine 20

E Details of the Search Cost Optimizations 21

- **E.1** Incremental checkpointing 21
- **E.2** Policy-value memoization 21
- **E.3** Branch-packed decoding 23
- **E.4** Action masking 24
- **E.5** Invalid-action filtering 26

F Synthetic Training Dataset 26

G Evaluation Boards 27

H Experimental Configuration 30

I Per-Checkpoint Results 31

J External Routing References 32

K Ablations 36

L Candidate Distributions 37

M Extended Discussion 38

N Board-wise search–sampling tradeoffs 39

A NOTATION

Key notation for the routing objective, calibrated values, and search. Auxiliary symbols are defined alongside their equations in Section 3 and Appendix B.

Table 4: Notation for routing and search.

Symbol	Meaning
Routing state and actions	
$\mathsf{NL},\ N$	netlist and an individual net
$\mathcal{B}$	fixed board data: outline, layers, obstacles, and design rules
s	board state, including routing geometry and active routing context
$a = (a_{\text{type}}, a_{\text{ptr}}, a_{\text{mode}})$	action type, pointer, and routing mode
$\mathcal{A}(s)$	candidates admitted by masking; execution may still fail
Objective and reward	
$\mathsf{U}(s),\ \rho(s)$	remaining connections and fraction of required connections realized
$\mathsf{WL}(s),\ \overline{\mathsf{WL}}(s)$	total wirelength and its normalization by the larger board dimension
$\mathsf{NV}(s)$	total via count
$\mathsf{DRV}(s),\ D(s)$	design-rule violation set and its concave penalty
$\Phi(s)$	routing potential, Eq. (1)
$r_t = \Delta\Phi_t$	step reward $\Phi(s_{t+1}) - \Phi(s_t)$
γ	discount factor shared by learning and search
Policy, critic, and calibration	
$\pi_\theta(a \mid s)$	pretrained policy prior over admissible actions
$V_\theta(s)$	critic prediction in normalized return units
$\sigma,\ \tilde{r}_t = r_t/\sigma$	running normalization scale and normalized training reward
G_t	discounted return from step t, in raw reward units
$\hat{g}$	board-wise calibrator mapping critic outputs to raw return units
$\mathcal{D}_{\text{fit}},\ \mathcal{D}_{\text{hold}}$	calibration transitions from disjoint fitting and hold-out episodes
$\hat{g}(V_{\text{done}})$	terminal anchor: median calibrated value at terminal rollout states
ρ_{cal}	held-out correlation of fitted values with a return proxy; gates bootstrapping
Search backup and selection	
R_{path}	discounted rewards along the root-to-leaf path
G_{sim}	simulation return including the gated, anchored leaf value, Eq. (4)
$N(s,a),\ Q(s,a)$	edge visit count and backed-up action value
$\hat{q}(s,a)$	completed action value, normalized within a node and scaled by visits, Eq. (11)
$\bar{Q}(s,a)$	tree-wide normalized action value for PUCT, Eq. (15)
$c_g,\ c_v$	scales of root Gumbel noise and the completed-value contribution
$\mathcal{C}(s),\ \mathcal{C}_{\text{final}}(s)$	initial and final root candidate sets for sequential halving
n_{sim}	simulation budget per decision; zero denotes policy-only execution

B DETAILS OF THE PROBLEM FORMULATION

Pads and layers. Every pad lies inside the board outline $\mathcal{E} \subset \mathbb{R}^2$, so $\text{pos}(p) \in \mathcal{E}$, and the layers it occupies—one or several, by pad type—are written $L(p) \subseteq L$, where L is the set of copper layers. For net N, the routing geometry is $s_N = T_N \cup V_N$, with T_N the intra-layer traces and V_N the inter-layer vias.

Action types and the candidate pool. The action space $\mathcal{A}$ consists of `net_select`, which picks the net to route; `start_route`, which starts a route at a point on a pad; `make_line`, which draws a segment to a point; `make_via`, which places a via there; and `net_end`, which closes the current net. The two that create routing take a routing mode—mark-obstacles, shove, walkaround—selecting how the engine resolves a conflict with existing copper. The pointer a_{ptr} indexes a candidate pool of board entities—pads, traces, ratsnest endpoints, vias—together with *directional* candidates, the eight compass directions from the route head at a fixed radius, which let a route leave a pad along a free

heading when no entity lies that way. The radius decides whether they land outside the pad without overlapping a neighbour.

Connectivity and routability. Two pads $p, p' \in N$ are electrically connected when a path alternating traces and vias of s_N joins them, and $c(N; s)$ is the number of connected components this induces on N: $c(N; s_0) = |N|$ before any segment is placed, and N is complete when $c(N; s) = 1$. Writing $\mathrm{conn}(N; s) = |N| - c(N; s)$ for the connections realized, board routability is the ratio of realized to required connections:

$$\rho(s) \;=\; \frac{\sum_{N \in \mathsf{NL}} \mathrm{conn}(N; s)}{\sum_{N \in \mathsf{NL}} \big(|N| - 1\big)} \;\in\; [0, 1] \tag{6}$$

Cost terms. The unconnected count, the total routing length, and the total via count are, respectively,

$$\mathsf{U}(s) = \sum_{N \in \mathsf{NL}} \big(c(N; s) - 1\big), \qquad \mathsf{WL}(s) = \sum_{N \in \mathsf{NL}} \sum_{t \in T_N} \mathrm{len}(t), \qquad \mathsf{NV}(s) = \sum_{N \in \mathsf{NL}} |V_N| \tag{7}$$

Here $\mathsf{U}(s)$ counts the ratsnest edges not yet connected and satisfies $\mathsf{U}(s) = \mathsf{U}(s_0)\,(1 - \rho(s))$, so $\mathsf{U}(s) = 0$ and $\rho(s) = 1$ are equivalent. The potential uses normalized wirelength,

$$\overline{\mathsf{WL}}(s) = \frac{\mathsf{WL}(s)}{\max(h_{\mathcal{B}}, w_{\mathcal{B}})}, \tag{8}$$

where $h_{\mathcal{B}}$ and $w_{\mathcal{B}}$ are the board height and width, measured in the same units as $\mathsf{WL}(s)$. The coefficients λ_{u}, λ_{wl}, and λ_{via} weight remaining connections, normalized wirelength, and via count in Eq. (1).

Design rule violation penalty. The penalty depends on how many nets carry a violation as well as on the total count. With $m(s)$ nets carrying one and $d_N(s)$ violations in net N, so that $\sum_N d_N(s) = |\mathsf{DRV}(s)|$,

$$D(s) \;=\; a \ln\!\Big(1 + \frac{m(s)}{o}\Big) \;+\; \sum_{N \in \mathsf{NL}} b \ln\!\Big(1 + \frac{d_N(s)}{o}\Big) \tag{9}$$

The first term charges how widely violations spread, the second how deeply they accumulate within one net, and $o > 0$ sets the knee. Both are concave, so the penalty does not grow linearly in the violation count. Training and search include error-severity and promoted violations in $\mathsf{DRV}(s)$; evaluation uses error-severity violations only. Clean pass requires complete routing and zero error-severity violations.

Completion bonus. The coefficient β of Eq. (1) is $\beta = \kappa \ln(1 + |\mathsf{NL}|)$, which keeps the completion bonus at a constant weight relative to the cost terms across boards of different size.

Reward modes. In dense mode every step yields $r_t = \Phi(s_{t+1}) - \Phi(s_t)$, whereas in sparse mode a reward is given only at termination, $r_T = \Phi(s_T)$.

Interior-node selection. At an interior node no new Gumbel noise is drawn, and the action executed is the one whose share of visits most lags the improved policy,

$$a_{\mathrm{int}} \;=\; \arg\max_{a \in \mathcal{A}(s)} \Big[\, \mathrm{softmax}\big(\log \pi_\theta(\cdot \mid s) + \hat{q}(s, \cdot)\big)(a) \;-\; \frac{N(s, a)}{1 + \sum_b N(s, b)} \,\Big] \tag{10}$$

where $N(s, a)$ is the visit count of the edge. Repeating this argmax drives the visit frequencies to $\mathrm{softmax}(\log \pi_\theta + \hat{q})$. At the root, the set $\mathcal{C}_{\mathrm{final}}(s)$ of Eq. (5) is $\{a \in \mathcal{C}(s) : N(s, a) = N_{\max}\}$, the candidates sequential halving leaves with the most visits; the root reuses the perturbed logit that formed $\mathcal{C}(s)$ and adds only $\hat{q}$.

Completed Q: $\hat{q}$. The quantity $\hat{q}$ of Eqs. (5) and (10) first fills unvisited children with the mixed value $v_{\mathrm{mix}}(s)$, then normalizes among the children and rescales,

$$\hat{q}(s, a) \;=\; c_v\big(\nu_0 + N_{\max}\big)\, \frac{\tilde{q}(s, a) - q_{\min}}{\max(q_{\max} - q_{\min}, \epsilon)}, \quad \tilde{q}(s, a) \;=\; \begin{cases} Q(s, a), & N(s, a) > 0 \\ v_{\mathrm{mix}}(s), & N(s, a) = 0 \end{cases} \tag{11}$$

where $q_{\min} = \min_b \tilde{q}(s,b)$ and $q_{\max} = \max_b \tilde{q}(s,b)$ are taken over the children b of s, $N_{\max} = \max_b N(s,b)$, and $\epsilon = 10^{-8}$ gives a zero normalized value when the range collapses. The mixed value is

$$v_{\text{mix}}(s) \;=\; \frac{1}{1+\sum_b N(s,b)}\Big(v(s) \;+\; \Big(\sum_b N(s,b)\Big)\frac{\sum_{b:N(s,b)>0}\pi_\theta(b|s)\,Q(s,b)}{\sum_{b:N(s,b)>0}\pi_\theta(b|s)}\Big) \tag{12}$$

where $v(s)$ is the value of the leaf node. This is the v_{mix} that Eq. (11) assigns to unvisited children. In the rescaling of Eq. (11) the coefficient $c_v(\nu_0 + N_{\max})$ grows as visit counts accumulate, so early in the search $\hat{q}$ does not overwhelm the $\log \pi_\theta$ and Gumbel noise terms, while later on the value dominates; ν_0 sets the lower bound when no visits have been made. That rescaling takes place on top of the min–max normalization among the children, and is therefore distinct from critic calibration, which is an earlier stage.

Computing the iterated Bellman calibration. The original formulation fits the calibrator on offline data with a doubly robust target (van der Laan & Kallus, 2026); here the target is formed from on-policy transitions collected on the board being routed. The 32 stochastic rollouts are split by episode, 16 into $\mathcal{D}_{\text{fit}}$ and 16 into $\mathcal{D}_{\text{hold}}$. Algorithm 1 gives the fit. Each iteration sorts by V_θ and solves an order-constrained least squares problem on the targets $y^{(j)}$, which PAVA answers exactly in $O(M)$ after sorting by merging adjacent blocks that violate monotonicity (Barlow & Brunk, 1972; Best & Chakravarti, 1990). Successor values use linear interpolation, clamped outside the fitted range. After convergence, the map is sampled at up to 64 distinct quantiles of critic outputs from all collected episodes. Hold-out rewards do not enter the fit. For knots (x_j, y_j), the export adds $\epsilon(x_j - x_1)$, where

$$\epsilon = \begin{cases} 0.01\,(y_L - y_1)/(x_L - x_1), & x_L > x_1,\ y_L > y_1, \\ 0.01, & x_L > x_1,\ y_L = y_1, \\ 0, & x_L = x_1. \end{cases} \tag{13}$$

This breaks flat-block ties while retaining monotonicity. The resulting interpolant defines the search map $\hat{g}$ in Eq. (4). Its median value on complete-routing anchors is subtracted from the stored ordinates; an empty anchor set gives offset zero. Thus inference reads an already anchored map, without subtracting the offset again.

The iteration alternates the sample Bellman operator with an order-constrained projection, as in fitted value iteration (Bellman, 1952; Munos & Szepesvári, 2008). We stop when $\|\hat{g}_{j+1} - \hat{g}_j\|_\infty < 10^{-4}$ on the fitting inputs; failure to converge within 4,000 iterations raises an error.

Held-out gate. Before compression and anchoring, the implementation measures Pearson correlation between the fitted map and the following diagnostic on held-out episodes:

$$G_t^{\text{gate}} = \sum_{k=t}^{H-1} \gamma^{k-t}\Delta\Phi_k + \gamma^{H-t}\mathbf{1}[\text{truncated}]\,V_\theta(s_H). \tag{14}$$

True terminal episodes have zero tail bootstrap. At a time limit, the implementation uses the uncalibrated critic to avoid evaluating the map against its own bootstrap. This is a return proxy: its tail is not rescaled from normalized critic units, so the gate diagnostic can depend on reward normalization. It is used only for the sign gate, not as the Bellman fitting target. In the iteration-400 artifacts, 149 of 150 board–training-seed gates are on; the remaining correlation is -0.00282. No separate gate ablation is reported.

A non-decreasing map cannot reverse an incorrect ordering, though it can merge predictions into ties (DeGroot & Fienberg, 1983). The exported adjustment separates such ties inside the knot range. Calibration and its gate precede the node-wise completed-Q normalization; c_v controls the later action-selection scale.

PUCT-family selection rules. The PUCT family used for comparison maximizes the sum of a normalized Q and a prior-based exploration term,

$$a_{\text{puct}} \;=\; \arg\max_{a\in\mathcal{A}(s)}\Big[\;\bar{Q}(s,a) \;+\; c_{\text{puct}}\,\pi_\theta(a \mid s)\,\frac{\sqrt{N(s)}}{1+N(s,a)}\;\Big] \tag{15}$$

Algorithm 1 Board-wise iterated Bellman calibration.

Require: Fitting transitions $\{(v_i, r_i, v'_i, \delta_i)\}_{i=1}^{M}$; critic outputs v_{all} from all collected episodes
Require: Hold-out transitions $\mathcal{D}_{\text{hold}}$; complete-routing critic values V_{done}
Require: Discount γ; tolerance $\tau = 10^{-4}$; iteration cap $J = 4000$
function CALIBRATE($\mathcal{D}_{\text{fit}}, \mathcal{D}_{\text{hold}}, v_{\text{all}}, V_{\text{done}}$)
  $\hat{g} \leftarrow 0$; $u \leftarrow \mathbf{0}$; converged $\leftarrow$ false
  for $j = 1, \ldots, J$ **do**
    $y_i \leftarrow r_i + \gamma(1 - \delta_i)\hat{g}(v'_i)$ for every i
    $(x, z) \leftarrow$ PAVA(v, y) ▷ exact fit at the observed inputs
    $\hat{g} \leftarrow$ LINEARINTERPOLATE(x, z) ▷ clamp to endpoint values outside x
    $u^+ \leftarrow \hat{g}(v)$; $\Delta \leftarrow \|u^+ - u\|_\infty$; $u \leftarrow u^+$
    if $j > 1$ and $\Delta < \tau$ **then**
      converged $\leftarrow$ true; **break**
    end if
  end for
  if not converged **then**
    raise non-convergence error
  end if
  $\rho_{\text{cal}} \leftarrow \text{Corr}(\hat{g}(v_{\text{hold}}), G^{\text{gate}}_{\text{hold}})$
  $\lambda \leftarrow \mathbf{1}[\rho_{\text{cal}} > 0]$
  $x \leftarrow$ UNIQUE(QUANTILES$(v_{\text{all}}, 64)$); $y \leftarrow \hat{g}(x)$
  $y_j \leftarrow y_j + \epsilon(x_j - x_1)$ using Eq. (13)
  $\hat{g} \leftarrow$ LINEARINTERPOLATE(x, y) ▷ clamp outside x
  $a \leftarrow \text{median}(\hat{g}(V_{\text{done}}))$, or 0 if $V_{\text{done}} = \varnothing$
  return $(x, y - a, \lambda)$ ▷ stored knots already include the anchor
end function
function PAVA(v, y)
  sort by v; pool ties using target means and multiplicities as weights
  for each pooled point in increasing v **do**
    append a block containing its fitted value and weight
    while the last two block values decrease **do**
      merge them using their weighted mean and summed weight
    end while
  end for
  return unique inputs and their fitted values expanded from the blocks
end function

where $\bar{Q}(s, a)$ is the tree-wide normalized action value, $N(s)$ is the visit count of the node and the Q of an unvisited child is completed with the normalized value of the parent. Reference implementations of AlphaZero and of MuZero are kept separately, and the two differ in where the tie-break is placed and in the form of the exploration coefficient. MuZero uses a coefficient that grows logarithmically in the visit count of the parent instead of a constant c_{puct}. Root-node Dirichlet noise, progressive widening, and the handling of a child that meets an invalid action during search are kept as add-on modules.

C Network Architecture and PPO Pretraining

Tokenization. Every entity of the observation becomes one token, and the entity class is part of the token: board, outline, net, pad, trace, via, ratsnest edge, route head, and the five candidate classes—pad point, trace endpoint, via center, ratsnest point, and directional. Continuous quantities enter through Fourier features (Tancik et al., 2020) of the coordinates rather than raw values, so that positions at different scales remain distinguishable after projection. Net membership is represented by a learned per-head attention bias on same-net pairs. The training configuration disables net-slot embeddings and design-rule-violation tokens.

Stack and attention zones. The stack is decoder-only over two zones. State tokens attend to one another in both directions, action tokens attend to the whole state zone, and the state zone never

attends to the action zone. This last rule is what makes the K/V of the prefix reusable: appending action tokens cannot change a single state hidden, which is the premise that Algorithm 3 relies on. Within the action zone, attention is causal by default and block-structured when branches are packed. Each residual sublayer is gated by a learned scalar initialized to zero, as in ReZero (Bachlechner et al., 2021), so the network starts as an identity map and the depth is engaged only as training makes use of it. Because the state zone is permutation-equivariant, no positional encoding is applied there; a learned positional embedding is injected on the action tokens alone, where the order of the three slots is meaningful.

Heads. The action type and pointer logits pass through a scaled $\tanh$ before the softmax. The bound this places on the logits suppresses the entropy collapse that a pointer over a large candidate set is otherwise prone to, and it keeps the gradient bounded when a single candidate begins to dominate. The critic is a small multilayer head on a summary of the prefix, and it is the V_θ that §3.2 calibrates per board.

Configuration. The saved training configuration sets the model width to 128, the feed-forward width to 512, the depth to 4 layers, and the number of attention heads to 8, with 32 Fourier frequencies for the coordinate features.

PPO pretraining (prior). We pretrain three PPO priors using seeds 42, 43, and 44 with the settings in Table 5. PPO standardizes advantages within each minibatch. Rewards are divided by the running standard deviation of discounted returns, without mean subtraction, and clipped to $[-10, 10]$; the running discounted returns reset at episode boundaries. Time-limit truncations bootstrap from the critic at the final state, while true terminal states have zero bootstrap. The value-loss coefficient multiplies half the mean squared return error, without value clipping. Training enables observation reflections, axis swaps, and virtual outline scaling around a sampled center; these augmentations are disabled during evaluation.

Table 5: PPO pretraining (prior) settings.

Setting	Value
Training seeds / iterations per seed	$42, 43, 44$ / 400
Parallel environments / steps per environment per iteration	32 / 512
Total environment steps per seed	$6{,}553{,}600$
PPO epochs per iteration / minibatch size	4 / 256
Optimizer / (β_1, β_2)	AdamW / $(0.9, 0.999)$
Optimizer ϵ / weight decay	10^{-5} / 10^{-4}
Learning rate	10^{-4}
Learning-rate schedule	20-iteration linear warm-up, then constant
PPO ratio clipping	0.2
Discount / GAE parameter	0.995 / 0.95
Value-loss / entropy coefficients	0.5 / 0.01
Gradient norm cap	0.5
Episode horizon	512 decisions
Routing corner angle / directional-candidate radius	45° / 1 mm
Engine random seed	77
Shove / follow-branch iteration limits	250 / 10^6

D Determinization of the CAD Engine

The individual items of the determinization summarized in §4.2 are as follows. In total, 16 nondeterminism and stability defects are fixed.

Object identifier issuance. The identifiers (UUIDs) assigned to objects were issued in an order that depended on memory addresses, so the same action sequence produced a different identifier assignment on every run. The issuance order is fixed independently of addresses, and the state of the identifier generator is rewound at every checkpoint and episode boundary, so the same point reproduces the same identifiers.

Address-based tie-break. Several algorithms inside the engine used C++ pointer addresses as the tie-break criterion when comparing candidates. Every tie-break is replaced by one based on identifiers, geometric information, and creation order, and the caches keyed by address are discarded.

Joint-link ordering. Both full and incremental restoration sort the links at routing joints by a stable, address-independent key. This removes the insertion-order dependence that otherwise allows a detour through another checkpoint to change subsequent routing.

Wall-clock time termination condition. Termination of the routing search was decided by a wall-clock time cutoff, so the same input was interrupted at different points depending on system load. The time cutoff is replaced by an iteration cap and time is retained for warnings only. Unnecessary recomputation inside an iteration is removed in the same pass.

Memory-safety defects. Paths that referenced already-released objects remained, so results varied across runs or the process terminated abnormally. These reference paths are blocked, the lifetime rules of the singletons are cleaned up, and the problem of released objects persisting as obstacles is removed.

E Details of the Search Cost Optimizations

E.1 Incremental checkpointing

Before realizing a new child, search restores its parent's checkpoint. A full restore replaces all tracks and vias, then rebuilds the routing world and connectivity, including obstacles derived from unchanged pads, footprints, zones, and the outline. The checkpoint and the current board are instead compared at the level of identifiers (UUIDs) and fields, and the change set splits three ways (Figure 4): objects with the same UUID and equal fields stay; all other current objects are removed, and their target counterparts are cloned and inserted. This includes replacing an object whose UUID is unchanged but whose geometry differs.

Three ordering constraints hold. The difference is computed before any deletion, since deleting first destroys the comparison target. Deallocation of removed objects is deferred until after the connectivity cleanup, since freeing first leaves the connectivity structure pointing at released objects. And the identifier generator is rewound to the restore point, without which the same action sequence receives different identifiers on each run and the determinism of §4.2 breaks. Connectivity recomputes only the ratsnest edges, the insertions and removals being already reflected in the graph, and the DRC state—the violation list and its signature—is stored with the checkpoint, without which the incremental DRC loses its baseline and runs a full check at the next step. The restore also clears address-keyed rule caches before replacing objects and canonicalizes the routing world's spatial index and joint-link order after applying the changes. The saved UUID-generator state is applied last, after routing-session setup, so allocations during restoration do not advance the replay stream. Pads, footprints, zones, and the outline are not replaced, and their routing obstacles are reused. Derived virtual vias are refreshed, while index canonicalization may still traverse the retained obstacles; this path avoids rebuilding the entire routing world.

E.2 Policy-value memoization

The cache key pairs a digest of the observation with the legal action set itself (Algorithm 2, line 3); the digest excludes the static part of the board, which is invariant within an episode. Carrying the legal set is not redundant: the provider reads the action, pointer, mode and net masks live and assembles its output over exactly the set it is handed, so two states sharing a digest but differing in either would be served each other's prior. The implementation includes the ordered list of legal-action triples in the key. The cache stores the prior over legal actions and the configured provider's leaf value. In the calibrated path, z is the output of the loaded isotonic map, whose knots incorporate the terminal-anchor shift. The calibration wrapper runs before memoization; the critic gate, path reward, and depth discount are applied later by search and are not cached. The cache lives for one episode (Figure 5).

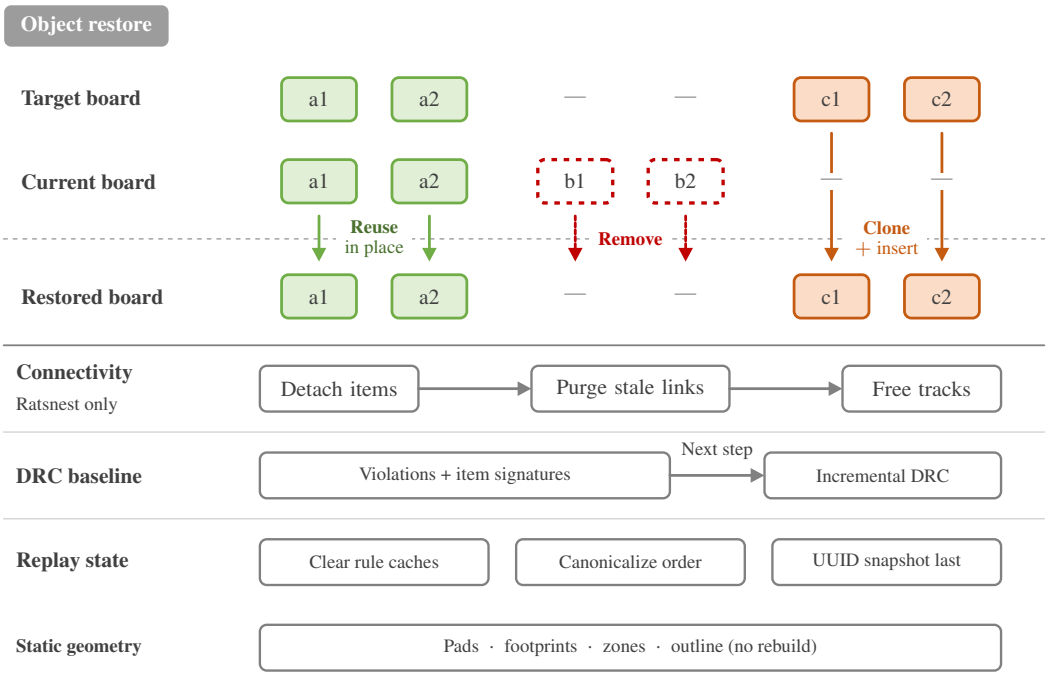


Figure 4: Incremental checkpointing. Columns align the same object across the target, current, and restored boards. The target is the board state saved in the selected checkpoint; dashes denote absent objects. The difference between the checkpoint and the current board is computed at the identifier and field level, and only the changed tracks and vias are inserted or removed. Same-UUID objects with changed fields are also replaced. Static geometry and its obstacle objects are retained; auxiliary indices are canonicalized.

Algorithm 2 Policy-value memoization with an episode-scoped cache and generational eviction.

state: cache $\mathcal{M}$ and touched set $\mathcal{T}$, initially empty
function POLICYVALUE($o, \mathcal{A}(s)$)
  $k \leftarrow (\text{FINGERPRINT}(o), \mathcal{A}(s))$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{k\}$
  if $k \in \mathcal{M}$ **then**
    return $\mathcal{M}[k]$ ▷ reuse the provider output
  end if
  $(\pi, z) \leftarrow \text{PROVIDER}(o, \mathcal{A}(s))$ ▷ one provider evaluation
  $\mathcal{M}[k] \leftarrow (\pi, z)$ ▷ before path reward and discount
  return (π, z)
end function

function NEWGENERATION( ) ▷ after each committed action
  $\mathcal{M} \leftarrow \{(k, v) \in \mathcal{M} : k \in \mathcal{T}\}$
  $\mathcal{T} \leftarrow \emptyset$
end function

A hit returns the cached provider output for the same observation and legal-action set. Restoration reinstates the observation and pointer tables saved with the node, or rebuilds them when that bundle is unavailable. An optional audit recomputes cache hits and checks for disagreement.

With a deterministic provider, identical masks, the same search randomness, and a fixed simulation budget, cache hits preserve the priors and leaf values and hence the search trajectory, apart from hash collisions. A wall-clock budget is different: saved computation can permit more simulations. The cache contains neither tree statistics nor transition results.

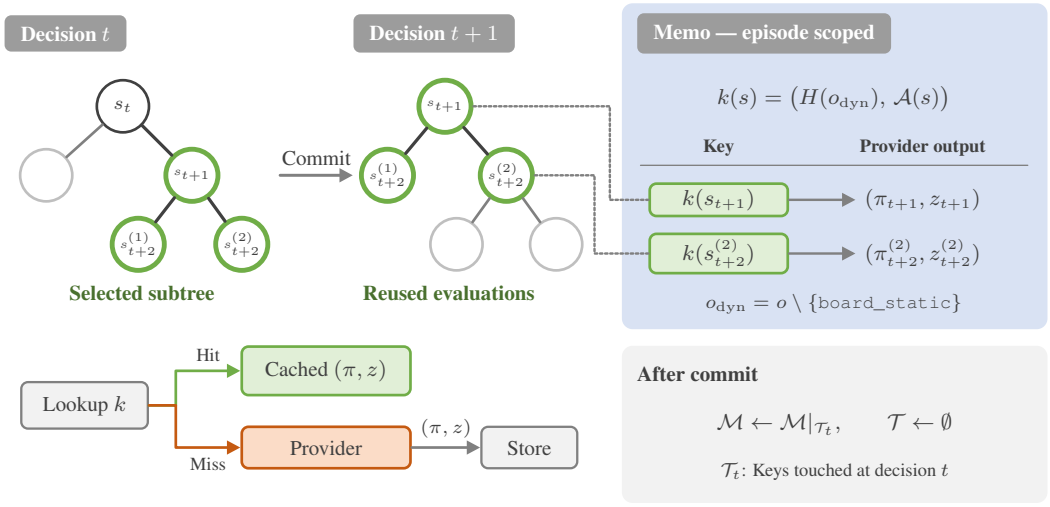


Figure 5: Policy-value memoization. Committing a_t makes the previously evaluated state s_{t+1} the next root; its simulated successors $s^{(j)}_{t+2}$ can reappear in the new search. Matching state labels identify reused legal-action priors π and provider values z, not transferred tree statistics. Superscript j indexes alternative successors. The cache key pairs the observation fingerprint with the legal-action set. After a commit, only entries touched during the preceding decision are retained.

Cross-decision hits arise when states evaluated below the committed action reappear in the next decision's newly built tree. The previous tree and its checkpoints are released; only matching provider evaluations are reused. Excluding the static part of the board from the key targets the same boundary: removing the large fields that do not change within an episode from the hash reduces the cost of computing the key without altering which lookups hit.

Entries are evicted generationally rather than by capacity (Algorithm 2, NEWGENERATION). The driver calls it once the decision commits an action, and it keeps only the keys that the just-finished decision touched. This retains recently evaluated states, including candidates beneath the committed action, while older entries expire. An evicted state is recomputed if revisited, so eviction affects cache hits rather than the provider's output.

E.3 BRANCH-PACKED DECODING

A model call proceeds in three stages (Figure 6). The state prefix of length L is first encoded once to obtain the K/V, action-type logits, and critic value. The prefix consists of the board, the nets, the pads, the tracks and vias, the candidates, and the route head. For prefix length L and hidden width d, the prefix attention work is $\Theta(L^2 d)$ per layer and is paid once per node.

The packed pass appends $T + PK_k$ tokens: one for every model action-type slot, including masked types, and one pointer token for each pair of an unmasked pointer-and-mode type and a candidate retained by the shared pointer mask. Here P counts these types and $K_k \leq K$ counts retained candidates. Type-specific masks, such as the off-layer restriction for `make_line`, are applied to the pointer logits; they do not prune the packed token list. Types that do not use a pointer-conditioned mode need no appended pointer token: their pointer logits, when needed, are read directly from the action-type hidden state and the encoded net or candidate tokens. Let H denote the encoded state prefix, $h^{(i)}_{\text{type}}$ the decoded state of the type-i token, and $h^{(i,j)}_{\text{ptr}}$ the decoded state of the candidate-j token in the branch under type i. For an action using both a pointer and a mode, with $i = a_{\text{type}}$ and $j = a_{\text{ptr}}$, the factorization can be written in terms of the representations used by the output heads as

$$P(a \mid s) = P(a_{\text{type}} \mid H) \cdot P(a_{\text{ptr}} \mid h^{(i)}_{\text{type}}, H) \cdot P(a_{\text{mode}} \mid h^{(i,j)}_{\text{ptr}}). \tag{16}$$

Each factor denotes the corresponding masked, softmax-normalized head output. The pointer head scores all targets from the type-token state and H; the pointer-branch state then supplies the full vector

Algorithm 3 Branch-packed decoding with one state encode and one incremental branch decode.

function DECODE(o, masks)
  $(H, \mathrm{KV}, \ell_{\mathrm{type}}, V_\theta) \leftarrow$ ENCODESTATE(o, masks) ▷ type logits and critic from the prefix
  $\mathcal{P} \leftarrow$ unmasked types using both a pointer and a mode; $P \leftarrow |\mathcal{P}|$
  $\mathcal{K} \leftarrow$ candidates surviving the pointer mask; $K_k \leftarrow |\mathcal{K}|$
  $E \leftarrow [\mathrm{at}(i)]_{i=0}^{T-1} \,\|\, [\mathrm{pt}(j)]_{p\in\mathcal{P}, j\in\mathcal{K}}$ ▷ T action types; $n = T + PK_k$ tokens
  $M \leftarrow -\infty$ in $\mathbb{R}^{n\times n}$; $M_{ii} \leftarrow 0$
  for all $(p, k) \in \mathcal{P} \times \mathcal{K}$ **do**
    $M_{\mathrm{idx}(p,k),\mathrm{idx}(p)} \leftarrow 0$ ▷ each pointer branch sees its own type token
  end for
  $h \leftarrow$ DECODEAPPENDED(KV, E, M) ▷ all branches also see the prefix
  read pointer logits from (h_{type}, H) and mode logits from h_{ptr}
  apply the corresponding pointer, net, layer, and mode masks
  return ℓ_{type}, pointer and mode logits, and the prefix value V_θ
end function

of routing-mode logits. No separate mode token is appended. Other action types omit whichever factors their slots do not use. The joint scores are subsequently normalized over the enumerated legal set: $\pi(a \mid s) = P(a \mid s) / \sum_{b\in\mathcal{A}(s)} P(b \mid s)$ for $a \in \mathcal{A}(s)$.

Finally, a block-structured mask over the appended tokens lets each branch attend to the prefix, to its own ancestors and to itself, but not to the other branches. That isolation is the correctness condition: were branches to see one another, a candidate's hidden state would depend on what was packed beside it and the same node would evaluate differently in different packs. The mask is a substructure of the full $(T + P \cdot K)^2$ mask rather than a causal one, and under it each branch sees exactly the context it would see alone. One call yields the action type logits, the pointer logits, pointer-conditioned mode logits and the critic value together, so each candidate's joint probability is assembled without replicating the prefix K/V.

Let $n = T + PK_k$. The implementation uses dense attention with an additive mask, so its per-layer attention work is $\Theta(L^2 d + n(L + n)d)$, including the $n^2 d$ work among appended tokens. When $n \ll L$, the incremental part is approximately $\Theta(nLd)$; the block mask enforces isolation but does not itself provide sparse computation. Projection and feed-forward work add $\Theta((L + n)d^2)$ for a fixed expansion ratio. The per-candidate reference repeats the prefix encoding for each legal action, incurring $\Theta(|\mathcal{A}(s)|L^2 d)$ prefix attention work. Pruning pointer branches reduces n; when no pointer-and-mode type is enabled, only the T type tokens are decoded.

One caveat: dropping the unreachable rows shortens the key axis, shifting the reduction order of the softmax and the matrix product, so the kept rows are mathematically unchanged but not bit-identical to a full decode. The residual is at the level of floating-point rounding and shrinks by orders of magnitude in double precision, which identifies rounding rather than semantics as its source. The action type tokens, which no pruning touches, come out bit-identical.

E.4 ACTION MASKING

Masking proceeds in two layers (Figure 7). The candidate generation layer asks the engine for the connected cluster holding the current route head and removes matching existing-copper targets while routing. Matching uses position and copper layer. Connecting within that cluster brings no connectivity gain, although it can still alter geometry. Directional proposals bypass this cluster test because they are generated coordinates rather than existing-copper anchors; they remain subject to the other candidate and action masks.

The action enumeration layer applies separate masks for action type, net, routing mode and candidate pointer, including the `make_line`-specific off-layer pointer mask, expressing the same conditions under which the policy drives the corresponding logits to $-\infty$ (Huang & Ontañón, 2022). That agreement is required because the prior must be normalized over the set the search branches on; if one mask is missing, the search expands an action the policy gives no probability, or the converse, and the two supports diverge. The legal action is a tuple of type, pointer and mode, the format the policy already encodes. The set size gives the number of available children; a widening rule may

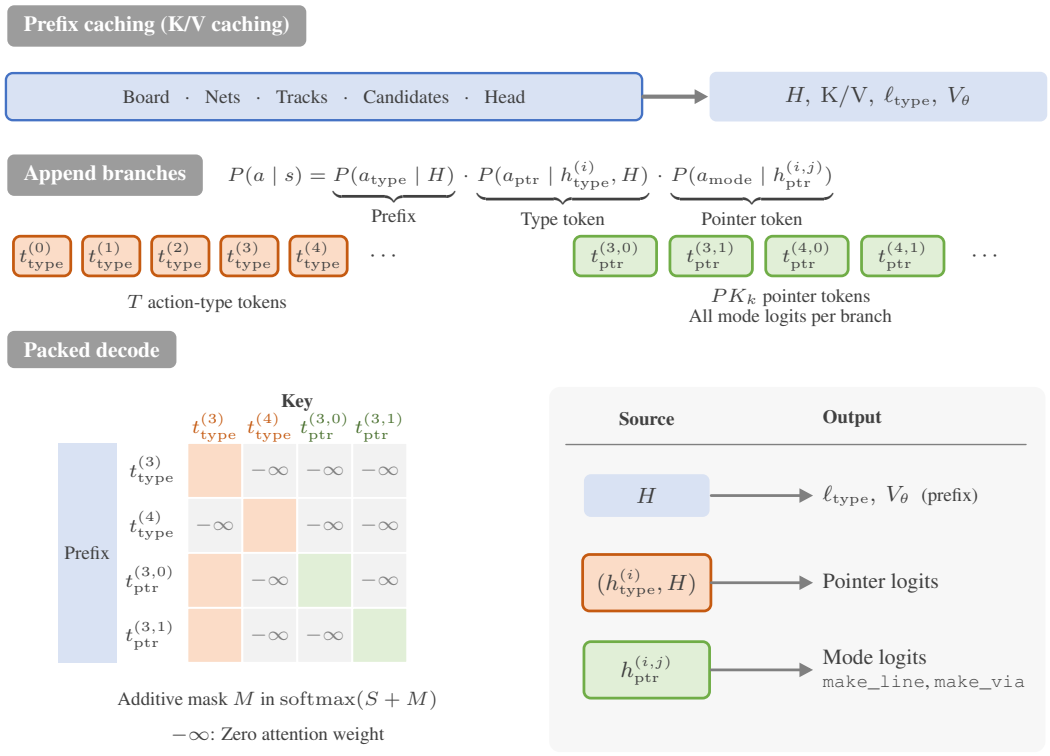


Figure 6: Branch-packed decoding. The prefix is encoded once, and branches are decoded together with cross-branch attention blocked. Tokens t and decoded states h are indexed by type i and candidate j. Colored mask cells have zero additive bias; orange and green denote type and pointer keys. Each pointer-branch state supplies all mode logits, without a separate mode token.

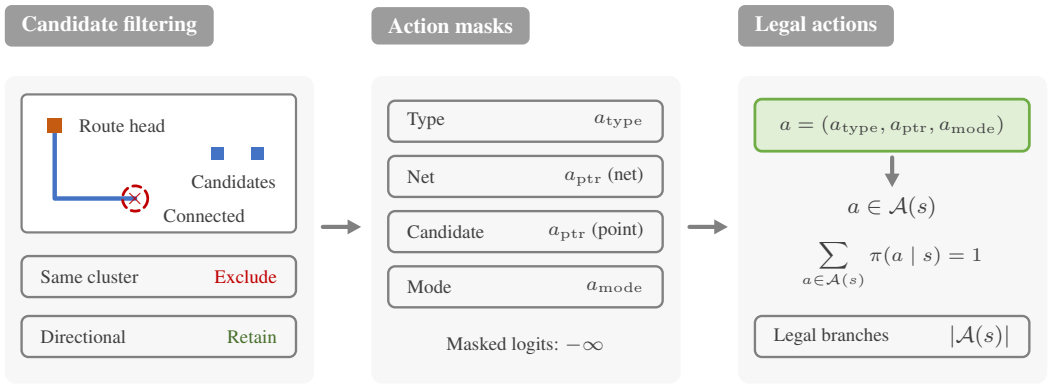


Figure 7: Action masking. The candidate generation layer excludes the connected cluster of the route head, while directional candidates bypass this cluster filter, and the action enumeration layer builds the legal action set from four kinds of masks. The masks act on the type, net-pointer, candidate-pointer, and mode slots; the policy prior and tree expansion use the same legal action set $\mathcal{A}(s)$. The candidate mask includes type-specific layer restrictions. $|\mathcal{A}(s)|$ counts available branches, before any widening rule.

expose only a subset for selection. Prior-computation cost also depends on prefix length and retained candidate tokens, so it is not simply proportional to $|\mathcal{A}(s)|$.

Algorithm 4 Invalid-action filtering after executing a newly selected child under the `pop` rule.

Require: Parent n at depth d; unrealized child c; discounted return R_{path} to n

function REALIZECHILD(n, c, R_{path}, d)
  RESTORE(n.state)
  $(r, \text{done}, \text{invalid}) \leftarrow$ STEP(c.action) ▷ $r = \Delta\Phi$; flags from the environment
  c.state $\leftarrow$ CHECKPOINT(); c.reward $\leftarrow r$
  $R' \leftarrow R_{\text{path}} + \gamma^d r$
  if done **then** ▷ termination takes precedence over invalidity
    mark c terminal with value R'; BACKUP(c, R')
    return DONE
  else if invalid **then**
    RELEASE(c.state); c.state $\leftarrow \varnothing$
    remove c from n.children ▷ no backup for the removed edge
    if n.children $= \emptyset$ **then**
      RESTORE(n.state); mark n terminal with value R_{path}
      BACKUP(n, R_{path})
      return DONE
    end if
    return RETRY ▷ reselect at n within the same simulation
  end if
  evaluate c and expand it unless the search depth cap is reached
  compute G_{sim} by Eq. (4) with path return R' and depth $d + 1$
  BACKUP(c, G_{sim})
  return DONE
end function

E.5 INVALID-ACTION FILTERING

Algorithm 4 handles a selected child that has not yet been executed (Figure 8). The parent is restored, the action is executed, and the child state is checkpointed. Terminal outcomes are backed up first. Otherwise, an action classified as failed or ineffective by the environment is removed and its checkpoint released without backup; search then reselects at the parent within the same simulation. A previously realized child is traversed using its stored reward and state, without re-executing the action.

The discarded attempt still pays for restoration, execution, checkpoint creation, and release. While siblings remain, it adds no backup, leaving N, Q, stored min–max statistics, and the completed-simulation count unchanged. Removing the edge changes the selectable child set, so subsequent selection scores can change even without a backup. The environment's invalidity flag, rather than a board-equality test, determines removal.

The removal loop terminates because the child list decreases monotonically: one removal deletes one child and creates none, so removals within a simulation are bounded by the children remaining along the descent. A node whose children are all removed is treated as terminal, backing up the discounted return to that node once; without this the search halts at an empty node and only consumes budget.

The alternatives change both tree contents and budget accounting. The `keep` rule retains a failed action as a penalized self-loop, allowing further expansion. The `penalize` rule backs up a penalized value once and removes the child, spending a simulation on that failure. The deployed `pop` rule instead retries without backing up the failed edge. Table 3 compares these three rules.

F SYNTHETIC TRAINING DATASET

The prior is trained with PPO (Schulman et al., 2017) on two-layer synthetic boards from the procedural generator in PCBWorld (Song et al., 2026). Table 6 gives the settings, with board scale and net composition informed by evaluation-set statistics. Figure 9 illustrates boards across pad-density deciles.

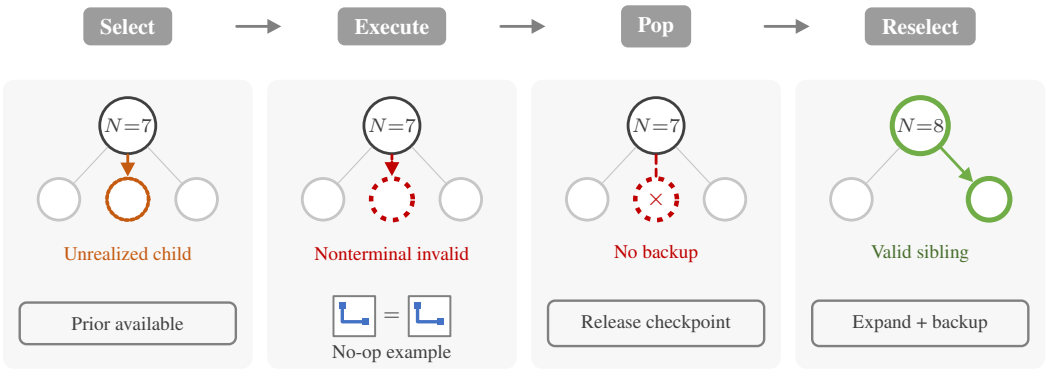


Figure 8: Invalid-action filtering. A nonterminal child classified as invalid is removed without backup, and the same simulation reselects among the surviving siblings. The unchanged board illustrates a no-op, one possible invalid outcome. While siblings remain, the discarded attempt adds no backup or completed simulation. The illustrated increase from $N = 7$ to $N = 8$ occurs only after the valid sibling's backup. If none remain, the parent is treated as a dead end and its path return is backed up.

Table 6: Procedural training-data settings. Board-scale and net-composition targets use evaluation-set statistics.

Generated quantity	How it is determined	Setting used
Nets per board	Discrete uniform draw, guided by evaluation nets with at least two pads	13–27 nets (mean 20)
Pads per net	Discrete power law of the evaluation set with a reinforced power-law tail	Zipf $s = 2.955$; $P(k \geq 16) = 1.8\%$
Spatial placement of pads	Pads of the same net placed adjacently, with locality decreasing as the net grows	locality 0.7; linear decay up to 10 pins
Board size	Pad counts sampled first, then the area set to match the total number of pads	Variable with the sampled pad count
Layer composition of pads	SMD and thru-hole pads mixed to generate nets that require vias and nets that do not	thru-hole probability 0.30
train/validation/test split	Training and validation of the PPO prior	Train 100,000 / validation 128 / test 128

The reference net-count mean is 19.06: 953 nonzero nets with at least two electrical pads across the 50 evaluation boards. Table 7 instead counts all engine net entries, including single-pad nets and the unassigned net 0 (1,302 entries; mean 26.04). Evaluation uses real boards outside the generated training set. Matching selected statistics does not establish that the generator captures realistic placement, obstacles, congestion, or layer usage. Improving this training distribution remains an untested direction for strengthening the prior.

G Evaluation Boards

The 50 evaluation boards come from PCBWorld-Bench's 287-board `medium` tier (Song et al., 2026), including all 10 official test boards (marked * in Table 7) and 40 others. None is used for policy training. The subset was fixed before these experiments. It contains 35–158 pads, 10–110 engine net entries, and 19–67 required connections per board. Median pad and net-entry counts are 76 and 23, versus 71 and 20 in the remaining 237 boards; the subset excludes the tier's largest designs, which reach 668 pads.

Both training and evaluation boards use two copper layers. The evaluation boards span 10–110 net entries and have more varied outlines and component placement than the synthetic examples in Figure 9. The policy-only difficulty groups below are separate from the benchmark's structural tiers.

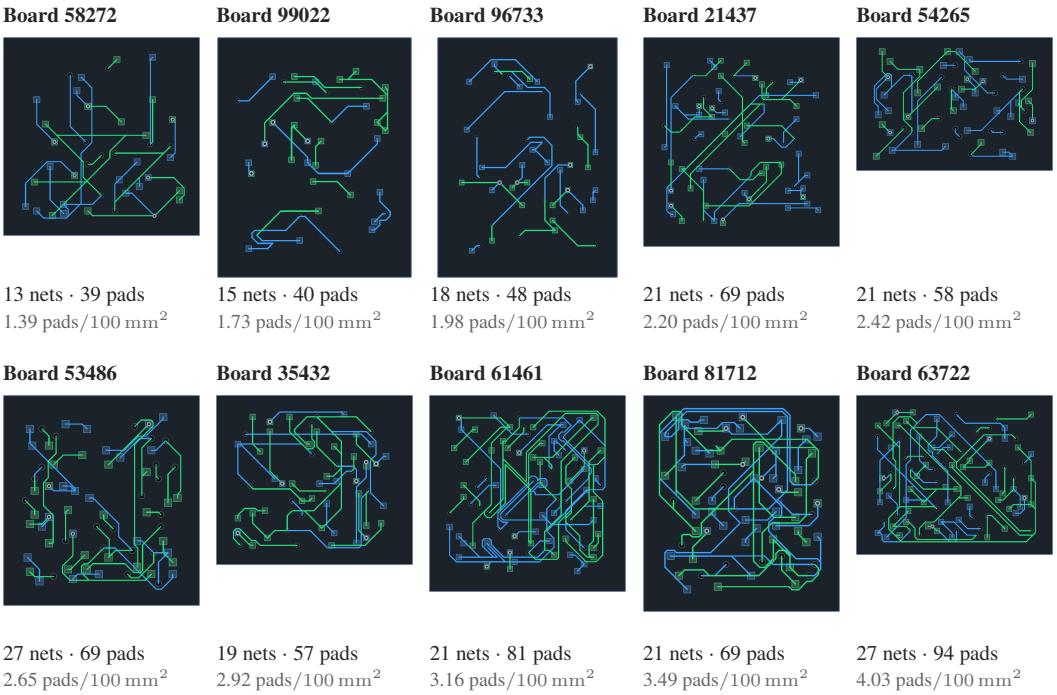


Figure 9: Synthetic training-board examples, ordered by pad density. From a random sample of 1,500 training boards, one board was selected near the median of each density decile. Images show selected PPO-routed layouts for illustration; training starts from the corresponding unrouted boards.

Difficulty assignment. We define difficulty using PPO's per-draw clean-pass rate, excluding the evaluated checkpoint 400. Let b identify a board, $c \in \mathcal{C} = \{50, 100, \ldots, 350\}$ a training checkpoint, $s \in \mathcal{S} = \{42, 43, 44\}$ a training seed, and $r \in \mathcal{R} = \{0, \ldots, 15\}$ a rollout seed. For each paired instance, $C_{b,c,s,r}$ is one if the *first* PPO rollout completes every required connection with zero error-severity design-rule violations, and zero otherwise. Additional PPO rollouts in the time-budgeted pool do not enter this indicator. The board's score is

$$\widehat{p}_b = \frac{1}{|\mathcal{C}|\,|\mathcal{S}|\,|\mathcal{R}|} \sum_{c\in\mathcal{C}} \sum_{s\in\mathcal{S}} \sum_{r\in\mathcal{R}} C_{b,c,s,r} = \frac{1}{336} \sum_{c,s,r} C_{b,c,s,r}. \tag{17}$$

All 336 observations receive equal weight; a higher score means the board is easier for the policy. We sort the 50 boards by decreasing $\widehat{p}_b$, breaking ties by increasing numeric board identifier. With j_b denoting the resulting one-based rank,

$$\text{difficulty}(b) = \begin{cases} \text{Easy}, & 1 \le j_b \le 17, \\ \text{Medium}, & 18 \le j_b \le 34, \\ \text{Hard}, & 35 \le j_b \le 50. \end{cases} \tag{18}$$

These are rank-based groups of 17, 17, and 16 boards, rather than fixed success-rate thresholds. Their observed score ranges are 0.506–1.000, 0.110–0.503, and 0.000–0.065, respectively. We fix the labels before stratifying iteration 400, yielding $17 \times 3 \times 16 = 816$, 816, and 768 paired instances in the three groups. Neither PW-MCTS outcomes nor iteration-400 outcomes enter the labels. The groups measure difficulty for the evaluated PPO policies; they are distinct from PCBWorld-Bench's structural tiers, whose `medium` tier contains all 50 boards here.

Table 7: The 50 evaluation boards. Net entries include single-pad nets and the unassigned net 0. `Req.` is the number of connections a clean pass requires, the denominator of routability (6). ⋆ marks the 10 boards of the benchmark's official test split. Difficulty follows Eqs. (17)–(18). Horizontal rules separate Easy (17), Medium (17), and Hard (16); boards within each group are ordered by numeric identifier.

Board	Pads	Net entries	Req.	Difficulty
0110_NodeMcu-v3-LiPoly-LiIon-Backpack_NodeMcuLiPolyBackpack	35	10	24	Easy
0113_maytal_Maytal⋆	73	43	20	Easy
0128_kitspace_4_switch_array	44	17	24	Easy
0140_eBUS-Adapter_Groeger	43	14	27	Easy
0144_ottawa-badges-2016_ottawa-badge-tagger-2016⋆	41	12	30	Easy
0152_mechkeys_isp	45	13	31	Easy
0170_hackaday_esp-14_power_meter__autosave-esp-14	44	23	22	Easy
0174_sensorboard_DiffIR⋆	47	16	32	Easy
0195_kitspace__autosave-postcard	49	13	37	Easy
0212_kitspace_trans_switch_volt_amp	158	110	36	Easy
0224_one-shift-register_one-shift-register	84	52	33	Easy
0237_Paperino_HW_Paperino_shield	64	25	36	Easy
0260_NiMH-Charger_NiMH Charger⋆	80	21	45	Easy
0268_m2-electronics_m2pa-instrument	69	10	34	Easy
0319_magic-table_etch-a-sketch_cyclone	102	25	54	Easy
0349_kitspace_8_switch_array	90	33	54	Easy
0359_kitspace_tt_nano_HAT_b2	96	33	56	Easy
0103_LizardBoard_lizard	37	14	19	Medium
0134_continuity-tester_continuity-tester	40	11	30	Medium
0158_opentilt_opentilt1	64	19	24	Medium
0182_Teensy-Hats_Temperature-Monitor-Hat	72	18	29	Medium
0190_LAUNCHXL-F28027-isolation-PCB_project1	48	13	35	Medium
0206_rgb-strip-controller__autosave-rgb-strip	58	17	39	Medium
0218_kitspace_GPSMux	71	20	36	Medium
0243_sonic3_feram_adapter_sonic3_feram_adapter	64	29	34	Medium
0273_omega-dock-new_omega-dock-new	93	42	49	Medium
0279_powerfreq-hardware_powerfreq	85	32	47	Medium
0288_raspberrypi-3-usb-hub_usb_hub⋆	97	33	56	Medium
0297_scimpy_crossover	76	22	51	Medium
0303_ESP-Breakout_ESP-Breakout	76	23	54	Medium
0355_imfr-schematics_Telescopio	116	33	55	Medium
0364_scimpy_volumebuffer	94	24	67	Medium
0380_Minitel_driver_board	105	43	63	Medium
0385_rgb-led_rgb-led-v2	104	35	65	Medium
0122_Usb-Serial-Breakout-Cp2102_cp2102	63	19	24	Hard
0164_yamsek.kicad_i2c_bridge_primary	48	11	34	Hard
0203_MOD-MPU9150_mod-mpu9150⋆	65	27	38	Hard
0232_ATtiny461Breakout_ATTiny461DevBoard⋆	58	21	38	Hard
0249_RN2483shield_RN2483shield	119	15	48	Hard
0255_audio_relay_input_switch_relay_switch	81	23	41	Hard
0291_C-BISCUIT_buck-reg-5v	76	17	55	Hard
0309_ESPLux_Board	87	28	54	Hard
0316_kicad-workshop_fancyboard⋆	101	47	55	Hard
0325_nanoSwinSidC_nanoSwinSidC	92	28	52	Hard
0329_NightClock_ESPBoard	94	28	63	Hard
0334_esp-serial-terminal_esp-com	108	23	59	Hard
0339_TTNEnschedeMote_ArduinoNanoRN2483	109	31	52	Hard
0344_mavbridge_mavbridge⋆	102	21	64	Hard
0370_Biscay_Blueeye_mcu	100	37	64	Hard
0376_cat-trainer_teensy_base_pcb⋆	98	28	67	Hard

H EXPERIMENTAL CONFIGURATION

Tables 1 and 2 compare PPO with PW-MCTS with $n_{\mathrm{sim}} \in \{4, 8, 16, 32\}$ simulations per decision. Within each training seed, all methods share the same pretrained policy; search also shares the fixed board-wise calibrator and terminal anchor of Eq. (4). PCBWorld (PPO)* denotes our implementation of the PPO baseline with this work's implementation optimizations, abbreviated PPO in the text. Table 8 summarizes the settings. The evaluation boards and fixed policy-only difficulty groups are specified in Appendix G. The separate PCBWorld column in Table 2 uses one LLM attempt per board (Appendix J) and does not change these quality cohorts.

Table 8: Settings for the equal-draw and matched-time comparisons. Candidate-pool sizes are per board and training seed.

Evaluation set	50 boards from PCBWorld-Bench
Training seeds / checkpoint	$42, 43, 44$ / iteration 400
Prior	PPO with wirelength-normalized reward
Search	Gumbel MCTS
Simulations / root candidates	$4/2$, $8/4$, $16/8$, $32/16$
Search depth cap	6
Gumbel / value scales	$c_g = 0.25$, $c_v = 0.05$, $\nu_0 = 50$
Horizon / discount	512 decisions / $\gamma = 0.995$
Invalid-action handling	`pop`
Ratsnest-stagnation stopping	50 steps for search; disabled for PPO
Calibration	32 policy rollouts; 16 fit / 16 hold-out
Calibrator export	64 quantile knots; strict adjustment 0.01; complete-routing anchor
Equal-draw rollout seeds	0–15; 2,400 instances per method
PW-MCTS (4), (8), (16), (32) pools	Up to 128/64/32/16 trajectories
PPO candidate pool	All recorded rollouts; truncated for Table 2
Common time limit	Sum of 16 PW-MCTS (32) trajectory durations
Timed region	Rollout execution; excludes initialization and calibration
Hardware / parallelism	One NVIDIA A10 node per training seed; 32 concurrent cells

Data collection and timing. Each original paired instance runs one PW-MCTS (32) trajectory, then samples PPO in the same process for that trajectory's measured duration. PPO finishes its final rollout, so the collected pool averages 257.1 trajectories per board and training seed and uses 4.1% more aggregate time. These full pools are retained for the diagnostics identified in Appendices I and L; Table 2 applies the stricter replay cutoff below. PW-MCTS (4), (8), and (16) are collected separately with extended rollout seeds. All times exclude startup, policy and calibrator loading, environment creation, and calibration collection and fitting. Replay sums recorded episode durations, rather than elapsed wall-clock time of parallel jobs.

Evaluation score and selection. Every candidate is scored before selection with wirelength penalty $0.3\,\mathsf{WL}/\max(h_{\mathcal{B}}, w_{\mathcal{B}})$, via penalty $0.1\,\mathsf{NV}$, and error-severity design-rule penalties. BoN selects the largest lexicographic key $(\Phi, \rho, -\mathrm{DRV}_{\mathrm{err}}, -\mathsf{WL}, -\mathsf{NV})$. A clean pass requires complete routing and $\mathrm{DRV}_{\mathrm{err}} = 0$. Additional promoted violations are diagnostic counts excluded from FP and clean pass. The same evaluation score is used for the invalid-action ablation; checkpoint-specific selection conventions are documented in Appendix I.

Equal draws. Table 1 pairs methods by board, training seed, and rollout seed. PPO contributes the first rollout ($r = 0$) from each collected instance; PW-MCTS contributes the trajectory with the corresponding rollout seed, using only seeds 0–15. CP, routability, and FP use all 2,400 instances, including failures. Entries are means and sample standard deviations over trajectories, without BoN selection. Every board contributes 48 instances, so no success-based filtering changes board weights.

Matched-time replay. For each board b and training seed s, Table 2 uses the sum of 16 PW-MCTS (32) durations as the time limit $T^{32}_{b,s}$. PPO is replayed in within-cell rollout-index order, then rollout-seed order; PW-MCTS follows ascending rollout seeds. Replay stops before the first trajectory whose completion would exceed the limit. Mean retained counts are 246.06 for PPO and

97.12/56.26/29.73/16.00 for PW-MCTS (4), (8), (16), and (32). The PW-MCTS (4), (8), and (16) recordings reach the time limit in 129, 87, and 79 of 150 board–seed pairs. Earlier-ending pools contribute their last observed incumbent without extrapolation: methods receive the same allowance but need not spend all of it. Figure 3 shows this replay over budget fractions; Appendix N gives the time-axis definition and board-wise curves.

Board-level aggregation. Table 2 selects a winner separately for every method, board, and training seed. CP and routability average the three selected outcomes within each board, then report the mean and sample standard deviation across equally weighted boards. Quality metrics use only training seeds in which all five PPO/PW-MCTS winners pass clean, with the same seed subset for every method before averaging within boards. The resulting quality cohorts contain 17 easy, 17 medium, and 10 hard boards, supported by 51, 51, and 18 jointly clean seed results. Pass requires at least one jointly clean seed, not success in all three seeds; Other means no jointly clean seed, not failure by every method. These quality statistics are conditional on success. Reference uses the same board cohorts. The six remaining hard boards (0122, 0249, 0255, 0291, 0309, and 0325) have no all-method-clean seed; their routability uses all three seeds with equal board weights. N averages completed candidate counts over every board and seed, regardless of success. FP leads below 0.01, the source score-log resolution, are not ranked in bold.

Calibration pool and cost. Calibration collects 32 PPO trajectories per board and training seed, totaling 4,800 trajectories, with at least one fully routed candidate on 39–40 of the 50 boards per seed. Table 9 adds this same pool to each method's retained candidates from Table 2. For each board and training seed, T^{32} is the time limit from 16 PW-MCTS (32) trajectories; shared collection time C shifts the allowance to $C + T^{32}$. Routed counts boards with at least one fully routed candidate, averaged over seeds; it is not CP or score-based BoN selection, since calibration DRC and final scores are unavailable. C is unmeasured and fitting overhead is excluded, so this is a candidate-reuse diagnostic, not an end-to-end matched-time comparison.

Table 9: Shared calibration candidates at the PW-MCTS (32) time cutoff.

	T^{32}		$C + T^{32}$	
Method	N	Routed (/50)	$N + 32$	Routed (/50)
PCBWorld (PPO)*	246.1	42.3	278.1	43.0
PW-MCTS (4)	97.1	44.7	129.1	45.0
PW-MCTS (8)	56.3	44.7	88.3	45.0
PW-MCTS (16)	29.7	44.0	61.7	44.3
PW-MCTS (32)	16.0	43.3	48.0	43.7

Additional sweeps and ablations. The checkpoint sweep evaluates iterations 50–400 in steps of 50 across all three training seeds, totaling 19,200 paired instances. System and invalid-action ablations use training seed 42 at iteration 400, with rollout seed 0 for system measurements and all 16 rollout seeds for invalid-action comparisons. The simulation-budget sweep uses all three training seeds. The calibration ablation pairs 16 rollout seeds on all boards and three training seeds at iteration 400, using 32 simulations. Results are in Appendices I and K.

I Per-Checkpoint Results

Figure 10 and Table 10 evaluate PW-MCTS (32) at eight checkpoints and three training seeds. Mean per-draw CP gains range from 11.8 to 23.7 percentage points; BoN coverage differences range from -5.0 to $+2.3$ boards. Refinement consistently improves individual attempts, while its time-budgeted benefit varies with the prior. The MDP is fixed throughout.

BoN retains the collection protocol of Appendix H. At checkpoints 50–350, board-level winners use the aligned score, but PPO candidates are restricted to the cell winners saved under the original score. Full PPO pools are unavailable for reselection; we therefore report only coverage and routability for these checkpoints. Iteration 400 uses aligned full-pool selection, before Table 2's strict time cutoff.

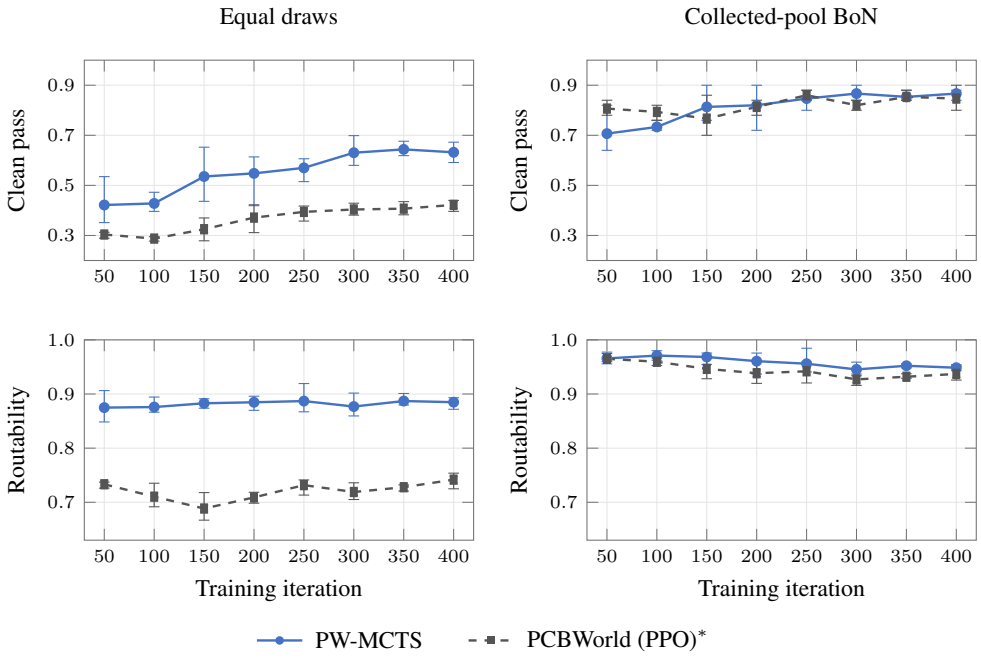


Figure 10: PW-MCTS (32) checkpoint sweep. Lines show three-seed means and bars their ranges. Columns compare per-draw outcomes and BoN from the collected pools. BoN follows the candidate-selection protocol above; CP counts clean selected winners among 50 boards.

J External Routing References

Table 11 places the PPO and PW-MCTS outcomes of Table 2 alongside external routing references: PCBWorld (Astra), Freerouting, and the original routed boards. The controlled comparison evaluates search against sampling from the same pretrained PPO policy; these references contextualize completion and route quality beyond that prior. Freerouting uses 16 attempts per board, all scored from full-precision records and selected using the same evaluation key (Appendix H). PCBWorld (Astra) uses GPT-6-astra with medium reasoning effort and one attempt per board: the v7 prompt, four-step history, a 512-step limit, and termination after 10 steps without geometry progress. All 50 boards start unrouted; 29 finish clean. Its exported evaluation FP is aligned to the paper's wire-length normalization, excluding promoted warnings. These reference runs are not time-matched to PPO/PW-MCTS; Freerouting runtimes are unavailable.

Panel (a) keeps the evaluation set fixed: FP reflects both incomplete routing and route quality. Freerouting has the highest CP and FP among the evaluated routers. Panel (b) describes completed layouts and their differences from the same original boards; direct comparisons between PPO and PW-MCTS retain the common clean cohorts of Table 2.

Table 10: PW-MCTS (32) checkpoint results. CP and routability use one search trajectory and the first PPO rollout per instance. Completed boards counts clean BoN winners out of 50 under the collection protocol and score conventions above.

			CP		Rout.		Completed boards	
Seed	Iteration	Instances	PCBWorld (PPO)*	PW-MCTS	PCBWorld (PPO)*	PW-MCTS	PCBWorld (PPO)*	PW-MCTS
42	50	800	0.310	0.535	0.737	0.906	40	41
42	100	800	0.295	0.472	0.735	0.894	40	38
42	150	800	0.279	0.436	0.718	0.891	37	38
42	200	800	0.311	0.419	0.719	0.896	39	36
42	250	800	0.357	0.515	0.741	0.919	42	40
42	300	800	0.401	0.699	0.716	0.902	42	45
42	350	800	0.435	0.676	0.732	0.880	42	44
42	400	800	0.440	0.672	0.754	0.893	45	44
43	50	800	0.289	0.351	0.726	0.848	39	33
43	100	800	0.278	0.396	0.704	0.867	38	36
43	150	800	0.326	0.517	0.680	0.874	35	39
43	200	800	0.422	0.614	0.710	0.888	42	45
43	250	800	0.417	0.589	0.741	0.875	44	43
43	300	800	0.381	0.613	0.705	0.869	40	42
43	350	800	0.404	0.636	0.731	0.901	42	42
43	400	800	0.396	0.632	0.725	0.890	40	44
44	50	800	0.312	0.379	0.737	0.870	42	32
44	100	800	0.290	0.415	0.692	0.866	41	36
44	150	800	0.370	0.652	0.667	0.884	43	45
44	200	800	0.380	0.611	0.698	0.870	41	42
44	250	800	0.407	0.606	0.713	0.867	43	44
44	300	800	0.429	0.580	0.736	0.860	41	43
44	350	800	0.383	0.619	0.721	0.880	44	42
44	400	800	0.430	0.591	0.747	0.872	42	42

Table 11: Overall performance and clean-route characteristics. (a) Mean (sample sd) across all boards, including failures, after averaging PPO/PW-MCTS seeds within boards. (b) Each method's clean boards; only clean seeds enter its board means. Counts indicate at least one clean outcome. Δ_{ref} is mean method minus original-board value on the same boards (WL in mm). Reference uses all 50 original boards. The differing cohorts in (b) describe routing characteristics, not method ranks.

(a) All-board performance

Subset	Method	CP	Rout.	FP (Φ)
All	PCBWorld (PPO)*	0.847 (0.324)	0.937 (0.184)	−5.263 (13.794)
	PW-MCTS (4)	0.893 (0.281)	0.952 (0.155)	−4.042 (11.438)
	PW-MCTS (8)	0.893 (0.289)	0.951 (0.168)	−4.333 (12.312)
	PW-MCTS (16)	0.880 (0.292)	0.950 (0.169)	−4.376 (12.322)
	PW-MCTS (32)	0.867 (0.301)	0.949 (0.163)	−4.597 (12.006)
	PCBWorld (GPT-6-astra-medium)*	0.580 (0.499)	0.830 (0.278)	−12.602 (20.354)
	Freerouting	0.920 (0.274)	0.995 (0.023)	−0.331 (3.390)
	Reference			−0.109 (1.446)
Easy	PCBWorld (PPO)*	1.000 (0.000)	1.000 (0.000)	0.520 (0.952)
	PW-MCTS (4)	1.000 (0.000)	1.000 (0.000)	0.521 (0.941)
	PW-MCTS (8)	1.000 (0.000)	1.000 (0.000)	0.500 (0.955)
	PW-MCTS (16)	1.000 (0.000)	1.000 (0.000)	0.483 (0.942)
	PW-MCTS (32)	1.000 (0.000)	1.000 (0.000)	0.419 (0.968)
	PCBWorld (GPT-6-astra-medium)*	1.000 (0.000)	1.000 (0.000)	0.783 (0.959)
	Freerouting	1.000 (0.000)	1.000 (0.000)	1.030 (0.722)
	Reference			0.804 (0.786)
Medium	PCBWorld (PPO)*	1.000 (0.000)	1.000 (0.000)	−0.243 (0.928)
	PW-MCTS (4)	1.000 (0.000)	1.000 (0.000)	−0.227 (0.859)
	PW-MCTS (8)	1.000 (0.000)	1.000 (0.000)	−0.296 (0.859)
	PW-MCTS (16)	1.000 (0.000)	1.000 (0.000)	−0.292 (0.858)
	PW-MCTS (32)	1.000 (0.000)	1.000 (0.000)	−0.359 (0.877)
	PCBWorld (GPT-6-astra-medium)*	0.588 (0.507)	0.829 (0.294)	−11.791 (20.597)
	Freerouting	0.882 (0.332)	0.996 (0.013)	−0.220 (2.012)
	Reference			0.098 (0.679)
Hard	PCBWorld (PPO)*	0.521 (0.421)	0.802 (0.286)	−16.742 (20.315)
	PW-MCTS (4)	0.667 (0.422)	0.850 (0.250)	−12.944 (17.351)
	PW-MCTS (8)	0.667 (0.439)	0.847 (0.274)	−13.758 (18.808)
	PW-MCTS (16)	0.625 (0.419)	0.844 (0.275)	−13.879 (18.769)
	PW-MCTS (32)	0.583 (0.413)	0.839 (0.261)	−14.430 (17.806)
	PCBWorld (GPT-6-astra-medium)*	0.125 (0.342)	0.650 (0.306)	−27.686 (21.292)
	Freerouting	0.875 (0.342)	0.990 (0.038)	−1.895 (5.287)
	Reference			−1.299 (1.783)

(b) Clean-route characteristics

Method	Clean boards	WL (mm)	ΔWL$_{\text{ref}}$	Via	ΔVia$_{\text{ref}}$
PCBWorld (PPO)*	45	702.0	+133.4	3.737	−1.241
PW-MCTS (4)	46	681.6	+121.4	3.957	−1.087
PW-MCTS (8)	46	687.1	+127.0	4.507	−0.536
PW-MCTS (16)	46	687.0	+126.8	4.283	−0.761
PW-MCTS (32)	46	698.7	+138.5	4.156	−0.888
PCBWorld (GPT-6-astra-medium)*	29	486.4	−22.5	4.414	+0.897
Freerouting	46	521.6	−45.9	2.783	−2.326
Reference	50	554.7	0	5.540	0

Routed layouts. Figure 11 shows selected BoN outcomes on four boards. PPO and PW-MCTS share a training seed and the common time limit of Appendix H within each row. Freerouting uses untimed BoN-16; Reference is the original routed board. Below each layout, N is the candidate count, "Seed" identifies the rollout seed, and r is PPO's within-run rollout index. The cases illustrate completion gains (0122), Freerouting's incomplete route (0134), via–length tradeoffs (0158), and remaining detours (0344). The shorter length of an incomplete route does not indicate better route quality.



Figure 11: Selected BoN routing layouts. Each row uses the same scale and framing across methods. Annotations give routability, wirelength in millimeters, and via count. Red marks routability below 1; bold marks the fewest vias on 0158 and 0344.

K ABLATIONS

Simulation budget. Table 12 uses 16 rollout seeds on all 50 boards and three training seeds at iteration 400, sharing calibrators within a training seed. The root candidate count is $m = \min(16, n_{\text{sim}}/2)$, so the sweep changes both simulations and candidates. The first 16 trajectories of every search setting also appear in Table 1; the extended pools enter the time-budget comparison.

Table 12: Simulation-budget sweep at iteration 400, mean (sample sd) over three training seeds, with 800 instances per setting and seed. `CP`, `Rout.`, and `FP` average all trajectories. Search time per decision is total search time divided by the number of searched decisions. Cost is the median paired episode-time ratio to PPO within each seed, then summarized across seeds.

$n_{\text{sim}} : m$	Visits/ cand.	`CP` (%)	`Rout.`	`FP` (Φ)	Search time/ decision (s)	Cost ($\times$PPO*)
0 (PPO*)	—	42.2 (2.3)	0.742 (0.015)	−19.31 (1.04)	—	1.00
4 : 2	2	55.8 (5.1)	0.849 (0.013)	−12.65 (0.87)	0.402 (0.043)	2.61 (0.14)
8 : 4	2	56.8 (3.6)	0.859 (0.004)	−11.98 (0.06)	0.619 (0.014)	4.51 (0.26)
16 : 8	2	60.9 (4.4)	0.875 (0.012)	−10.58 (0.52)	1.123 (0.043)	8.44 (0.48)
32 : 16	2	63.2 (4.1)	0.885 (0.011)	−9.86 (0.69)	2.201 (0.143)	16.65 (0.96)

All search settings exceed PPO in per-draw CP, routability, and FP in every training seed. Increasing simulations from 4 to 32 raises mean CP from 55.8% to 63.2%, while the median paired episode-cost ratio rises from 2.61 to 16.65. Gains need not be monotone within a seed: seed 42 drops from 61.4% to 60.4% CP between 4 and 8 simulations. These per-draw results do not determine the best allocation of a time budget; Figure 3 addresses that tradeoff.

Calibration and anchoring. Table 13 replaces the gated, calibrated and anchored leaf estimate in Eq. (4) with $\sigma_{\text{ckpt}} V_\theta(s_d)$:

$$G_{\text{sim}}^{\text{scale}} = R_{\text{path}} + \gamma^d (1 - \delta_d)\, \sigma_{\text{ckpt}} V_\theta(s_d). \tag{19}$$

The checkpoint-wide reward-normalization scales are 14.201, 14.473, and 14.469 for training seeds 42, 43, and 44, with no board-specific map or terminal offset. Checkpoints, boards, rollout seeds, simulation counts, and evaluation scores match. This compares the complete value-calibration procedure with scaling alone; it does not isolate the map, gate, or anchor.

Table 13: Calibration ablation at 32 simulations per decision. Mean (sample sd) over 2,400 paired trajectories; WL and Via use the 1,195 pairs where both pass clean. The final row averages BoN-16 clean-board counts over three training seeds. Δ is calibrated minus scale-only. These are equal-draw comparisons; episode times exclude calibration collection and fitting.

Metric	Scale only	Calibrated	Δ
CP	0.581 (0.493)	0.632 (0.482)	+0.051
Rout.	0.874 (0.243)	0.885 (0.235)	+0.011
FP (Φ), all outcomes	−10.848 (17.534)	−9.860 (17.154)	+0.988
Episode time (s)	326.0 (465.9)	266.8 (315.4)	−59.2
WL (mm), both clean	673.5 (435.1)	669.3 (426.4)	−4.2
Via, both clean	3.840 (5.192)	3.644 (5.137)	−0.196
BoN-16 clean boards (of 50)	41.3 (1.2)	43.3 (1.2)	+2.0

Calibration raises CP by 5.1 percentage points and reduces mean episode time by 59.2 seconds. The per-draw CP gains are 1.1, 10.3, and 3.8 points on easy, medium, and hard boards. BoN-16 gains two clean boards in each training seed. The time reduction concerns search execution, not the end-to-end cost of fitting the calibrator.

Invalid-action handling. Table 3 compares 800 paired instances for training seed 42. The `pop`, `keep`, and `penalize` settings reach 67.2%, 51.9%, and 64.5% CP at 1.987, 1.135, and 1.013 seconds per searched decision, respectively. Relative to `pop`, aligned FP is lower by 3.43 for `keep`

and 0.29 for `penalize`. Retrying rejected actions changes both work and outcomes; profiling is enabled in all three runs, so these timings are comparable within this ablation.

Implementation optimizations. On 50 boards with training seed 42 and rollout seed 0, cumulative optimizations reduce total search time by 7.95×. Search time per searched decision falls from 24.936 to 3.140 seconds (7.94×), and mean per-board peak RSS falls from 9,899 to 671 MB. Packing accounts for a 6.78× reduction from the preceding memoization configuration; these measurements do not isolate each technique's standalone speedup. The reported speedups compare cumulative configurations on the same board set.

L CANDIDATE DISTRIBUTIONS

Figure 12 shows the original PW-MCTS (32) and PPO candidate pools before BoN selection. Search shifts routability and FP toward higher scores, particularly on hard boards. PPO includes its full collected pool, so these distributions provide per-candidate context rather than the time-limited BoN outcomes in Table 2.

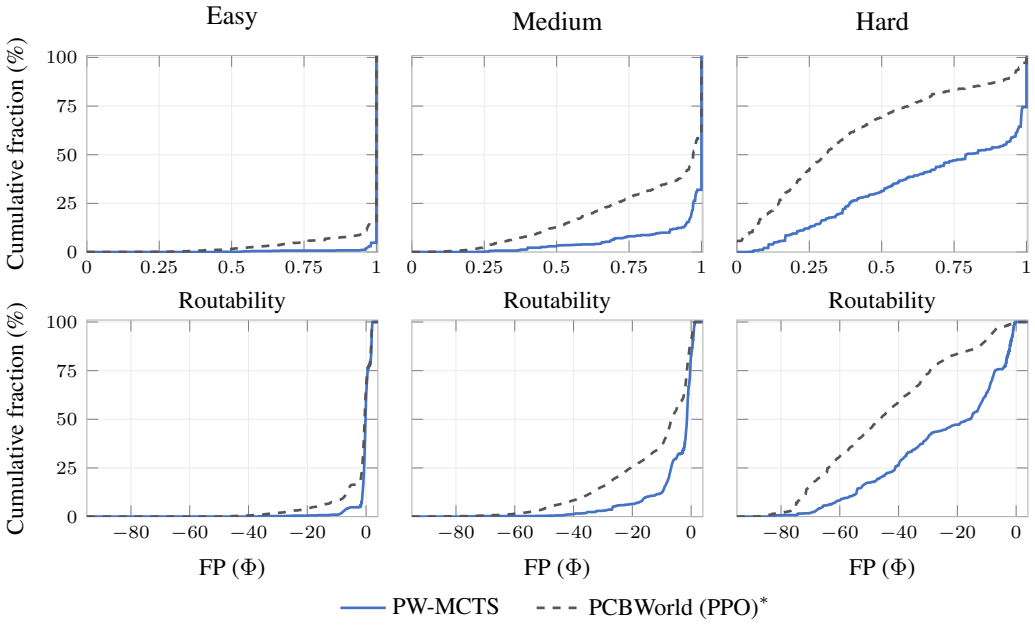


Figure 12: Candidate distributions by difficulty, before BoN selection. PW-MCTS (32) uses 16 trajectories per board and training seed; PPO uses the full collected pool. Empirical cumulative distributions give equal weight to each board and training seed, including incomplete and non-clean outcomes. Rightward curves indicate higher scores; FP reflects both completion and route quality.

M EXTENDED DISCUSSION

Table 14 supplements Table 2 with all metrics under the same time limits and all-method-clean quality cohorts. Additional promoted violations, $\max(0, \mathrm{DRV}_{\mathrm{prom}} - \mathrm{DRV}_{\mathrm{err}})$, are diagnostic counts excluded from FP and clean pass.

Table 14: Complete matched-time BoN results for PPO and PW-MCTS (4), (8), (16), and (32). Entries are mean (sample sd) across equally weighted board means; quality uses only the jointly clean seeds of Table 2. N is the mean completed candidate count; Pass counts boards with at least one such seed. DRV (prom.) counts additional promoted violations, excluded from evaluation FP and clean pass. Bold compares the five PPO/PW-MCTS settings.

Subset	Metric	PCBWorld (PPO)*	PW-MCTS (4)	PW-MCTS (8)	PW-MCTS (16)	PW-MCTS (32)
All	N	246.1	97.1	56.3	29.7	16.0
	CP	0.847 (0.324)	**0.893** (0.281)	**0.893** (0.289)	0.880 (0.292)	0.867 (0.301)
	Rout.	0.937 (0.184)	**0.952** (0.155)	0.951 (0.168)	0.950 (0.169)	0.949 (0.163)
Pass (44)	FP (Φ)	−0.548 (1.779)	**−0.468** (1.691)	−0.543 (1.751)	−0.511 (1.718)	−0.572 (1.705)
	WL (mm)	690.2 (428.8)	**676.9** (412.9)	681.9 (416.7)	687.5 (430.5)	697.1 (430.3)
	Via	3.595 (5.057)	**3.564** (4.562)	4.045 (5.538)	3.633 (4.675)	3.712 (4.819)
	DRV (prom.)	0.735 (1.301)	1.511 (2.371)	1.814 (2.901)	**0.693** (1.019)	1.076 (2.376)
Easy	N	287.2	110.5	59.7	30.4	16.0
	CP	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)
	Rout.	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)
Pass (17)	FP (Φ)	0.520 (0.952)	0.521 (0.941)	0.500 (0.955)	0.483 (0.942)	0.419 (0.968)
	WL (mm)	**545.8** (303.1)	548.6 (307.1)	551.3 (309.4)	555.3 (305.2)	567.0 (314.5)
	Via	1.451 (2.017)	**1.373** (1.683)	1.412 (1.801)	**1.373** (1.863)	1.471 (1.990)
	DRV (prom.)	0.333 (0.565)	0.392 (0.700)	0.431 (0.840)	0.294 (0.470)	**0.275** (0.429)
Medium	N	211.9	87.2	54.5	29.2	16.0
	CP	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)
	Rout.	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)
Pass (17)	FP (Φ)	−0.243 (0.928)	**−0.227** (0.859)	−0.296 (0.859)	−0.292 (0.858)	−0.359 (0.877)
	WL (mm)	752.3 (492.6)	**741.2** (479.7)	746.5 (481.3)	759.5 (503.7)	763.0 (499.0)
	Via	**2.804** (4.165)	3.098 (4.395)	3.451 (4.587)	3.235 (4.690)	3.510 (4.898)
	DRV (prom.)	**0.549** (0.645)	1.431 (1.747)	1.333 (1.155)	0.961 (1.013)	0.882 (1.369)
Hard	N	238.7	93.5	54.4	29.5	16.0
	CP	0.521 (0.421)	**0.667** (0.422)	**0.667** (0.439)	0.625 (0.419)	0.583 (0.413)
	Rout.	0.802 (0.286)	**0.850** (0.250)	0.847 (0.274)	0.844 (0.275)	0.839 (0.261)
Pass (10)	FP (Φ)	−2.881 (1.896)	**−2.560** (1.990)	−2.737 (2.048)	−2.573 (2.131)	−2.618 (2.065)
	WL (mm)	830.1 (464.0)	**785.9** (431.1)	794.2 (440.0)	789.7 (462.8)	806.5 (461.1)
	Via	8.583 (6.827)	8.083 (5.279)	9.533 (7.493)	8.150 (5.152)	**7.867** (5.705)
	DRV (prom.)	1.733 (2.303)	3.550 (3.684)	4.983 (4.623)	**0.917** (1.502)	2.767 (4.343)

Variation across training seeds. Under the common time limit, the clean-board differences from PPO for training seeds 42/43/44 are $0, +5, +2$ for PW-MCTS (4), $+1, +4, +2$ for PW-MCTS (8), $+1, +4, 0$ for PW-MCTS (16), and $-1, +4, 0$ for PW-MCTS (32). After averaging seeds within boards, PW-MCTS (4) improves CP on five boards and ties on 45; PW-MCTS (8) improves on six but worsens on two. Their equal aggregate CP therefore reflects different board-level outcomes. Together with Appendix I, this variation connects prior dependence with the allocation of test-time computation.

N BOARD-WISE SEARCH–SAMPLING TRADEOFFS

The board-wise curves use the matched-time replay in Appendix H. For board b and training seed s, define

$$T^{32}_{b,s} = \sum_{i=1}^{16} \tau^{32}_{b,s,i}, \qquad \overline{T}^{32}_{b} = \frac{1}{3} \sum_{s} T^{32}_{b,s}. \tag{20}$$

At fraction f, seed s contributes its selected outcome at $fT^{32}_{b,s}$; the minute axis shows $f\,\overline{T}^{32}_{b}$, then outcomes are averaged across seeds. This aligns the endpoints without treating the longest seed duration as a shared wall-clock time. Exhausted recordings retain their observed incumbents. PW-MCTS (4), (8), (16), and (32) use pools of 128/64/32/16 trajectories per board and seed, as in Figure 3. FP starts once every seed has a completed candidate; routability is zero for a seed with no candidate. Vertical dotted lines mark the earliest clean pass across seeds in budget fraction. End labels follow final values, breaking ties by the earliest routability-1 event. Horizontal references are untimed Freerouting BoN-16 and original-board (Reference) outcomes; Reference FP is not an upper bound.

Cases exceeding reference outcomes. Figure 13 selects two kinds of favorable cases. All four PW-MCTS settings produce clean winners in all three seeds on 0134 and 0279, where Freerouting remains incomplete. On 0158 and 0218, they exceed the original board's FP with clean winners; PW-MCTS (16) gains 1.47 and 0.20 in three-seed mean FP, respectively. PPO also exceeds the corresponding references, so these cases complement the aggregate router comparison in Appendix J.

Board-wise cases. Figure 14 contrasts completion and partial progress. PW-MCTS (32) completes 0122 and 0203 in all three seeds, versus zero and one for PPO. Neither completes 0325 or 0249: PW-MCTS (32) has higher routability and FP on 0325, whereas PPO has higher values on 0249. These selected cases show both directions of the board-level variation.

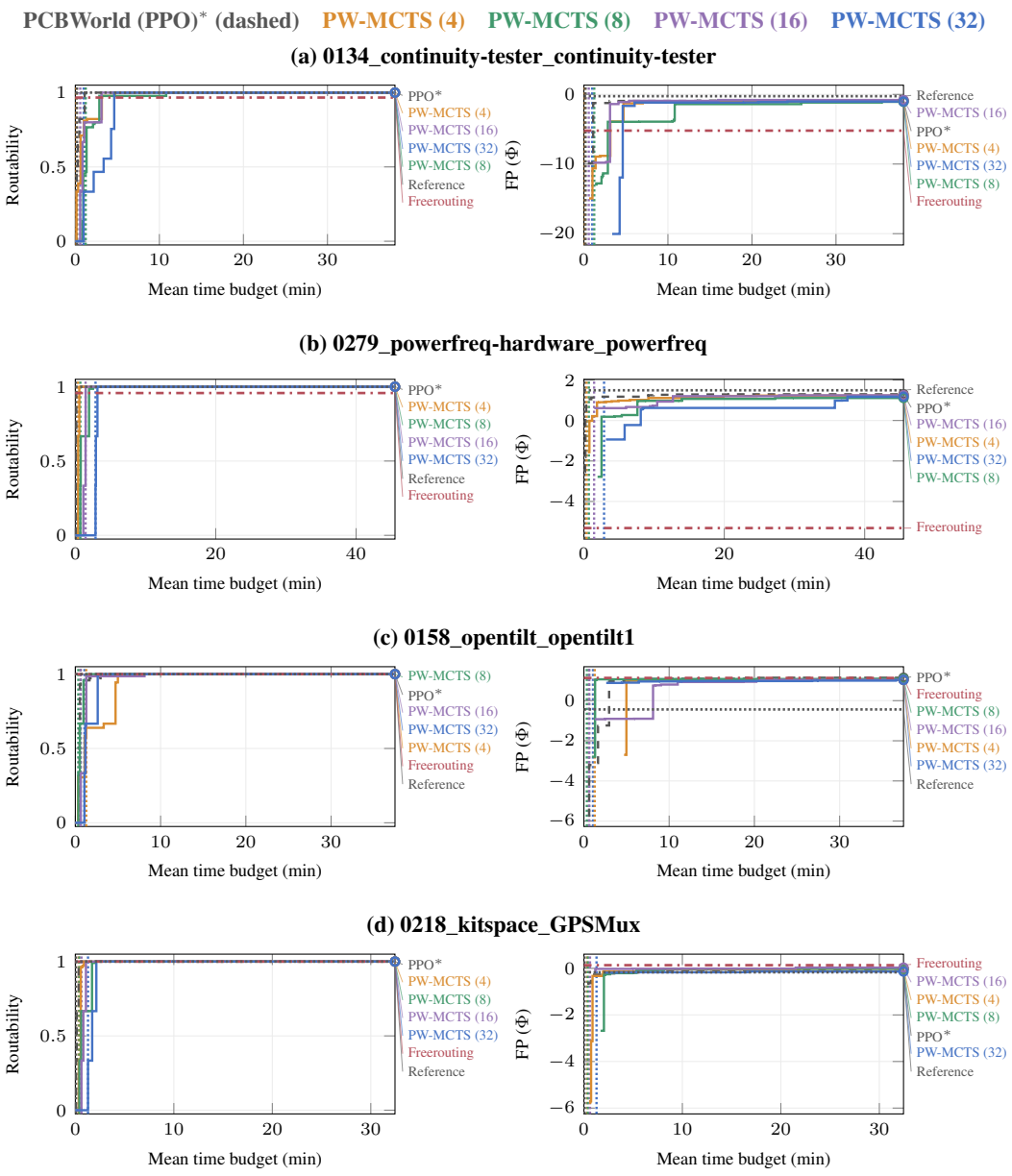


Figure 13: Selected favorable cases: completion beyond Freerouting on 0134/0279 and FP beyond the original board on 0158/0218. Curves average three training seeds at matched budget fractions, displayed as $f\,\overline{T}_b^{32}$ minutes. Vertical lines mark earliest clean pass; horizontal references are untimed.

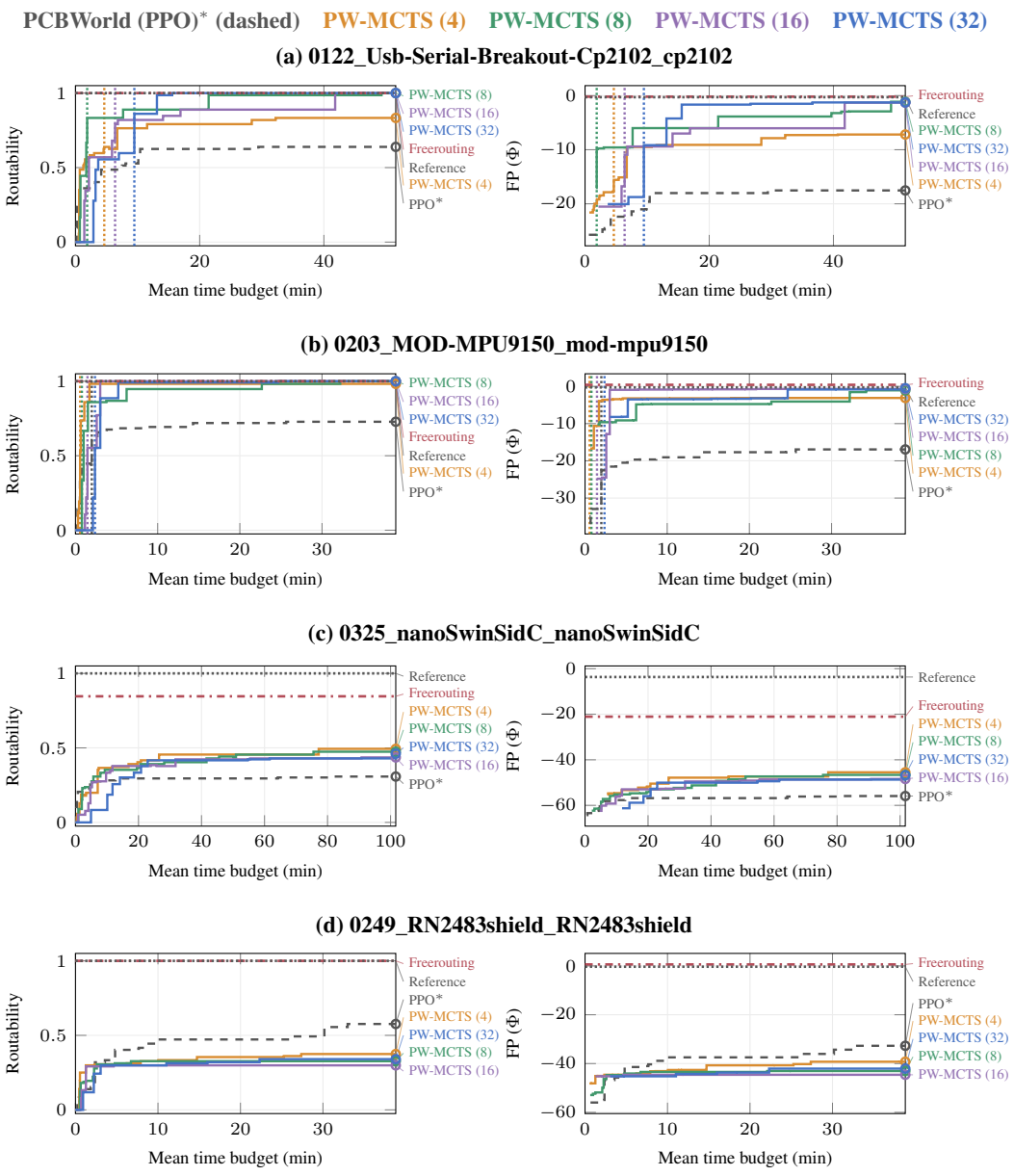


Figure 14: Board-wise BoN showing both directions of the search–sampling tradeoff. Curves are three-seed means at matched budget fractions, displayed as $f\,\overline{T}_b^{32}$ minutes. Observed incumbents are retained when recordings end early. Vertical lines mark earliest clean pass; horizontal references are untimed.