Coupling-Aware Planner Exploration for Shared-Workspace Multi-Manipulator Motion Planning

Rahul Vimalkanth
Indian Institute of Technology Madras
ee23b061@smail.iitm.ac.in

***Abstract*—Shared-workspace multi-manipulator planning repeatedly faces a representation decision: exploit efficient decoupled/hybrid planning or escalate to coupled composite-space search. We cast this as coupling-aware exploration over planning representations: the system probes interaction structure before deciding whether to exploit cheap decoupled/hybrid planning or explore a more expensive coupled composite-space representation. Our framework logs interpretable interaction descriptors, routes tasks among decoupled, hybrid, and joint-space modes, and evaluates these policies on 4,000 PyBullet tasks and 32,000 planner runs. Coupling-aware routing improves paired solve outcomes over random and pure joint-space routing, while matching the best fixed-mode solve rate in the selection ablation. The main contribution is a regime-level account of escalation under interaction uncertainty: coupling descriptors indicate when hybrid exploitation is sufficient, when deeper coupled search is warranted, and where selector overhead remains.**

***Index Terms*—multi-manipulator planning, planner selection, motion planning, shared-workspace robotics, adaptive planning, exploration**

I. Introduction

Shared-workspace robot cells rarely fail because one arm cannot move. They fail because several individually plausible motions compete for the same volume, at the same time, under a finite planning budget. A pair of collision-free single-arm paths may still collide when executed synchronously; a conservative joint-space planner may avoid that interaction but spend its budget in a much larger search space. This tension is central to practical multi-manipulator planning.

Most systems resolve the tension with fixed engineering choices. Decoupled and prioritized planners are fast when interaction is weak, but brittle when timing matters. Fully coupled planners represent robot–robot constraints directly, but their cost grows with the composite configuration space. Hybrid stacks sit between these extremes and are often effective in practice, yet they leave open a policy question: when should the system escalate its representation of coupling?

We study that question as coupling-aware planner exploration. The selector computes inexpensive interaction probes and routes each task among decoupled, hybrid, and joint-space modes. The goal is to make escalation measurable: identify regimes where routing choices fail and quantify robustness–runtime tradeoffs.

This work was accepted for poster presentation at Xplore @ ICRA 2026, the ICRA 2026 Workshop on Cross-Disciplinary Aspects of Exploration in Robotics, Reinforcement Learning and Search.

Figure 1 summarizes the central empirical observation: the benchmark organizes into planning regimes in coupling-feature space rather than around a single globally dominant routing mode. We therefore frame the result as an empirical study of escalation policy rather than a leaderboard.

Here, *planner exploration* means choosing how deeply to search over planning representations under limited budget: cheap decoupled/hybrid structure versus more expressive but expensive coupled search.

The contributions are:

- A coupling-aware formulation of shared-workspace planning as representation exploration over decoupled, hybrid-stack, and joint-space routing.
- A coupling-feature telemetry framework that logs synchronized overlap, crossing stress, proximity, congestion, and conflict density.
- A benchmark-scale empirical comparison with fixed seeds and paired tasks (4,000 tasks, 32,000 benchmark runs, 24,000 selection-ablation runs).
- A telemetry-based regime and failure analysis that explains escalation behavior beyond leaderboard ranking.

II. Related Work

Recent work has renewed interest in multi-robot and multi-manipulator planning under shared-workspace constraints. Online coordination for high-DOF manipulators emphasizes scalable trajectory adjustment and replanning [18], [25], while scheduling-centric hybrid coordination exploits individual plans before escalating to stronger coupled reasoning [19], [22], [23]. Hierarchical optimization approaches for confined multi-robot manipulation [24] and industrial multi-robot TAMP pipelines [26] further support this systems perspective.

Learning-based multi-robot planning has also advanced quickly. Diffusion-based methods generate diverse multi-robot trajectories and combine learned priors with feasibility handling [20], [21]. Our work is complementary: instead of replacing classical planners with a learned generator, we study coupling-aware routing over decoupled, hybrid, and joint-space modes.

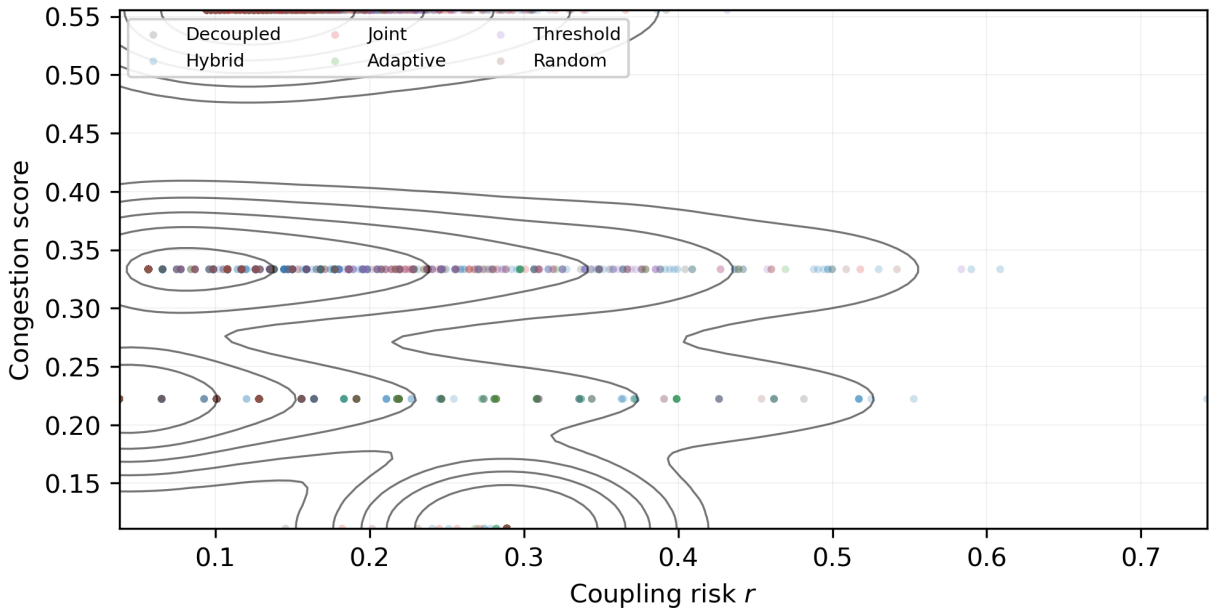

Fig. 1. Anchor regime map from the selection ablation. Axes show coupling risk and congestion score; point color indicates the fastest successful policy per task; contour lines indicate task density. The figure shows regime-dependent behavior rather than a single globally dominant routing mode.

A. Sampling-Based Manipulation Planning

Sampling-based planning remains central in high-dimensional manipulation: RRT/RRT-Connect/PRM [1], [2], [3], optimal variants such as RRT* [4], and informed/batch/tree refinements [5], [6], [7]. Our work does not propose a new primitive; it studies when to explore different planning representations for shared-workspace multi-arm tasks.

B. Multi-Robot and Multi-Manipulator Coordination

Multi-robot planning balances decoupled coordination against coupled search. Prioritized methods [8], [9] are efficient but order-sensitive, while coupled formulations scale poorly; MAPF/CBS formalize this tradeoff in discrete settings [10]. Hybrid systems exploit cheap representations first and escalate when needed. We make this escalation explicit and evaluate it as a routing policy.

C. Planner Portfolios and Algorithm Selection

Planner portfolios choose algorithms from problem features [11], [12]. In robotics this appears as planner switching and budgeted routing; the bottleneck is informative descriptors for interaction-heavy tasks. We therefore use compact coupling descriptors and evaluate policy behavior directly. Related learning-guided and optimization-based planning lines [13], [14], [15], [16], [17] provide context for this routing perspective.

D. Empirical Robotics Evaluation

Empirical planning claims depend on paired comparisons, held-out splits, confidence intervals, and failure analysis. We report Wilson intervals, paired McNemar tests, and bootstrap runtime intervals on both-solved tasks.

III. Problem Formulation

Consider N manipulators operating in a shared workspace. Robot i has configuration $q_i \in \mathcal{C}_i$, start configuration s_i, and goal configuration g_i. The composite configuration is

$$Q = (q_1, \dots, q_N) \in \mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_N. \tag{1}$$

A synchronized multi-robot path is a set of discrete trajectories

$$\Pi = \{\pi_i(0 : T_i)\}_{i=1}^{N}, \tag{2}$$

with execution evaluated at synchronized index t by holding shorter paths at their final waypoint:

$$Q(t) = \big(\pi_1(\min(t, T_1)), \dots, \pi_N(\min(t, T_N))\big). \tag{3}$$

The path is valid if each robot begins at its start, reaches its goal, respects single-robot collision constraints, and satisfies

$$\text{collide}\big(Q(t)\big) = 0 \quad \forall t, \tag{4}$$

including interpolated edge checks between waypoints.

We study planner selection rather than only path generation. Let $\mathcal{A} = \{a_D, a_H, a_J\}$ denote decoupled, hybrid, and joint-space modes. Given task $x = (E, S, G, N)$, a selector chooses

$$a = \pi_\theta(\phi(x)), \tag{5}$$

where $\phi(x)$ is a vector of coupling features. We evaluate solve rate, runtime on solved tasks, path length, collision checks, and failure type rather than optimizing a single scalar. This reflects the systems tradeoff: small speed gains can be unacceptable if they introduce unsolved tasks, while predictable overhead may be acceptable when it improves robustness.

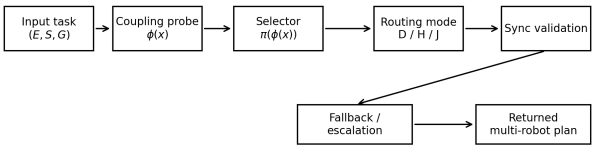

Fig. 2. HyCoS routing pipeline used in this paper: task input, coupling probe, selector routing (decoupled / hybrid / joint), synchronized validation, and fallback escalation when validation fails.

IV. Coupling-Aware Planning Framework

A. Planning Modes

We refer to the hybrid coordination stack as HyCoS, short for *Hybrid Coordination Search*. HyCoS is not a new primitive sampling planner; it is a coordination stack that combines direct single-robot attempts, per-robot sampling-based planning, synchronized validation, scheduling, and fallback escalation. In the experiments, HyCoS denotes the fixed always-hybrid policy. AdaptiveHyCoS denotes an earlier adaptive baseline included for comparison. CA-HyCoS, short for *Coupling-Aware Hybrid Coordination Search*, denotes the final coupling-aware selector studied in this paper.

The framework routes each task among three planning modes. The decoupled mode plans individual robot motions and relies on the HyCoS scheduling component to validate synchronized execution. It is appropriate when independent motions do not create shared occupancy conflicts.

The hybrid mode uses the full HyCoS stack: direct attempts, local sampling-based planning, scheduling, and fallback. It is fast because many tasks are solved before composite search is necessary, yet it retains escape routes when simple decoupling fails.

The joint mode invokes a composite-space RRT-style planner. It represents inter-robot collision constraints directly, but is most exposed to dimensionality and sampling burden. In our implementation, joint mode may fall back to hybrid planning after a small joint budget is exhausted.

B. Coupling Features

The selector uses features computed before planning from sampled straight-line probes between starts and goals. These features are not physical probabilities; they are normalized stress indicators designed to expose likely interaction structure.

For a pair of robots (i, j), let

$$q_i(\alpha) = (1-\alpha)s_i + \alpha g_i, \tag{6}$$

and similarly for $q_j(\alpha)$. The synchronous path-overlap feature estimates the fraction of paired samples for which both robots moving at the same interpolation parameter produce a composite collision:

$$f_{\text{sync}} = \frac{1}{|\mathcal{P}|K} \sum_{(i,j)\in\mathcal{P}} \sum_{k=1}^{K} \mathbf{1}\{\text{collide}(Q_{ij}(\alpha_k, \alpha_k))\}. \tag{7}$$

The asynchronous crossing feature samples independent interpolation parameters:

$$f_{\text{cross}} = \mathbb{E}_{\alpha,\beta}\left[\mathbf{1}\{\text{collide}(Q_{ij}(\alpha,\beta))\}\right], \tag{8}$$

which is meant to detect ordering-sensitive conflicts that may not occur under perfectly synchronized motion.

We also compute a midpoint proximity feature f_{mid}, a congestion proxy f_{cong} based on robot count and obstacle count, and a joint conflict density term f_{dens} that scales path overlap by normalized motion magnitude. The aggregate risk score used for threshold routing is

$$\begin{aligned} r = {} & 0.28 f_{\text{sync}} + 0.22 f_{\text{cross}} + 0.18 f_{\text{mid}} \\ & + 0.17 f_{\text{cong}} + 0.15 f_{\text{dens}}. \end{aligned} \tag{9}$$

The weights in (9) are heuristic calibration constants, selected during pilot runs to keep each normalized feature on a comparable influence scale while preserving monotonic risk ordering. We do not claim these coefficients are universal or physically derived; they are benchmark-specific operating values used for transparent thresholding and visualization. The learned CA-HyCoS selector uses seven inputs: the five coupling terms in Eq. (9), plus normalized robot count and normalized motion magnitude. The scalar risk r is used only for visualization, threshold routing, and regime analysis.

C. Telemetry

Every run logs selected mode, fallback usage, collision checks, coupling features, solve status, runtime, and failure reason. These logs separate first-choice success from escalation recovery and drive all figures/tables.

V. Planner Selection Policy

A. Learned and Threshold Policies

CA-HyCoS is the learned coupling-aware selector used as the final adaptive policy. We also evaluate a threshold-only policy over the scalar risk score r. The learned policy is a random forest trained on empirical oracle labels: the fastest successful logged mode mapped to decoupled, hybrid, or joint classes.

The second policy is a two-threshold rule over r:

$$\pi(r) = \begin{cases} a_D, & r < \tau_{\text{low}}, \\ a_H, & \tau_{\text{low}} \le r < \tau_{\text{high}}, \\ a_J, & r \ge \tau_{\text{high}}. \end{cases} \tag{10}$$

The calibrated thresholds are $\tau_{\text{low}} = 0.05$ and $\tau_{\text{high}} = 0.75$, with held-out threshold accuracy 0.6495 on the mode-label objective. A conservative default rule is used when no learned model or calibrated thresholds are available. Algorithm 1 summarizes the online routing and fallback logic used in our implementation.

B. Fallback Logic

Routing is not a hard commitment. If joint mode fails within budget, the planner falls back to the hybrid stack; if decoupled mode fails, it escalates to hybrid planning. This keeps early routing reversible at the cost of extra overhead.

Algorithm 1 CA-HyCoS adaptive selector (runtime policy)

Require: task (E, S, G), feature probe $\phi(\cdot)$, selector π, modes $\{a_D, a_H, a_J\}$

compute coupling features $z \leftarrow \phi(E, S, G)$ and risk summary r
choose routing mode $a \leftarrow \pi(z)$ ▷ learned or threshold policy
run planner in mode a and obtain candidate plan Π
if Π is missing or invalid under synchronized validation **then**
    escalate/fallback (e.g., $a_D \rightarrow a_H$ or $a_J \rightarrow a_H$), recompute Π
end if
return Π (or failure)

TABLE I
ENVIRONMENT DETAILS FOR THE 4,000-TASK BENCHMARK.

Env.	Robots	Layout	Coupling character	Tasks
1	1	Separated workspace	No inter-robot coupling	1000
2	2	Partial overlap	Moderate coupling	1000
3	3	Central interaction region	Medium-high coupling	1000
4	5	Dense shared workspace	High coupling	1000

TABLE II
SELECTION ABLATION OVER 4,000 PAIRED TASKS. SOLVE INTERVALS ARE WILSON 95% INTERVALS. MEDIAN RUNTIME IS COMPUTED ON SOLVED TASKS.

Policy	Solve rate	Med. rt (s)
CA-HyCoS	99.95 [99.82, 99.99]	0.0374
Decoupled	99.88 [99.71, 99.95]	0.0518
Hybrid	99.95 [99.82, 99.99]	0.0124
Joint	97.72 [97.22, 98.14]	0.0374
Threshold	99.95 [99.82, 99.99]	0.0356
Random	99.03 [98.67, 99.29]	0.0323

TABLE III
FULL BENCHMARK SUMMARY OVER 32,000 RUNS. HYCOS AND CA-HYCOS MATCH THE HIGHEST SOLVE RATE; HYCOS HAS THE LOWEST MEDIAN RUNTIME ON THIS SUITE.

Planner	Solve (%)	Med. rt (s)
BasicRRT	71.35	0.0099
RRTConnect	74.75	0.0097
Prioritized	96.93	0.0312
Sequential	94.85	0.0389
JointSpaceRRT	97.63	0.0155
HyCoS	99.95	0.0057
AdaptiveHyCoS	97.80	0.0185
CA-HyCoS	99.95	0.0194

C. Why the Selector Is Not an Oracle

We define the oracle mode label as the mode class of the fastest successful planner on that task among the logged candidate planners. The learned selector's oracle-match accuracy is modest because fastest-planner labels are noisy and often separated by small runtime differences. In the held-out environment split, oracle match is 31.7% with 100.0% solve rate and median runtime gap 0.0087 s. Its value is robust escalation behavior, not exact oracle imitation.

VI. EXPERIMENTAL SETUP

A. Benchmark

Experiments use PyBullet shared-workspace scenes with UR5-like manipulators. The full benchmark contains 4,000 tasks from four environments, 100 seeds, and 10 start–goal pairs per seed. Each task is evaluated with eight planners, producing 32,000 rows. The selection ablation evaluates six routing policies on the same tasks, producing 24,000 rows.

The full benchmark planners are BasicRRT, RRTConnect, Prioritized, Sequential, JointSpaceRRT, HyCoS, AdaptiveHyCoS, and CA-HyCoS. The selection-ablation policies are Adaptive, Decoupled, Hybrid, Joint, Threshold, and Random under identical task seeds and validation.

B. Metrics

We report solve rate with Wilson confidence intervals, median runtime on solved tasks, path length, collision-check counts, fallback frequency, and failure type. For paired solve-rate comparisons, we use McNemar tests between CA-HyCoS and each selection baseline. For runtime, we report bootstrap confidence intervals on paired differences over tasks solved by both planners. This both-solved runtime statistic is useful but incomplete: it excludes tasks on which one planner fails, so it must be interpreted alongside solve rate.

VII. RESULTS

A. Finding 1: Coupling-Aware Routing Avoids Bad Extremes

The first result is that coupling-aware routing reduces the most damaging planner choices. In Table II, CA-HyCoS solves 99.95% of tasks, matching AlwaysHybrid and ThresholdOnly within the reported precision. Its clearest paired gains are against AlwaysJoint and RandomSelection: 90 adaptive-only successes versus one joint-only success over 4,000 tasks ($p = 2.84 \times 10^{-20}$), and 37 adaptive-only successes versus no random-only successes ($p = 3.25 \times 10^{-9}$).

The lesson is not that joint-space planning is weak; it is that pure joint-space commitment is a poor default under a finite budget. Coupling-aware selection improves robustness by deciding when coupled search is worth invoking, rather than invoking it everywhere.

B. Finding 2: Hybrid Planning Is the Dominant Default

AlwaysHybrid matches adaptive solve rate and has lower median runtime. In the full benchmark, HyCoS reaches 99.95% solve with median solved runtime 0.0057 s versus 0.0194 s for CA-HyCoS. This reinforces the regime interpretation in Fig. 1: hybrid exploitation is often sufficient, while coupling-aware escalation helps avoid poor routing extremes.

C. Finding 3: Selection Buys Robustness, Not Free Speed

On tasks solved by both planners, CA-HyCoS is slower on average than the fixed baselines. For example, AlwaysHybrid is faster by a mean of 0.470 s on the both-solved subset, with bootstrap 95% interval $[0.398, 0.551]$ s. This runtime gap is the price of feature computation, conservative routing, and

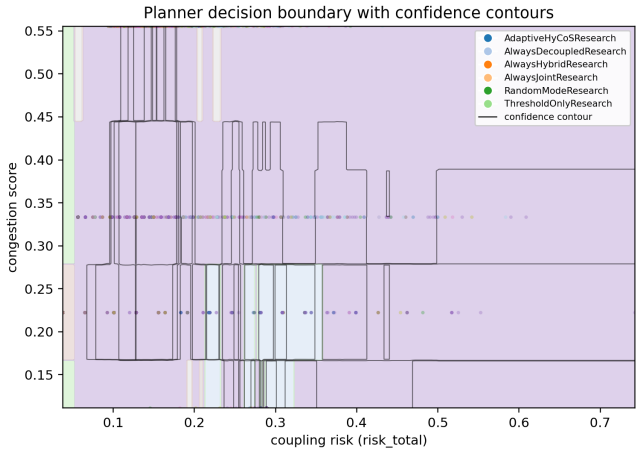

Fig. 3. Selector boundary in coupling-risk and congestion space with median-fixed remaining features. Colored regions indicate predicted planner class and black contours indicate selector confidence levels.

TABLE IV
STRESS SUBSETS FROM THE SELECTION ABLATION. VALUES ARE SOLVE RATES.

Subset	n	CA-HyCoS	Hybrid	Joint / Random
High coupling (r high-risk bin)	1196	99.92%	99.92%	97.74% / 99.00%
Large robot count (≥ 3)	2000	99.95%	99.95%	97.50% / 98.85%

occasional recovery. This mean gap is larger than the median gap because fallback-heavy cases create a long runtime tail.

Coupling-aware selection improves robustness against poor routing but is not yet speed-optimal against the strongest fixed hybrid policy.

D. Finding 4: The Selector Is Conservative Rather Than Oracle-Like

The oracle analysis reinforces the same point. AlwaysHybrid has the smallest mean runtime gap to the fastest successful planner, with an oracle-close rate of 84.9% under a 0.01 s threshold. CA-HyCoS is less often runtime-close to the oracle (48.6%), but has a near-zero solve gap.

Held-out splits show similar behavior. Training on environments 1–3 and testing on environment 4 yields a 100.0% selector solve rate and zero solve gap relative to the oracle, but only 31.7% exact oracle-planner match. The selector often chooses a reliable mode rather than the fastest logged option, which is acceptable for a robustness-oriented system but leaves room for budget-aware selection.

VIII. WHERE SELECTION MATTERS

A central question is whether explicit selection is necessary when the hybrid stack is already strong. On this benchmark, the necessity argument is not "adaptive beats hybrid everywhere." It is that adaptive routing avoids poor escalation extremes in coordination-stress subsets while preserving the same high solve regime as hybrid.

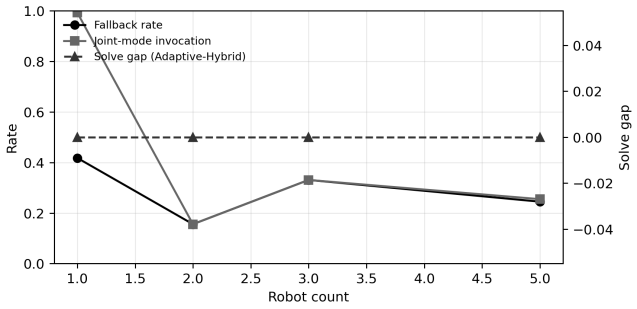

Fig. 4. Robot-count stress trends for the adaptive selector. Fallback and joint-mode invocation vary with robot count, while the adaptive selector maintains near-zero solve gap relative to hybrid.

Table IV shows the pattern directly: in high-coupling and larger-robot subsets, adaptive and hybrid preserve near-identical solve performance, while pure joint-space and random routing drop. This is where selection is useful as a systems safeguard: it avoids repeatedly selecting weak extreme modes under coordination stress.

A. Feature Sensitivity and Runtime-Cost Transparency

To test fragility, we perturb Eq. (9) weights by $\pm 20\%$ one term at a time and recompute risk ranking offline. The perturbed scores remain strongly aligned with baseline risk (Pearson correlation ≥ 0.997 across all perturbations), which indicates the regime ordering is not dominated by a single coefficient choice.

For runtime cost, telemetry shows adaptive median runtime 0.0375 s, median 276.5 collision checks, and fallback rate 28.8%. These are not component timers, but indicate overhead from feature computation, validation, and fallback.

B. Why Hybrid Dominance Does Not Invalidate the Study

Hybrid dominance supports, rather than contradicts, the paper's premise: escalation policy is central. The hybrid stack succeeds because it encodes effective escalation behavior. The adaptive selector's role is to make this decision process explicit, auditable, and analyzable by regime, instead of treating escalation as opaque implementation detail.

IX. PLANNING REGIME ANALYSIS

A. Risk Bins

We divide tasks into tertiles by aggregate coupling risk: low risk $r \leq 0.1008$, medium risk $0.1008 < r < 0.2453$, and high risk above that threshold. AlwaysHybrid is dominant in all three bins when dominance is defined as highest solve rate with median-runtime tie-break. In low-risk tasks, decoupled and hybrid policies solve all tasks, while AlwaysJoint drops to 98.7%. In high-risk tasks, adaptive and hybrid both solve 99.9%, while AlwaysJoint remains around 97.2%.

Thus, higher heuristic coupling risk does not imply "use joint-space planning everywhere." In this benchmark, high-risk tasks often benefit from trying cheap structured solutions before escalating. Coupling risk is best interpreted as a signal

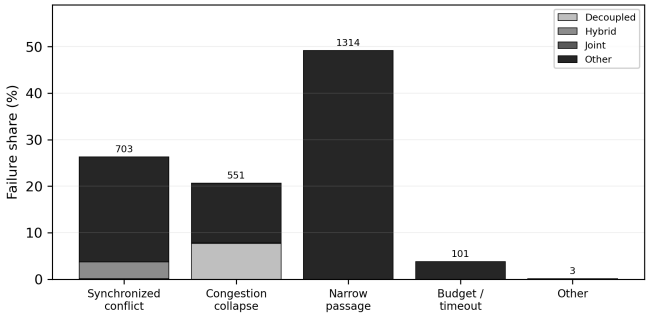

Fig. 5. Log-derived failure taxonomy from planner telemetry. Bars summarize dominant failure categories: synchronized conflict, congestion collapse, narrow passage failures, and budget-related failures.

for careful escalation, not a command to search the full composite space.

B. What Coupling Risk Captures

The coupling features are most informative when they describe how independent motions interact in time. Synchronous overlap catches collisions under matched progress. Asynchronous crossing stress catches shared workspace use at different progress values. Midpoint proximity captures arms meeting near fixtures or central obstacles, while congestion and joint conflict density provide scene context.

Feature ablations show small drops in oracle-mode prediction accuracy when removing individual features. Congestion score produces the largest drop, 0.0035, followed by normalized robot count, proximity risk, and joint conflict density. The robust finding is not that any single feature predicts the winner, but that the feature vector supports interpretable regimes and stable avoidance of poor routing.

C. Escalation Boundary

The regime boundary analysis finds a heuristic joint-versus-rest split around $r = 0.3903$, but no sharp phase transition. This is expected: continuous manipulation regimes are shaped by obstacle layout, robot count, start–goal geometry, and timing. The practical role of r is to order tasks by coupling stress, while the full feature vector and fallback logic handle ambiguity near boundaries.

X. Failure Analysis

The failure logs show four recurring patterns: synchronized occupancy conflict, congestion-driven no-path outcomes, narrow-passage interpolation failure, and scheduler failure. All four are coordination failures rather than simple single-arm planning failures.

These failures explain both sides of the escalation tradeoff. Pure decoupling can miss synchronized conflicts; pure joint-space planning can spend its budget in a high-dimensional space when a structured hybrid solution would have been found quickly.

XI. Discussion

Coupling is an operational measure of how strongly each robot's success depends on the timing and geometry of the others. The proposed features approximate this dependency before full planning: cheap and imperfect, but useful for escalation guidance.

Hybrid planning performs well because many tasks do not require composite-space search from the first sample. Direct paths, individual-arm RRT-Connect attempts, and scheduling solve many cases; fallback recovers some tasks that defeat pure decoupling. This explains why AlwaysHybrid is the strongest fixed baseline.

The adaptive selector should therefore be viewed as a baseline for coupling-aware routing, not a final algorithm. Its value is to make escalation observable, auditable, and analyzable by regime. It improves robustness over poor routing choices, but does not beat the hybrid stack's runtime.

XII. Limitations

This study is simulator-only. Experiments use PyBullet, fixed robot models, and a finite set of shared-workspace environments. We do not evaluate perception noise, calibration error, controller tracking error, object uncertainty, or hardware execution; the results are planning-system evidence, not deployment evidence.

The environments are fixed rather than procedurally generated. The coupling metrics are heuristic descriptors rather than calibrated physical probabilities, and runtime comparisons remain implementation-sensitive. The selector has no completeness or optimality guarantee.

XIII. Conclusion

This paper presented an empirical study of coupling-aware planner selection for shared-workspace multi-manipulator planning. The central result is not universal planner dominance. It is that coupling-aware escalation provides a useful way to reason about when planning should remain decoupled or hybrid and when it should invoke coupled search.

Across 4,000 tasks and 32,000 full benchmark runs, coupling-aware selection improves paired solve outcomes over random and pure joint routing while matching the best fixed-mode solve rate. The always-hybrid stack remains the fastest fixed policy, pointing to the next step: budget-aware selectors that learn when to trust a strong hybrid default. More broadly, multi-manipulator planning papers should report not only which planner wins, but why the winning regime changes with task coupling.

References

[1] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Technical Report TR 98-11, Computer Science Department, Iowa State University, 1998.

[2] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2000, pp. 995–1001.

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[5] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2997–3004.

[6] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. IEEE International Conference on Robotics and Automation*, 2015, pp. 3067–3074.

[7] L. Janson, E. Schmerling, A. A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[8] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, pp. 477–521, 1987.

[9] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robotics and Autonomous Systems*, vol. 41, no. 2–3, pp. 89–99, 2002.

[10] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[11] J. R. Rice, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[12] C. P. Gomes and B. Selman, "Algorithm portfolios," *Artificial Intelligence*, vol. 126, no. 1–2, pp. 43–62, 2001.

[13] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2018, pp. 7087–7094.

[14] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2021.

[15] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[16] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[17] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems*, 2013.

[18] S. Zhang and F. Pecora, "Online and scalable motion coordination for multiple robot manipulators in shared workspaces," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 3, pp. 2657–2676, 2024.

[19] W. Guo, Z. Kingston, K. Hang, and L. E. Kavraki, "Efficient multi-robot motion planning for manifold-constrained manipulators by randomized scheduling and informed path generation," *IEEE Robotics and Automation Letters*, 2026, doi: 10.1109/LRA.2026.3662639.

[20] Y. Shaoul, I. Mishani, S. Vats, J. Li, and M. Likhachev, "Multi-robot motion planning with diffusion models," in *Proc. International Conference on Learning Representations*, 2025.

[21] J. Liang, J. K. Christopher, S. Koenig, and F. Fioretto, "Simultaneous multi-robot motion planning with projected diffusion models," in *Proc. International Conference on Machine Learning*, 2025.

[22] Y. El Ghazi, K. Subrin, H. Mouchere, and O. Cardin, "High-level multi-robot coordination in a shared workspace for pick-and-place operations using MILP optimization and heuristic for conflict resolution," *Robotics and Autonomous Systems*, 2026, Art. no. 105465, doi: 10.1016/j.robot.2026.105465.

[23] Y. El Ghazi, K. Subrin, H. Mouchere, and O. Cardin, "Coupled vs. decoupled motion planning for four-robotic arm manipulation in a shared workspace: An empirical study," in *Proc. 7th International Conference on Control and Robotics (ICCR)*, Kyoto, Japan, Dec. 2025, pp. 13–19, doi: 10.1109/ICCR67607.2025.11372088.

[24] Z. Jin, J. Yu, Y. Liang, Y. Wang, Z. Wang, and C. Hu, "CO-DOSP: A hierarchical optimization-based motion planner for multi-robot manipulation in confined and task-constrained workspace," *Advanced Engineering Informatics*, vol. 69, Art. no. 103923, 2026, doi: 10.1016/j.aei.2025.103923.

[25] D. Sun and Q. Liao, "Real-time coordination of multiple robotic arms with reactive trajectory modulation," *IEEE Transactions on Robotics*, vol. 41, pp. 200–219, 2025, doi: 10.1109/TRO.2024.3502223.

[26] A. Shaarawy, C. Erdogan, R. Stolkin, and A. Rastegarpanah, "Multi-robot vision-based task and motion planning for EV battery disassembly and sorting," arXiv preprint arXiv:2509.21020, 2025.