CADENCE: Predicting Realized MAPF Execution Time Beyond Sum of Costs

Abhishek S* *BuildMachineLabs* mail

Badrikanath Praharaj* *BuildMachineLabs* mail

Sreeram M.V.* *BuildMachineLabs* mail

Mohan Prakash V *IISc, Bengaluru* mail

***Abstract*—Multi-Agent Path Finding (MAPF) algorithms are increasingly used to plan motion for robot teams in industrial warehouses and robotic shared workspaces, but standard MAPF algorithm evaluation metrics, such as Sum of Costs (SoC), makespan, and planner runtime, can obscure how planner choices translate into realistic execution performance. We present CADENCE (Coordination- and Action-Driven Estimation for Networked Continuous Execution), a hardware study of this evaluation gap on a fixed 7×7 workcell with seven differential-drive robots, asking which features available before execution can best predict final wall-clock completion time. We compare SoC, total planned travel cost, primitive motion burden (how much basic motion the plan requires, such as makespan, turns, consecutive moves, and start–stop transitions), and interaction-aware coordination structure (how much inter-robot coordination the plan induces, such as dependency links, interacting robot pairs, dependency depth, and crowding exposure). To test this, we generate 120 plans across 15 scenarios - 5 Empty, 5 Medium-Random, and 5 Bottleneck—and execute each plan four times, yielding a 480-trial hardware corpus. Using both a scenario-held-out ridge model and a trial-level mixed-effects model, we find that SoC alone is informative but incomplete, while primitive motion burden gives the strongest improvement, reducing held-out error by about 48.6%–59.8% in MAE and 44.2%–61.4% in RMSE relative to SoC-only models. Interaction-aware coordination features add smaller, less uniform gains, most clearly in the mixed-effects analysis. Across both models and uncertainty checks, primitive motion burden is the most reliable additional signal beyond SoC, suggesting that much of the execution time gap is already visible in the offline plan before any robot starts moving.**

***Index Terms*—Multi-Agent Path Finding, Multi-Robot Coordination, Execution Time Prediction, Real-Hardware Evaluation, Plan Quality Metrics, Sum of Costs**

I. Introduction

MAPF algorithms are deployed in warehouse-style multi-robot systems, where many robots navigate shared aisles to reach goals without collisions. In these environments, collision avoidance and efficient goal attainment are critical. A plan may seem efficient during offline computation, but its execution on physical hardware can be slower due to factors beyond the nominal plan cost, such as motion dynamics and coordination during deployment. Effective deployment requires considering both planned costs and the dynamics of motion and coordination among robots. This motivates the central research question: Once an MAPF plan is executed on physical robots, to what extent can the execution time be predicted before deployment? The study examines how much of the observed execution time is evident in the pre-execution plan.

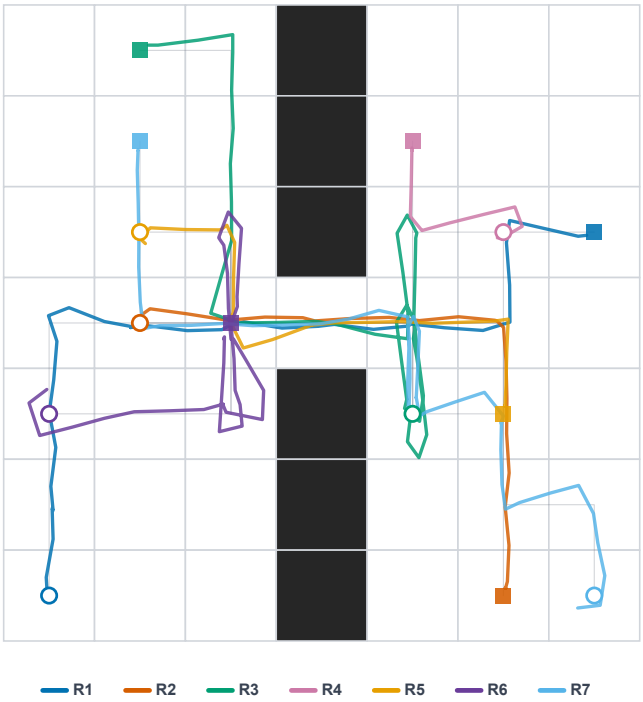

Fig. 1. **Hardware OptiTrack trace.** Color-coded trajectories for Robots 1–7 on a representative 7×7 bottleneck execution. Circles mark starts; squares mark goals.

This question matters because physical execution introduces structure that scalar plan costs suppress. Robots must preserve the same-agent action order, satisfy cross-robot precedence constraints, pass through locally contested spaces, and often incur stop–go delays under an execution monitor [1], [2], [3]. As a result, plans with similar SoC can produce materially different wall-clock completion times on hardware. Figure 1 shows one representative bottleneck execution in which the realized trajectories reflect these effects. This motivates the layered view used in this paper: SoC captures a coarse plan-cost trend, primitive motion burden captures how much basic motion a plan demands, and interaction-aware coordination structure captures how much inter-robot coordination load the plan induces.

We study this as a hardware transfer question using Pololu 3Pi+ validated using Optitrack Motion Capture system Prime 13W. The same agent action order requires each robot to

execute its planned actions sequentially; cross-robot precedence constraints may require one robot to wait until another completes a dependent move. Local contention occurs when multiple robots compete for access to the same narrow region or shared space. Repeated waiting behavior can lead to cumulative stop-and-go delays during execution. Figure 2 illustrates the hardware platform and the three scenario families used to represent these interaction regimes. Subsequently, a four-model ladder is evaluated using scenario-level held-out validation to determine which plan side descriptors, computable prior to robot movement, best predict the realized execution time on this fixed hardware stack.

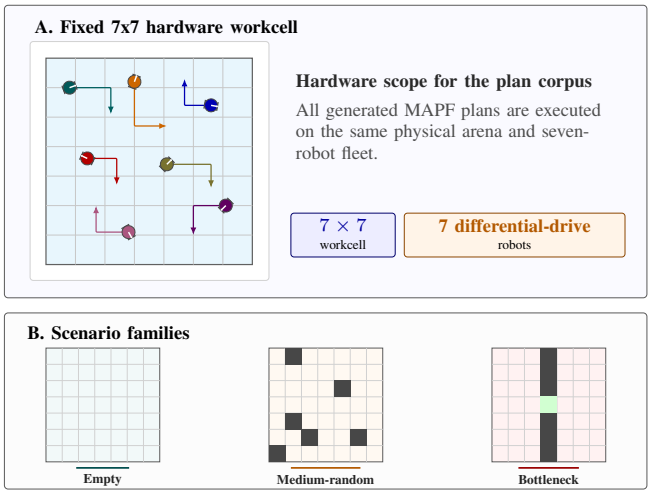

Fig. 2. **Hardware platform and scenario-family overview.**

This paper makes three contributions:

- A real hardware execution time corpus for offline MAPF plans on a fixed multi-robot platform, built with an interaction-regime-stratified scenario library.
- A disciplined three-layer decomposition of plan-side predictors of realized execution time. We separate plan descriptors into three analytically distinct layers: (i) SoC, the standard MAPF plan quality scalar; (ii) primitive motion burden total turn count, consecutive-move runs, and start–stop transitions; and (iii) interaction-aware plan structure cross-robot precedence count, precedence depth, and crowding exposure
- A cross-scenario test of where the execution-time signal lives beyond SoC. In scenarios withheld entirely from training, Primitive motion burden cuts held-out MAE from 2.76 s to 1.42 s beyond SoC; interaction-aware plan structure contributes a smaller, consistent additional predictive signal beyond primitive burden .

II. Related Work and Positioning

MAPF algorithm evaluation is often plan-centric: benchmarks and surveys typically compare solvers. These comparisons often utilize metrics such as SoC, planned makespan, planner runtime, and success rate. Bounded-sub-optimality is also assessed in standardized instances [4]. These metrics are appropriate when the objective is to analyze search algorithms and planner trade-offs. However, the present inquiry differs: given a plan intended for execution on a physical robotic system, which properties of that plan account for realized execution time beyond its nominal cost? optimal solvers, such as Conflict-Based Search (CBS) [5], generate plans with well-defined cost properties. However, cost minimality does not necessarily result in minimal realized execution time when factors such as robot motion, precedence constraints, and cross-robot interaction structure are taken into account [6], [1].

The closest conceptual precedent is Yan et al. [7], who use SMART realistic simulation to study how MAPF planner-design choices translate into execution performance. Their results show that SoC captures the trend of average execution time, that execution related structure improves execution time estimates, and that model accuracy can matter more than aggressive refinement of a simplified objective. We ask the corresponding hardware question: on a physical robot stack, do pre-execution plan descriptors beyond SoC improve held-out prediction of realized wall-clock completion time? The present work, therefore, differs in three ways. First, it measures physical hardware rather than simulated execution; a diagnosis that holds in simulation need not survive motor dynamics, tracking noise, and real contention. Second, it decomposes the missing signal into primitive motion burden and interaction-aware coordination structure. Third, it tests a compact feature ladder under scenario-held-out validation, so the claim is a platform-scoped hardware transfer result

More broadly, this study relates to realistic execution and dependency-graph methodologies for MAPF [1], [2], [8], [9], [3], [10]. These works show that precedence constraints, congestion, and delay propagation can change realized execution performance after a plan is computed. Building on this execution-structure perspective, we ask a hardware-transfer question: whether primitive motion burden and interaction-aware plan structure, measured before execution, provide held-out predictive value for realized wall-clock execution time beyond SoC.

III. Study Design, Scenario Library, and Plan Corpus

A. Platform and task scope

The study measures realized execution time on a fixed 7×7 workcell with seven differential-drive Pololu 3Pi+ robots. The target is not nominal planner quality but total wall-clock completion time under the execution system described in Section IV. By holding the platform, executor, and task geometry fixed, the analysis isolates which plan-side descriptors transfer to hardware on this system.

B. Scenario library

The scenario library contains 15 scenarios, five from each of three declared families: Empty, Medium-Random, and Bottleneck. The Empty family acts as an obstacle-free control with

minimal structural routing constraint. The Medium-Random family introduces moderate routing restriction through fixed random obstacle layouts that create corridor sharing and local detours without collapsing all motion into a single choke point. The Bottleneck family concentrates narrow-passage dependency pressure through a fixed contested passage geometry. We use this library to span distinct interaction regimes rather than to approximate a broad benchmark suite [4], [2], [8], [9]. This family balance is the declared corpus-design choice. Within each family, candidate start–goal assignments are generated by seeded validity rules, with CBSH2-RTC used as the scenario-admission feasibility oracle before final scenario identities are frozen. The resulting 15-scenario library is a pre-specified interaction-regime sample for this platform, fixed before hardware execution and regression fitting.

TABLE I
STUDY PROTOCOL AND HARDWARE CORPUS.

Element	Value
Platform	Seven differential-drive robots on a 7×7 physical workcell
Scenario library	15 scenarios: 5 Empty, 5 Medium-Random, 5 Bottleneck
Plan-generation menu	1 CBSH2-RTC; 4 EECBS weights; 3 LaCAM seeds per scenario
Corpus construction	Seeded scenario freeze; fixed menu for every scenario
Plans per scenario	8 planned MAPF solutions
Plan corpus	120 planned MAPF plans
Hardware corpus	480 scheduled executions; four per plan
Hardware repetitions	4 scheduled executions per plan
Evaluation target	Plan-level mean execution time
Execution monitor	Precedence-faithful continuous executor
Measured target	Total execution wall-clock time
Held-out evaluation	Scenario-level family-balanced 5-fold split
Estimators	Ridge plan-mean ladder and trial-level mixed-effects ladder
Metrics	Held-out MAE and RMSE in seconds

C. Plan-corpus construction

For each frozen scenario, we instantiate the same eight plan-generation slots. The fixed menu contains one optimal reference plan from CBSH2-RTC, four bounded-suboptimal plans from EECBS at multiple weight settings, and three seed-varied plans from LaCAM3 [11], [12], [13], [14], [15]. Applied uniformly to all hardware corpus whose prediction target is plan-level mean wall-clock time rather than a single run. The declared generator menu induces cost and structural variation under a feasible hardware budget while keeping the construction auditable: every scenario receives the same treatment. Accordingly, the study asks whether progressively richer plan descriptors explain realized execution time better than SoC alone, not which planner should be declared best.

IV. Execution System, Feature Ladder, and Held-Out Evaluation

A. Hardware execution system

Hardware trials are executed under a precedence-faithful continuous executor (i.e., an executor that preserves the dependency order implied by the plan, so each stage starts only after the robot's previous stage and all required cross-robot predecessor stages have completed). The MAPF plan is treated as a shared logical schedule whose same-agent stage order and cross-robot precedence constraints must be preserved during execution. However, execution is not barrier-synchronized by a global timestep release clock. A stage may begin only after the previous stage of that robot has completed and all incoming cross-robot dependencies have been satisfied [1], [2], [3]. This choice keeps the execution semantics close to the logical coordination structure of the plan while still exposing real wall-clock delay, local contention, and stop-go behavior on hardware, as shown in Figure 3. It also makes the executor an explicit part of the studied system rather than a hidden implementation detail. Accordingly, all predictive claims are executor-conditioned: alternative release policies or switchable passing-order Controllers may reshape realized waiting and should be evaluated as separate execution stacks.

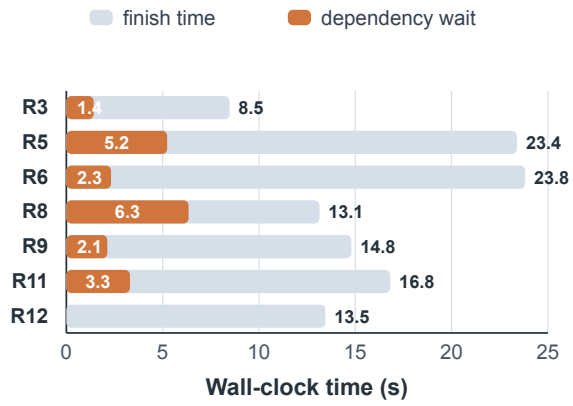

Fig. 3. **Executor dependency-wait diagnostic.** In one representative hardware trial, dependency-gated waiting accounts for a visible share of several robots' wall-clock finish times. This illustrates the execution mechanism rather than a corpus-level result.

B. Pre-execution feature extraction

All predictors are computed before hardware execution from the MAPF plan alone; no hardware timing outcome enters the extraction pipeline. For each normalized plan, we first compute the nominal plan-cost and primitive-motion features directly from the per-robot path sequences:SoC, planner-side makespan, total turn count, consecutive-move count, and start–stop transition count.

We then compile the same plan into an Action Dependency Graph (ADG) [2] that the precedence-faithful continuous executor of Section IV consumes at run time. In this ADG, nodes are per-robot stages, intra-robot edges encode the same-agent sequential order, and inter-robot edges encode cross-robot precedence induced by shared vertex occupancy and edge swaps under the planned ordering. The executor releases each stage only after its intra-robot predecessor and all incoming inter-robot predecessors have completed.

The interaction-aware features are then read directly off this ADG: the number of inter-robot precedence edges, the number of distinct robot pairs connected by those edges, and the depth of the longest inter-robot dependency chain. Crowding exposure is reported separately and is not an ADG edge count; it is computed from the co-occupancy of planned robot states over active portions of the trajectories, and is included as a plan-trace proximity descriptor that complements the ADG-derived edge features.

C. Feature ladder

We evaluate a predefined four-level model ladder:

- *Model 0 (null baseline):* a baseline with no plan-side predictive features, used only as a reference for the richer models.
- *Model 1 (SoC only):* uses Sum of Costs (SoC, the total planned travel cost summed across all robots) as the only predictor.
- *Model 2 (SoC + primitive motion burden):* augments SoC with four execution-aware motion features that capture how much basic motion the plan demands: planner-side makespan (the logical number of timesteps until the last robot reaches its goal), total turn count (the total number of direction changes across all robots), consecutive-move count (how often robots continue moving across successive steps without pausing), and start–stop transition count (how often robots switch between moving and waiting).
- *Model 3 (SoC + primitive motion burden + interaction-aware coordination load):* further adds four coordination features that capture how much inter-robot interaction the plan induces: the number of cross-robot precedence edges (dependency links where one robot must wait for another), the number of distinct robot pairs they connect (how many different robot pairs are involved in such dependencies), the depth of the longest cross-robot dependency chain (the length of the longest multi-robot wait sequence), and a crowding-exposure count (how much planned motion passes through shared space at the same logical time).

Planner-side makespan is used here as a logical plan-horizon feature in timesteps rather than as a direct estimate of physical seconds; the prediction target remains realized wall-clock execution time.

D. Outcome and validation

The primary target is realized total execution wall-clock time, measured per trial from the hardware logs. Each plan is executed four times, but The four repeats of one plan are not treated as four independent test rows: between-plan variance accounts for 99.4% of total execution-time variance and the median within-plan coefficient of variation is 1.6%, so the plan-level mean is a stable summary and the Repeats serve as a variability estimate rather than independent confirmatory evidence. The confirmatory unit is therefore the plan mean.

Held-out validation is then performed at the *scenario* level, not the plan level. All plans and all repeats sharing a scenario must fall on the same side of the train/test split, otherwise the model would see the same start–goal layout in both training and evaluation. We use a 5-fold family-balanced scenario split: each fold holds out three scenarios (one Empty, one Medium-Random, one Bottleneck), the ridge ladder is fit on the remaining 12, and MAE/RMSE are computed on the held-out plan means.

We report two trained estimators over the same M0–M3 feature ladder. The ridge model is a ridge-regularized linear ladder fit to plan-mean outcomes, with fold-local feature standardization, an unpenalized intercept, and ridge penalty selection by scenario-grouped inner cross-validation inside each outer training fold. The mixed-effects model is fit at the trial level over repeated hardware executions and is evaluated on held-out plan means so that its MAE and RMSE are directly comparable to the ridge model. Both estimators use the same scenario-level family-balanced 5-fold outer holdout. The held-out unit is the scenario, so all plans and repetitions derived from one scenario remain in the same fold. With the 15-scenario library, each outer fold holds out three scenarios: one Empty, one Medium-Random, and one Bottleneck.

The staged comparisons are the same for both trained models: first, whether primitive execution burden improves on SoC alone, and second, whether interaction-aware structure improves on the primitive-motion tier. We report MAE and RMSE in seconds. For each stepwise comparison, scenario-blocked Bootstrap intervals and paired scenario-level t intervals summarize the reliability of the held-out error change. Deltas are defined as higher-tier error minus lower-tier error, so negative values indicate improvement. All interpretations remain platform-scoped and executor-conditioned.

V. Results

This section reports held-out predictive performance for the same M0–M3 feature ladder under two trained models on the 480-trial hardware corpus. The ridge model evaluates scenario-held-out plan-mean prediction. The mixed-effects model fits repeated trial-level executions and is evaluated on held-out plan means for the same MAE and RMSE readout. The two central comparisons are $M_1 \rightarrow M_2$, which tests whether primitive motion burden adds signal beyond SoC, and $M_2 \rightarrow M_3$, which tests whether interaction-aware structure adds held-out improvement beyond the primitive-motion tier.

Tables II and III present the main readout. The shared pattern is clear: SoC alone is incomplete, and the primitive-motion tier gives the largest reduction in held-out error. In the ridge model, M_3 gives a small favorable point estimate whose intervals cross zero.

In the mixed-effects model, M_3 gives a larger additional reduction, with bootstrap intervals below zero and scenario-level t intervals close to zero. Figure 1 anchors these results in one measured hardware execution from the same corpus.

TABLE II

HELD-OUT MODEL-LADDER ERRORS FOR THE RIDGE AND MIXED-EFFECTS MODELS ON THE HARDWARE CORPUS. BOTH PANELS USE THE SAME M0–M3 FEATURE LADDER. DELTA COLUMNS ARE RELATIVE TO THE PREVIOUS MODEL TIER; NEGATIVE VALUES INDICATE BETTER PREDICTION.

A. Held-out ridge error by model tier

Model	Feature set	MAE	ΔMAE	RMSE	ΔRMSE	Interpretation
M_0	Null	4.9998	–	6.4507	–	baseline
M_1	SoC only	2.7626	−2.2372	3.4118	−3.0389	informative but incomplete
M_2	SoC + primitive motion burden	1.4196	−1.3430	1.9029	−1.5089	large improvement
M_3	SoC + motion + interaction-aware structure	1.3969	−0.0228	1.7513	−0.1516	small point gain

B. Held-out mixed-effects error by model tier

Model	Feature set	MAE	ΔMAE	RMSE	ΔRMSE	Interpretation
M_0	Null	5.0352	–	6.4750	–	baseline
M_1	SoC only	4.7933	−0.2419	6.3315	−0.1434	informative but incomplete
M_2	SoC + primitive motion burden	1.9273	−2.8660	2.4469	−3.8846	large improvement
M_3	SoC + motion + interaction-aware structure	1.4310	−0.4963	1.7885	−0.6584	additional reduction

TABLE III

STEPWISE HELD-OUT ERROR CHANGES FOR THE RIDGE AND MIXED-EFFECTS MODELS ON THE HARDWARE CORPUS. NEGATIVE DELTAS MEAN THE RICHER MODEL HAS BETTER PREDICTION. BOOTSTRAP INTERVALS RESAMPLE SCENARIO BLOCKS; t INTERVALS USE ONE PAIRED DELTA PER SCENARIO.

A. Tested ridge increments with two interval estimates

Comparison	Metric	Δ error	Bootstrap 95% CI	t 95% CI	Interpretation
M_2 vs. M_1	MAE	−1.3430	[−2.2096, −0.5878]	[−2.3043, −0.4268]	supported
M_2 vs. M_1	RMSE	−1.5089	[−2.4946, −0.5780]	[−2.3284, −0.4421]	supported
M_3 vs. M_2	MAE	−0.0228	[−0.4995, 0.4740]	[−0.5751, 0.5188]	not confirmed
M_3 vs. M_2	RMSE	−0.1516	[−0.6296, 0.4421]	[−0.6756, 0.4298]	not confirmed

B. Tested mixed-effects increments with two interval estimates

Comparison	Metric	Δ error	Bootstrap 95% CI	t 95% CI	Interpretation
M_2 vs. M_1	MAE	−2.8660	[−4.6168, −1.4106]	[−4.8200, −1.0468]	supported
M_2 vs. M_1	RMSE	−3.8846	[−5.5813, −1.8878]	[−4.6962, −1.0334]	supported
M_3 vs. M_2	MAE	−0.4963	[−1.0122, −0.0385]	[−1.0527, 0.0493]	bootstrap below zero; t near zero
M_3 vs. M_2	RMSE	−0.6584	[−1.0969, −0.1936]	[−1.1181, 0.0075]	bootstrap below zero; t near zero

A. Primitive motion burden recovers the largest shared signal

Primitive execution burden produces the largest shared improvement beyond SoC. In the ridge model, M_2 reduces MAE from 2.7626 to 1.4196 s (down by 1.3430 s, or about 48.6%) and RMSE from 3.4118 to 1.9029 s (down by 1.5089 s, or about 44.2%). The ridge M_2-M_1 intervals are below zero for both metrics: MAE bootstrap 95% CI $[-2.2096, -0.5878]$ and t 95% CI $[-2.3043, -0.4268]$; RMSE bootstrap 95% CI $[-2.4946, -0.5780]$ and t 95% CI $[-2.3284, -0.4421]$, supporting the interpretation that primitive motion burden adds reliable held-out predictive value beyond SoC. The mixed-effects model shows the same feature-family message with even larger stepwise reductions: MAE falls from 4.7933 to 1.9273 s (down by 2.8660 s, or about 59.8%) and RMSE from 6.3315 to 2.4469 s (down by 3.8846 s, or about 61.4%), again with both bootstrap and t intervals below zero. Because these features are computable from the offline plan before execution begins, this result identifies primitive motion burden as the strongest missing layer between SoC and realized hardware execution time.

B. SoC is informative but incomplete

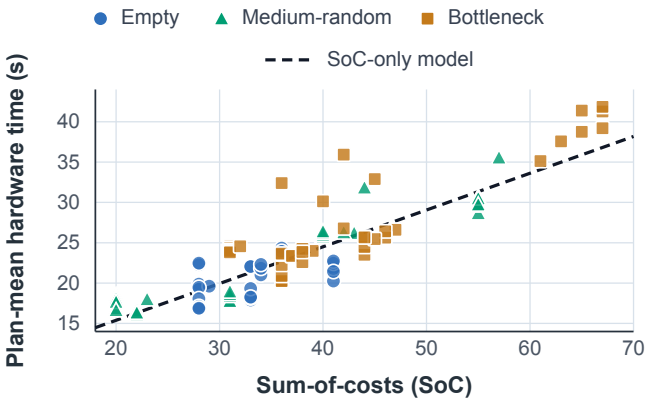

Fig. 4. **SoC-only model.** Each point is a plan mean; the dashed line is the SoC-only model. The vertical spread shows why SoC alone is not a complete predictor of hardware time.

In the ridge model, SoC reduces held-out MAE from

4.9998 to 2.7626 s and RMSE from 6.4507 to 3.4118 s. In the mixed-effects model, the SoC-only tier remains close to the null baseline, with MAE 4.7933 s and RMSE 6.3315 s. The combined reading is not that SoC is irrelevant. Rather, SoC is too coarse to serve as a complete predictor for realized wall-clock cost on this platform, especially once a repeated trial structure is modeled.

C. Interaction-aware structure under the current corpus

Adding the interaction-aware feature tier on top of the primitive-motion tier moves both estimators in the same direction. (For reference, the primitive-motion tier model M2 adds per-plan turn count, start–stop count, and total commanded path length on top of SoC; the interaction-aware tier model M3 adds, on top of M2, schedule-level coordination quantities such as cross-robot dependency count, dependency-chain depth, and crowding exposure.) Under ridge, MAE drops from $1.4196\,\mathrm{s}$ to $1.3969\,\mathrm{s}$ ($-0.0228\,\mathrm{s}$, $\approx 1.6\%$) and RMSE from $1.9029\,\mathrm{s}$ to $1.7513\,\mathrm{s}$ ($-0.1516\,\mathrm{s}$, $\approx 8.0\%$); both held-out intervals cross zero. Under the mixed-effects model, MAE drops from $1.9273\,\mathrm{s}$ to $1.4310\,\mathrm{s}$ ($-0.4963\,\mathrm{s}$, $\approx 25.8\%$) and RMSE from $2.4469\,\mathrm{s}$ to $1.7885\,\mathrm{s}$ ($-0.6584\,\mathrm{s}$, $\approx 26.9\%$), with bootstrap intervals below zero on both metrics, while the scenario-level t intervals span MAE $[-1.0527,\, 0.0493]$ and RMSE $[-1.1181,\, 0.0075]$.

We therefore report the interaction-aware tier as an additional, directionally consistent layer of the model ladder rather than as a settled effect-size claim. The evidence is aligned in direction across both trained models, but the present corpus does not support a precise statement about how large this additional coordination-structure signal is on the platform. Sizing that layer cleanly is the natural target for a larger held-out scenario corpus.

D. Reading the two estimators together

The estimator comparison defines the current evidence boundary. Both ridge and mixed-effects analyses show that sum-of-costs is too coarse for realized hardware execution time and that primitive-motion burden materially improves prediction. For the interaction-aware tier, the estimates are aligned in direction but not equally resolved by the present corpus: ridge on held-out plan means does not separate the interaction-aware tier from the primitive-motion tier with intervals away from zero, whereas the mixed-effects analysis estimates a larger reduction after using the repeated executions. Given the current corpus size, we are not able to certify the reliability or precise signal magnitude of the additional interaction-aware features. The next empirical step is therefore to expand the scenarios. The joint reading is therefore that primitive motion burden is the supported missing layer between SoC and realized execution time on this platform; interaction-aware structure carries a directionally consistent additional signal whose resolution under the primary held-out comparison is bounded by the information content of the present corpus.

VI. Discussion and Limitations

How can these predictions be used in deployment, and what remains uncertain? Because these descriptors are computable from the offline plan before any robot moves, they can be used to score or compare candidate plans using a quantity that is closer to realized execution time than SoC alone on this hardware stack. In practice, the predicted values are best used for plan selection, or deployment-side screening among candidate plans generated for the same platform and executor, rather than as universal execution-time guarantees. The main uncertainty is scale transfer: all claims here remain platform-scoped and executor-conditioned, so transfer to larger fleets or different execution policies requires separate hardware validation.

Limitations.

- *Controlled hardware scope.* The findings are conditioned on seven robots, a 7×7 arena, and one precedence-faithful continuous executor, so transfer to larger fleets or different execution stacks requires new hardware evidence.
- *Compact scenario support.* The 15 scenarios span Empty, Medium-Random, and Bottleneck regimes, but they do not constitute a broad benchmark suite.
- *Declared corpus design.* The fixed eight-slot plan-generation menu makes the corpus auditable and repeatable, but not representative of all MAPF tasks, solvers, or deployment settings.
- *Interpretable rather than maximal models.* The ridge and mixed-effects ladders are designed as disciplined, interpretable tests of feature families, not as upper bounds on predictive performance.

Acknowledgment

The authors thank Prof. Pavankumar Tallapragada, Associate Professor at the Robert Bosch Centre for Cyber-Physical Systems (RBCCPS), Indian Institute of Science (IISc), Bengaluru, for his guidance and the resources that made this work possible. The authors also acknowledge BuildMachineLabs, a research-centric laboratory based in Bengaluru, for the initial motivation and continued support for this effort.

VII. Conclusion

This study concludes that standard MAPF plan quality is insufficient to predict realized execution time on a physical multi-robot workcell and additional plan-side structure is needed before execution. This is validated using a seven-robot, 7×7 hardware platform under a precedence-faithful continuous executor, SoC captures an important trend but does not fully explain wall-clock completion time. The clearest additional signal is primitive motion burden: adding planner-side makespan, turns, consecutive moves, and start–stop transitions to SoC gives the largest and most reliable improvement across both trained analyses, reducing held-out MAE from $2.7626\,\mathrm{s}$ to $1.4196\,\mathrm{s}$ in the ridge model and from $4.7933\,\mathrm{s}$ to $1.9273\,\mathrm{s}$ in the mixed-effects model.The Interaction-aware coordination structure points to a further layer of execution signal, but the

present hardware corpus does not settle its reliability or magnitude. The interaction-aware tier improves the point estimates in both analyses, while the strictest held-out comparison does not separate it from the primitive-motion tier with intervals away from zero. The resulting design implication is that execution-aware MAPF evaluation should not rely on SoC alone: even in a compact hardware study, much of the missing execution-time signal is captured by primitive motion descriptors before any robot moves.

REFERENCES

[1] H. Ma, T. K. S. Kumar, and S. Koenig, "Multi-agent path finding with delay probabilities," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[2] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and robust execution of mapf schedules in warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.

[3] Y. Su, R. Veerapaneni, and J. Li, "Bidirectional temporal plan graph: Enabling switchable passing orders for more efficient multi-agent path finding plan execution," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 17 559–17 566.

[4] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, R. Bartak, and E. Boyarski, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the Annual Symposium on Combinatorial Search*, 2019.

[5] 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.

[6] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Magazine*, vol. 29, no. 1, pp. 9–20, 2008.

[7] J. Yan, Z. Li, W. Kang, S. F. Smith, and J. Li, "Analyzing planner design trade-offs for mapf under adg-based realistic execution," 2025.

[8] S. Varambally, J. Li, and S. Koenig, "Which mapf model works best for automated warehousing?" *Proceedings of the International Symposium on Combinatorial Search*, vol. 15, no. 1, pp. 190–198, 2022.

[9] Z. Chen, D. Harabor, J. Li, and P. J. Stuckey, "Traffic flow optimisation for lifelong multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 18, 2024, pp. 20 674–20 682.

[10] T. Duhan, C. He, and G. Sartoretti, "P3gasus: Pre-planned path execution graphs for multi-agent systems at ultra-large scale," *IEEE Robotics and Automation Letters*, vol. 11, no. 2, pp. 1274–1281, 2026.

[11] J. Li, A. Felner, E. Boyarski, H. Ma, and S. Koenig, "Improved heuristics for multi-agent path finding with conflict-based search," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019, pp. 442–449.

[12] J. Li, W. Ruml, and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, 2021, pp. 12 353–12 362.

[13] K. Okumura, M. Machida, X. Defago, and Y. Tamura, "Lacam: Search-based algorithm for quick multi-agent pathfinding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 10, 2023, pp. 11 655–11 662.

[14] K. Okumura, R. Kuroiwa, and Y. Tamura, "Engineering lacam*: Towards real-time, large-scale, and near-optimal multi-agent pathfinding," in *International Conference on Autonomous Agents and Multiagent Systems*, 2024.

[15] K. Okumura, "Engineering LaCAM: Towards real-time, large-scale, and near-optimal multi-agent pathfinding," 2024, presented at AAMAS-24. [Online]. Available: https://arxiv.org/abs/2308.04292