A Experiment on Seed

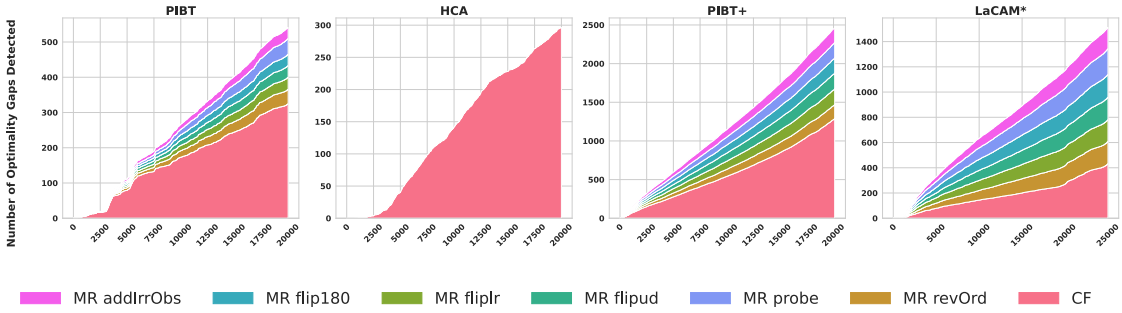

Figure 10: Cumulative number of Completeness Optimality Failures with maze seed, τ = 25%.

Setup. To further evaluate the influence of seed composition on fuzzing performance, we extended the seed set of MapfFuzz by including maze maps in addition to the original seed maps. Other experimental settings are the same as Table 1. For each solver, the experiments were repeated five times.

Results and Analysis. The results show that the inclusion of maze maps introduces diverse effects across solvers. For PIBT, the number of detected Optimality Failures decreases from 925.4 ± 590.2 to 541.2 ± 286.0, indicating no improvement. This reduction stems from PIBT's inability to produce valid solutions in maze-like environments—it tends to fail completely when faced with tight corridors or dead ends. Since our framework discards instances if the original test case fails, such failures are not counted, resulting in an apparent drop in the detection count. Thus, maze-type maps are unsuitable as fuzzing seeds for PIBT.

For HCA, the detection count also drops, from 2052.8 ± 3254.7 to 296.4 ± 93.2. Note that all of these cases are completeness failures. This finding is consistent with the MET-MAPF study, where maze benchmarks also failed to produce MR violations for HCA.

In contrast, PIBT+ and LaCAM* both show substantial improvements with maze seeds (PIBT+: 824.2 ± 482.8 → 2467.4 ± 407.7; LaCAM*: 500.6 ± 1026.5 → 1511.8 ± 629.6). These solvers are more resilient to complex layouts and can still generate feasible solutions within mazes, allowing MapfFuzz to uncover deeper suboptimal behaviors. This trend aligns with the findings from MET-MAPF, where both algorithms were able to detect MR violations in maze settings. The challenging structures—such as corridors, bottlenecks, and junctions—stimulate diverse failure patterns, thereby enhancing the fuzzing framework's defect coverage.

This experiment highlights the importance of matching the difficulty level of seed maps with solver capabilities. Overly difficult maps, such as mazes, may reduce effective coverage for solvers prone to infeasibility (e.g., PIBT and HCA), producing mostly completeness failures rather than meaningful violations. Such maps are better reserved for controlled benchmark testing rather than fuzzing-based mutation. Conversely, including difficult maps can significantly enhance detection performance for robust solvers like PIBT+ and LaCAM*. Therefore, introducing maps of appropriate complexity as seeds can improve the overall defect-detection ability of MapfFuzz.

B Questionnaire Study

Setup. To validate the failures detected by MapfFuzz through independent assessors, we conducted a small questionnaire-based study. The questionnaire contained 10 randomly selected test cases, covering both completeness and optimality failures across four MAPF solvers. Participants were asked to determine whether each case represented a genuine defect and to specify its type. At the end of the questionnaire, they were also asked whether such cases could be useful for improving solver design and whether the 25% violation threshold adopted in this paper was reasonable.

In total, we collected six responses from MAPF solver designers (Experts) and eight from participants familiar with MAPF but not directly involved in solver development (Informed Participants).

Results. Table 3 shows the results of our questionnaires. In our analysis, we report the proportion of "Yes" answers among respondents who gave a definite yes/no judgment, and separately indicate the fraction of "Unsure" responses.

All six experts unanimously confirmed that every sampled case represents a genuine defect, while a small fraction of "Unsure" responses (1/6) appeared in a few cases. Among the eight informed participants, the majority also validated the correctness of the detected failures, with agreement rates ranging from 86% to 100%. Both groups consistently identified the failure types, except several divergences between deadlocks and livelocks.

All participants agreed that the discovered cases could be useful for guiding algorithmic improvements. Typical comments noted that certain failures reveal extreme behaviors. For instance, algorithmic inefficiencies in LaCAM that lead to exponential cost growth, and

deadlocks in PIBT that could be mitigated by introducing dedicated handling mechanisms. These findings indicate that the failures produced by MapfFuzz are not artificial artifacts but *actionable insights* for solver developers.

Regarding the additional questions, both experts and informed participants unanimously agreed that the 25% of MD is a reasonable criterion for defining performance-deficient instances. Two experts selected Unsure and provided further comments. One expert noted that, while the MD metric is reasonable, there is no definitive evidence supporting a perfect violation threshold—lower thresholds may still indicate meaningful defects. The other expert agreed that the MD metric is useful but not sufficiently comprehensive, suggesting that it would be better to estimate both the lower and upper SOC bounds of a solver across equivalent test cases. We regard this as an ideal extension of our current framework and plan to explore it in future work.

Overall, the results confirm that the failures detected by MapfFuzz are credible, interpretable, and practically relevant from both expert and practitioner perspectives.

Table 3: Questionnaire Result

Question	Experts			Informed Participants		
	Q1 Case Solvability [#Yes/#No/#Unsure]	Q2 Defect Confirmation [#Yes/#No/#Unsure]	Q3 Defect Type [Type]	Q1 Case Solvability [#Yes/#No/#Unsure]	Q2 Defect Confirmation [#Yes/#No/#Unsure]	Q3 Defect Type [Type]
Case 1	6/0/0	6/0/0	6 Deadlock	8/0/0	8/0/0	8 Deadlock
Case 2	6/0/0	6/0/0	3 Deadlock 3 Livelock	7/0/1	8/0/0	8 Deadlock
Case 3	6/0/0	6/0/0	6 Large Optimal Gap	7/0/1	7/0/1	7 Large Optimal Gap
Case 4	6/0/0	6/0/0	6 Unsolved	8/0/0	8/0/0	8 Unsolved
Case 5	6/0/0	5/0/1	5 Large Optimal Gap	8/0/0	7/0/1	7 Large Optimal Gap
Case 6	6/0/0	6/0/0	5 Deadlock 1 Livelock	8/0/0	7/0/1	7 Deadlock
Case 7	5/0/1	5/0/1	3 Deadlock 2 Livelock	8/0/0	8/0/0	8 Deadlock
Case 8	6/0/0	6/0/0	6 Large Optimal Gap	8/0/0	8/0/0	5 Large Optimal Gap 3 Deadlock
Case 9	6/0/0	6/0/0	6 Large Optimal Gap	7/0/1	6/1/1	6 Large Optimal Gap
Case 10	6/0/0	5/0/1	4 Unsolved 1 Large Optimal Gap	8/0/0	6/1/1	4 Unsolved 2 Large Optimal Gap
Useful for solver improvement	6/0/0			8/0/0		
25% SOC violation threshold reasonable	4/0/2			7/0/1		