offline or online solver can also be used.

The implementation is in Java. Experiments were run on an AMD® EPYC 7702P 64-Core with 32 GB of RAM.

Experiment Design and Metrics: For every problem in our benchmark we run each of the evaluated algorithms 50 times. Every such run was terminated when either all agents have declared that their goal has been reached or when the overall number of steps exceeded 200. In the latter case, we forced all remaining agents to perform a declare action.

The main performance metric we considered is the average sum of discounted rewards (ADR), which is a common solution quality measure in the POMDP literature. We also report the standard error of these averages, the average runtime in seconds, and the *success rate*, which is the percentage of runs where all agents reach their respective goal states and performed the declare action appropriately. ADR is not directly correlated with the success rate: when one agent reaches the goal and the other does not, declaring done will produce a positive ADR will not be considered a success. This can be seen in our results, where some cases with 0% success rate still have positive ADR.

P	Alg	ADR	RT	% S	#C	#R
S_1	Joint-FSVI	80 ± 4	671	85%	-	-
	Joint-POMCP	56±6	78	48%	-	-
	POMCP+FL	55 ± 6	3	68%	0.04±0.03	0.04±0.03
	FSVI	80 ± 4	3	88%	0.5±0.1	0.5±0.2
	FSVI+MF	76 ± 4	2	80%	0.3±0.1	0.4±0.1
	FSVI+FL	$\mathbf{79 \pm 4}$	3	88%	0.5±0.1	0.5±0.2
	FSVI+MF+FL	74 ± 5	5	76%	2±2	4±4
S_2	Joint-FSVI	62 ± 6	692	72%	-	-
	Joint-POMCP					
	POMCP+FL	63 ± 4	106	68%	130±28	131±28
	FSVI	61 ± 5	11	64%	5±0.8	7±1
	FSVI+MF	49 ± 5	23	40%	6±3	10±4
	FSVI+FL	64 ± 6	41	80%	27±5	10±11
	FSVI+MF+FL	48 ± 7	50	62%	41±6	80±13
S_3	Joint-FSVI	3 ± 4	3516	10%	-	-
	Joint-POMCP					
	POMCP+FL	0 ± 4	331	12%	90±6	170±10
	FSVI	21 ± 1	17	0%	2.7±0.1	4.3±0.2
	FSVI+MF	17 ± 3	21	0%	2.4±0.1	3.9±0.2
	FSVI+FL	11 ± 5	68	18%	66±7	128±13
	FSVI+MF+FL	3 ± 4	93	10%	70±6	138±13
S_4	Joint-FSVI	57 ± 6	4735	63%	-	-
	Joint-POMCP					
	FSVI	22 ± 1	9	0%	2±0	3±0
	POMCP+FL	-12.2 ± 0	422	0%	99.7±0.06	199±0.1
	FSVI+MF	22 ± 0.6	5	0%	2±0	3±0
	FSVI+FL	21 ± 5	57	36%	58±6	112±13
	FSVI+MF+FL	-12 ± 0	203	0%	100±0	200±0

Table 2: Results for small problems (S_1, S_2, S_3, and S_4).

Results The results for small problems are shown in Table 2 . Columns "P", "RT", and "%S" specify the problem name, runtime in seconds, and the success rate, respectively. We also report the number of potential collisions detected and the number of times an agent replanned, denoted as "#C" and "#R", respectively. $\#C$ and $\#R$ are always zero for *Offline*, since its initial policy is conflict-free.

As expected, Joint-FSVI could only solve in reasonable time the smaller 5x5 problems, S_1, S_2, S_3, and S_4. On the other hand, Joint-FSVI managed in some problems to provide the best policy in terms of success rate and ADR. This is expected, as an optimal POMDP solver on the joint problem, given enough runtime, can return an optimal policy. Joint-POMCP could not scale beyond the smallest problem, due to the large number of joint actions and observations.

The online algorithms perform well on S_1 and S_2, where the agents' shortest path to their goals do not conflict. However, they perform poorly in S_3 and S_4, which represent very difficult problems, where agents must move close together in a tight space. As such, the number of conflicts between the shortest paths is very high, many potential conflicts will be detected and the agents will replan very often. This is clearly observable in the "#C" and "#R" results, which are high and are inversely correlated with the ADR.

Another observation from these results is that when many conflicts are expected (S_3 and S_4), the modified FSVI technique is not beneficial, but the forced localization works better than the vanilla FSVI. This is because in such small grids localizing is very helpful, allowing agents to move at least one step without colliding, and then localizing again. Looking at the run time of the different algorithms, as expected, computing the joint policy is much more computationally demanding, requiring many orders of magnitude additional run time compared to the online algorithms. Indeed, the offline algorithm could not scale to any of the larger problems.

The POMCP+FL is slower than all other online methods. This is because even the single agent problems have a relatively large number of actions (movements + pinging beacons), and many observations. Thus, the branching factor is difficult to handle. On the other hand, POMCP+FL creates in some cases good policies, and in S_2, even the best policies.

Results on large problems Table 3 lists the results for the larger problems (M_1, M_2, M_3, M_4, M_5, L_1, L_2, and L_3), where the joint problem is too difficult for FSVI and POMCP. In these problems, the benefit of the forced localization (FL) method are obvious. For example, without FL and MF the success rate of the vanilla FSVI was 0 for M_3, M_4, M_5, and L_2, while FSVI+FL and FSVI+MF+FL often succeed. This advantage is also shown in the ADR. For example, in L_1 using FL approximately doubled the ADR achieved.

The benefit of modifying FSVI (MF) to include more pings, however, is less pronounced. Only in M_1 FSVI+MF provides better results than FSVI+FLm while in most cases, using only FSVI+MF in not as good as FSVI+FL. On the other hand, the combined FSVI+FL+MF provides additional leverage, at least in the largest problem, L_3.

POMCP-FL provided competetive results in some cases, but in most problems did not work as well as FSVI-FL. On the largest problems, POMCP-FL failed to provide strong policies, probably because the longer needed planning horizon requires deeper trees, which take a long time to construct.

In some sense, M_5 is the hardest of the large problems, as the single-agent path of each agent to its goal is conflicting in M_5, requiring much coordination for navigating the narrow corridors without collisions. Additional research is needed to develop methods that can better handle such problems that require strong synchronization. It may be that in such corridors, solving a small, short range joint problem is better.

Summary For very small problems, finding a solution to the joint problem provides the best policy, but takes a very long time. As such, only OPP was able to scale beyond very small problems, especially when using the FL method. This demonstrates that solving the naive joint MAPF POMDP problem is not practical, and hence, one would need to use some

type of decoupling technique, creating single agent problems that allow for much better scaling up. We also see here that domain specific techniques can help POMDP solvers in creating better policies, with respect to the synchronization of policies. Our localization additions were successful because, when considering a single agent, localization is less important, unless we are near the goal. However, when other agents are nearby, localization can reduce significantly the possibility for collisions, and hence, the need to replan, and the constraints during replanning.

6 Related Work

MAPF with unexpected delays has been studied in different forms (**????**). Yet limiting stochastic effects to delays simplifies the problem compared to SMAPF-PO, and these works all assumed perfect observability. UM^* (**?**) does support uncertainty over agents' locations but it allows agents to risk conflicts as long as their probability of occurring is beneath a given threshold. Similarly, prior work on multi-robot navigation assigned cost to collision and aim to minimize it. The SMAPF-PO is fundamentally different since we require avoiding any chance for having a conflict. Recent work (**?**) on MAPF with partial observability differs from SMAPF-PO in that they assumed control is not centralized and agents' visibility is based on line of sight. The latter assumption is different from our beacons-based observation model, precluding empirical comparison.

MAPF under movement uncertainty is a special case of Multi-Agent MDP (MMDP) (**?**). General-purpose MMDP solvers (**?**) do not exploit the specific structure of the SMAPF-PO problem, such as the limited interactions between agents. Similarly, online approaches for solving large MDPs, such as FF-Replan(**?**), are not expected to be effective in our domain and do not deal with partial observability. Dec-POMDP (**?**) is often used to model multi-agent problems under partial observability, where agents plan jointly, but execute their policies independently. This is different from our setting, where agents are controlled in a centralized manner and share their observations. Also, Dec-POMDP algorithms tend to scale poorly unless the interactions between the agents are limited to small predefined areas in the state space (**?**). Multiagent POMDP (**?**), where both planning and control are centralized, is an appropriate model for SMAPF-PO. **?** suggest an algorithm based on POMCP which exploits the locality of agent interactions. Their algorithm given singleton sets is similar to our POMCP, which does not scale as well as FSVI. Our approach can be viewed as a particular implementation of FT-POMCP leveraging the structure of the SMAPF-PO problem for factorization, and using prioritized planning to ensure independence. Factorizing the decision-making process in POMDPs has long roots (**??**, e.g.), and is particularly useful given several independent tasks. Factorization is less effective, however, when the coupling between the tasks increases, as happens in our problem where agents may collide. In tasks with high uncertainty concerning the current state, such coupling is common, and useful factoring is difficult. We used FSVI and POMCP but other alternatives for offline and online POMDP solvers exist (**???**, e.g.). Our contribution is not in an adaptation of a particular POMDP solver to

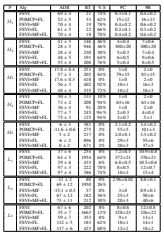

Table 3: Results for medium and large problems.

SMAPF-PO, but in our factoring and prioritization schemes. Replacing POMCP with, e.g., ABT, is likely to scale up only slightly, while improvements to the factorization can make a huge difference.

7 Conclusion

We studied the Stochastic MAPF with Partial Observability (SMAPF-PO) problem, which is a generalization of MAPF in which actions have stochastic outcomes and agents do not have perfect observability of the current state. We focused on a centralized control setup. While SMAPF-PO can be modeled as a single-agent POMDP problem, solving this POMDP is intractable even for small problems. We introduced the OPP approach, an online adaptation of Prioritized Planning. OPP has several non-trivial components which we describe, and we also propose two extensions that encourage the agents to actively localize. The results showed that OPP solves larger problems than an offline baseline. Yet, solving larger problems is still an open challenge for future work. Possimus placeat accusamus corporis officia temporibus, harum consequatur quidem.A pariatur quo maxime blanditiis porro dolores quos, eligendi odit nobis veritatis cumque vitae accusamus molestias eos est blanditiis, molestiae commodi totam temporibus optio perspiciatis itaque tenetur cupiditate?Consequuntur voluptas tempora voluptatem, nisi aliquam minus deserunt deleniti alias illo minima quis, assumenda vero perferendis aperiam culpa amet quasi dolorum officia alias?Nemo porro saepe dolores rerum exercitationem, quo magnam amet totam, nisi enim vitae aliquid officia iste tempore explicabo, eaque deleniti iusto odit similique dolor, ex dicta exercitationem adipisci distinctio alias deserunt qui dolore aut.